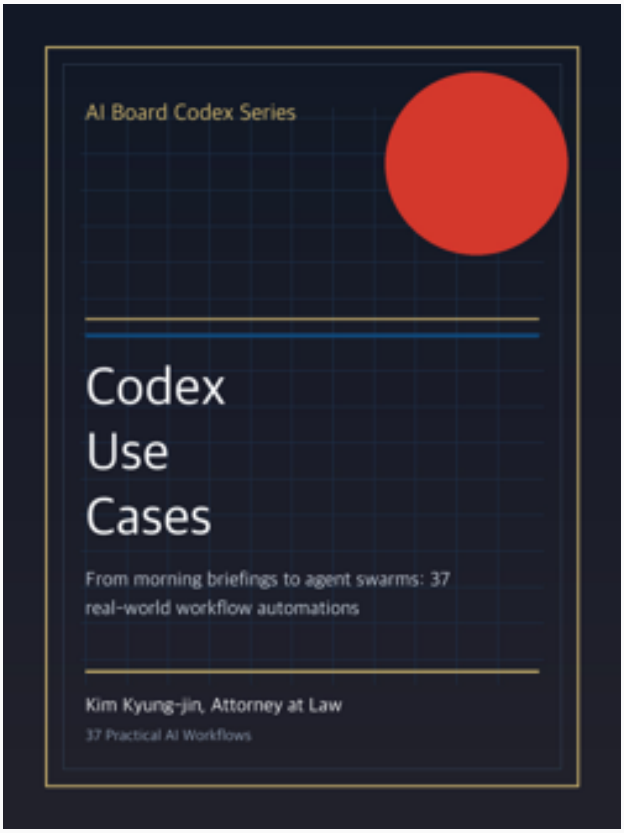
KIMKJ AI Library

37 Concrete Codex Use Cases

From morning briefings to agent swarms: 37 real-world workflow automations



Kim Kyung-jin

English PDF edition

37 Concrete Codex Use Cases

Kim Kyung-jin

This guide gathers 37 ways to connect Codex and AI agents to real work: personal routines, data processing, marketing, sales, documents, development, and browser control.

Table of Contents

37 Concrete Codex Use Cases - Chapter 1. Personalized Morning Briefing and Schedule Automation

Next Chapter

Chapter 1. Personalized Morning Briefing and Schedule Automation

Use AI agents (for example, Claude Co-work,

Codex

) to receive a fully personalized briefing every morning on your desktop or smartphone. In the past, you had to open each newsletter one by one and check your calendar, but now AI collects and summarizes all information in the background at a set time and sends it to Slack or your desktop. The core of this workflow is the use of scheduled tasks and connectors. If you grant the AI access to your Gmail, Google Calendar, and Slack, it can scan emails and understand schedules on its own. It gathers the latest industry news and trends through web search and filters out only the insights that are valuable to you.

Core setup and workflow:

Connector setup: In the AI settings menu, activate the Gmail, Google Calendar, and Slack connectors.

Automation scheduling: Set a scheduled task to run every morning at a specific time, such as 7:00 AM. Because the computer needs to be on, keep background execution enabled, or use a remote server like a Mac Mini.

Example prompt (enter this in the prompt field or save it as a skill):

"Create an automated workflow called 'Morning Briefing' that runs every morning at 7:00 AM. First, check my Google Calendar and summarize today’s key schedules and meetings in chronological order. Next, scan my Gmail inbox and list any emails from clients or team members marked as 'urgent' or 'important'. Finally, search the web and find three popular news articles related to 'AI automation and

SaaS

trends,' then summarize the key insights. Format all results into clean, well-structured text and send it immediately to my Slack #daily-brief channel."

Once you save this prompt once, the agent will read email and scrape news while you are asleep, then produce a complete morning report.

Next Chapter

37 Concrete Codex Use Cases - Chapter 2. Large-scale Receipt and Invoice Processing Workflow

Previous Chapter

Next Chapter

Chapter 2. Large-scale Receipt and Invoice Processing Workflow

For a finance team or a solo entrepreneur, organizing dozens or hundreds of receipts and invoices is a major time drain. Using an AI agent, receipts piled up in a folder in PDF and image format can be perfectly sorted and have their data extracted with a single command. The key is that the AI in this workflow does not read every document one by one as a single agent; it spawns sub-agents in parallel matching the number of documents and distributes the work. If there are 50 receipts, 50 mini-agents are deployed at the same time and the job is finished in just a few minutes.

Core setup and workflow:

Folder setup: Prepare a source folder named invoices on the desktop, an empty CSV file (log_invoices.csv) to store results, and a processed folder where completed files will be moved.

Define the extraction skill (Skill): Creating a skill.md in advance that states exactly which fields the AI should extract from receipts, such as issue date, sender, and amount, greatly improves accuracy.

Example prompt:

"Run the /process_invoices skill. Scan all PDF and image receipt files currently in the invoices folder (about 50 files) using parallel sub-agents. From each receipt, extract the 'Invoice Number', 'Sender', 'Invoice Date', and 'Total Amount' and write each value accurately into a new row in log_invoices.csv. Move processed files into the processed folder by automatically creating subfolders organized by payment date (YYYY-MM). When the job is finished, summarize the total invoiced amount and list files with missing data."

Given this instruction, the AI can finish the repetitive data-entry work in only 3 to 5 minutes and return a CSV file plus a cleanly organized folder structure.

Previous Chapter

Next Chapter

37 Concrete Codex Use Cases - Chapter 3. Perfectly Cloning and Building a Website or Landing Page

Previous Chapter

Next Chapter

Chapter 3. Perfectly Cloning and Building a Website or Landing Page

This is a workflow for coding a fully working HTML/React webpage using only a captured image from a design site (such as Dribbble) or a third-party website URL. It combines AI's strong vision analysis with frontend coding capability, so you can get a high-quality layout immediately without writing code from scratch. The key in this chapter is avoiding reckless "vibe coding" that hands design entirely to AI, and instead supplying your own design system documents or templates, such as

design.md

, in advance so the result keeps a consistent brand tone.

Core setup and workflow:

Build a design system: create a

design.md

file in the project root folder and predefine the brand color (Hex codes), typography (Geist, etc.), and button layout rules.

Provide visual context: capture a landing page layout you like from Dribbble and attach the image to the prompt window using drag and drop.

Example prompt:

"Create landing page code based on the latest Next.js and Tailwind CSS that perfectly replicates the layout and structural layout of the attached website UI capture image. However, ignore the capture's colors and fonts, and strictly apply the brand design system from the

design.md

file in this project root (button colors, typography, dark mode support, etc.). Place a navigation bar at the top and a responsive Tally.so embed form in the center for collecting user email addresses. When the code is complete, open a Playwright browser and run a visual QA check yourself to verify the screen renders correctly."

After AI interprets the design and writes the code, it validates the rendered output by taking screenshots through built-in browser control permissions, and if there is any error, it debugs on its own and completes the final code.

Previous Chapter

Next Chapter

37 Concrete Codex Use Cases - Chapter 4. Building Full-Stack SaaS and Web Applications

Previous Chapter

Next Chapter

Chapter 4. Building Full-Stack SaaS and Web Applications

Beyond a simple static website, this chapter covers workflows for building complex

SaaS

applications or interactive browser games like Chrome Dino Run, where user authentication (Auth), a backend database, and payment system integration are required. For complex projects, it is important to use Plan mode so that the AI is led to design and get approval for the database schema and architecture first, instead of writing code directly.

Core setup and workflow:

Forced technology stack (

agents.md

): In

agents.md

, clearly state, for example, to always use Next.js client components, Clerk for authentication,

Supabase

for the database, and Tailwind CSS for UI, so the AI cannot use outdated patterns or arbitrary libraries.

Dependency installation and scaffolding: Grant the AI permission to run terminal commands so it can install packages automatically.

Example prompt (SaaS and browser game integration example):

"We are going to build a full-stack

SaaS

application with user authentication in an empty directory. Start in

/plan

mode and first create an implementation plan. The requirements are: 1) Implement social login using Clerk. 2) Connect

Supabase

and design a database schema to store each user's top score. 3) Implement a playable HTML/Canvas-based clone of the Chrome Dino Run game inside a dashboard that only logged-in users can access. 4) Record the score to

Supabase

when the game is over. Start from Next.js project scaffolding (initial setup) according to the rules in my

agents.md

, and validate each milestone with test code to confirm it works."

After planning, the AI links the user authentication backend, then implements a full-stack app by sequentially building the JavaScript-based game loop with a physics engine and collision detection.

Previous Chapter

Next Chapter

37 Concrete Codex Use Cases - Chapter 5. Mobile app (iOS/Android) planning and development

Previous Chapter

Next Chapter

Chapter 5. Mobile app (iOS/Android) planning and development

Even without any coding knowledge, you can build fully working mobile applications (for example, a music player, a dopamine-blocking app) from just idea sketches and planning documents, then run and verify them directly in the local Mac Xcode simulator. With

Codex

’s powerful

/goal

command, you can throw a roadmap document loaded with dozens of tasks and instruct the AI to implement the entire roadmap autonomously for hours without human intervention.

Key setup and workflow:

Roadmap and PRD prep: Prepare detailed docs/prd.md (product requirements document) and product_roadmap.md (task list) at the project root.

Autonomous driving (

/goal

) execution: Give the AI a goal and run an endless loop in which it codes and debugs on its own until all task list checkboxes are marked complete.

Example prompt:

"Write the full Swift code for the iOS mobile app 'Math Block' defined in

/goal

docs/prd.md. This is a dopamine-blocking app that requires the user to solve a math problem before allowing scrolling in SNS apps like Instagram. Follow the 60 tasks in product_roadmap.md in order, and mark each completed task (easy/medium/hard level choices, UI rendering, etc.) with X in the roadmap document when each milestone is done. For tasks at Medium difficulty or higher, integrate RevenueCat and implement a paywall screen that takes users to a $5-per-month payment flow. Keep working in Auto-mode autonomously and do not stop until all tasks are complete and Xcode build errors are zero."

After receiving this command, the AI sets up the project by itself, writes thousands of lines of code, and fixes bugs it introduces. After about one to two hours, coding is usually done, and the user can open local Xcode in the terminal, launch the simulator, and immediately check the finished mobile app running on a phone-shaped screen.

Previous Chapter

Next Chapter

37 Concrete Codex Use Cases - Chapter 6. Make Your Own Chrome Extension: Build a Personal Productivity Tool in 5 M

Previous Chapter

Next Chapter

Chapter 6. Make Your Own Chrome Extension: Build a Personal Productivity Tool in 5 Minutes

When we work every day, the web browser Google Chrome gives us endless ways to extend its functionality through extensions. In the past, creating one Chrome extension required a high barrier to entry: you had to understand JavaScript, HTML, CSS, and Chrome’s complex manifest permission model. Now, with AI agents like

Codex

or

Claude Code

, even a beginner with no coding background can build and install a Chrome extension in just 5 minutes.

In this chapter, we will practice building an extension that converts a webpage to Markdown and then downloads it as a clean PDF file. That ability is useful for saving web content as context for AI learning or for scraping documents for offline use.

1. Prepare the workspace

You do not need heavy development tools to build an extension. Start by creating an empty folder on your desktop.

Create an empty folder named Chrome_Extension on your desktop.

Open a terminal (or command prompt), move to that folder, and run the

codex

command to launch the AI agent. Your blank canvas is now ready.

2. Give instructions to the agent (example prompt)

When writing the prompt, the key is to state exactly what you want and add that you are a beginner and want installation instructions too.

Example prompt input: "Please create a Google Chrome Extension (Chrome Extension) that converts the content of the currently open webpage to Markdown format and saves it to my computer's

Downloads folder. I am a coding beginner, so I need you to guide me step by step through the full process of how to deploy and install this extension in Chrome so I can use it myself. Keep it simple and implement it so it runs without unnecessary complexity."

3. Review the AI workflow and result

Once you issue the command, the AI will quickly confirm that this task is feasible and begin planning and writing the needed files, starting with manifest.json, then HTML for the popup, and JavaScript for Markdown conversion.

This usually takes about 1 to 2 minutes. You can watch in real time through the terminal as the AI automatically generates about four or five files in your folder. When it finishes, it will provide a friendly guide saying, "The extension is ready. Install it as follows."

4. Install and deploy directly in Chrome (beginner guide)

Registering the files the AI created in Chrome is done in just a few clicks. Follow these steps to install it.

Open Chrome and go to chrome://extensions/ in the address bar to open the extension management page.

Click the Developer mode toggle in the top right and turn it on.

On the top left, click the button labeled Load unpacked.

Select the Chrome_Extension folder on your desktop where the AI has just generated the files.

Installation is complete. Click the puzzle-piece icon in the upper right of Chrome and pin the extension you just made.

5. Test and extend it

Now go to a news article or technical document page with heavy content and click the icon for the extension you just created. With one click, unnecessary ads and menus disappear, and you can confirm that the remaining text and core structure are saved to your Downloads folder as a clean Markdown file. This Markdown format is the easiest for AI to consume when feeding documents.

Practical tip: this skeleton can be expanded endlessly. Try adding prompts such as, "Build me an extension that scrapes LinkedIn profiles and sends them directly to my

CRM

," or "Modify this into an extension that saves selected Gmail content to Google Drive with one click." You can create your own custom tools without paying for expensive productivity software.

Previous Chapter

Next Chapter

37 Concrete Codex Use Cases - Chapter 7. Automatic Cleaning and Preprocessing of Messy CSV Data: Handling Data wit

Previous Chapter

Next Chapter

Chapter 7. Automatic Cleaning and Preprocessing of Messy CSV Data: Handling Data with Natural Language without Regular Expressions

If you are a data analyst, marketer, or sales manager, you know the worst task: cleaning messy Excel (CSV) data received from coworkers or outside partners one row at a time. Dates may be mixed as 2024-01-01, Jan 1st, and 01/01; company names come in random casing; phone number formats differ; and sales amounts may be buried inside free-text Notes.

Before, solving this meant working with Python and the Pandas library, or writing head-splitting regular expressions. But now, if you simply tell an AI in natural language what to clean, it can generate a preprocessing script and produce a fully cleaned file in a few minutes.

1. Defining the environment and the problem

Assume you have a raw_contacts.csv file in a project folder on the desktop. It has the following issues.

Phone numbers are a mess: 1234567890, 123-456-7890, (123) 456 7890, and so on.

Company names are chaotic with mixed case and spacing, such as Apple Inc., apple llc, APPLE.

Notes: Sentences like "The estimate for this deal is around $45,000" include numbers in text, so only the amount should be extracted.

2. Instructing the agent (example prompt)

Beginners do not need to write complex Python code. You only need to list the data problems and the exact shape of the final output you want.

Example prompt input: "The raw_contacts.csv file in this folder is a very messy contact list. Analyze this file and clean it perfectly based on the following rules.

Phone column: The formats are all inconsistent; normalize all phone numbers to (XXX) XXX-XXXX using Regex.

Company name column: Standardize irregular casing to Title case and remove unnecessary punctuation at the end of the name, such as periods or commas.

Notes column: The text contains dollar signs or amounts mixed inside. Extract only numeric values as 'estimated_deal_value' and keep only the largest number if there are multiple. If no amount exists, set

it to Null (blank). Write a Python script to perform all these cleaning steps automatically and save the result to a new file called clean_raw_contacts.csv.",

3. AI workflow and result verification

After you enter the prompt, the AI reads the data and starts writing its own Python code. It identifies complex phone number patterns using Regex and writes and runs a script of over 100 lines using the Pandas library.

About one to two minutes later, the agent finishes and reports completion while leaving a clean_raw_contacts.csv file in the folder. Opening it in Excel, you can see every phone number is neatly standardized to (XXX) XXX-XXXX, the messy company names are normalized, and the number 45000 has been accurately extracted from a long text sentence and placed in the new column.

Advice for beginners: Regex is a strict grammar that can easily waste hours as you debate where to place each parenthesis. If you provide natural language examples like Before/After and hand it over, it will write accurate code without typos. You can experience a process that cuts data preprocessing time from five hours to just five minutes.

Previous Chapter

Next Chapter

37 Concrete Codex Use Cases - Chapter 8. Merging multiple spreadsheet datasets and creating pivot tables: context-

Previous Chapter

Next Chapter

Chapter 8. Merging multiple spreadsheet datasets and creating pivot tables: context-aware AI data joining

In a business setting, data is rarely gathered neatly in one place. For example, Team A works with customer data downloaded from a

CRM

platform, Team B handles billing data, and Team C manages webinar participation data in Excel. Your manager tells you, "Merge these three files and create one consolidated report that shows each customer's billing status and marketing response rate at a glance."

Using common Excel functions such as VLOOKUP or INDEX/MATCH requires a perfect matching identifier key like customer ID. But the marketing file only has company name, and even then names are not exact; one file may list Apple Inc. while the marketing file lists Apple. In the past, this required manual verification and hand-crafted matching, but AI can merge, or join, these records intelligently by understanding context and meaning.

1. Set up the working environment

Assume the project folder already contains three files.

crm

_data.csv (customer ID, company name, contract size, account executive)

billing_data.csv (customer ID, past-due invoice amount flag)

marketing_engagement.csv (company name, webinar attendance, NPS satisfaction score)

2. Instruct the agent to merge the data (example prompt)

The key is to explain the mismatch across files to the AI clearly and ask for context-aware merging.

Example prompt input (data merge): "There are three files in the current folder:

crm

_data.csv, billing_data.csv, and marketing_engagement.csv. Merge these files into one so each customer has a single view showing contract value, billing delinquency status, and marketing NPS

score. Note that the marketing file does not use Customer ID; it uses Company Name as the identifier. Also, company names in the

CRM

and marketing files may not be identical, for example Apple Inc. versus Apple. Use contextual understanding to merge similar company names smartly. Save the merged result as customer_combined_view.csv."

When you send the prompt, the AI writes a Python script using similarity-based matching rather than exact matching and merges all three files into one in about one minute.

3. Create a pivot table and analyze performance with the merged data

Once the data is merged, it is time to extract a management-ready summary pivot table and insights. Ask follow-up questions in the same chat.

Example prompt input (pivot table and analysis): "Great work. Now use the customer_combined_view.csv you just created to make something like an Excel pivot table. Put 'Account Executive' in Rows, 'Product Type' in Columns, and the sum of 'Total Revenue' in Values, then summarize it. Also analyze this table to identify which account executives are missing their monthly quota and how large the gap is, and summarize that in text."

4. AI output and practical use

The AI can instantly show the result as text and code form, such as Pandas pivot_table, instead of requiring mouse clicks in Excel. The output table cleanly summarizes, by salesperson names such as James and Angela, how much revenue was generated in each product category like hardware and software.

It also returns deeper insights as requested in the prompt. "Analysis results: only one account executive, Marcus Johnson, failed to reach the target. Marcus had a target of 960K but has reached only 822K, leaving a gap of 137K, or about 15%. James Wilson and Angela are posting strong results, but they are focused only on software licenses."

If you use this workflow, even beginners can achieve the same effect as hiring a senior data engineer and a data analyst without writing code. You gain the speed to open your email and immediately send a report to your manager saying, "Marcus Johnson is 15% behind on revenue, so action is needed."

Previous Chapter

Next Chapter

37 Concrete Codex Use Cases - Chapter 9. Preemptive Outlier and Missing Data Detection in Reports: Running a Perfe

Previous Chapter

Next Chapter

Chapter 9. Preemptive Outlier and Missing Data Detection in Reports: Running a Perfect AI Audit Right Before Executive Meetings

For beginners: why AI data audit is needed. Every data analyst or practitioner has faced this moment. You prepare an Excel report for an executive meeting or a major presentation, press send, and five minutes into the session someone says, "Hold on, this revenue figure looks wrong." If a report goes out with absurdly high revenue, negative profit values, or missing data, trust in the data collapses.

Before, people had to scan through thousands of Excel lines by eye or build complex filters and formulas to spot outliers. Now we can use AI agents like

Claude Code

and

Codex

as a second set of eyes. You can simply ask in natural language, "Find anything suspicious in this data," and the AI will flag logical errors before they reach the report.

Practical use case: imagine a file named Q1_revenue_report.csv on the desktop for an urgent Q1 revenue audit. The file holds tens of thousands of rows with customer name, region, product, revenue, account owner, and margin percentage. There are six critical issues hidden in it that are hard to catch by eye.

A normal revenue level is around 10,000 dollars, but one row contains 980,000 dollars due to a typo.

The Revenue column contains a negative (-) value.

It is a Q1 (January to March) report, yet some rows contain dates incorrectly entered as December 15, 2024.

There is a missing margin value represented as a blank cell.

Some rows include duplicated customer IDs.

Prompt guidance for beginners: the prompt you issue to an AI agent in this situation should be specific and rich in context. Clearly state urgency and the exact types of errors you want it to look for.

Example prompt: "There is a file named Q1_revenue_report.csv in the current folder. I need to present this Q1 revenue report to my executive manager in one hour. I am about to enter the meeting, and I need you to audit this data now so I can be sure there are no suspicious or logically incorrect values before I present. Can you do that?"

Prioritize scanning for these items:

Outliers that are unreasonably high or low, such as abnormal revenue values.

Missing values and empty fields.

Duplicate rows.

Incorrect date formats that are not part of Q1 (January to March).

Any other conditions that do not match business logic.

If you find issues, summarize them in a list, including which row each issue is on and why you judged it to be a problem."

What happens when the AI runs: once you submit this prompt, the AI agent immediately reads the CSV and starts scanning. In only one to two minutes, it outputs results like these.

Critical issue 1 (Outlier): "I found an outlier of $980,000 in the Revenue column. Given that other average revenue values are between $5,000 and $15,000, this looks like a serious input error."

Critical issue 2 (Logical error): "I found a revenue value marked as negative (-). In normal sales data, revenue cannot be negative."

Critical issue 3 (Incorrect date): "This is a Q1 report by fiscal quarter, but it contains a date recorded as December 15 (12/15). That record should not be in the Q1 dataset."

Warning (Missing data): "The Margin percentage field is missing and blank in row 12."

A practical tip for beginners on prompt iteration: the AI is strong, but sometimes it still misses things even with one instruction. If it skips detection of duplicates in the output above, do not lose confidence. Ask it again and press for detail.

Follow-up prompt: "Thanks! But I do not see any mention of duplicates in your result. It looks like duplicate values may have been missed. Can you check again thoroughly for duplicate records and any other mismatches?"

When you ask again, the AI can respond, "I am sorry. After rescanning more deeply, I found duplicate billing entries with the same customer ID. I also found another outlier: row 11 has a Units value of 500 units, while the normal range is usually 20 to 50 units." This way, five minutes before the meeting, AI helps you fix critical errors in advance and submit a clean, reliable report.

Previous Chapter

Next Chapter

37 Concrete Codex Use Cases - Chapter 10. SQL query debugging, speed optimization, and automatic View conversion:

Previous Chapter

Next Chapter

Chapter 10. SQL query debugging, speed optimization, and automatic View conversion: using AI to fully refine slow and complex queries

For beginner-level understanding: when you work with data in

SQL

that executes but is still a 'bad'

SQL

query, you inevitably face

SQL

itself. A major frustration for beginners is when there is no syntax error and execution succeeds, yet the result is completely wrong or the query takes more than 4 minutes to run. Searching through dozens of lines of JOIN and SUBQUERY to find what went wrong is painful. There is also the annoying routine of manually updating hard-coded dates such as '2024-01-01' in a weekly-Monday query every week. With an AI agent, you can quickly debug

SQL

logic errors, optimize speed, and convert them into clean automated views in no time.

Use case 1 for practice: Suppose there is a file called slow_query.

sql

in the company database for data extraction. The result is accurate, but it is written inefficiently and takes over 4 minutes every time it runs.

Example prompt input: "The query in slow_query.

sql

in the current folder returns the correct result, but it takes more than 4 minutes to run. There seems to be an inefficient part in how it was written. Analyze this query and explain what exactly is slowing it down. Then rewrite it into a fully optimized query that runs much faster. Finally, estimate the expected speedup percentage compared with the original."

AI workflow and output: After analyzing the code for about 1 to 2 minutes, the AI returns an analysis report and a revised query.

Bottleneck cause: "The analysis shows the main cause is a Correlated Subquery inside an EXISTS clause. This shape reruns the subquery for every row in the main query, so as data grows the runtime slows down exponentially (cubic blowup). That is why it took 4 minutes."

Solution and optimized query: The AI provides an optimized

SQL

query rewritten with a much cleaner JOIN structure by removing unnecessary DISTINCT and heavy subqueries.

Performance expectation: "Based on real production datasets, this optimized query should run about 20 to 50 times faster than the original."

Use case 2 for practice: Converting hard-coded queries into dynamic views. There are three

SQL

files that produce weekly active users (WAU), revenue target achievement rate, and related metrics every Monday. All include manually hard-coded dates, so they must be edited each week. I want to make them into automated views for continuous use.

Example prompt input: "I have three

SQL

query files that I run manually every Monday morning. They work, but they are messy.

Convert each of these three queries into

SQL

CREATE VIEW code that produces clean view tables. The important part is to remove all hard-coded dates in the queries (for example, '2024-03-01') and replace them with dynamic date functions like CURRENT_DATE so that running on every Monday automatically returns the last 7 days of data. Add thorough comments explaining what each part of the code does."

AI workflow and output: The AI reads all three files, identifies the issues, and writes the view creation scripts. It identifies hard-coded dates and replaces them completely with dynamic functions such as WHERE order_date > CURRENT_DATE - 7. Even beginners can then build a strong automated setup where they just run SELECT * FROM weekly_active_users and always get the latest correct data without opening a complicated query window and editing dates.

Previous Chapter

Next Chapter

37 Concrete Codex Use Cases - Chapter 11. Legacy code and database schema automatic documentation (Data Dictionary

Previous Chapter

Next Chapter

Chapter 11. Legacy code and database schema automatic documentation (Data Dictionary): decoding the inheritance of the past like a cipher with AI

To explain this concept for beginners: when you are moved to a new department or inherit tasks left by a departing predecessor, the uncomfortable moment comes when you face legacy output with no documentation at all. You open a Python script and there is not a single comment; variable names are x, y, temp_data. You open the database and see columns like col_mrr_24, sub_tier, acq_src_1, written like codes, so you cannot tell what data they represent. In the past, you had to decode this by stepping through old code line by line and debugging or by asking around in other teams. With AI, though, you can convert that coded, opaque code and schema into a complete reference manual, a data dictionary.

Practical case 1: decode and refactor a nameless legacy Python script. There is an unattended undocumented_script.py file sitting alone in the project folder.

Example prompt input:

"There is a file called undocumented_script.py in the current folder. It was written months ago, and honestly there are no comments and the variable names are a mess, so I cannot remember exactly what this code does.

Please read this script from start to finish and explain what purpose it serves in plain English terms that a beginner can understand. Then rename all meaningless variable names like x and df1 to intuitive, descriptive names that match each role the code performs. Finally, add appropriate comments to each functional block and add a comprehensive top-level docstring explaining how to use this script, then save it as clean code that can be put into production immediately."

AI workflow and result: AI immediately analyzed the code and gave a plain-language summary in the terminal, such as, "This script imports user data and aggregates revenue in six steps." Then it replaced the unknown df1 variable with monthly_revenue_data, inserted helpful comments and documentation blocks throughout the code, and saved a fully regenerated Python script. That is the moment a dormant legacy script comes back as modern code anyone can understand.

Practical case 2: document a cipher-like database schema as a full Data Dictionary. A data engineer dropped a database schema dump file and left. Table and column names are abbreviated, so their meaning is unclear. As a data analyst, you need a compiled Data Dictionary so you know what each column means.

Example prompt input:

"There is a schema dump file (schema.

sql

) in the folder that I got from the engineer. It contains many tables with abbreviated column names that look like ciphers.

Please analyze the entire schema file and write a complete Data Dictionary for me in Markdown table format. Important note: assume this database is for analytics in a

SaaS

business. Using that business context, infer in a logical way what each cipher-like column likely means in real operations, and add explanations in the table."

AI workflow and result: telling the model that "this is a

SaaS

business database" in the prompt was the key move. With that business context, AI used insight to build the dictionary.

AI found the MRR column and, using the

SaaS

context, interpreted and documented it not just as a token but as "Monthly Recurring Revenue: a core

SaaS

metric."

It inferred the acq_source column as "Customer Acquisition Source" and added that explanation.

When AI saw the discount_applied column, it added a sharp business tip at the bottom of the Markdown document: "Note: based on this schema alone, it is ambiguous whether MRR is pre-discount or post-discount; please confirm this with the data engineer."

With AI, the dictionary that once required following an engineer around all day for questions can now be completed in 2 to 3 minutes as a clean Markdown table and put into real work immediately.

Previous Chapter

Next Chapter

37 Concrete Codex Use Cases - Chapter 12. Competitor websites, SEO, pricing policy, and review scraping analysis:

Previous Chapter

Next Chapter

Chapter 12. Competitor websites, SEO, pricing policy, and review scraping analysis: a single view of our business weaknesses and opportunities

Foundation for beginners: Why hand competitor analysis to AI? When you are launching a new business or entering a competitive markets, such as landscape firms or web design agencies, it is essential to analyze how competitors get traffic, how they set prices, and what customer reviews say. But visiting dozens of competitor websites one by one to collect data and organize it into spreadsheets is hard labor that can take weeks to months. With AI agents like

Claude Code

and

Codex

, you can scrape competitors' SEO status (backlinks, organic traffic), social media awareness, and pricing structure with one prompt and summarize everything in a complete dashboard. The AI even analyzes the data and gives strategic advice, such as "This competitor has strong traffic but slow mobile speed, so we should focus on mobile optimization."

Practical use case: building a competitor website SEO and health check dashboard. Suppose you run a voice AI agent agency and want to analyze your competitors' SEO condition.

1. Prepare the environment and configure APIs. To fetch competitor data, it is best to use an API such as Data for SEO, a credential or access path to retrieve data from specific software. In the project folder, create an agents.md file and pre-load the AI with the instruction: "Use the Data for SEO API for SEO analysis."

2. Give instructions to the agent (example prompt). For beginners, give clear instructions about the role the AI should play and the outcome you expect.

Example prompt input: "I am planning to launch a new landscape company. I will provide URLs of competitors I want to benchmark, such as competitor-landscape.com. Please scrape and analyze the current business status of this site.

Collect the following items using Data for SEO API or web scraping tools such as Firecrawl, and visualize them in an HTML dashboard format.

SEO health check: check organic traffic, backlink count, and whether there are broken links.

Marketing strategy: monitor whether Google Ads or Facebook Ads are running and evaluate social media awareness.

Pricing structure and reviews: summarize the pricing policy listed on the site and analyze the available customer reviews.

Once collection is complete, based on the analyzed data, also prepare an executive report that includes the competitor’s Weaknesses, Improvements, and Keyword suggestions for targets we should pursue."

Once the prompt describing the AI workflow and output is entered, the AI immediately activates the scraper and explores the competitor website. In about one to two minutes, it produces and shows a dashboard with polished UI. The dashboard presents intuitive findings such as "Competitor Score: 0/100 (no backlinks, organic traffic exists)." What is more useful is the analysis the AI provides: "Competitor weakness: the pricing policy is unclear, which may cause customer churn. Recommendation: place a transparent pricing table on the main page to differentiate us." You can get this level of business consulting output in a few minutes, equivalent to the productivity of hiring ten employees.

Previous Chapter

Next Chapter

37 Concrete Codex Use Cases - Chapter 13. Google Maps Lead collection and custom cold email writing: the automatio

Previous Chapter

Next Chapter

Chapter 13. Google Maps Lead collection and custom cold email writing: the automation spell behind effective sales

Foundation for beginners: At the core of hyper-personalized (Enrichment) B2B sales, far beyond basic lead collection, is finding high-quality potential customers (Lead) and sending cold emails with high reply rates. Most people open Google Maps, search for "Roofer in Los Angeles," and copy company names and phone numbers into Excel. But an AI agent not only fully automates this, it also performs the magic of "data enrichment."

It does not just scrape addresses. It automatically visits a company’s TikTok, Instagram, Facebook, and YouTube, then collects the owner’s name and recent activity. On that basis, it can generate refined cold email drafts in unlimited quantity, far more polished than text written manually.

Practical use case: lead collection and sales automation for a roof repair company (Roofer) in the LA area. Imagine you are a freelancer building websites or a sales rep.

1. Give instructions to the agent (prompt example): connect Google Maps scraping capabilities with personalized email creation.

Prompt input example: "I run a website design agency. Starting now, build a pipeline dashboard that collects leads and writes sales emails.","

"Scraping: When I enter the query 'Roofers' and the location 'Los Angeles,' use Apify or the Google Maps Scraper API to collect at least 20 business records. You must capture company name, review rating, website, and phone number.","

"Enrichment: Scan each company website and social channels (such as Facebook) to identify the business’s latest activity and characteristics.","

"Hyper-personalized icebreaker: based on the collected data, create one icebreaker sentence per company for use as the first line in sales calls or cold emails, and organize it in Google Spreadsheet. The sentence must not feel mechanical and must be fully personalized."

AI workflow and output: The AI agent activates the scraper and identifies companies with 18 websites in the dataset. It then records the collected data in the spreadsheet and writes outstanding custom email drafts (icebreakers) like these.

"Collected data example: a business with 80 years of tradition and a record of completing more than 130 repairs.","

"Cold email draft written by AI: CEO, I was deeply impressed by your strong legacy of over 80 years and the more than 130 successful roofing repair jobs. I would like to learn more about your philosophy in the LA market and have a brief discussion about a full website refresh that can reflect it.","

Even to a beginner, this email is in a completely different league from basic copy-and-paste spam emails. In another example, it also generates hyper-personalized, storytelling-style emails automatically, such as, "I saw your Facebook post and noticed you drove for 40 minutes through the rain in the Paramount area to repair a burst pipe at 2 a.m.; that level of commitment must have won over your customers." At this point, you only need to press the send button.

Previous Chapter

Next Chapter

37 Concrete Codex Use Cases - Chapter 14. Meta ads library scraping and ad performance dashboard: Unpacking the se

Previous Chapter

Next Chapter

Chapter 14. Meta ads library scraping and ad performance dashboard: Unpacking the secret behind winning ads

For beginners, an advertiser trying to sell products or promote services usually asks, What kind of ad should I build? The place to watch the ads competitors are currently spending on is the Meta Ads Library. Yet, scanning hundreds of ads by eye to sort out which ones are video, image, or caption strategies is nearly impossible. With AI, you can simply enter a competitor's ad library link and build a professional dashboard that scrapes dozens of ads and lets AI analyze copy and hooking points by itself.

Practical use case: Batch analysis of 40 competitor ads for a Med Spa

1. Instructing the agent (example prompt): After preparing the Meta ad library URL for competitors, instruct the agent to scrape it and build an analysis dashboard.

Example prompt input: "Build me an 'ad analysis dashboard' that analyzes Facebook ads (Meta Ads). When I provide a Meta ad library URL for a specific brand (for example, a U.S. Med Spa company), use a scraper API to extract 40 currently active ads that are running recently."

The dashboard should include these features.

Media format classification: Separate the 40 collected ads by format, and show how many are Video, Image, and Carousel, so it is clear which format is the core of the campaign.

AI ad performance analysis: When a specific ad is clicked, AI should evaluate the ad's Effectiveness. It should assess whether the opening hook, target audience settings, and call to action (CTA) copy are appropriate.

Improvement suggestions: For ads expected to perform poorly, provide textual feedback with stronger copy or specific suggestions.

AI workflow and results: In the built dashboard, after entering the URL and pressing Scrap Run, you get a complete analysis in just a few minutes.

Ad strategy insight: "Among 40 ads, the analysis shows 19 Carousel ads, 12 Image ads, and 9 Video ads, so the competitor, Vagar Pro, is primarily using Carousel ads." That gives us a clear hint that we also need to test Carousel ads.

Copy analysis and fact-grounded critique: After AI analyzes the copy of a specific ad, it gives expert-level feedback like, "CTA analysis: In a Med Spa service business, the phrase 'Contact us' is too vague and not effective. To improve performance, it should be made more specific, such as 'Book a consultation' or 'Schedule treatment'." This is a workflow that lets almost any beginner build a strong ad analysis strategy without paying a marketing agency millions of won.

Previous Chapter

Next Chapter

37 Concrete Codex Use Cases - Chapter 15. YouTube Channel Strategy Planning and Thumbnail/Asset Generation Automat

Previous Chapter

Next Chapter

Chapter 15. YouTube Channel Strategy Planning and Thumbnail/Asset Generation Automation: A Never-Ending Content Burst Generator

Concept for beginners: finish YouTube strategy through design with AI. To grow a YouTube channel, you need to analyze competitor view counts, write hook scripts that pull clicks, and make thumbnails that grab attention. In the past, this meant running analytics tools, writing scripts with ChatGPT, then opening Photoshop or Canva and editing thumbnails by hand. Now an AI agent connects all of this into one pipeline. It scrapes competitor data to find opportunities (planning), writes scripts (content), and automatically creates responsive thumbnails as design assets that change layout with text length in just a few minutes.

Practical use case: analyzing competing YouTube channels and batch-generating dynamic thumbnails.

1. Instructing the agent (example prompt): Tell the AI to handle strategy planning and asset generation at the same time. (It is useful to place an image in the project folder that serves as a guideline for the thumbnail template beforehand).

Example prompt input: "Build me a \"YouTube strategist agent\" pipeline for growing my YouTube channel."

Step 1: Competitor analysis. If I give you the names of rival channels (for example, Jack Roberts channel), analyze their views, subscriber count, and engagement-rate metrics and compare them with my channel. Then, based on their top-performing content, pull out the \"content gaps\" I need to fill and five new video ideas I should create. For each idea, include a high-viral-potential Hook and a title. Summarize this analysis in Google Slides format and share it as a URL.

Step 2: When the brand asset (thumbnail) production idea is finalized, use my photo (avatar) and guest logo/photo in the project folder to automatically design a YouTube thumbnail. Write the code so that the font size automatically adjusts based on title text length, and if there are two photos of me and the guest, the layout dynamically and cleanly adapts to include both.

AI workflow and output

Google Slides report: The AI delivers a Google Slides URL within minutes. Open the slides and you will see top-performance metrics of competitors like Jack Roberts, plus five breakout title options and the first 3-second Hook for each script written professionally for my channel.

Dynamic thumbnail auto design: I tell the agent, \"The next week\'s workshop topic is \"

Claude Code

for Beginners\" and the hosts are me and Alex.\" The AI then uses design skills to align font sizes perfectly and immediately generates a thumbnail with the two avatar images cleanly composited. Beginner YouTubers and marketers no longer need to stay up all night doing thumbnail edits and title writing every week. They now have a strong system that can output every asset from YouTube planning to design with a single command.

Previous Chapter

Next Chapter

37 Concrete Codex Use Cases - Chapter 16. Social-media repurposing from YouTube and meeting notes (LinkedIn/Twitte

Previous Chapter

Next Chapter

Chapter 16. Social-media repurposing from YouTube and meeting notes (LinkedIn/Twitter): an endless content generator

Conceptual primer for beginners: Why should you let AI handle content repurposing? For creators, solo entrepreneurs, and marketers, posting on social media platforms such as LinkedIn and Twitter is mandatory. But creating new ideas and writing something fresh every day carries the pain of production. You already own many strong raw content assets: YouTube videos you recorded, podcast transcripts, and even insight-rich conversations from your weekly video meetings captured in recording apps such as Gong or Granola.

In the past, repurposing this material meant rewatching videos from the start, typing everything out, summarizing to fit Twitter length, and adjusting tone to match LinkedIn style, a process that took many hours of manual work. With AI agents now, you can scan your existing content, evaluate it by Viral Score, and generate limitless publication-ready post drafts by replicating your own tone.

Case in practice: Every Saturday at 8 AM, build an automatic content planning dashboard. The core of this system is teaching AI with "your best performing past content." Instruct the agent to adopt your distinctive style, for example direct language, presenting counter-positions, using concrete numbers, and using short sentences.

1. Environment setup. Create a folder named Content_Machine on your desktop. Gather your YouTube scripts and podcast transcript files in it. Then create a my_style.md file and paste in samples of your past strong LinkedIn posts.

2. Instruct the agent (example prompt). Give very specific instructions to create a repurposing skill.

Example prompt input: "I am a business owner who must post on social media every week. Please scan all my recent YouTube scripts in this folder and all weekly meeting note text files recorded in the Granola app.",

"Insight extraction: From these texts, extract five core topics, insights, and quotes with the highest viral likelihood and assign a score from 1 to 100. Prioritize contrarian takes, especially opinions that run against what most people assume.",

"Hyper-personalized writing: Based on those five topics, write one Twitter thread of 6 tweets and one professional LinkedIn post for each.",

"Tone enforcement: When writing, you must mirror my tone defined in my_style.md at 100%. That tone is direct, with minimal emojis, and uses concrete numbers.",

"Create an automation schedule to send a summary of all these outputs as a Google Doc or HTML dashboard to my email or Slack every Saturday at 8 AM.",

AI workflow and output: The AI immediately reads the scripts and meeting notes, and then captures your meeting line, "If we only do SEO optimization, we are already behind." From that, it writes highly hooking Twitter threads and LinkedIn posts. You just drink coffee, read what AI wrote, pick what you like, and copy it for scheduled posting. It is a reliable way to escape idea drought forever.

Previous Chapter

Next Chapter

37 Concrete Codex Use Cases - Chapter 17. Fully Automated Video Editing and Branding Asset Injection: Automated Vi

Previous Chapter

Next Chapter

Chapter 17. Fully Automated Video Editing and Branding Asset Injection: Automated Video Editing from Start to Finish

For beginners: what if a coding agent edits video? "AI that helps with coding can edit video?" Beginners may find this odd. But video editing is also a workflow that can be controlled through code. If you hand off a 1-hour webinar, Zoom meeting, or interview video to an editor so it becomes five 1-minute core clips for TikTok or YouTube Shorts with captions and an outro logo video, it usually takes tens of thousands of won per piece and several days of time. But if you give the agent permission to use FFmpeg or assign it as a Skill, it can read the transcript, decide that this section is the most engaging, and then cut the footage cleanly based on exact timestamps.

Use case in practice: extracting 5 LinkedIn Shorts from a 1-hour live workshop

1. Prepare the environment. Create a Video_Highlights folder. Put the original workshop files (workshop_speaker_view.mp4 and workshop_screenshare_view.mp4) and the transcript text file (transcript.txt) inside. Also add the branded outro video to append at the end: 9x_outro.mp4.

2. Give instructions to the agent (example prompt). Direct it with the level of detail a director would use to control scene transitions.

Example prompt input: "Build me a video editing pipeline that analyzes the workshop recording and full transcript in the current folder."

"Highlight extraction: Read the entire transcript and find five 1-minute highlight segments that contain core topics or punchlines likely to keep viewers interested."

"Automatic video cuts: Identify start and end timestamps for the five selected segments and trim the source footage accordingly by writing and running an FFmpeg script."

"Dynamic cutting: When the presenter reads from on-screen text, show screenshare_view.mp4, and when they give general explanations, switch to speaker_view.mp4, using cross-cut editing."

"Branding asset blending: After each of the five clips ends, append the 9x_outro.mp4 branded outro from the folder so it flows naturally. Finally, create a Social_Clips folder and save five finished MP4 files."

AI process and results: The AI writes its own video-processing script and completes the job in about five minutes. When you open the folder, you find five perfectly cut MP4 files. When you play them, the impact line appears when the presenter says, "This skill is only basic text!" and the speaker view is

shown at that moment, then the company logo appears cleanly at the end. What used to take 1 to 2 hours of manual clip editing each week now finishes in about 5 minutes.

Previous Chapter

Next Chapter

37 Concrete Codex Use Cases - Chapter 18. Real-time synchronization of Notion documents to Community (Circle) post

Previous Chapter

Next Chapter

Chapter 18. Real-time synchronization of Notion documents to Community (Circle) posts: automating tedious publishing work

Concept primer for beginners: Stop using broken copy-and-paste formatting. Teams usually write tutorials or manuals in

Notion

beautifully. There are bullet points, bold text, and many screenshot images embedded throughout. But when moving this into the company's official community (for example, Circle, WordPress, Naver Cafe) for publication, the nightmare starts. If you copy and paste everything, the text comes through, but the images fail to copy and appear as broken boxes, and YouTube embed links break. In the end you have to open the editor and download dozens of images again, then insert each one manually in the right place. If you use AI, you can migrate the

Notion

document with just one link and do it cleanly.

Practical use case: automatic publishing when a

Notion

status changes to "Ready to publish"

1. Prepare the environment. Use a Notion database and get an API key from a community with API integration (Circle), then provide it to the agent.

2. Instruct the agent (example prompt)

Example prompt input: "Create a /publish_

notion

_to_circle skill. This workflow should perfectly migrate and auto-publish our

Notion

tutorial documents to our company Circle community.

If I give you a link to a specific

Notion

document, download all text data and formatting (bold, headings, and so on) exactly as-is from that page.

Download every screenshot image file inserted throughout the

Notion

document to local storage, then upload them to the community server via the Circle API and insert each image back into the correct position in the body.

If there is a YouTube video link, convert it into a perfect embed format that works in the community.

Once all conversions are complete, publish this document immediately as a new post in Circle's AI Tutorial board, and send me a link confirming completion."

After I send the prompt with the task details and expected output, the agent writes a Python script that connects

Notion

and the community APIs. After opening the community link that confirms the job is done, you can see that all screenshots, YouTube videos, and text formatting from the original

Notion

page have been replicated and posted without a single line out of place. The energy spent on copy-paste labor can now be focused on writing better content.

Previous Chapter

Next Chapter

37 Concrete Codex Use Cases - Chapter 19. Automatically classifying incoming emails (triage) and drafting tailored

Previous Chapter

Next Chapter

Chapter 19. Automatically classifying incoming emails (triage) and drafting tailored replies: the AI assistant steals my email tone

Understanding concepts for beginners: an AI email assistant that knows my tone better than I do. In the morning, when I open my inbox, I usually see junk newsletters, spam, and promotional emails hiding alongside a client’s urgent request or a partner’s meeting proposal. I can spend the start of the day just reading and filtering dozens of messages. By letting AI scan my Gmail or Outlook inbox, I can ignore spam and extract only the emails that require my action. Even more useful, an AI trained on my past emails can draft responses that closely mimic my own signature and writing style.

Practical use case: email sorting and Slack alerts every morning at 6 a.m.

1. Prepare the environment, then grant Gmail permissions in the agent's Connector settings. In the memory.md file, define your email writing principles, for example: write briefly in a direct, top-down style, and make the closing salutation "Thank you, Lab Manager.".

2. Giving instructions to the agent (example prompts)

Example prompt input: "Set up an email triage automation schedule that runs at 6 a.m. every morning."

Use the Gmail connector to scan my inbox for the last 24 hours. Ignore all newsletters, spam, and notification emails.

Create a summary list by filtering only the important emails that I need to handle myself, such as meeting requests, customer support inquiries, payment issues, etc.

For each important email you have selected, write a reply draft by strictly applying the 'Email Writing Style and Tone of Voice' rules saved in memory.md in my project folder.

Send all these results to my Slack, and record the email details in the

Airtable

database.

In AI's workflow and outputs, when you turn on your computer at 7 a.m. and open Slack, you get a message from AI: "I scanned a total of 28 emails and found one new important email." That message is a meeting request from your partner company, and right under it is a polite, short reply draft written

in exactly the same style as your usual emails. All you have to do is read it and click the "Send" button.

Previous Chapter

Next Chapter

37 Concrete Codex Use Cases - Chapter 20. Creating a professional executive report from customer feedback survey d

Previous Chapter

Next Chapter

Chapter 20. Creating a professional executive report from customer feedback survey data: visualizing thousands of responses in one minute

For beginners: the hidden customer voice (VOC) workshop, events, and post-launch Google Forms or Typeform feedback is a valuable asset. But when you open the CSV file full of responses, the 1-to-5 rating numbers are mixed with long open-ended comments, and analysis feels overwhelming. If you hand this Excel file straight to executives or partners, no one reads it. With an AI agent, you can analyze thousands of responses, quantify positive and negative factors, and generate a professional, polished PDF or Word report instantly.

Case study: automatic PDF feedback report generation after legal and finance workshops.

1. Preparation: place the customer response file feedback_survey.csv in the Feedback_Reports folder. The file contains fields such as customer department, satisfaction score, what was best (open-ended), and improvement suggestions (open-ended).

2. Prompt the agent (example prompt).

Example prompt input: "Analyze the feedback_survey.csv data in the current folder and create a professional feedback report that stakeholders and executives can read and understand immediately. Put core metrics (high-level stats) at the top, including overall attendee satisfaction score average and AI adoption intent ratio. Analyze the open-ended comments in 'best points (positive factors)' and 'improvement points (negative factors)', group and summarize the top three most frequently mentioned patterns for each. Based on this data, suggest clear 'key insights' on which workshop topics to add next, using the Use cases demanded lens. Then create a Word (Docx) or PDF document in the folder formatted as a complete 'Executive Summary' report."

Core data crunch: calculate key metrics such as average overall satisfaction and AI adoption intent ratio, and place them at the top.

Open-ended sentiment analysis: analyze customer open-text responses for 'best points' and 'improvement points', then group and summarize the top three most common patterns in each category.

Insights and next steps: based on this data, propose insightful 'key insights' about which topics should be added in the next workshop.

Report generation: generate and save a Word (Docx) or PDF document in the folder, formatted as a complete 'Executive Summary' report for executives."

AI workflow and output: the AI uses Python (Pandas, etc.) to compute values from the spreadsheet and runs NLP-based sentiment analysis. In just a few minutes, a polished file named Feedback_Report_AI_Workshop.pdf appears in the folder. When opened, it is organized cleanly by sections with insights like "Overall satisfaction: 4.8/5.0," "Strengths: prompt examples ready for immediate practical use," and "Improvements: more hands-on time is needed." If you send this document directly via executive email, your results will be recognized immediately.

Previous Chapter

Next Chapter

37 Concrete Codex Use Cases - Chapter 21. Weekly CEO Review and Core Business Metric Tracking Report: Executive Re

Previous Chapter

Next Chapter

Chapter 21. Weekly CEO Review and Core Business Metric Tracking Report: Executive Reporting Automated with AI Memory

Concepts for beginners: Giving AI a memory. Every Monday morning, you usually have to summarize last week's performance for executives or team members. You pull website sessions, new sign-ups, and revenue from Excel, compare those with the prior week to calculate change, and write a report. This same repetitive task had one blocker if you wanted to hand it to AI. A normal AI chatbot cannot remember 'last week's data,' so you must upload historical data again each time.

With Claude Co-work's 'Co-work Projects' feature, that flips completely. This project workspace has memory, so it recalls prior conversations and data each time the task runs. If you only provide this week's data, AI can produce a polished weekly report on its own and say, 'Last week revenue was 1,000,000 KRW, this week is 1,200,000 KRW, so that is a 20% increase, a green signal.'

Practical use case and setup

Project creation: In the Claude Co-work desktop app, create a new 'Co-work Project' and name it Weekly CEO Review.

Context instruction setup: In the project setup file (for example,

agents.md

or project instructions), assign the role: 'You are our company data analyst. Every week, when I provide the latest metrics, compare them with last week's numbers and write the report.'

Example prompt for beginners (command to run every week):

"Run this week's report."

"[Paste this week's raw data or attach the CSV file]"

"Based on this data, generate a 'Secretly CEO Weekly Review' report in the format below."

"Headline: summarize this week's business performance in one line."

"Key Numbers: compare with the 'last week's values' you remember and show the percentage change (Delta)."

"Green Flags: 3 metrics that grew significantly or look positive versus last week."

"Watch List: 2 metrics that declined or need corrective action."

"Focus for next week: based on this data, list the action items we should prioritize next week."

When you run the execution prompt, AI loads last week's context in seconds, cross-checks last week against this week, and displays a fully formatted executive report. You then copy that output and send it to the team via Slack or email. You can compress what used to take two hours every week for data collection and analysis into about 10 seconds.

Previous Chapter

Next Chapter

37 Concrete Codex Use Cases - Chapter 22. Extracting Missing Action Items Through Meeting-Notes Comparison: An AI

Previous Chapter

Next Chapter

Chapter 22. Extracting Missing Action Items Through Meeting-Notes Comparison: An AI Assistant That Does Not Miss Anything

For beginners: Cross-check human notes against AI notes. When a meeting ends, we usually clean up minutes in

Notion

. But among the many points raised during discussion, there will always be parts missed when a person manually writes down who is doing what and by when. In contrast, AI-based voice transcription apps that many teams use now, such as Gemini, Gong, and Granola, keep a full text transcript of the meeting without dropping wording.

The core of this chapter is to instruct the AI: \"Compare the full conversation recorded by AI with the minutes summarized by humans in

Notion

, and find any promises or tasks that should have been written in

Notion

but were forgotten.\"

Practical use case and connector integration

Connector setup: Go into the \"Connectors\" area of Claude Co-work settings, grant the AI access to Google Drive, where voice transcripts are stored, and

Notion

, where meeting notes live.

Working-folder setup: Set a working folder that the AI can access.

Example prompt for beginners:

\"Read the \"Gemini transcript\" document from the marketing meeting this morning stored in Google Drive, and use the

Notion

connector to pull the \"Marketing Meeting Summary Notes\" page I just wrote.\"

\"Cross-reference these two documents carefully and do the following:\"

Missing-task extraction: Find all hidden action items where someone committed in the conversation to do something but the item is missing from my

Notion

minutes, i.e., it fell through the cracks.

Assignee and deadline assignment: For each action item you find, indicate who owns it and by what date the person agreed to complete it.

Notion

update: Automatically update the missing items into the \"Additional Action Items\" section at the bottom of my

Notion

page.\"

AI process and result: The AI quickly reads a multi-page meeting transcript from Google Drive and pulls the

Notion

notes, then cross-checks the two documents. It then reports, \"Analysis result: Sarah promised to send the landing page draft by next Wednesday, but this is missing from the

Notion

meeting notes.\" This creates a strong workflow in which the AI fills in all the gaps a human manager might miss, with strictness and precision.

Previous Chapter

Next Chapter

37 Concrete Codex Use Cases - Chapter 23. Churn risk prediction and upsell (expansion) opportunity monitoring: fin

Previous Chapter

Next Chapter

Chapter 23. Churn risk prediction and upsell (expansion) opportunity monitoring: finding money in call recordings like magic

Concept guide for beginners: In subscription services (

SaaS

) or B2B businesses, a customer canceling a service, known as churn, is a serious loss for the company. Before a customer suddenly cancels, they will usually show some sign of dissatisfaction during calls with sales or the CS team. At the same time, calls often hide upsell opportunities, where a customer asks questions like, "Do you also have this feature?"

Executives or managers cannot listen to hundreds of sales calls every day, even when recordings come from Gong, Fireflies, and similar systems. At this point, if you connect AI with

CRM

tools through Model Context Protocol features, AI can scan hundreds of call transcripts each day, detect customer sentiment and intent, and summarize the findings on a dashboard.

Practical case and setup: in tools such as Cursor or

Claude Code

, connect HubSpot (customer

CRM

) and Gong (call recording tool)

MCP

integrations as plugins. Then AI can access

CRM

data directly using natural language commands.

Example prompt for beginners:

"Using the HubSpot and Gong

connectors, please run a Customer Health Monitoring System that analyzes all sales and CS call scripts with existing customers completed this week."

"Find the following two signals and report them in table format:"

"Churn Risk Detection: identify moments in the call content where the customer expressed 'frustration' or 'disappointment,' or pointed out that we are 'not proactive enough.' For each account, calculate churn probability as a percentage, and propose Immediate actions we should take right away."

"Upsell/Expansion: if the customer sounds satisfied and shows signs of budget expansion by asking about additional marketing channels or solutions, capture and summarize those opportunities."

AI workflow and results: AI reviews the context in call scripts. It surfaces highly specific insights such as, 'The CMO at customer company A said, "Are we doing something wrong? The conversion rate is dropping." Churn probability is 75%. Immediate actions: the account manager should schedule a review meeting tomorrow.' By sharing this warning directly with the team, the company can protect millions of won in revenue that was about to leave and also secure extra sales opportunities.

Previous Chapter

Next Chapter

37 Concrete Codex Use Cases - Chapter 24. CRM Data Integration to Revive Lost Deals: a Sales Representative AI Tha

Previous Chapter

Next Chapter

Chapter 24. CRM Data Integration to Revive Lost Deals: a Sales Representative AI That Resuscitates Dead Contracts

For beginners: the ultimate money-making AI, Deal Revivor. Many contracts end up as Lost Deals when active sales were abandoned with reasons like "we do not have budget right now" or "the timing does not fit," and they never close. Since people have short memories, they easily forget to contact customers who failed a year ago.

We call this workflow "Deal Revivor (contract resurrection)." AI scans the

CRM

(such as HubSpot), and pulls out very large deals that failed 1 to 3 years ago. Then it searches the web for the latest news about those companies and sends you a highly natural, unnervingly personalized follow-up email draft in Slack.

Real-world use case and setup: connect

Claude Code

or a specific AI dashboard to the Slack and HubSpot APIs. When AI suggests in Slack, "Should I send this person an email?" you build a Human-in-the-loop flow where you only press the "Send Email" button.

Example prompt for beginners:

"Build me an automated workflow for 'Deal Revivor (sales opportunity revival).'

CRM

data mining: scan HubSpot

CRM

and find all large deals of 100,000,000 KRW ($100K) or more that were marked as 'Lost' between 10 months and 50 months ago.

Latest information scraping: based on the company name and owner (for example, Jason), run web searches and identify recent news or achievements of that company (for example, funding, new product launches).

Personalized cold email writing: using that, write a follow-up email draft in the following tone. (Sample tone): 'Hi Jason, I saw the latest [recent news) update. Congratulations on that. It has been about [N] months since we last spoke. There seems to have been a lot of change in the company, so I wanted to ask if this might be a good time to discuss our solution again in a quick call?'

Send this draft to Slack with a 'Send Email' button."

AI process and result: AI immediately searches HubSpot and surfaces a 750,000,000 KRW ($750K) deal that was missed 43 months ago. Then it combines the contact's promotion or the company's latest news and posts a near-perfect email draft in Slack. You can be drinking coffee, look at the Slack alert, skim the draft, and just click "Send." Even without any coding knowledge, this is a system that can recover hundreds of millions of won in lost revenue.

Previous Chapter

Next Chapter

37 Concrete Codex Use Cases - Chapter 25. Building a New Product GTM (Go-to-Market) Strategy and Writing an Email

Previous Chapter

Next Chapter

Chapter 25. Building a New Product GTM (Go-to-Market) Strategy and Writing an Email Sequence: A Marketing Launch Plan in 15 Minutes

For beginners, a concept guide: when marketing teams launch a new product or service (such as

SaaS

software, an online course, and so on) to market with AI in one click, they still need a solid strategy. The work of planning and writing an email marketing sequence (consecutive emails) that guides customers from awareness and signup to paid purchase often takes several weeks, and firms may pay agency fees in the tens of millions of won.

But if you use

Claude Code

, you can build a complete GTM strategy, from target customer analysis to tightly tailored email copy by User Journey, with a single prompt and just 15 minutes.

In practical use cases and setup, place documentation about the new product (for example, readme.md, product_info.txt) in the project folder so the AI can learn in advance what value the product delivers.

Example prompt for beginners:

"Build the GTM (Go-to-Market) strategy and the complete onboarding email sequence for our newly launched product (for example, ClickFlow). Read the product documents in the folder and identify the core value proposition of our product.

Then write perfect marketing email copy (subject, body, and call to action) for each user journey stage that will be sent:

Onboarding/Welcome: An email that guides a user who has just started a 14-day free trial after signup on how to get started, such as connecting a website or publishing first content.

Abandoned Signup: An email to bring back users who dropped out in the middle of signing up.

Trial Canceled/Churn: A retention-focused email to users who canceled after their trial ended, expressing regret and asking for feedback.

Cross-sell/Upsell: An email to loyal customers who are using the product well, recommending an upper-tier plan or additional managed services (Managed Services)."

When you send the AI workflow request and expected output prompt, the AI designs your product Funnel Architecture on its own and produces dozens of emails with psychologically tuned copy for each stage. It includes practical guidance like, "Welcome! Please check whether our product is right for you over the next 14 days. First, connect Google Search Console and publish your first content." You only need to review the email drafts for about 10 to 15 minutes, revise roughly 5%, and hand them to your marketing team; launch preparation is done. It is the best way to massively reduce the cost and time of hiring experts.

Previous Chapter

Next Chapter

37 Concrete Codex Use Cases - Chapter 26. Interactive slideshow and presentation auto-design: producing a brand-ta

Previous Chapter

Next Chapter

Chapter 26. Interactive slideshow and presentation auto-design: producing a brand-tailored PPT in 10 minutes

[Concept: background for beginners] People who have tried making presentations with AI in the past usually end up disappointed. When they give AI a bare request like "make some slides" and use the so-called "Vibe Designing" style, they usually receive results with mismatched fonts, awkward colors, and inconsistent layouts. They then keep repeating endless adjustment prompts until they eventually give up and build the PPT themselves.

But when you use the strong Template and Design System features in Claude Design or Claude Co-work, the story changes. If you first teach AI your company font, logo, brand colors, and the frame of a 20-slide layout, the AI can generate a flawless slideshow in a flash as soon as it reads your provided summary text or script, fitting everything into the pre-approved layout like copy and paste.

[Preparation and setup]

Build the design system: In the Claude Design system tab, set your company name, brand colors (hex codes), fonts, and so on. If you are new to this, you can use the Design System Creator Skill; by entering just the website URL, you can let AI pull the logo and brand assets from the site and set up the system on its own.

Template registration: Upload 20-plus PPT slides you built well before, or screenshots of beautiful layouts found online (for example, Dribbble or Canva) to Claude and instruct it to "save these formats as slide templates" to turn them into reusable templates.

[Specific example prompt]

"I am preparing a visual slideshow for a YouTube video and seminar on 'AI Automation Trends' next week. Please apply only the 'pre-approved 28-page slide templates' stored in my project folder and our company brand design system."

"Read the attached Word document, which is the seminar script summary, and do the following: 1. Slide mapping: Select and assign the most fitting template layouts for each topic in the script (for example, a three-column split layout, a full-screen image layout). 2. Copywriting: Write concise and impactful headlines and body copy for each slide. 3. Rendering: Combine all of this and immediately generate a perfectly formatted interactive slideshow."

[AI workflow and near-magical result] When you send this prompt, the AI (using the Visual Layout Skill and similar tools) analyzes the script and plans how to proceed, saying, "I will apply template 14 to slide 1 and write the headline this way." After your approval, Claude immediately renders the slides using HTML/CSS form or internal presentation skills. The result shows 20 completed slides with your brand colors applied. If you do not like a text size or line of copy, you can click directly on slide elements to edit text and adjust sizes by pixels. It is a workflow that turns what used to take 5 hours into expert-level slides in just 10 minutes.

Previous Chapter

Next Chapter

37 Concrete Codex Use Cases - Chapter 27. Conducting a professional UX audit of desktop apps and websites: hiring

Previous Chapter

Next Chapter

Chapter 27. Conducting a professional UX audit of desktop apps and websites: hiring AI as our company QA tester

[Concept primer for beginners] When launching a new website or app, testing whether users can move smoothly from sign-up to payment is essential. That process, known as UX audit and QA, is the core of release quality. But pressing every button to find bugs needs huge patience and time. Recently, Claude and

Codex

include the innovative features called

Computer Use

and

Browser Use

. They let AI see your monitor screen directly and move the mouse and type on the keyboard like a person, so it can navigate the browser. You can hire AI as a QA quality assurance tester, let it sweep through your app, identify friction points, and turn them into a professional report.

[Preparation and setup]

In the Claude desktop app, go to Settings and turn on

Computer Use

permission. On a Mac, you must allow Screen Recording and Accessibility permissions in System Settings.

For security, it is recommended to set Ask before acting mode first so AI cannot click around randomly.

[Specific example prompt]

"Run a professional UX Audit for my new desktop web app (for example, your zone of genius app). Open a browser, enter the app URL, and access it."

Then run a full user flow test from the perspective of a real first-time user: 1. Login flow: enter my email on the login page, go to my inbox and copy the one-time password (OTP), then return to the app and complete login successfully. 2. Interface testing: click through various tabs and buttons on the dashboard at random, and run stress tests by typing into the search bar or changing filters to check for friction points and broken links. 3. Report writing: find UI defects or awkward elements that might confuse users, and generate a document called the "UX Audit Summary Report" with recommendations on how to fix them."

[AI workflow and almost magical results] Once given the instruction, the AI opens the browser and actually moves the mouse cursor. It types the email into the login field, switches tabs to copy the code from email, and completes sign-in. Then it moves around the app and pinpoints bugs and UI improvements such as "the search function is too rigid," "clicking an external YouTube link does not open in a new tab," and "the empty state screen is too bare." About 15 minutes later, AI compiles every test step into a complete executive audit report with "four UX defects that should be fixed first." Even without hiring an expensive QA team or external consultants, you can get top-level feedback.

Previous Chapter

Next Chapter

37 Concrete Codex Use Cases - Chapter 28. n8n, Make.com, and other automation workflow (JSON) conversion and build

Previous Chapter

Next Chapter

Chapter 28. n8n, Make.com, and other automation workflow (JSON) conversion and building: Perfect fusion of visual tools and coding

[Concept overview: background for beginners]

n8n

,

Make.com

, Zapier, and other visual automation tools are popular with beginners because you can connect apps by dragging and dropping blocks. But when the automation grows complex, the blocks get tangled like spaghetti and maintenance becomes difficult. The platform's paid execution quota (Task quota) can also run out, creating extra cost. In this case, you can use AI to convert a visual

n8n

workflow into a fully independent Python script that can run permanently on your local PC or a free server. Conversely, you can ask AI to analyze a Python codebase and generate a JSON file that you can paste directly into

n8n

.

[Preparation and environment setup]

From an existing

n8n

workflow you already built (for example, a flow that automates YouTube live stream creation, Luma event registration, and Zoom meeting setup), click 'Download' in the upper right corner of the screen and save it as a JSON file in your desktop folder.

Run

Claude Code

or

Codex

in that folder.

[Specific example prompt]

"I have an

n8n

workflow file (JSON) downloaded in this folder. This file contains logic that automatically links Luma, Zoom, and

Notion

when I prepare a weekly live workshop.

1. Document the logic: Read this JSON file carefully and write a documentation in Markdown so that even beginners can understand which trigger starts this automation and what happens at each step. 2. Convert to Python code: Now convert this visual workflow into a complete Python script so I can run it permanently on my own computer. Split the API logic connecting Zoom, Luma, Notion, and YouTube into separate Python files, and write a main execution script that controls all of them. 3. Generate command: Finally, create a custom slash command so this code can be run easily from this chat as a shortcut command."

[AI workflow and magical outcome] AI analyzes the complex machine-readable JSON file and in just one minute produces a clean document: "This workflow is an automation that creates Zoom meetings and records data in

Notion

." Then AI splits and writes Python code that connects each app's API, packaging it into a reusable module. At this point you do not need to use an expensive subscription-based automation platform. By entering one command in the terminal, you can make AI run the Python code and process the same automation in the background for free.

Previous Chapter

Next Chapter

37 Concrete Codex Use Cases - Chapter 29. Online Shopping and Web Automation via Browser Control (Computer Use): S

Previous Chapter

Next Chapter

Chapter 29. Online Shopping and Web Automation via Browser Control (Computer Use): Setting Up an AI Personal Shopping Assistant

[Conceptual overview for beginners] "If you send a text on your phone after getting home, 'The coffee beans are out, please order the same one I usually buy on Amazon,' and the computer at home wakes up and puts it in the cart by itself." It sounds like Iron Man's Jarvis from the movies, but this is a workflow you can set up right now with Claude Co-work's Dispatch feature and

Computer Use

. Give the AI eyes to see the monitor and hands to move the mouse, then give it instructions remotely from your smartphone. Even everyday repeated web surfing, like shopping, can be fully delegated to AI.

[Preparation and environment setup]

On your desktop, open Claude Co-work, then enable

Computer Use

in settings. Allow browser access permissions.

Security tip: When you give shopping permission to AI, it is strongly recommended to issue virtual cards with a monthly limit (for example, $50) from a service like Privacy.com and register them in your Amazon account, to prevent financial incidents from hacking or prompt injection (misleading instructions).

[Concrete example prompt (send from smartphone iMessage or the Dispatch app)]

"My spring water at home is gone. Open Amazon.com in the browser on my desktop.

Follow the next instructions and handle the shopping for me:

Search for 'unlabeled 2L spring water with high ratings.'

From the search results, find one best-value item with at least a 4.5-star rating and a price at or below $15.

Open that item's detail page and click the 'Add to Cart' button with the mouse.

For safety, do not press the final checkout (Buy now) button. Take a screenshot showing the item is in the cart and send it to my smartphone (iMessage or Dispatch). After I confirm, I will complete the payment myself."

[AI workflow and near-magical outcome] While you are on the move, the AI on your home desktop wakes up immediately. The AI cursor moves on its own, opens Chrome, and types the search terms on Amazon. It checks review ratings, picks a suitable water, and clicks the 'Add to Cart' button precisely. Within a few minutes, a message arrives on your smartphone saying, 'I added the item you requested to your cart,' along with a confirmation screenshot. You do not need to go home; tapping approval to pay from your phone completes the routine shopping task.

Previous Chapter

Next Chapter

37 Concrete Codex Use Cases - Chapter 30. Remote Desktop Work via Smartphone/Smartwatch Dispatch: Breaking Time an

Previous Chapter

Next Chapter

Chapter 30. Remote Desktop Work via Smartphone/Smartwatch Dispatch: Breaking Time and Distance Constraints

[Concept Overview: Background for Beginners] You do not have to sit in front of a computer for work to move forward. You may be eating, working out at the gym, or lying down before sleep when you suddenly think, "Tomorrow I need to collect competitor data for a meeting." In this case, with Claude's Dispatch feature, you can send instructions to an agent from your phone or Apple Watch in your pocket and keep doing your own tasks. Dispatch turns a smartphone into a remote control, letting the AI agent on the desktop at home run heavy data analysis or web scraping jobs in the background for hours.

[Preparation and Environment Setup]

In the Claude app settings on your desktop (such as a Mac Mini), enable Dispatch and link it to the Claude app on your phone.

If the computer enters sleep mode while the agent is working, the job is interrupted, so make sure to enable the Keep Awake option in settings.

[Concrete Example Prompt (Send from Phone While Waiting in a Coffee Shop)]

"I need a performance analysis of the YouTube channel for tomorrow morning's marketing meeting. Using the desktop

Computer Use

permission, run the following background tasks.

Open my browser and access the YouTube Analytics page.

Extract all data for my 10 most recently uploaded videos, including Views, CTR, and Average View Duration.

Based on this data, generate a cleanly formatted Excel spreadsheet (.xlsx) file, and add a summary of the core insights at the end of the document explaining why this week's performance was strong.

When file creation and analysis are complete, send the summary to my smartphone Dispatch thread. I'll check it on my desktop when I return tomorrow morning."

[AI Workflow and the Result] In the moment you order coffee and put the phone in your pocket, the remote desktop AI opens a browser, connects to YouTube Studio, and begins collecting complex data. It gathers thousands of numbers and arranges them into clean charts and tables in Excel. Once you finish your coffee and get home, or when you power on the desktop the next morning, you will find a neatly organized Excel file and analysis summary waiting calmly at the center of the screen. Combining the portability of mobile devices with the desktop AI's strong computing power, this is a final workflow that builds a true AI assistant working for me around the clock, even while I sleep.

Previous Chapter

Next Chapter

37 Concrete Codex Use Cases - Chapter 31. Custom batch creation of invoices and documents from Airtable data: comp

Previous Chapter

Next Chapter

Chapter 31. Custom batch creation of invoices and documents from Airtable data: complete automation of manual paperwork tasks

[Beginner concept primer: Why combine

Airtable

and AI?] If you are a freelancer, an agency, or a finance manager in a small company, there is a painful monthly task that comes back at the end of every month. It is creating invoices, receipts, and contracts to send to tens or hundreds of customers. Checking each customer's name, billed amount, date, and details in Excel, then copying and pasting them one by one into Word or a design tool template such as Canva, converting to PDF, and saving again is pure repetitive labor and a high risk of typos.

Until now, people tried to solve this with automation tools like Zapier or Make, but they hit limits when fine-tuning template design or rendering complex tables. But when you connect AI agents like

Claude Code

or

Codex

to

Airtable

, the situation changes. The AI can read the

Airtable

database in real time, generate branded HTML/CSS invoices, and then batch-convert them into high-quality PDFs and upload them automatically into

Airtable

's Attachment field.

[Preparation and environment setup]

Airtable

base setup: prepare an

Airtable

base containing customer information and billing records. The core setup uses two tables.

Invoices table: create fields for customer name, email, total amount, status (such as To Create, Ready to Send), and an empty Attachment column.

Line Items table: record the line-level details for each invoice, for example website development fee, maintenance fee.

Teach the agent with templates: place sample_invoice.pdf or a design guideline document in the project folder so the AI prelearns our brand colors, logo placement, and font style.

API permission: set a Personal Access Token in environment variables (.env) so the agent (such as

Claude Code

) can access

Airtable

.

[Specific agent instruction prompt for beginners] Once all setup is done, switch the

Airtable

status to To Create, open the agent terminal, and run the following prompt.

"Run the /create_invoices automation skill. This task is to generate bulk invoice PDFs through

Airtable

integration.

Automate this fully using the following steps:

Data fetch: call the

Airtable

API and read all records from the Invoices table with Status set to To create. At the same time, load all related Line Items data.

HTML invoice rendering: using the sample_invoice style guide in the folder, dynamically map each customer's data and render it as a beautiful HTML/CSS invoice.

PDF conversion: convert the rendered HTML into high-resolution PDF files using Puppeteer or the Playwright library.

Temporary hosting and upload:

Airtable

APIs accept attachment uploads only through URLs. Upload the generated PDFs briefly to temporary hosting (for example AWS S3 or temporary storage like imgbb), then use that URL to upload the PDF to

Airtable

's Attachment column.

Status update: once the upload succeeds, set the record status to Ready to Send and end the workflow."

[AI workflow and near-magic results] After you send the prompt, the agent immediately writes and runs Python or Node.js scripts on its own.

After 1 minute: the agent successfully reads the list of 50 customers in To Create status from

Airtable

.

After 3 minutes: the agent opens a browser (Playwright) in the background and parallelizes HTML-to-PDF conversion.

After 5 minutes: if you are watching

Airtable

on a dual monitor, a scene like this unfolds. In the Attachment column for 50 customers, which had been empty, PDF icons appear one per second as if dropped in rhythm, and they are uploaded automatically. At the same time, the status column turns green as it changes to Ready to Send.

If you click one of the generated PDFs, you see the company logo perfectly embedded and a neatly calculated table of each customer's line items. This is a workflow that produces hundreds of proposals, invoices, and contracts in 5 minutes—work that once took the finance team all week—and does so with zero errors.

Previous Chapter

Next Chapter

37 Concrete Codex Use Cases - Chapter 32. Building a stock/crypto portfolio analysis and trading-automation dashbo

Previous Chapter

Next Chapter

Chapter 32. Building a stock/crypto portfolio analysis and trading-automation dashboard: Hiring your own AI private banker

[Beginner guide: understand the concept by using

MCP

(Model Context Protocol) to master brokerage APIs] Investors usually manage their stocks or cryptocurrencies by looking at MTS (Mobile Trading System) or HTS (Home Trading System). "How much is the Samsung Electronics I bought worth right now?" "Is my portfolio too concentrated in technology stocks?" To answer these, you often open Excel, copy numbers over, and build complicated charts.

But with the

MCP

(Model Context Protocol) approach, you can give an AI agent an interpreter that lets it talk directly to brokerage systems. If you connect a major brokerage API, such as India-based Zerodha or a Korean brokerage API, or a cryptocurrency exchange API (such as Binance or Upbit), to the AI, the AI can check your account balance, returns, and holdings in real time and generate a polished portfolio analysis dashboard at the level of a management report. If the logic is right, you can also build your own "AI private banker" that can execute buy or sell orders with one click.

[Preparation and environment setup]

API key issuance: Obtain the API Key and Secret Key from the developer portal of the brokerage or exchange you use (such as Zerodha), and store them in environment variables (.env).

MCP

server connection: Add an

MCP

server in the settings of Claude desktop app or Cursor IDE. (For example, register a simple Python script that communicates with the brokerage API as an

MCP

server.) Through this, the AI gets safe permission to read your account.

[Beginner-friendly agent instruction prompt]

"Build a 'portfolio analysis dashboard' that syncs perfectly with my stock account using the Zerodha

MCP

server connector.

Follow these instructions:

Account scan and return analysis: Read all current holdings and positions in my account. Calculate total invested principal, current value, and unrealized P&L, then show these as large figures at the top.

Sector and exposure visualizations: Research and classify each holding into its industry group (for example, technology, infrastructure, finance, international ETF). Then visualize a pie chart at the center of the dashboard so I can check whether my assets are too concentrated in any single sector.

Benchmark and individual stock review: Highlight the top winner and top loser in my portfolio, the positions with the highest gains and the highest losses.

AI trading automation button (order execution): If I click a specific stock in the dashboard and press the AI Sell button, write a script that directly calls the

MCP

order API to place a market order sell for that stock. (Insert a safety guard that always asks for my final Yes/No approval via Slack or terminal before any actual fill.)"

[AI workflow and the almost magical result]

Analysis and visualization: The agent immediately calls the API and pulls 14 holdings from my account. The user had forgotten what the stock 'MAFANG' refers to, but the AI classifies it politely as "MAFANG is an overseas ETF investing in large-cap US technology stocks."

Dashboard creation: A polished dashboard opens in the local browser after a few minutes. It shows a briefing such as "Total portfolio assets: KRW 20,000,000, unrealized gain: +KRW 4,200,000. Biggest loss position: PTM (-38% drop). 18% of assets are exposed to infrastructure, 12% to international technology." with a pie chart drawn beside it.

Order execution test: Click the position with the sharpest decline and run the 'sell script'. The AI asks in the terminal, "Current balance checked. Should I send a market sell order?" When 'Yes' is pressed, the order is sent to the broker server through the API and a fill notification appears. (Caution: for beginners, set up trading so every trade goes through a small-size test and a Human-in-the-loop manual approval to prevent wrong orders from AI Hallucination.)

Previous Chapter

Next Chapter

37 Concrete Codex Use Cases - Chapter 33. Multilingual (translation) and parallel multi-design development with Gi

Previous Chapter

Next Chapter

Chapter 33. Multilingual (translation) and parallel multi-design development with Git Worktree: Building an AI dev team that works like it has many hands

[For beginners: understanding the difference between Branch and

Worktree

] When building a website or app, there are times you want to try several versions at once. For example, "Why not make this landing page in Korean, French, and German?" or "What if we code three layouts at the same time: one modern Swiss design, one brutalist look, and one cute style."

In the traditional Git Branch model, you can check out only one branch in a single folder. So while Agent 1 spends 30 minutes coding the Swiss design, I cannot try another design at the same time and must wait, creating a bottleneck. But when you use Git

Worktree

, a very different workflow opens up. You can create multiple folders, physically separate clones, that share the original repository history instantly. That lets us place three AI agents in different

Worktree

folders at the same time for parallel processing, so we can build three fully independent apps without any code conflicts.

[Preparation and setup]

Project directory setup: create a main project folder on your desktop (for example, App_Main). It must be initialized as a Git repository (git init).

Worktree

management folder creation: beside the main folder, create one empty folder named

Worktree

s to hold the

Worktree

copies.

[For beginners: concrete agent instruction prompt] Run

Codex

or

Claude Code

in the main folder from the terminal and issue a multi-agent instruction.

"Based on this main project, I want to test three different frontend designs in parallel. Set up the work using Git

Worktree

."

1. Create three Worktree folders: use the git worktree add command to create three separate Worktree folders named design-swiss, design-brutalism, and design-japanese inside the neighboring Worktrees folder. Each must be linked to an independent branch.

2. Deploy three parallel agents:

In the design-swiss folder, place an agent that rewrites the landing page in a Swiss International Typographics style.

In the design-brutalism folder, place an agent that codes in a raw and bold brutalism style.

In the design-japanese folder, place an agent that codes in a Japanese minimalism style where ma, or negative space, stands out.

3. Run parallel local servers: to let me compare the three designs visually in real time, start local servers with npm run dev at the same time on ports 5171 for Swiss, 5172 for brutalism, and 5173 for Japanese style.

[AI workflow and the near-miraculous result]

Clone-like execution: the main agent generates three physical folders in an instant and places three separate Sub-agent sessions into each folder. On screen, three agents write different CSS and UI code at once, and progress rises like three bars moving together.

Independent conflict-free development: each agent does not overwrite or interfere with other agents' files. Each one codes and tests inside its own

Worktree

world.

Real-time review and merge: after 15 minutes, all coding is done. You open your browser and load tabs for localhost:5171, localhost:5172, and localhost:5173. All three apps with completely different moods run perfectly. If you decide, "The Swiss design at 5171 is the best," you then tell the agent, "Merge the Swiss design branch into the main branch, and remove the other

Worktree

s."

A multilingual translation job that normally takes days, or large UI/UX design work for A/B testing, can be completed in just 30 minutes through AI

worktree

cloning, which feels like the ultimate one-person development move.

Previous Chapter

Next Chapter

37 Concrete Codex Use Cases - Chapter 34. Code Review, Test-Driven Development (TDD), and Automatic Architecture R

Previous Chapter

Next Chapter

Chapter 34. Code Review, Test-Driven Development (TDD), and Automatic Architecture Refactoring: Hiring a Strict Senior Engineer to Redo My Own Code

[For beginners: Understanding Vibe Coding pitfalls and the magic of TDD] When you do "Vibe Coding" by giving vague instructions such as "make a board feature" to AI, the result is often code that works on the screen for now but becomes spaghetti code internally. Business logic and database wiring are packed into one file, as a monolithic bundle, there is no test code, and the code can be full of security flaws like

SQL

injection. This kind of code causes a full app failure when you try to change even one feature later.

What you need at that point is to transform AI into a strict senior engineer. Give AI Audit and TDD skills. TDD means you write a failing test code (Red) before writing production code, then write only the minimum code needed to pass that test (Green), and repeatedly do refactoring to tidy it up. If AI runs this loop autonomously, you can get enterprise-grade software with no errors and strong maintainability.

[Preparation and environment setup]

Custom skill (Skill.md) definition: In the project root folder, create TDD_Skill.md and Security_Audit.md as the instruction guides the agent must follow. For example, "Always follow the Red-Green-Refactor loop" and "Follow OAWSP standards for security vulnerability scanning."

Grill Me skill setup: Configure a Grill_Me.md skill so AI cannot start coding blindly. The skill forces AI to ask sharp questions before development, helping you build a solid architecture first.

[Concrete agent instruction prompts for beginners] Assume you already have a badly written auth.js (user login) code.

"Run a senior engineer-level architecture Audit and a security vulnerability scan across the entire codebase of my project."

Step 1. Code and security audit (Audit):

Scan all files, including auth.js, for violations of the Single Responsibility Principle (SOLID), including God Object patterns that do too much and spaghetti code.

If there are hardcoded passwords, leaked API keys, or

SQL

injection vulnerabilities, summarize them in a perfect markdown audit report with severity scored from 1 to 10.

Step 2. Automatic refactoring based on TDD (Red-Green-Refactor):

For the critical defects you found, such as database connections being bundled inside the router, refactor the code following the TDD_Skill.md instructions.

Before changing code, always write and run failing unit tests first to confirm errors (Red).

Then write the code by clearly separating business logic and service layers (Green) so the tests pass.

After refactoring is complete, complete your own verification loop by running E2E tests with Playwright in a real browser environment to ensure behavior works end to end."

[AI workflow and outcomes]

Step 1: Ruthless fact checking (Audit output): In just two minutes the agent scans the code deeply and prints a report. "Severity 10/10: auth.js is a giant spaghetti module that combines database queries, JWT token generation, and routing in a single file. The email field is vulnerable to

SQL

injection." It perfectly diagnoses the flaws in a beginner-built implementation.

Step 2: Running the TDD loop: To fix the defects, the agent acts on its own. First it creates test_auth.js and runs attack-simulation tests to confirm a failing state (Red). Then it neatly splits the code into three files—route.js, auth_service.js, and db_repository.js—for clean layering (Abstraction).

Step 3: Test pass and final approval: Running tests on the split code passes perfectly (Green). Finally, it opens Playwright E2E browser tooling in the background and validates, without errors, a virtual user entering ID and password and logging in.

Through this process, junk code built by a beginner is transformed like magic in 20 to 30 minutes into a robust and secure enterprise-grade architecture that looks like it was built by a Silicon Valley senior engineering team over several weeks.

Previous Chapter

Next Chapter

37 Concrete Codex Use Cases - Chapter 35. Automatically split and summarize huge PDF and book data by chapter: an

Previous Chapter

Next Chapter

Chapter 35. Automatically split and summarize huge PDF and book data by chapter: an AI workflow that breaks the 400MB document barrier

[Concept for beginners: Why it does not work on the ChatGPT website?] If you are a graduate student, a researcher, a legal professional, or a workplace person handling a massive manual, you may face PDF files that run into hundreds of pages and hundreds of MB in size. If you try to upload such a file to the ChatGPT or Claude website, upload is rejected with a warning message that says "The file is too large (limit 30MB)". Even if you try to paste the text manually, it stops in the middle because it exceeds the AI's context window limit.

But if you use a desktop agent like Claude Co-work or

Codex

installed on your computer with direct access to local files, this limit disappears completely. The AI agent reads a 400MB PDF on its own, logically identifies the table of contents (contents pages or heading text), and splits it into chapters (Chunking), saving each one as a separate PDF or text file. It then creates a clean folder-level workflow that also generates a core summary for each split chapter so you can read the output right away.

[Preparation and environment setup]

Create a workspace folder: create an empty folder named PDF_Master on the desktop. (In Claude Co-work, it is safe to set this as the 'Playground' folder.)

Prepare the large file: put the file you want to split inside this folder, for example 2026_Global_Economic_Report_400MB.pdf.

Agent execution permissions: allow the agent to access local files and run Python scripts (such as the PyPDF2 library) with read and write permissions.

[Concrete agent instruction prompt for beginners]

"Process the massive PDF file called 2026_Global_Economic_Report_400MB.pdf in the current folder.

Follow these steps in order: 1. Identify the logical structure: scan the document to find the table of contents and natural chapter boundaries (natural section breaks). 2. Chapter-level file splitting (Chunking): write and run a Python script yourself to split this large PDF into chapters. Then create a

subfolder called Chapters in the folder and save each split piece with intuitive, descriptive file names such as '01_Introduction.pdf', '02_Market_Trends.pdf'. 3. Generate individual summaries: once file splitting is complete, read each split chapter file again and write a one-page executive summary as a Markdown (.md) file. When everything is finished, report back a summary list so I can find everything at a glance."

[AI workflow and magical outcomes] Once the prompt is sent, the agent calmly writes and runs a Python script and starts analyzing the document without being overwhelmed by its size.

About 1 minute later: "I found 12 core chapters in the document. Starting file splitting."

About 3 minutes later: 12 lightweight PDF files are perfectly split and saved in the Chapters folder on the desktop.

About 5 minutes later: A summary file named 01_Introduction_Summary.md is automatically generated next to each chapter.

Now you do not need to open a heavy PDF viewer and scroll endlessly. You can quickly read the chapter-level summaries the agent has already sorted, then pull out and read only the original sections you need. This is the moment document work becomes dozens of times more efficient.

Previous Chapter

Next Chapter

37 Concrete Codex Use Cases - Chapter 36. Building YouTube Comment Sentiment Analysis and a Custom Dashboard: Turn

Previous Chapter

Next Chapter

Chapter 36. Building YouTube Comment Sentiment Analysis and a Custom Dashboard: Turning the Customer Voice Into a Data App

[Understanding the concept for beginners: a Streamlit data dashboard beyond Excel] YouTube creators and marketers read hundreds of comments on a video to gauge audience reaction. But once comments pass 200 or 500, it is impossible to read all of them and say, in a quantitative way, that the response is 70% positive and complaints are mainly about audio quality.

Usually, you can imagine exporting this to Excel, but if you take one more step and use Streamlit, the Python library, you can build your own interactive web dashboard app that runs immediately on your computer in just a few minutes. Streamlit is a powerful tool that lets you create a web app with complex pie charts, filter functions, and data tables using only a few lines of Python code. Instruct your AI once to handle comment collection, sentiment analysis, and dashboard coding.

[Preparation and environment setup]

Create the working folder: make a Youtube_Dashboard folder.

API and environment setup: provide the agent with a YouTube API key, or web scraping tool permissions, so it can collect comments, and set it in the environment variable file (.env). Python must be installed on the computer.

[Specific agent prompt for beginners]

"Collect the latest 200 comments from my most recent YouTube upload and build a Streamlit dashboard data app after running sentiment analysis on them.

Do the following steps flawlessly and run them: 1. Comment collection and sentiment analysis: write a Python script to fetch 200 YouTube comments. Then classify each comment into three categories: Feedback, Question, Complaint/Bug, and assign sentiment scores as positive, negative, and neutral, then save the results to comments_data.csv locally. 2. Build a Streamlit dashboard: write a Streamlit web app called app.py that visually displays this CSV data. The dashboard should include:

Top: a summary scorecard showing the overall positive percentage and total comment count.

Middle: a pie chart showing distribution by category (Feedback/Question/Complaint).

Bottom: a data table with search and filter controls to select by specific keywords or only negative comments. 3. Execution: once the code is complete, automatically run streamlit run app.py in the

terminal so the dashboard opens in my browser."

[AI workflow and almost magical outcome]

Data analysis: the agent immediately calls the YouTube API, pulls 200 comments, and classifies "The content was very useful" as positive feedback and "The audio is too low, I cannot hear it" as negative complaint through natural language processing (NLP), then saves the data to a CSV file.

Dashboard execution: after two to three minutes, the agent finishes app.py and enters the streamlit run app.py command in the terminal itself.

Result check: your Chrome browser opens automatically and a powerful data dashboard appears on local web hosting at localhost:8501. The screen shows a clean summary card with "Positive response 65%, complaints 12" and a vivid pie chart. If you check "Complaint" in the search box, only 12 comments pointing out audio issues are neatly filtered into the table. What used to require hiring both a developer and a data analyst, a result worth millions of won, appears in one prompt and about five minutes of waiting.

Previous Chapter

Next Chapter

37 Concrete Codex Use Cases - Chapter 37. Using an Agent Swarm to automate resume writing and job hunting: changin

Previous Chapter

Next Chapter

Chapter 37. Using an Agent Swarm to automate resume writing and job hunting: changing the game of job preparation

[Concept for beginners: a 300-AI swarm, not a single AI] So far, we learned how to get one smart AI assistant (Agent) to do work. But the latest AI models (for example, Kimi 2.6) and agent frameworks provide a new layer:

Agent Swarm

technology.

An

Agent Swarm

means that when you give one instruction, an AI main manager analyzes it and instantly creates and deploys dozens up to 300 Sub-agents in parallel. Let us use job preparation as an example. Updating your resume, searching job sites, finding 100 companies that match your profile, and tailoring your resume and cover letter to each company profile would take a human many weeks or months. With an

Agent Swarm

, you can finish all that in about 20 minutes.

[Preparation and environment setup]

Master file setup: create a Job_Search folder and place your base resume file (base_resume.pdf or .md) and a preferences.md file listing your career history, desired salary, and preferred work mode (remote, hybrid, commute, etc.).

Swarm-support platform setup: prepare an environment that can run multiple agents in parallel, such as Kimi K2.6,

Codex

multi-

worktree

, or Claude with Sub-agents creation.

[Concrete agent prompt for beginners]

"Read base_resume.pdf and preferences.md in my folder (remote work, front-end developer, desired salary 80 million won, etc.), then activate a 'large-scale job support

Agent Swarm

' for me.",

Process the following steps with Parallel agents: 1. Large-scale research: send 10 research sub-agents first to scrape and find 100 open job postings on platforms such as LinkedIn or Indeed that perfectly match my conditions. 2. Create 100 dedicated agents: assign one dedicated sub-agent each (total 100) to each of the 100 companies you found, all at once. 3. Personalized document drafting: each agent should deep-research the JD and recent company news for its assigned company, then automatically produce a hyper-personalized resume and Cover Letter by reshaping my base experience to fit that company’s talent profile. 4. Organize outputs: after completion, create a subfolder for each of the 100 companies inside an Applications folder, and save the finished custom resumes and application links neatly.

[AI workflow and magical output] The moment you issue the instruction, a remarkable scene unfolds on screen.

The master agent launches dozens of terminal threads at once with a message saying, 'Deploying agents,' and they start pinging in parallel.

Each agent emits logs like 'Researching Company A,' 'Analyzing job opening for Company B,' 'Generating customized resume for Company C,' and performs all 100 tasks through perfect Parallel Processing.

About 20 to 25 minutes later, the entire swarm returns and the work ends.

Opening the folders, you see folders named for 100 companies. When you open the cover letter in the Company A folder, it includes a highly specific hyper-personalized introduction letter saying, 'My front-end experience aligns exactly with the AI project recently launched by Company A.' All you do is click the application links in the organized Excel dashboard and submit with the custom resume attachment the agents prepared. This is the ultimate way to scale job searching, not by finding a job alone, but by hiring hundreds of HR experts and assistants at once to blitz the market.

Kim Kyung-jin, Attorney at Law

Attorney at Law · Former National Assembly Member · AI Policy Researcher

kimkj.com

#KimKyungjin #AttorneyKimKyungjin #Codex #OpenAI #AIAgent #AIWorkflowAutomation #AIBoard

Previous Chapter

Next Chapter

Support This AI Library Book

If this book was useful, you can support future translations, public editions, and open AI knowledge projects through PayPal.

PayPal

https://paypal.me/kimkjkj

Scan the QR code or open the link above.