ChatGPT Cheat Sheet

For Data Science

Learn Data Science Online at
www.DataCamp.com

How to use this cheatsheet?

1

What is ChatGPT?

ChatGPT is a language model developed by OpenAI that has gained significant attention due to its remarkable ability to generate human-like responses to prompts given to it. ChatGPT is useful for a variety of language-based tasks, including language translation, question answering, text completion, and a lot more. It's also very useful for **data science workflows**.

2

How to get started?

To get started, simply head to OpenAI's site and start inputting prompts. Register for free or upgrade to a paid version for priority access.

https://chat.openai.com/chat

3

What to expect in this cheat sheet?

In this cheat sheet, you will find specific examples of using ChatGPT for common data science tasks, including data visualization, data analysis, machine learning, and more.

Our guide for great prompts

1 **Start prompts with a description of the AI's role.** For example, "You are a {x}" or "I want you to act as {x}". If you aren't sure, try "You are a helpful assistant". *e.g., You are a data scientist at OpenAI, you are working on large language models. Explain to me how GPT-3 works.*

2 **Use it for common tasks.** The more common the task, the better the chances of success with ChatGPT.

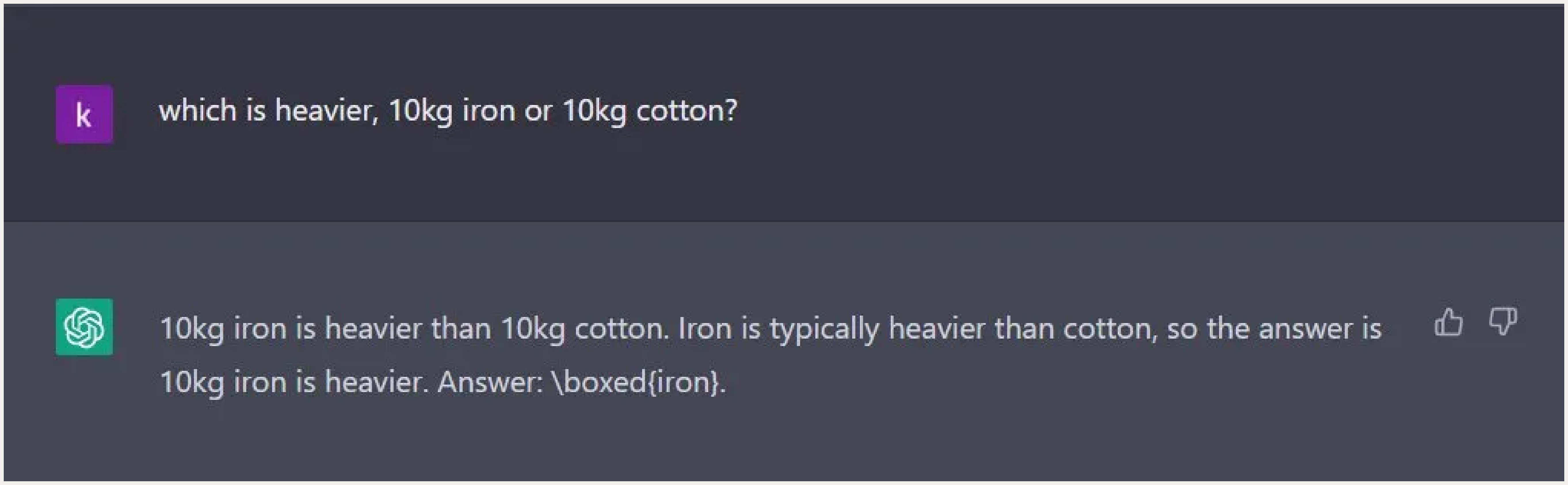
3 **Offer context.** Offering context when posing a question to ChatGPT allows it to provide specific answers. Providing context allows the model to understand the nuances of the problem and generate more informed responses.
e.g., You are a data scientist at OpenAI, you are working on large language models. Explain how GPT-3 works to a business executive with no technical background.

4 **Be precise.** Being precise is another best practice that can minimize the need for tweaking the output. When posing a question, be clear about the input (if any) and the desired outcome.

5 **Keep trying.** Do not be afraid to try multiple prompts. Using different prompts can provide different perspectives on a problem and enable the model to generate a variety of responses.

Limitations of ChatGPT

However, ChatGPT is not perfect – it can generate wrong information, or “hallucinate”. ChatGPT can provide incorrect answers in an authoritative tone. As such, it’s very important to fact check all of its answers.



k

which is heavier, 10kg iron or 10kg cotton?

10kg iron is heavier than 10kg cotton. Iron is typically heavier than cotton, so the answer is 10kg iron is heavier. Answer: \boxed{iron}.

Source: Gary Marcus

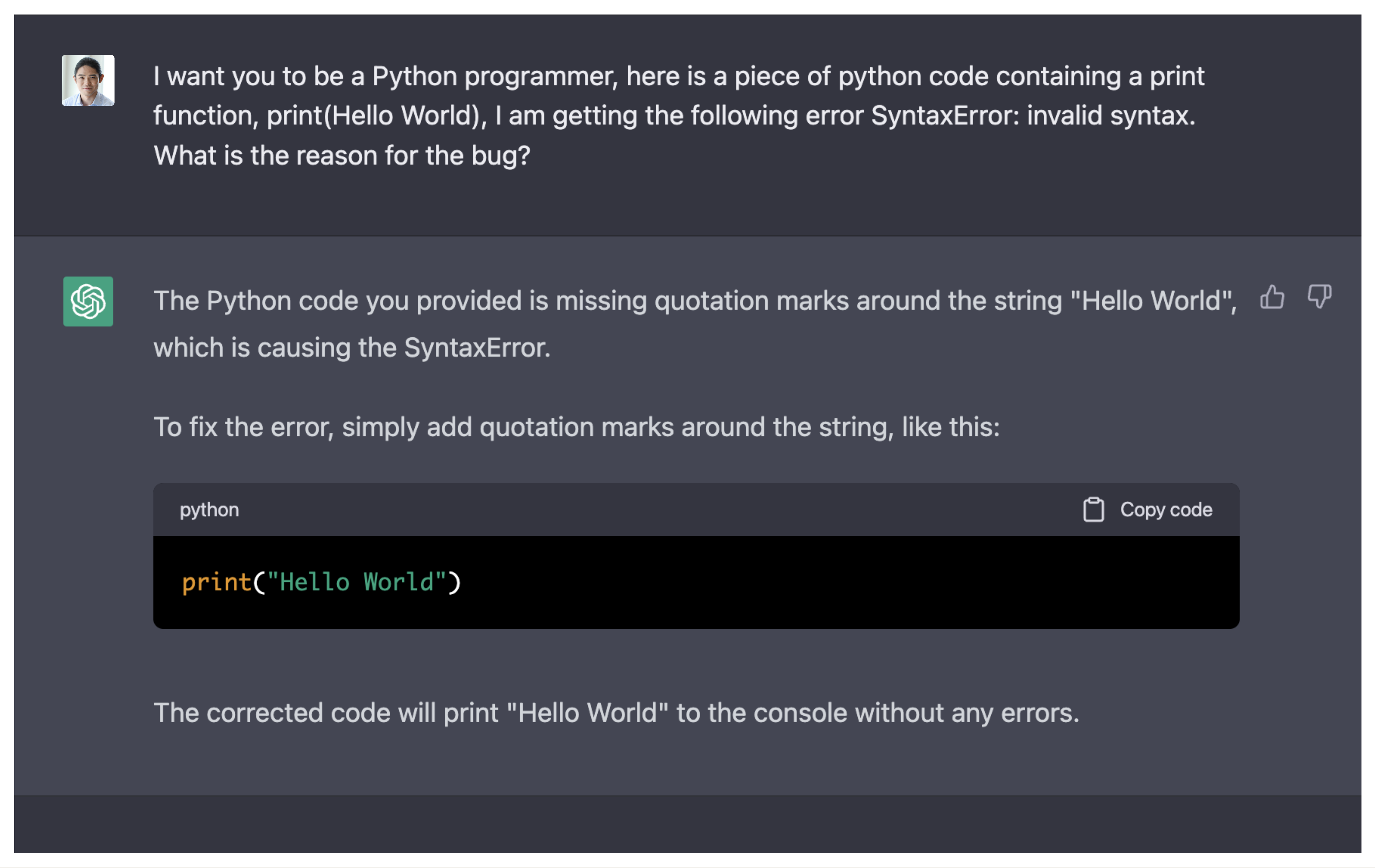
datacamp

Prompts for general coding workflows

Debugging Python code

PROMPT STRUCTURE

I want you to be a Python programmer, here is a piece of Python code containing {problem} — {insert code snippet} — I am getting the following error {insert error}. What is the reason for the bug?



I want you to be a Python programmer, here is a piece of python code containing a print function, print(Hello World), I am getting the following error SyntaxError: invalid syntax. What is the reason for the bug?

The Python code you provided is missing quotation marks around the string "Hello World", which is causing the SyntaxError.

To fix the error, simply add quotation marks around the string, like this:

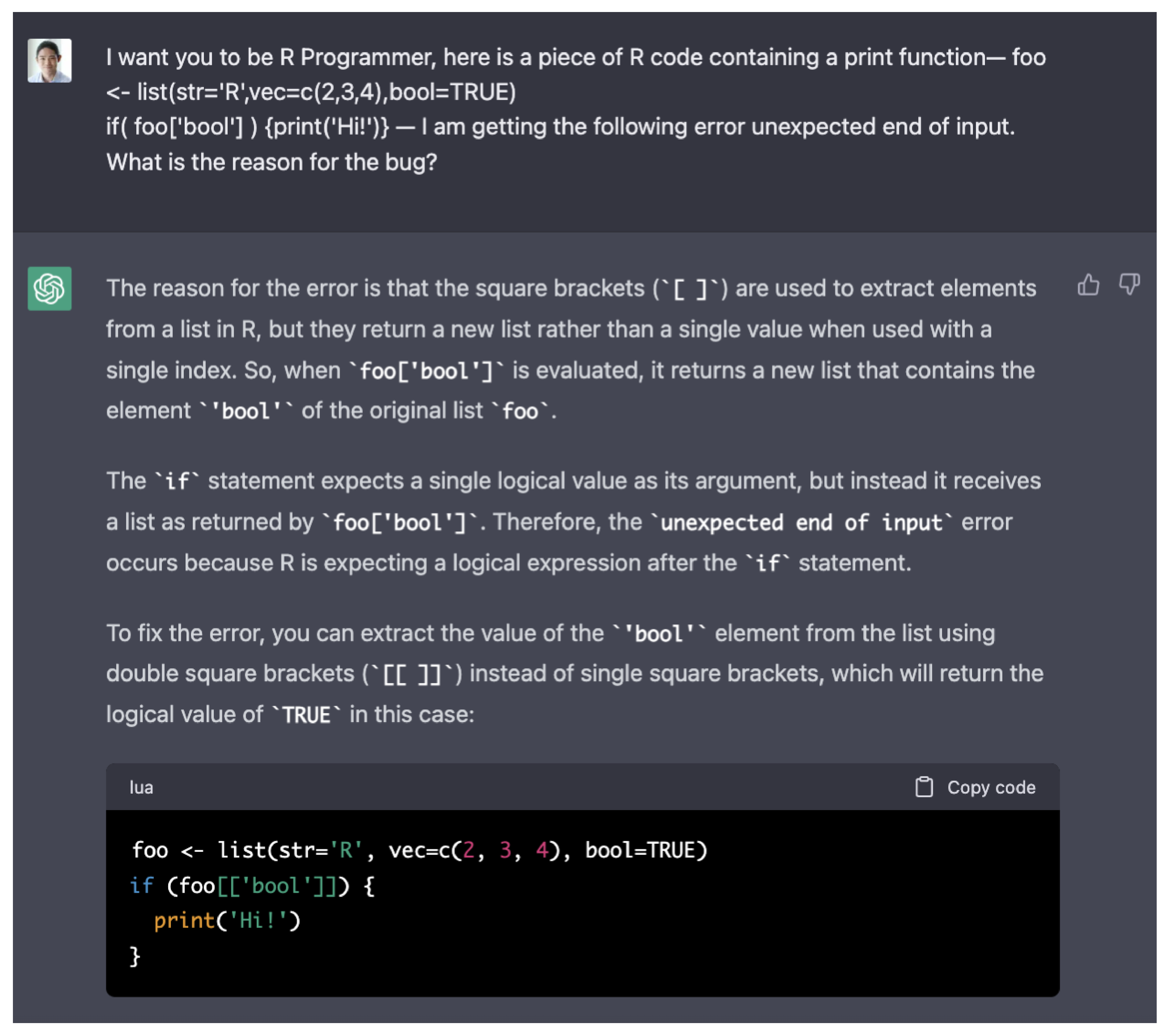
```python
print("Hello World")
```

The corrected code will print "Hello World" to the console without any errors.

Debugging R code

PROMPT STRUCTURE

I want you to be an R programmer, here is a piece of R code containing {problem} — {insert code snippet} — I am getting the following error {insert error}. What is the reason for the bug?



I want you to be R Programmer, here is a piece of R code containing a print function— foo <- list(str='R',vec=c(2,3,4),bool=TRUE)
if(foo['bool']) {print('Hi!')} — I am getting the following error unexpected end of input. What is the reason for the bug?

The reason for the error is that the square brackets (`[ ]`) are used to extract elements from a list in R, but they return a new list rather than a single value when used with a single index. So, when `foo['bool']` is evaluated, it returns a new list that contains the element `'bool'` of the original list `foo`.

The `if` statement expects a single logical value as its argument, but instead it receives a list as returned by `foo['bool']`. Therefore, the `unexpected end of input` error occurs because R is expecting a logical expression after the `if` statement.

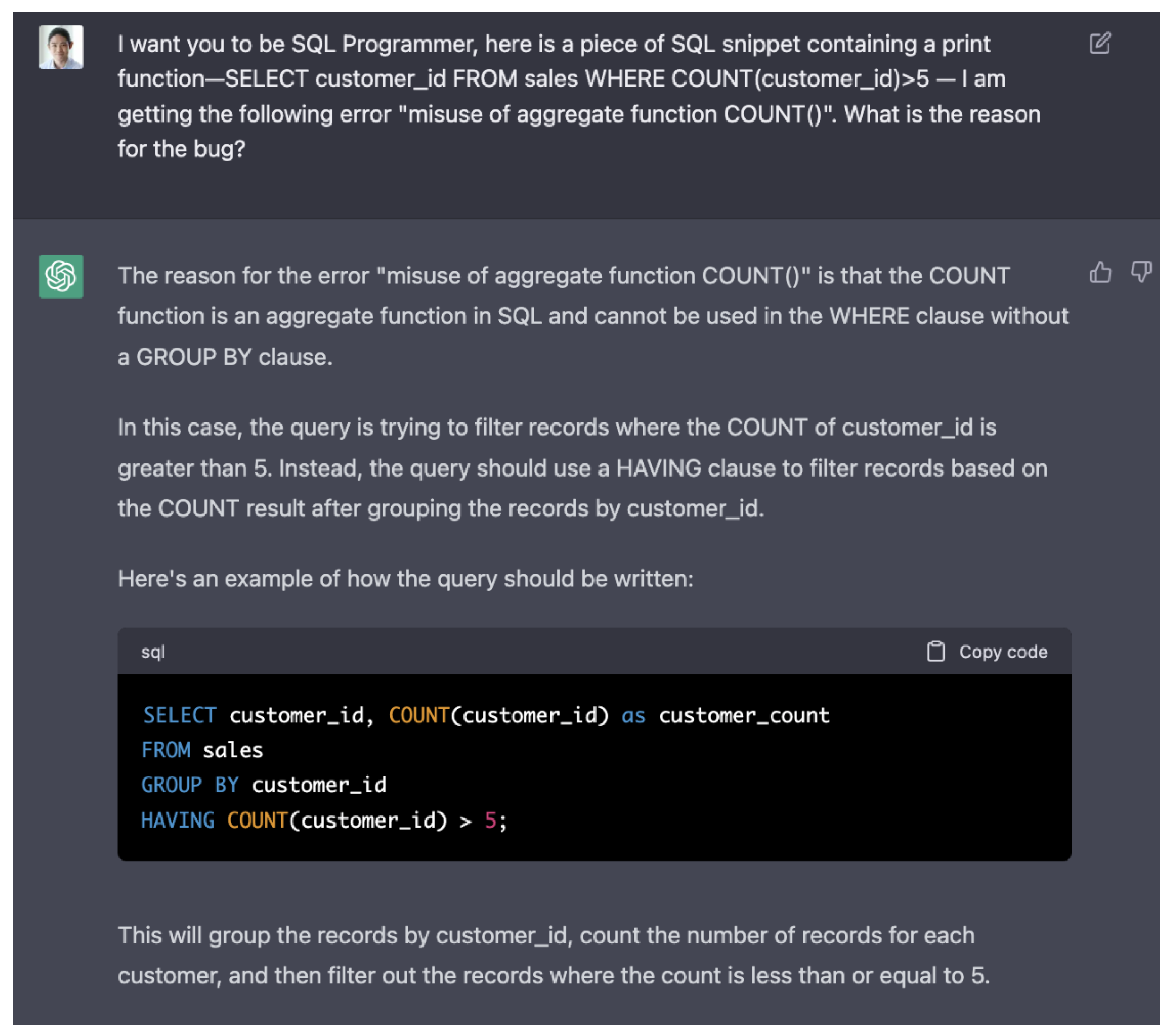
To fix the error, you can extract the value of the `'bool'` element from the list using double square brackets (`[[ ]]`) instead of single square brackets, which will return the logical value of `TRUE` in this case:

```lua
foo <- list(str='R', vec=c(2, 3, 4), bool=TRUE)
if (foo[['bool']]) {
  print('Hi!')
}
```

Debugging SQL code

PROMPT STRUCTURE

I want you to be a SQL programmer, here is a piece of SQL code containing {problem} — {insert code snippet} — I am getting the following error {insert error}. What is the reason for the bug?



I want you to be SQL Programmer, here is a piece of SQL snippet containing a print function—SELECT customer_id FROM sales WHERE COUNT(customer_id)>5 — I am getting the following error "misuse of aggregate function COUNT()". What is the reason for the bug?

The reason for the error "misuse of aggregate function COUNT()" is that the COUNT function is an aggregate function in SQL and cannot be used in the WHERE clause without a GROUP BY clause.

In this case, the query is trying to filter records where the COUNT of customer_id is greater than 5. Instead, the query should use a HAVING clause to filter records based on the COUNT result after grouping the records by customer_id.

Here's an example of how the query should be written:

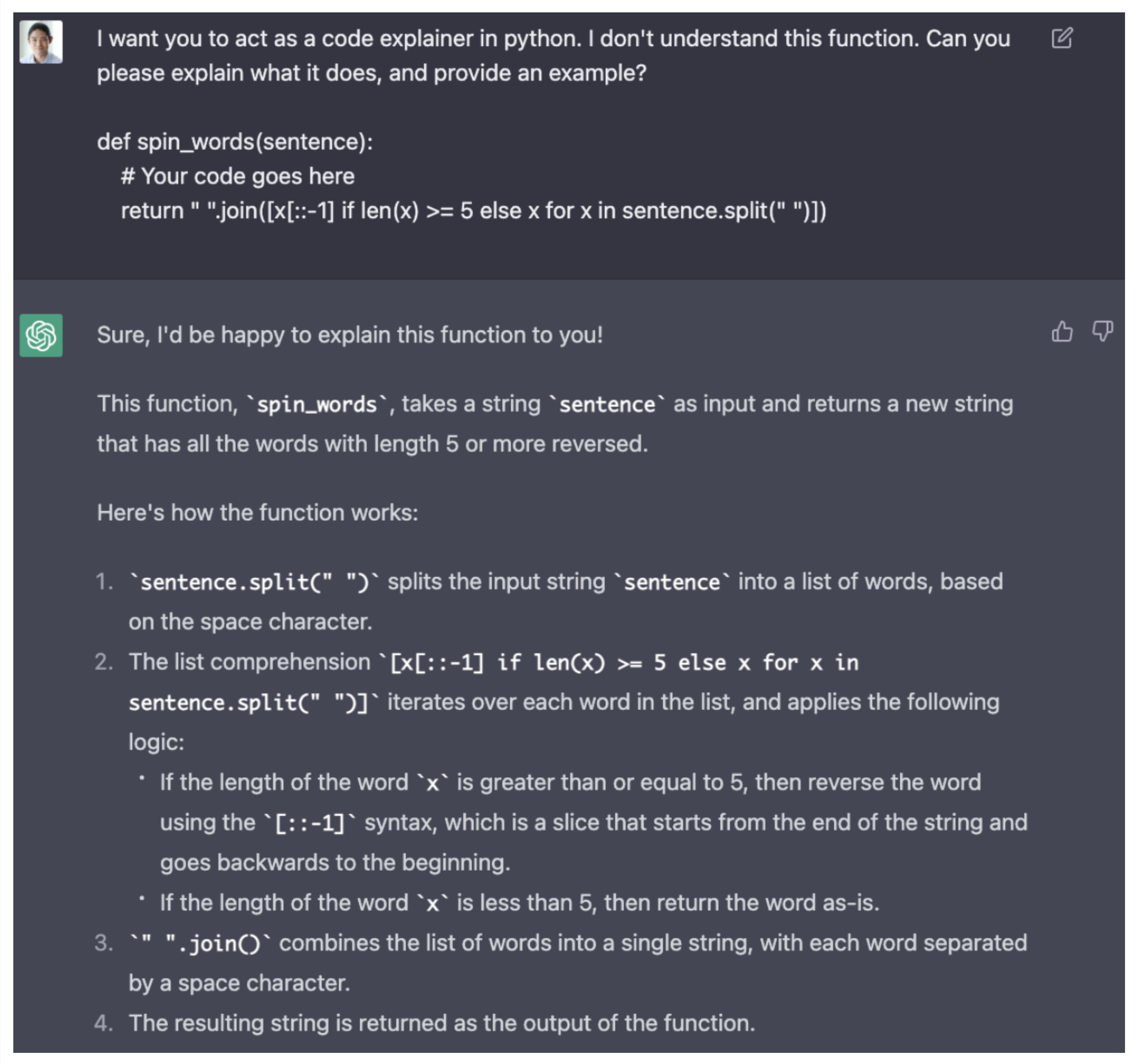
```sql
SELECT customer_id, COUNT(customer_id) as customer_count
FROM sales
GROUP BY customer_id
HAVING COUNT(customer_id) > 5;
```

This will group the records by customer_id, count the number of records for each customer, and then filter out the records where the count is less than or equal to 5.

Python code explanation

PROMPT STRUCTURE

I want you to act as a code explainer in Python. I don't understand this function. Can you please explain what it does, and provide an example? {Insert function}



I want you to act as a code explainer in python. I don't understand this function. Can you please explain what it does, and provide an example?

```
def spin_words(sentence):
    # Your code goes here
    return " ".join([x[::-1] if len(x) >= 5 else x for x in sentence.split(" ")])
```

Sure, I'd be happy to explain this function to you!

This function, `spin_words`, takes a string `sentence` as input and returns a new string that has all the words with length 5 or more reversed.

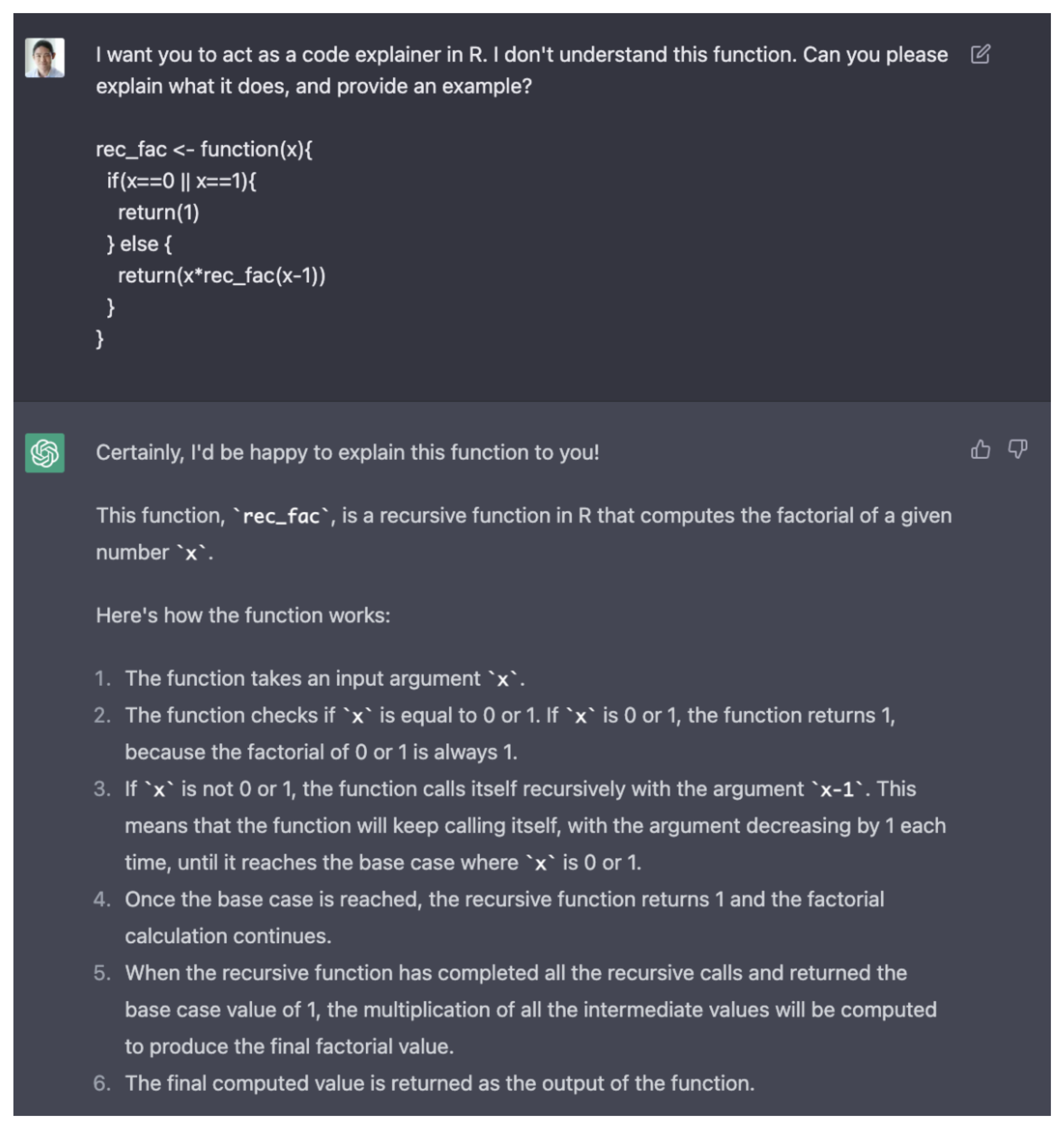
Here's how the function works:

1. `sentence.split(" ")` splits the input string `sentence` into a list of words, based on the space character.
2. The list comprehension `[x[::-1] if len(x) >= 5 else x for x in sentence.split(" ")]` iterates over each word in the list, and applies the following logic:
   - If the length of the word `x` is greater than or equal to 5, then reverse the word using the `[::-1]` syntax, which is a slice that starts from the end of the string and goes backwards to the beginning.
   - If the length of the word `x` is less than 5, then return the word as-is.
3. `" ".join()` combines the list of words into a single string, with each word separated by a space character.
4. The resulting string is returned as the output of the function.

R code explanation

PROMPT STRUCTURE

I want you to act as a code explainer in R. I don't understand this function. Can you please explain what it does, and provide an example? {Insert function}



I want you to act as a code explainer in R. I don't understand this function. Can you please explain what it does, and provide an example?

```
rec_fac <- function(x){
 if(x==0 || x==1){
  return(1)
 } else {
  return(x*rec_fac(x-1))
 }
}
```

Certainly, I'd be happy to explain this function to you!

This function, `rec_fac`, is a recursive function in R that computes the factorial of a given number `x`.

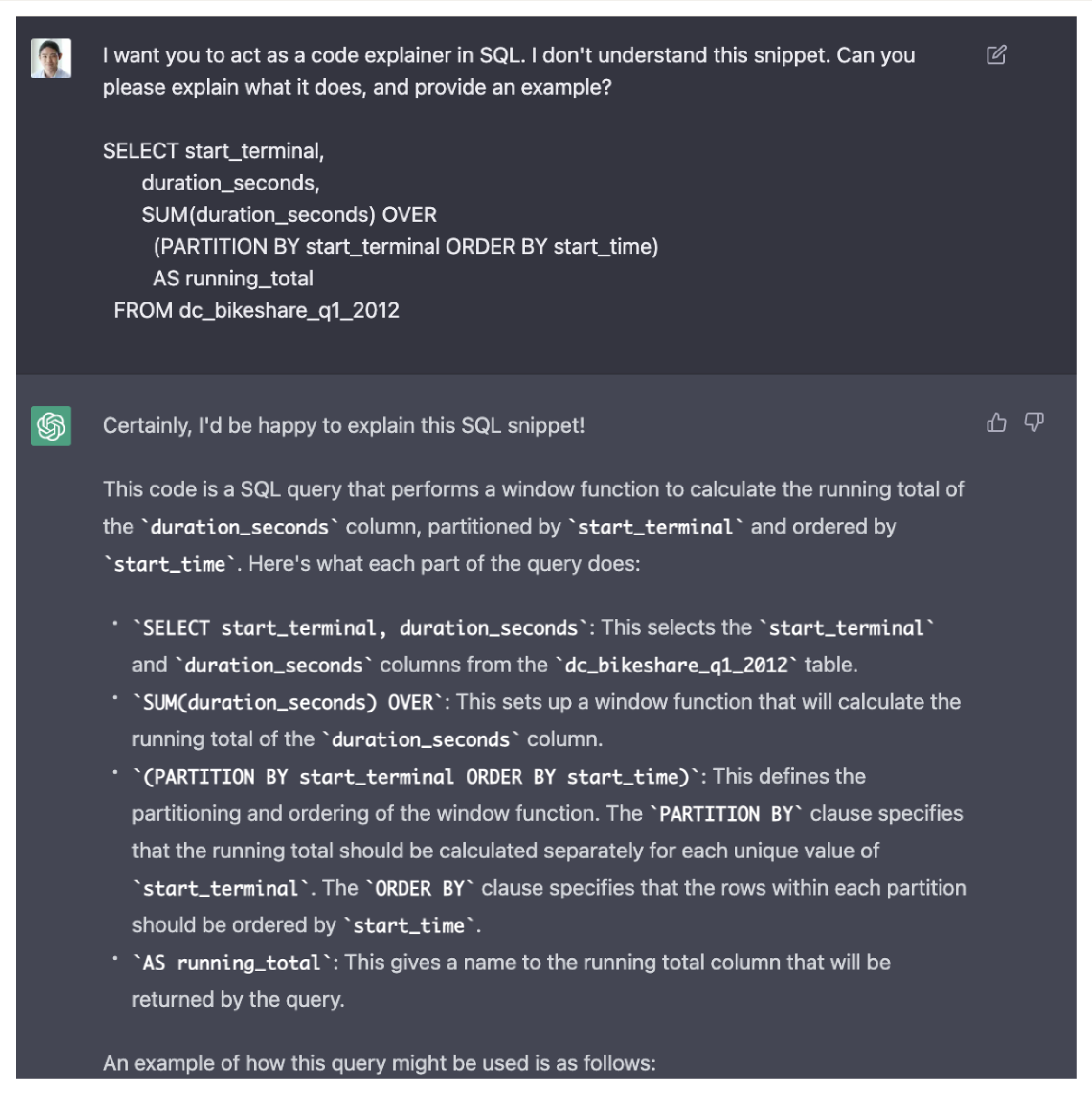
Here's how the function works:

1. The function takes an input argument `x`.
2. The function checks if `x` is equal to 0 or 1. If `x` is 0 or 1, the function returns 1, because the factorial of 0 or 1 is always 1.
3. If `x` is not 0 or 1, the function calls itself recursively with the argument `x-1`. This means that the function will keep calling itself, with the argument decreasing by 1 each time, until it reaches the base case where `x` is 0 or 1.
4. Once the base case is reached, the recursive function returns 1 and the factorial calculation continues.
5. When the recursive function has completed all the recursive calls and returned the base case value of 1, the multiplication of all the intermediate values will be computed to produce the final factorial value.
6. The final computed value is returned as the output of the function.

SQL code explanation

PROMPT STRUCTURE

I want you to act as a code explainer in SQL. I don't understand this snippet. Can you please explain what it does, and provide an example?
{Insert SQL query}



I want you to act as a code explainer in SQL. I don't understand this snippet. Can you please explain what it does, and provide an example?

```
SELECT start_terminal,
       duration_seconds,
       SUM(duration_seconds) OVER
         (PARTITION BY start_terminal ORDER BY start_time)
         AS running_total
  FROM dc_bikeshare_q1_2012
```

Certainly, I'd be happy to explain this SQL snippet!

This code is a SQL query that performs a window function to calculate the running total of the `duration_seconds` column, partitioned by `start_terminal` and ordered by `start_time`. Here's what each part of the query does:

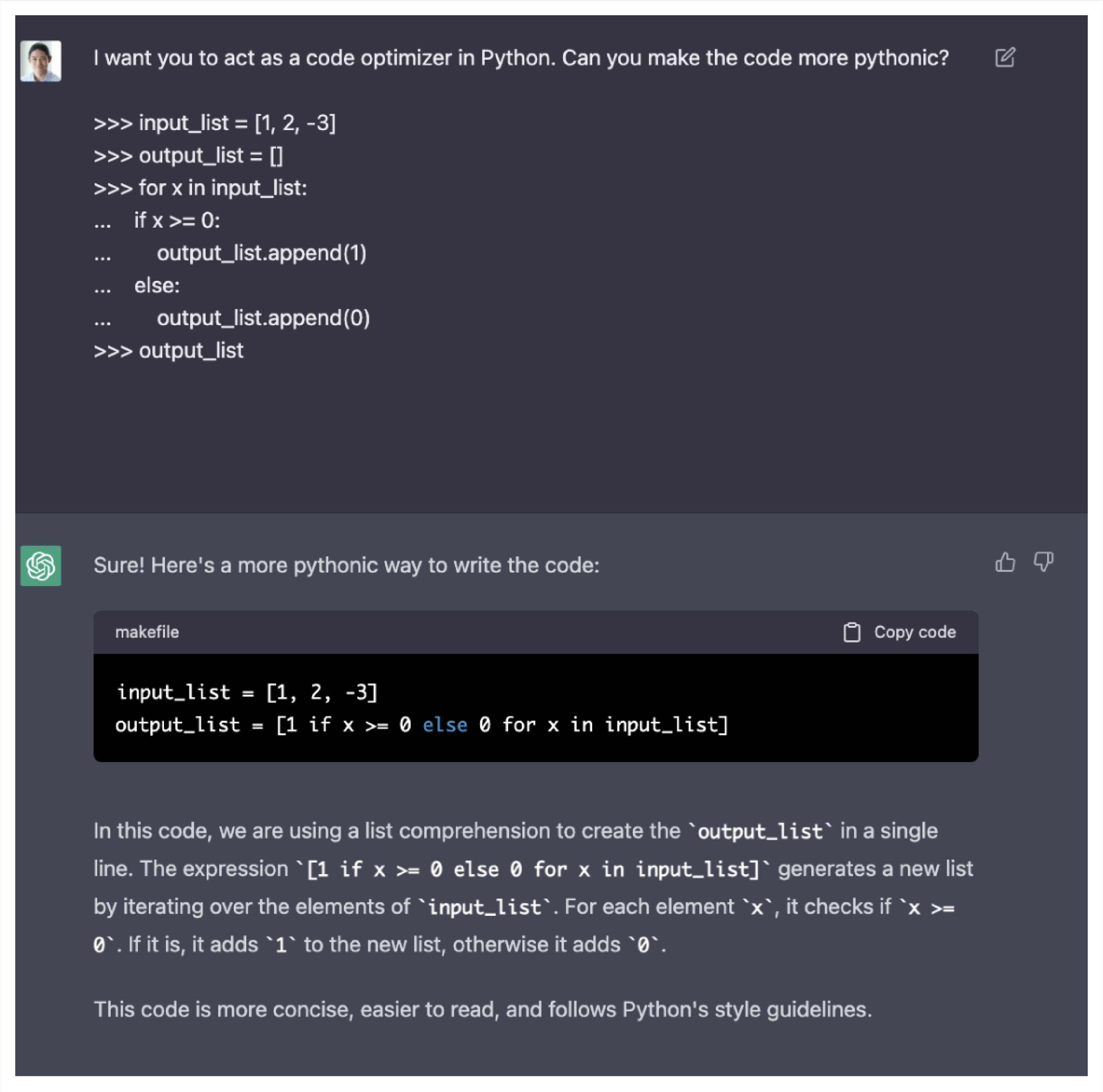
- `SELECT start_terminal, duration_seconds`: This selects the `start_terminal` and `duration_seconds` columns from the `dc_bikeshare_q1_2012` table.
- `SUM(duration_seconds) OVER`: This sets up a window function that will calculate the running total of the `duration_seconds` column.
- `(PARTITION BY start_terminal ORDER BY start_time)`: This defines the partitioning and ordering of the window function. The `PARTITION BY` clause specifies that the running total should be calculated separately for each unique value of `start_terminal`. The `ORDER BY` clause specifies that the rows within each partition should be ordered by `start_time`.
- `AS running_total`: This gives a name to the running total column that will be returned by the query.

An example of how this query might be used is as follows:

Python code optimization

PROMPT STRUCTURE

I want you to act as a code optimizer in Python. {Describe problem with current code, if possible}. Can you make the code {more Pythonic/cleaner/more efficient/run faster/more readable}? {Insert Code}



I want you to act as a code optimizer in Python. Can you make the code more pythonic?

```
>>> input_list = [1, 2, -3]
>>> output_list = []
>>> for x in input_list:
...    if x >= 0:
...        output_list.append(1)
...    else:
...        output_list.append(0)
>>> output_list
```

Sure! Here's a more pythonic way to write the code:

```
input_list = [1, 2, -3]
output_list = [1 if x >= 0 else 0 for x in input_list]
```

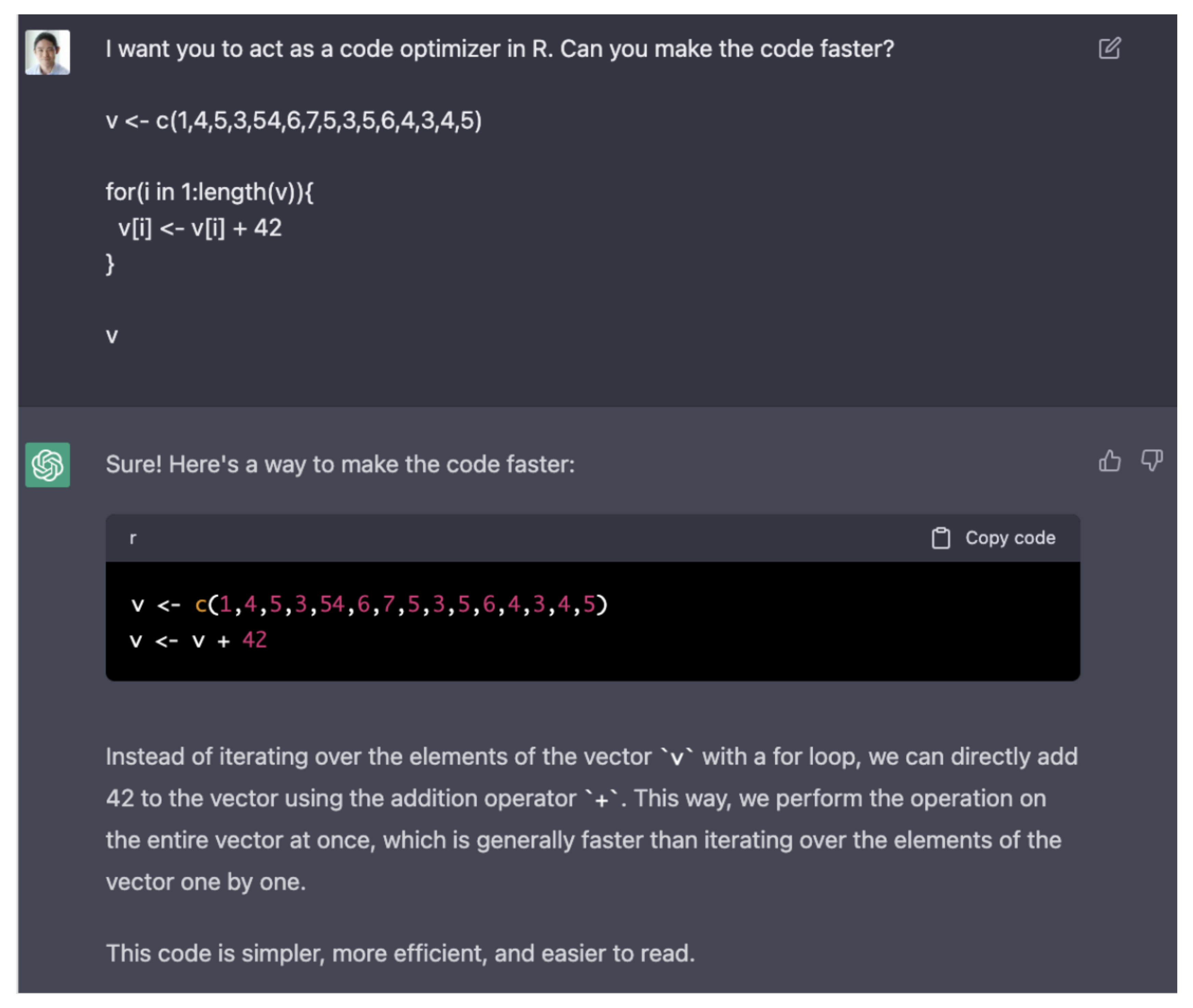
In this code, we are using a list comprehension to create the `output_list` in a single line. The expression `[1 if x >= 0 else 0 for x in input_list]` generates a new list by iterating over the elements of `input_list`. For each element `x`, it checks if `x >= 0`. If it is, it adds `1` to the new list, otherwise it adds `0`.

This code is more concise, easier to read, and follows Python's style guidelines.

R code optimization

PROMPT STRUCTURE

I want you to act as a code optimizer in R. {Describe problem with current code, if possible}. Can you make the code {cleaner/more efficient/run faster/more readable}? {Insert Code}



I want you to act as a code optimizer in R. Can you make the code faster?

```
v <- c(1,4,5,3,54,6,7,5,3,5,6,4,3,4,5)

for(i in 1:length(v)){
  v[i] <- v[i] + 42
}

v
```

Sure! Here's a way to make the code faster:

```r
v <- c(1,4,5,3,54,6,7,5,3,5,6,4,3,4,5)
v <- v + 42
```

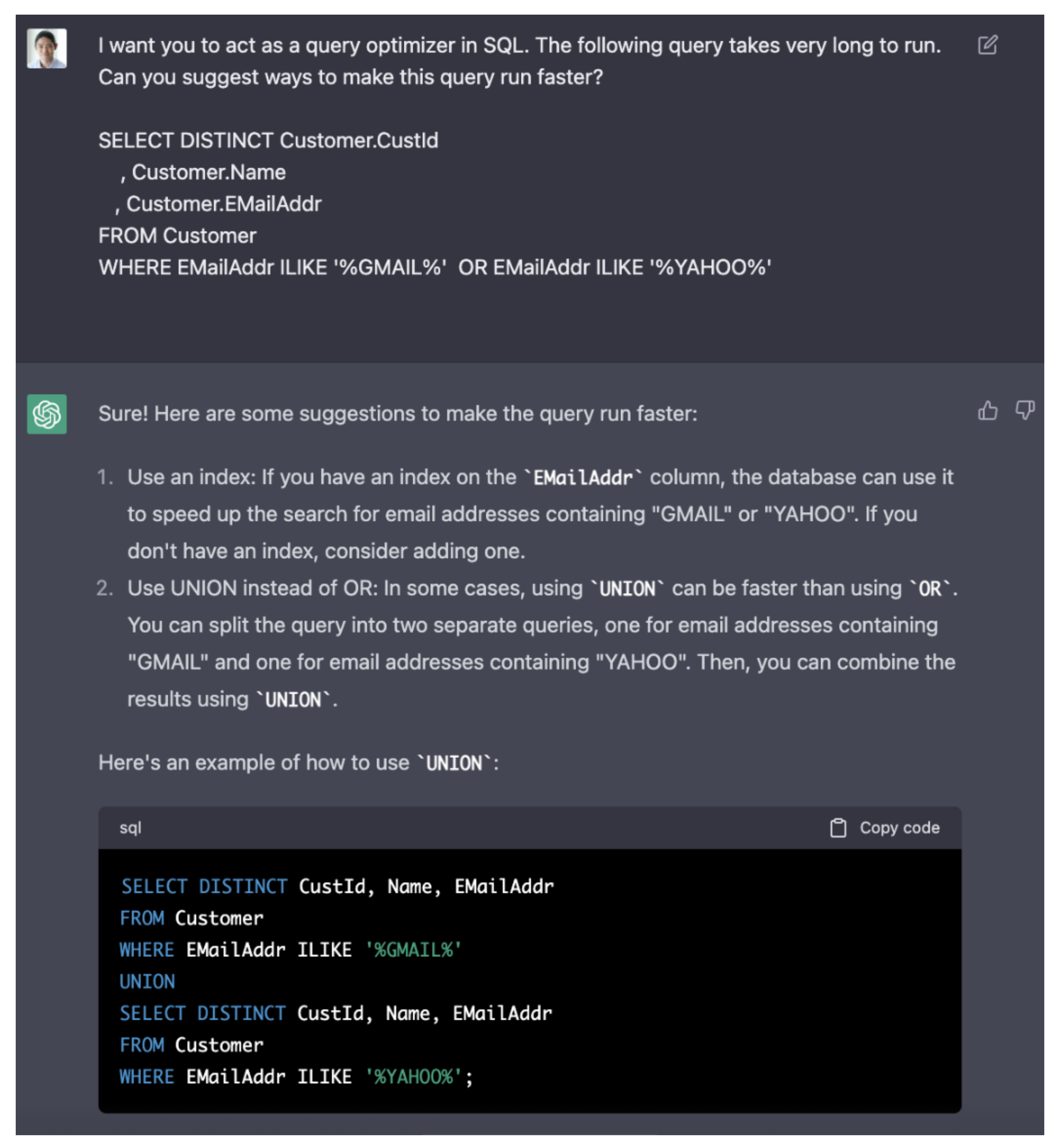
Instead of iterating over the elements of the vector `v` with a for loop, we can directly add 42 to the vector using the addition operator `+`. This way, we perform the operation on the entire vector at once, which is generally faster than iterating over the elements of the vector one by one.

This code is simpler, more efficient, and easier to read.

SQL code optimization

PROMPT STRUCTURE

I want you to act as a query optimizer in SQL. {Describe problem with current code, if possible}. Can you suggest ways to make the query {run faster/more readable/simpler}? {Insert Code}



I want you to act as a query optimizer in SQL. The following query takes very long to run. Can you suggest ways to make this query run faster?

```
SELECT DISTINCT Customer.CustId
    , Customer.Name
   , Customer.EMailAddr
FROM Customer
WHERE EMailAddr ILIKE '%GMAIL%'  OR EMailAddr ILIKE '%YAHOO%'
```

Sure! Here are some suggestions to make the query run faster:

1. Use an index: If you have an index on the `EMailAddr` column, the database can use it to speed up the search for email addresses containing "GMAIL" or "YAHOO". If you don't have an index, consider adding one.
2. Use UNION instead of OR: In some cases, using `UNION` can be faster than using `OR`. You can split the query into two separate queries, one for email addresses containing "GMAIL" and one for email addresses containing "YAHOO". Then, you can combine the results using `UNION`.

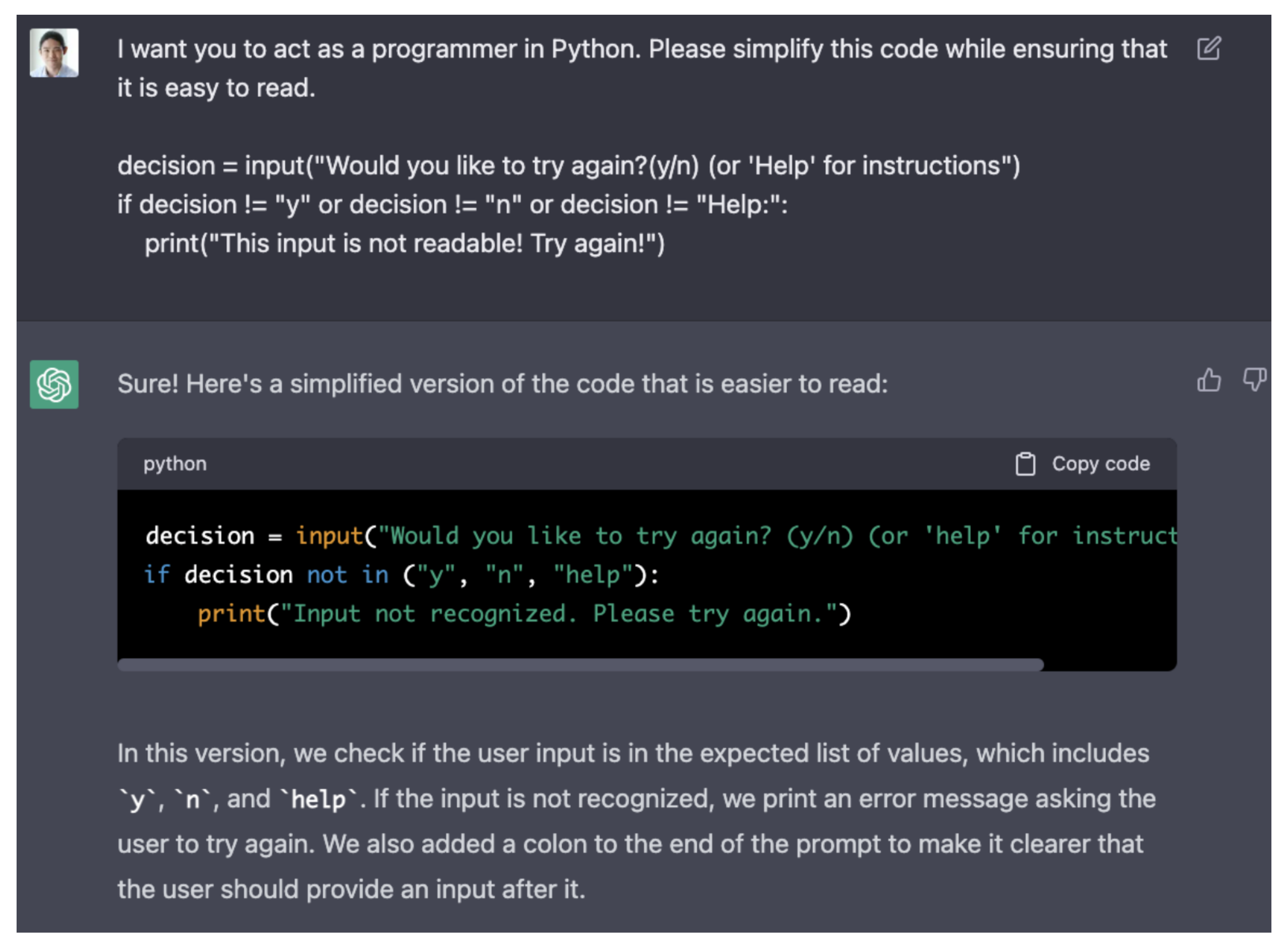
Here's an example of how to use `UNION`:

```sql
SELECT DISTINCT CustId, Name, EMailAddr
FROM Customer
WHERE EMailAddr ILIKE '%GMAIL%'
UNION
SELECT DISTINCT CustId, Name, EMailAddr
FROM Customer
WHERE EMailAddr ILIKE '%YAHOO%';
```

Python code simplification

PROMPT STRUCTURE

I want you to act as a programmer in Python. Please simplify this code while ensuring that it is {efficient/easy to read/Pythonic}? {Insert Code}



I want you to act as a programmer in Python. Please simplify this code while ensuring that it is easy to read.

```
decision = input("Would you like to try again?(y/n) (or 'Help' for instructions")
if decision != "y" or decision != "n" or decision != "Help:":
   print("This input is not readable! Try again!")
```

Sure! Here's a simplified version of the code that is easier to read:

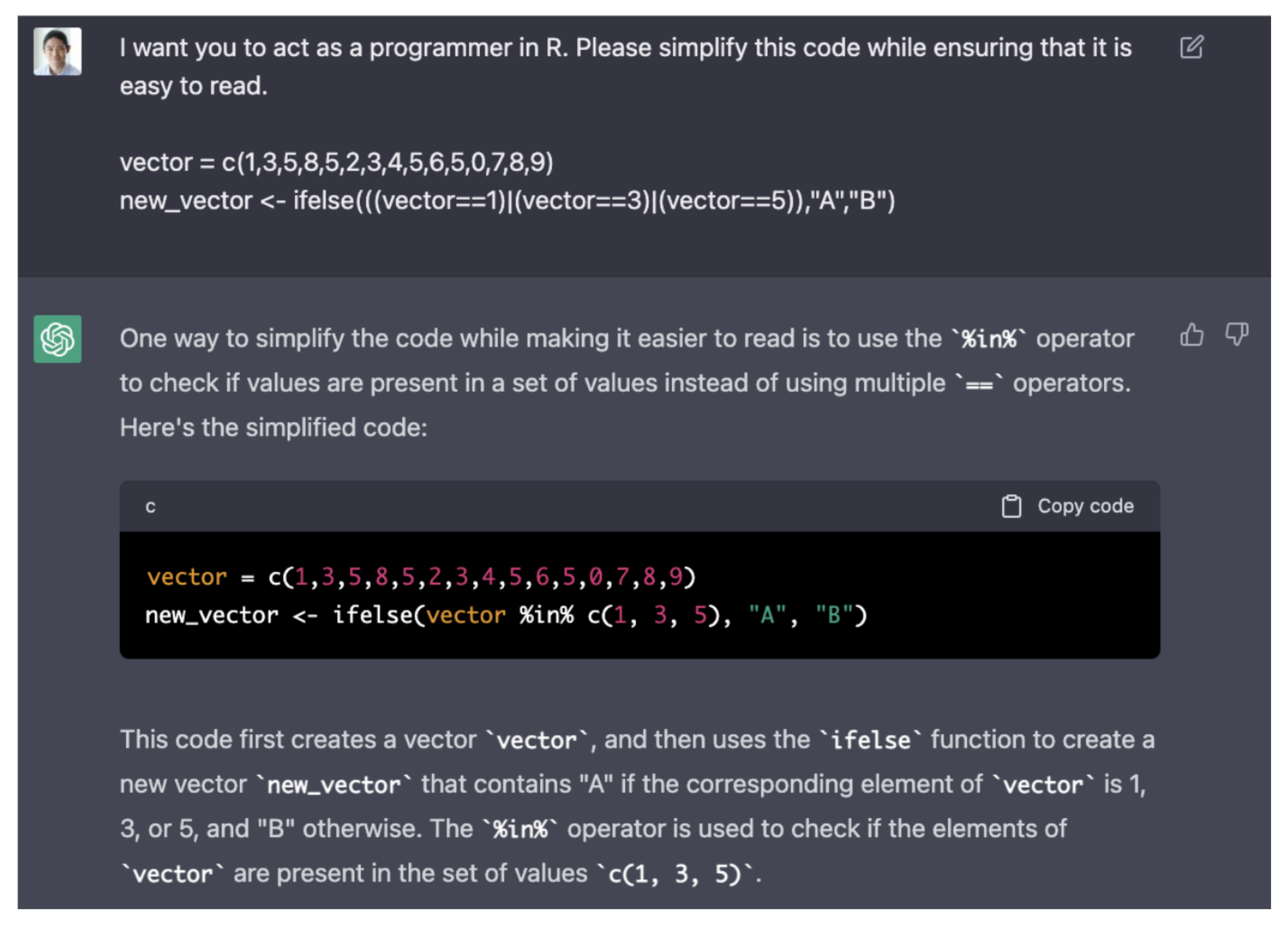
```python
decision = input("Would you like to try again? (y/n) (or 'help' for instruct
if decision not in ("y", "n", "help"):
    print("Input not recognized. Please try again.")
```

In this version, we check if the user input is in the expected list of values, which includes `y`, `n`, and `help`. If the input is not recognized, we print an error message asking the user to try again. We also added a colon to the end of the prompt to make it clearer that the user should provide an input after it.

R code simplification

PROMPT STRUCTURE

I want you to act as a programmer in R. Please simplify this code while ensuring that it is {efficient/easy to read}? {Insert Code}



I want you to act as a programmer in R. Please simplify this code while ensuring that it is easy to read.

vector = c(1,3,5,8,5,2,3,4,5,6,5,0,7,8,9)
new_vector <- ifelse(((vector==1)|(vector==3)|(vector==5)),"A","B")

One way to simplify the code while making it easier to read is to use the `%in%` operator to check if values are present in a set of values instead of using multiple `==` operators. Here's the simplified code:

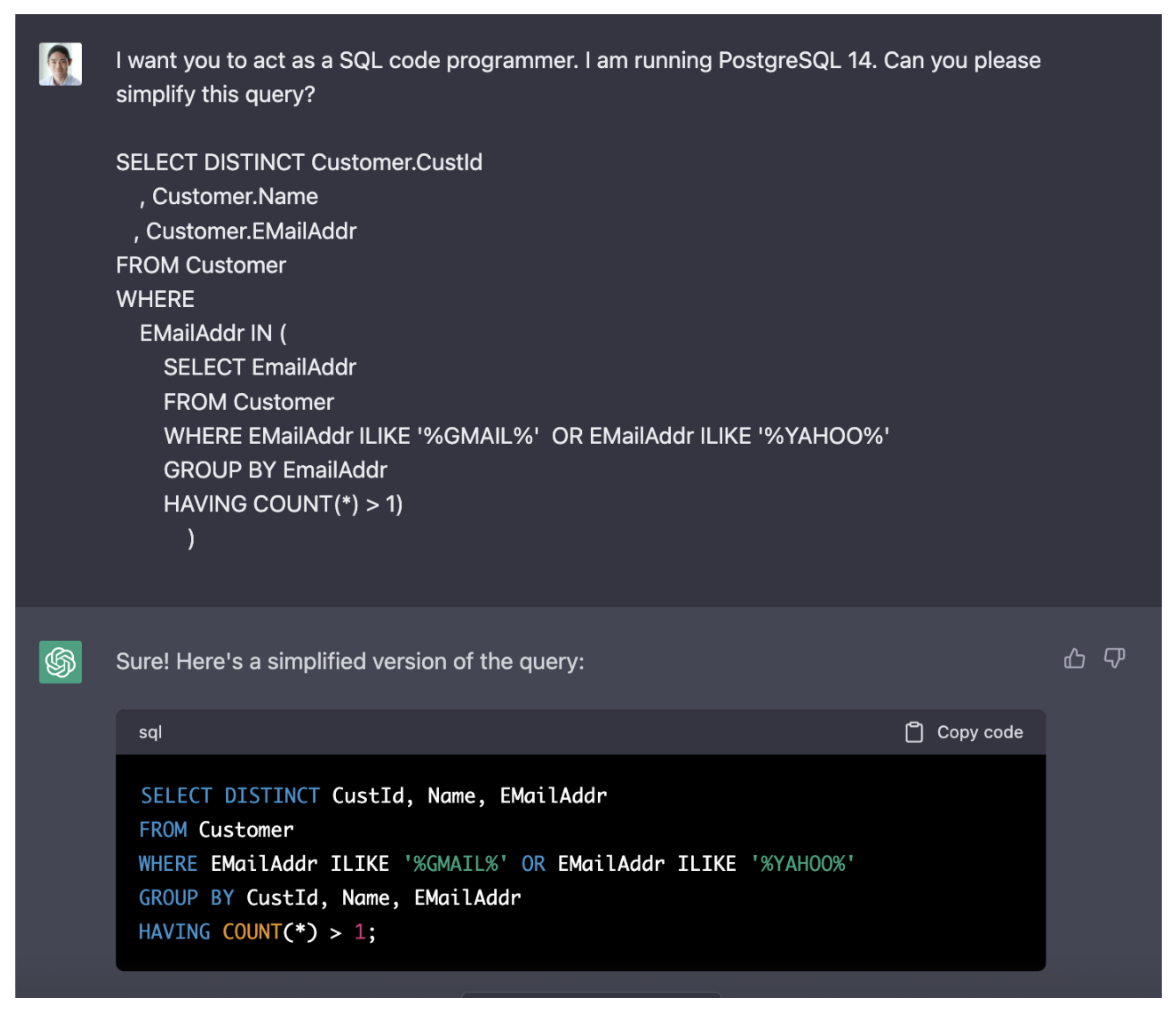
```
vector = c(1,3,5,8,5,2,3,4,5,6,5,0,7,8,9)
new_vector <- ifelse(vector %in% c(1, 3, 5), "A", "B")
```

This code first creates a vector `vector`, and then uses the `ifelse` function to create a new vector `new_vector` that contains "A" if the corresponding element of `vector` is 1, 3, or 5, and "B" otherwise. The `%in%` operator is used to check if the elements of `vector` are present in the set of values `c(1, 3, 5)`.

SQL code simplification

PROMPT STRUCTURE

I want you to act as a SQL programmer. I am running {PostgreSQL 14/MySQL 8/SQLite 3.4/other versions.}. Can you please simplify this query {while ensuring that it is efficient/ easy to read/insert any additional requirements}?



I want you to act as a SQL code programmer. I am running PostgreSQL 14. Can you please simplify this query?

```
SELECT DISTINCT Customer.CustId
    , Customer.Name
   , Customer.EMailAddr
FROM Customer
WHERE
    EMailAddr IN (
        SELECT EmailAddr
        FROM Customer
        WHERE EMailAddr ILIKE '%GMAIL%'  OR EMailAddr ILIKE '%YAHOO%'
        GROUP BY EmailAddr
        HAVING COUNT(*) > 1)
            )
```

Sure! Here's a simplified version of the query:

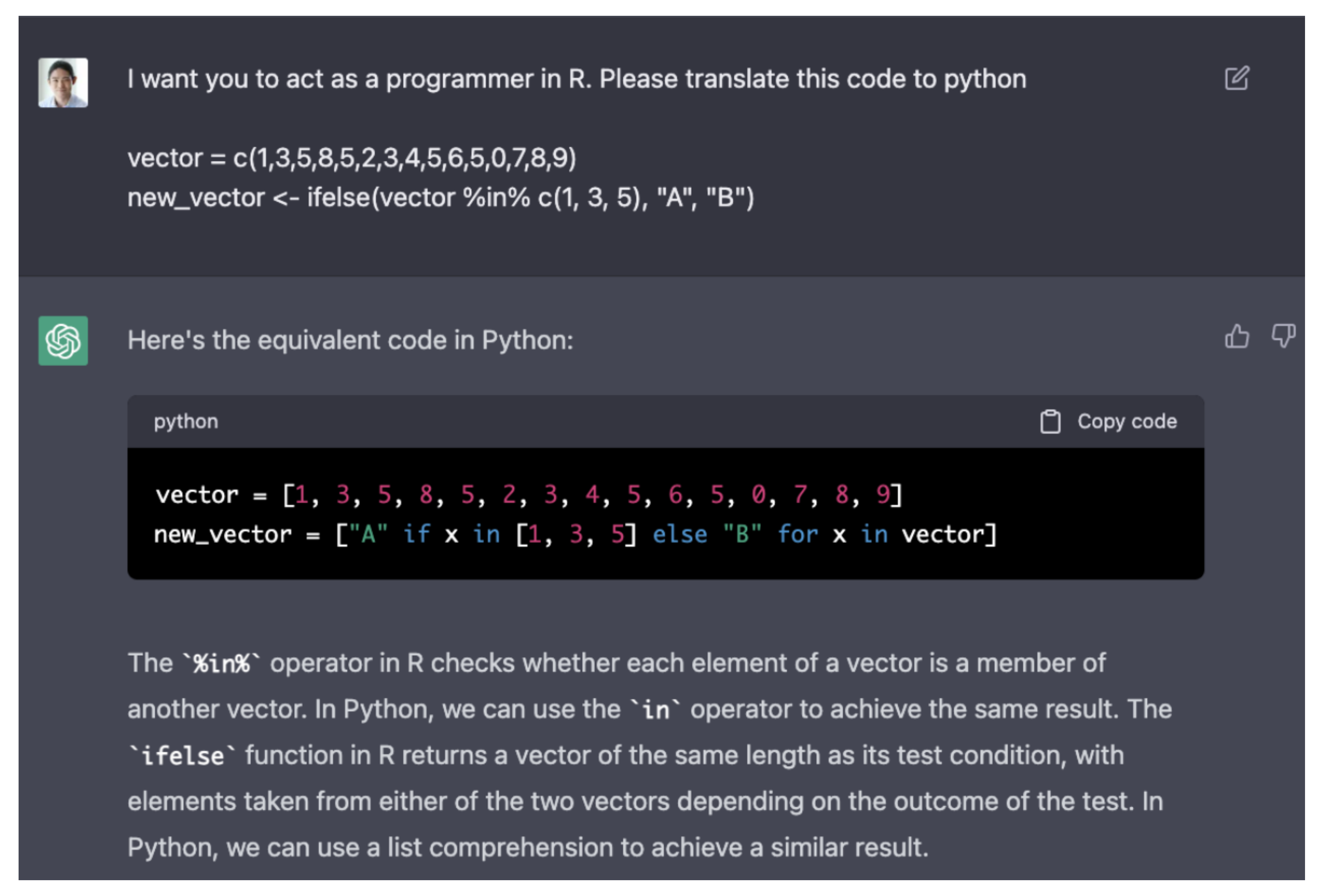
```sql
SELECT DISTINCT CustId, Name, EMailAddr
FROM Customer
WHERE EMailAddr ILIKE '%GMAIL%' OR EMailAddr ILIKE '%YAHOO%'
GROUP BY CustId, Name, EMailAddr
HAVING COUNT(*) > 1;
```

datacamp

From R to Python code translation

PROMPT STRUCTURE

I want you to act as a programmer in R. Please translate this code to Python. {Insert code}



I want you to act as a programmer in R. Please translate this code to python

vector = c(1,3,5,8,5,2,3,4,5,6,5,0,7,8,9)
new_vector <- ifelse(vector %in% c(1, 3, 5), "A", "B")

Here's the equivalent code in Python:

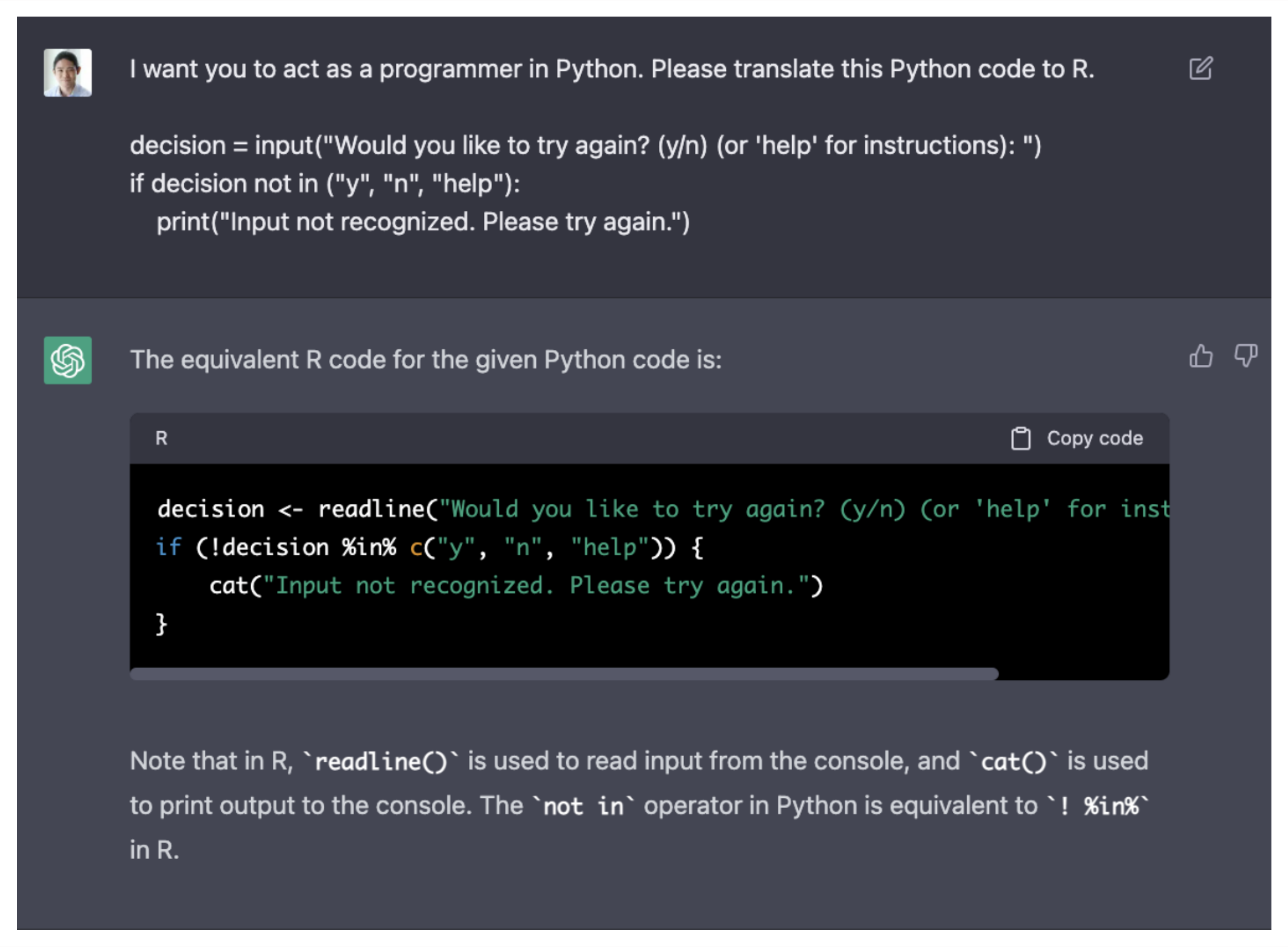
```python
vector = [1, 3, 5, 8, 5, 2, 3, 4, 5, 6, 5, 0, 7, 8, 9]
new_vector = ["A" if x in [1, 3, 5] else "B" for x in vector]
```

The `%in%` operator in R checks whether each element of a vector is a member of another vector. In Python, we can use the `in` operator to achieve the same result. The `ifelse` function in R returns a vector of the same length as its test condition, with elements taken from either of the two vectors depending on the outcome of the test. In Python, we can use a list comprehension to achieve a similar result.

From Python to R code translation

PROMPT STRUCTURE

I want you to act as a programmer in Python. Please translate this code to R. {Insert code}



I want you to act as a programmer in Python. Please translate this Python code to R.

```
decision = input("Would you like to try again? (y/n) (or 'help' for instructions): ")
if decision not in ("y", "n", "help"):
   print("Input not recognized. Please try again.")
```

The equivalent R code for the given Python code is:

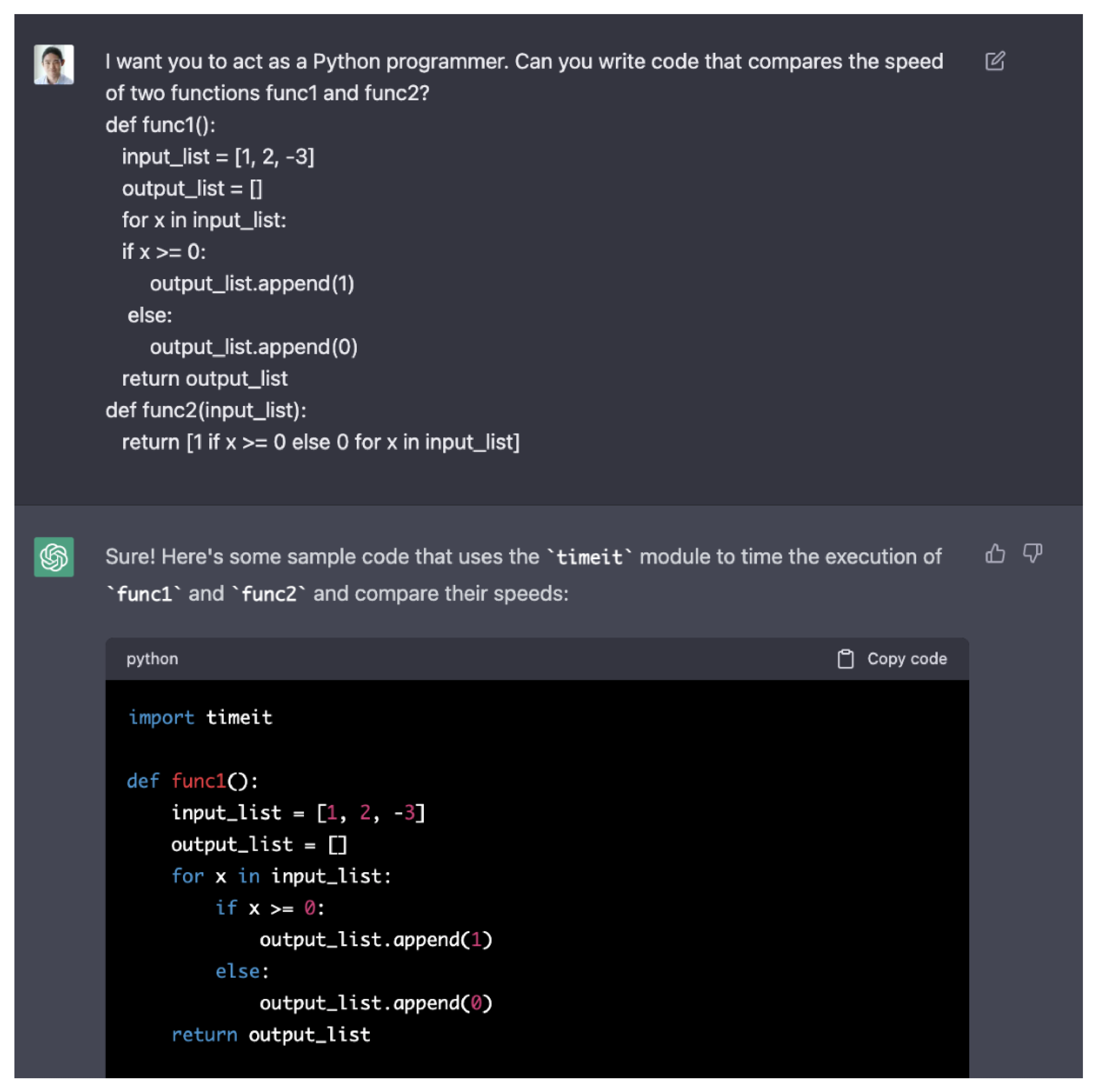
```r
decision <- readline("Would you like to try again? (y/n) (or 'help' for inst
if (!decision %in% c("y", "n", "help")) {
    cat("Input not recognized. Please try again.")
}
```

Note that in R, `readline()` is used to read input from the console, and `cat()` is used to print output to the console. The `not in` operator in Python is equivalent to `! %in%` in R.

Compare function speeds in python

PROMPT STRUCTURE

I want you to act as a Python programmer. Can you write code that compares the speed of two functions {functionname} and {functionname}? {Insert functions}



I want you to act as a Python programmer. Can you write code that compares the speed of two functions func1 and func2?

```
def func1():
  input_list = [1, 2, -3]
  output_list = []
  for x in input_list:
  if x >= 0:
      output_list.append(1)
   else:
      output_list.append(0)
  return output_list
def func2(input_list):
  return [1 if x >= 0 else 0 for x in input_list]
```

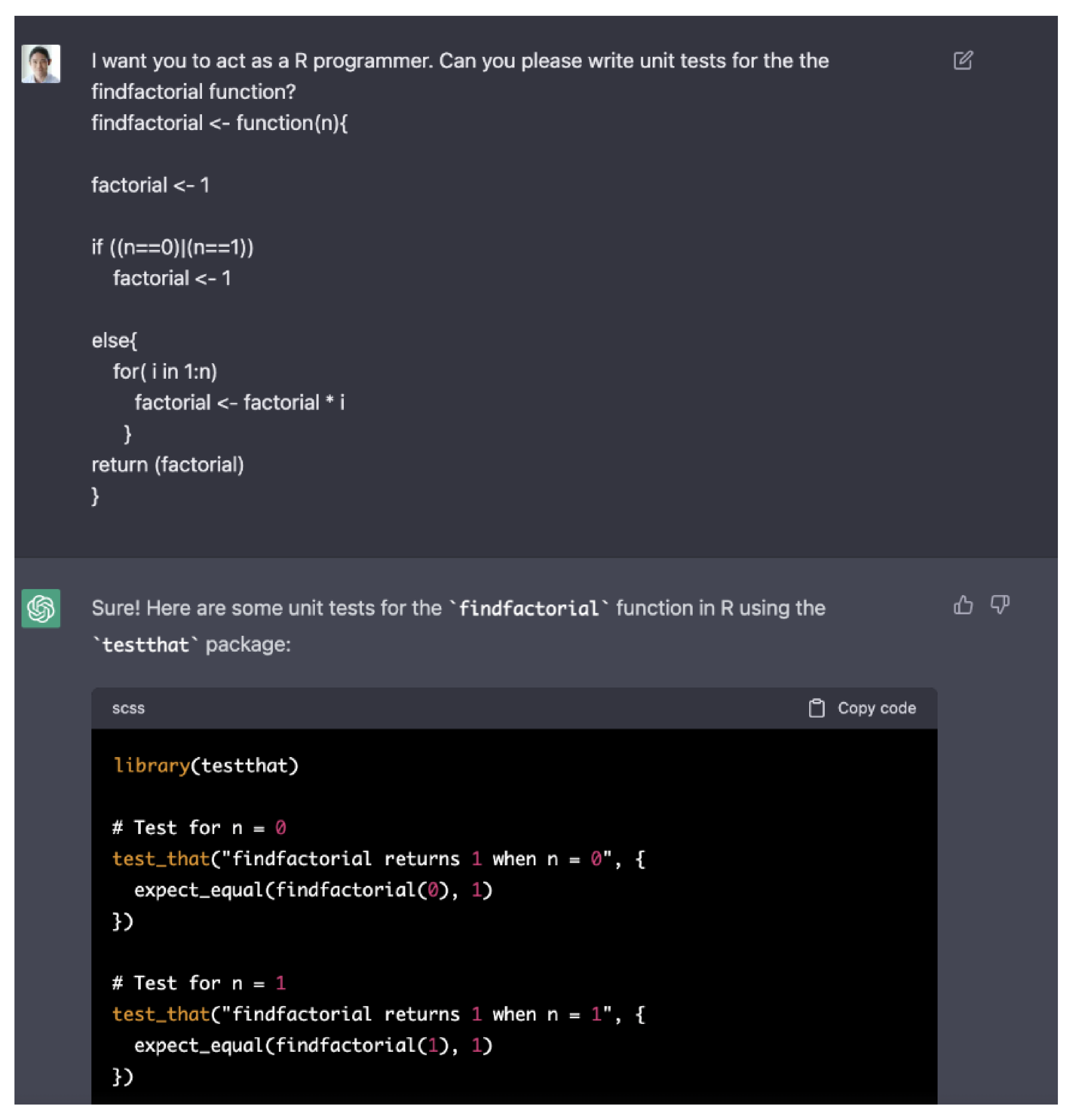
Sure! Here's some sample code that uses the `timeit` module to time the execution of `func1` and `func2` and compare their speeds:

```python
import timeit

def func1():
    input_list = [1, 2, -3]
    output_list = []
    for x in input_list:
        if x >= 0:
            output_list.append(1)
        else:
            output_list.append(0)
    return output_list
```

Write unit tests in R

PROMPT STRUCTURE

I want you to act as a R Programmer. Can you please write unit tests for the function {functionname}? {Insert requirements for unit tests, if any} {Insert code}



I want you to act as a R programmer. Can you please write unit tests for the the findfactorial function?

```
findfactorial <- function(n){

factorial <- 1

if ((n==0)|(n==1))
   factorial <- 1

else{
   for( i in 1:n)
      factorial <- factorial * i
    }
return (factorial)
}
```

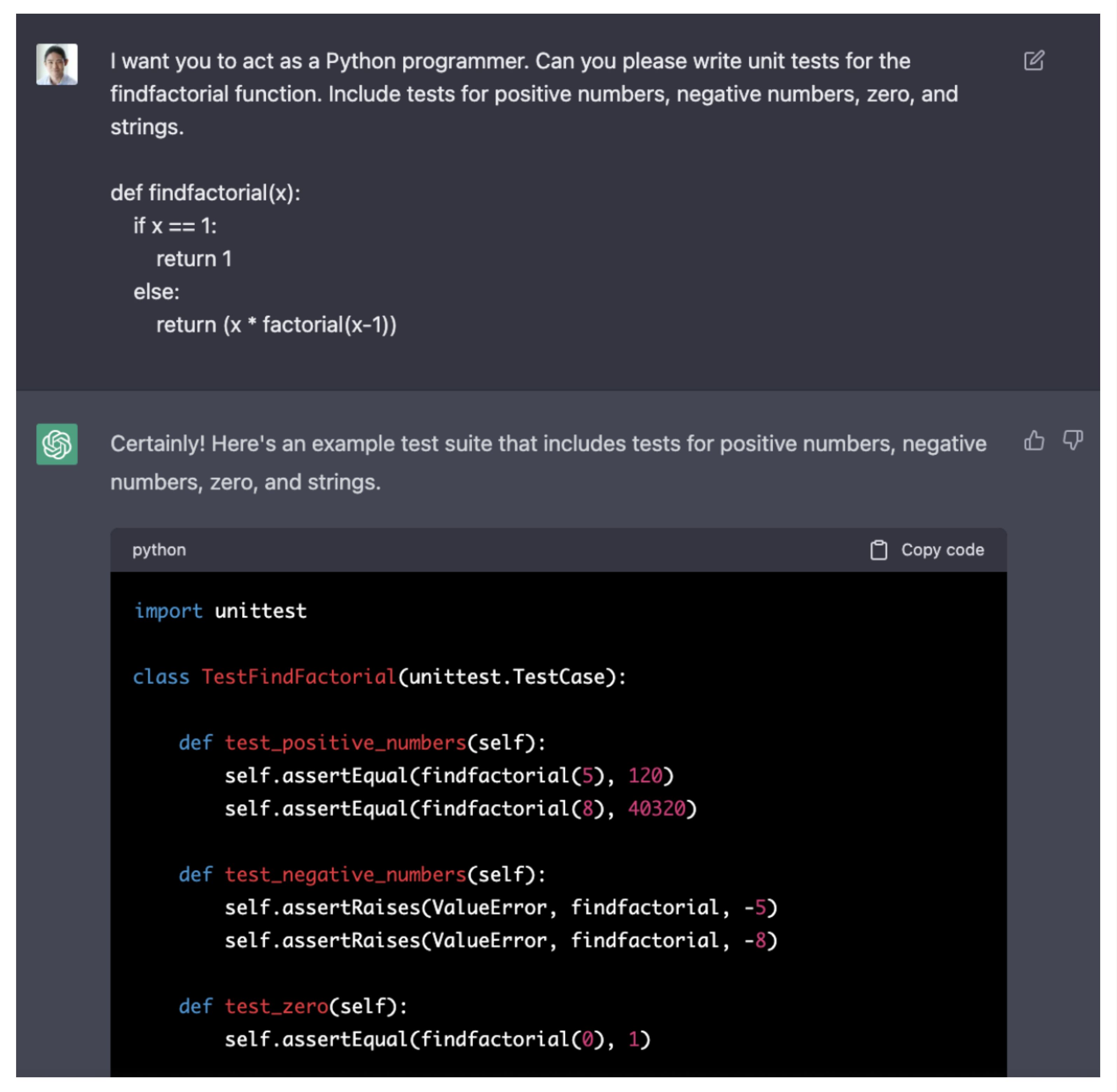
Sure! Here are some unit tests for the `findfactorial` function in R using the `testthat` package:

```
library(testthat)

# Test for n = 0
test_that("findfactorial returns 1 when n = 0", {
  expect_equal(findfactorial(0), 1)
})

# Test for n = 1
test_that("findfactorial returns 1 when n = 1", {
  expect_equal(findfactorial(1), 1)
})
```

Write unit tests in Python

PROMPT STRUCTURE

I want you to act as a Python Programmer. Can you please write unit tests for the function {functionname}? {Insert requirements for unit tests, if any} {Insert code}



I want you to act as a Python programmer. Can you please write unit tests for the findfactorial function. Include tests for positive numbers, negative numbers, zero, and strings.

```
def findfactorial(x):
    if x == 1:
        return 1
    else:
        return (x * factorial(x-1))
```

Certainly! Here's an example test suite that includes tests for positive numbers, negative numbers, zero, and strings.

```python
import unittest

class TestFindFactorial(unittest.TestCase):

    def test_positive_numbers(self):
        self.assertEqual(findfactorial(5), 120)
        self.assertEqual(findfactorial(8), 40320)

    def test_negative_numbers(self):
        self.assertRaises(ValueError, findfactorial, -5)
        self.assertRaises(ValueError, findfactorial, -8)

    def test_zero(self):
        self.assertEqual(findfactorial(0), 1)
```

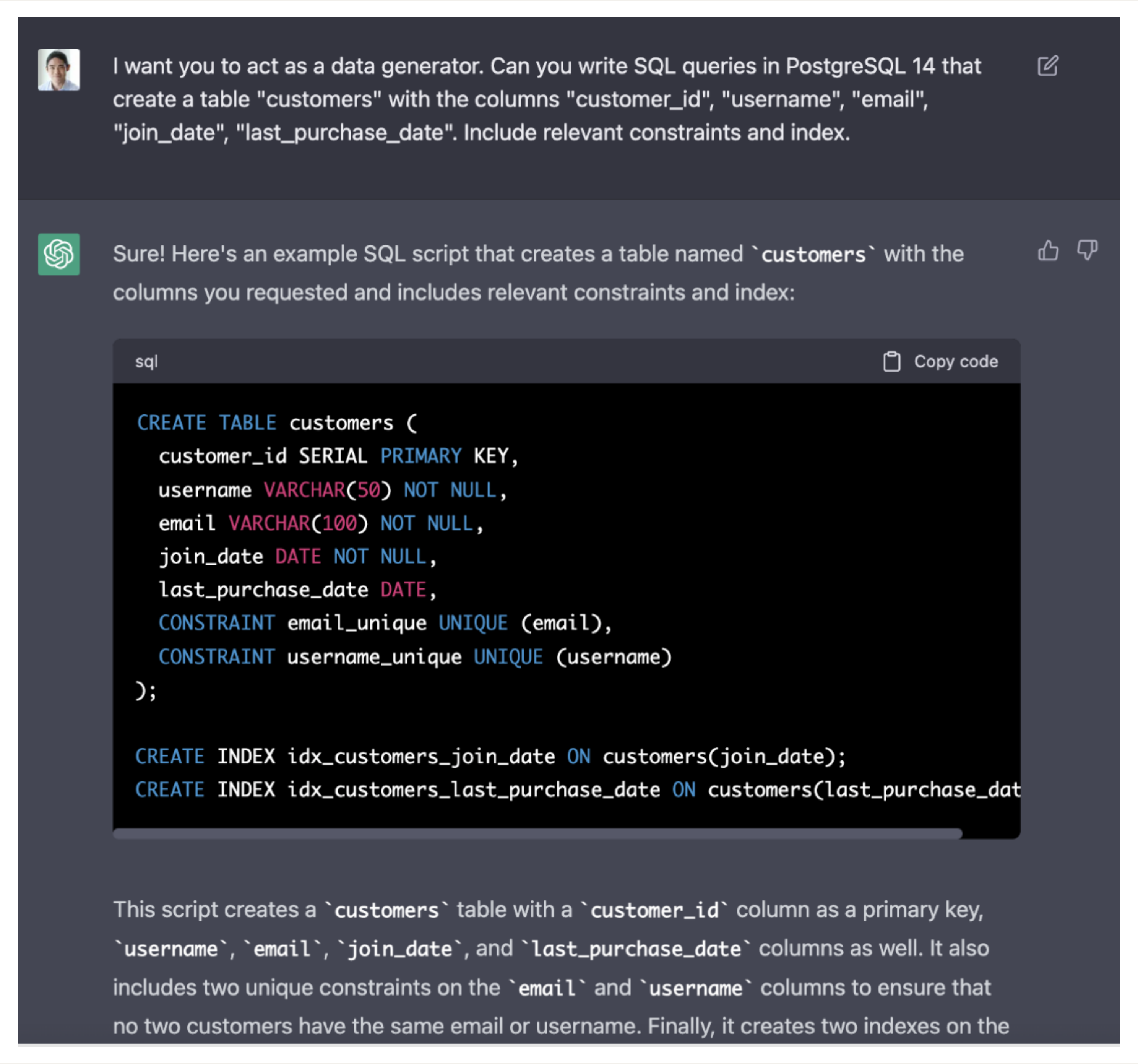
datacamp

Prompts for data analysis workflows

Data generation & creating tables

PROMPT STRUCTURE

I want you to act as a data generator. Can you write SQL queries in {database version} that create a table {table name} with the columns {column name}. Include relevant constraints and index.



I want you to act as a data generator. Can you write SQL queries in PostgreSQL 14 that create a table "customers" with the columns "customer_id", "username", "email", "join_date", "last_purchase_date". Include relevant constraints and index.

Sure! Here's an example SQL script that creates a table named `customers` with the columns you requested and includes relevant constraints and index:

sql

Copy code

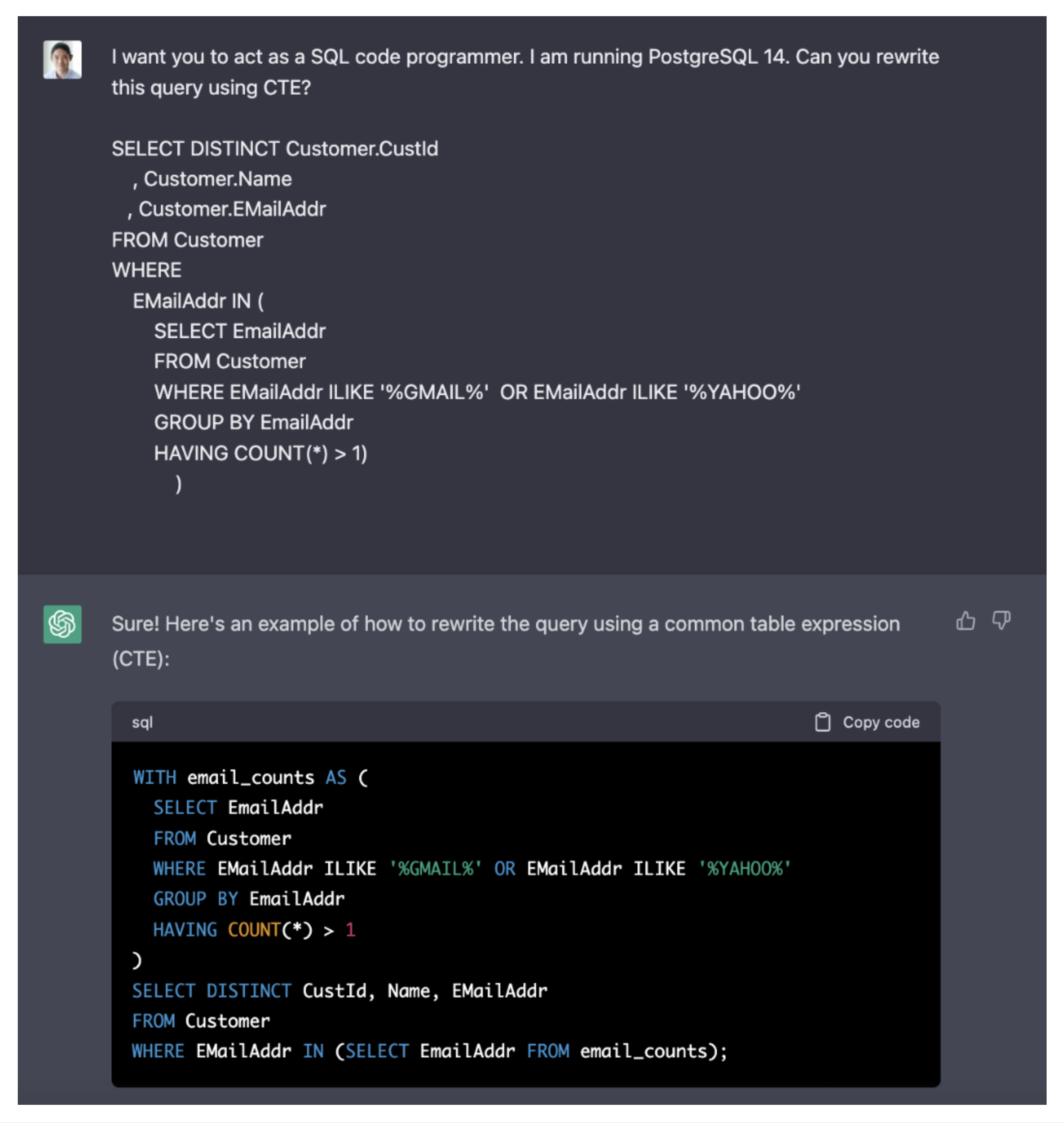
```sql
CREATE TABLE customers (
  customer_id SERIAL PRIMARY KEY,
  username VARCHAR(50) NOT NULL,
  email VARCHAR(100) NOT NULL,
  join_date DATE NOT NULL,
  last_purchase_date DATE,
  CONSTRAINT email_unique UNIQUE (email),
  CONSTRAINT username_unique UNIQUE (username)
);

CREATE INDEX idx_customers_join_date ON customers(join_date);
CREATE INDEX idx_customers_last_purchase_date ON customers(last_purchase_dat
```

This script creates a `customers` table with a `customer_id` column as a primary key, `username`, `email`, `join_date`, and `last_purchase_date` columns as well. It also includes two unique constraints on the `email` and `username` columns to ensure that no two customers have the same email or username. Finally, it creates two indexes on the

Common table expressions

PROMPT STRUCTURE

I want you to act as a SQL code programmer. I am running {database version}. Can you rewrite this query using CTE? {Insert query}



I want you to act as a SQL code programmer. I am running PostgreSQL 14. Can you rewrite this query using CTE?

```
SELECT DISTINCT Customer.CustId
   , Customer.Name
  , Customer.EMailAddr
FROM Customer
WHERE
   EMailAddr IN (
      SELECT EmailAddr
      FROM Customer
      WHERE EMailAddr ILIKE '%GMAIL%'  OR EMailAddr ILIKE '%YAHOO%'
      GROUP BY EmailAddr
      HAVING COUNT(*) > 1)
         )
```

Sure! Here's an example of how to rewrite the query using a common table expression (CTE):

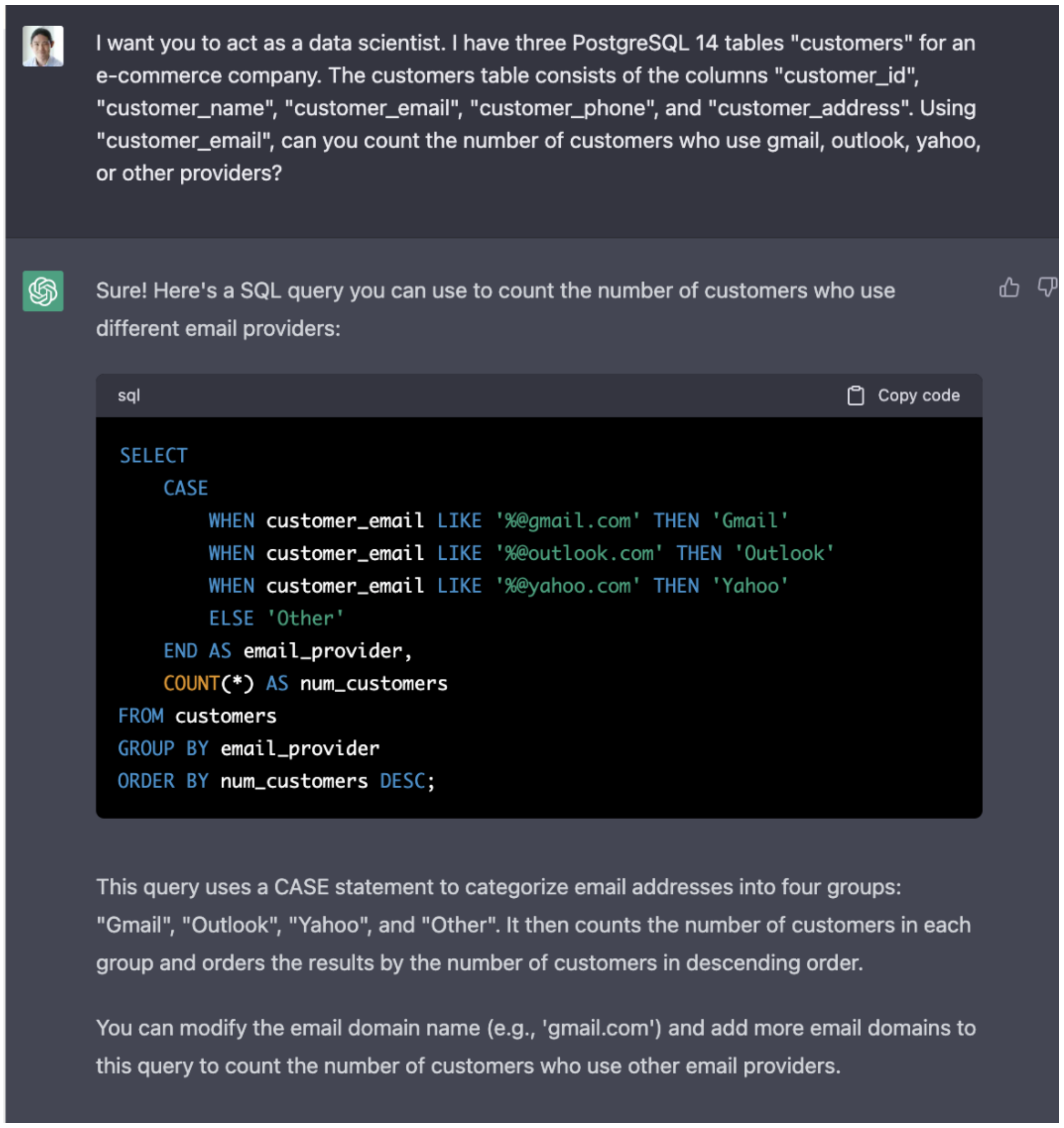
```sql
WITH email_counts AS (
  SELECT EmailAddr
  FROM Customer
  WHERE EMailAddr ILIKE '%GMAIL%' OR EMailAddr ILIKE '%YAHOO%'
  GROUP BY EmailAddr
  HAVING COUNT(*) > 1
)
SELECT DISTINCT CustId, Name, EMailAddr
FROM Customer
WHERE EMailAddr IN (SELECT EmailAddr FROM email_counts);
```

Write SQL queries from natural language

PROMPT STRUCTURE

I want you to act as a data scientist. {Insert description of tables}. Can you {count/sum/take average} of {value} which are {insert filters}

Example: Data aggregation in SQL



I want you to act as a data scientist. I have three PostgreSQL 14 tables "customers" for an e-commerce company. The customers table consists of the columns "customer_id", "customer_name", "customer_email", "customer_phone", and "customer_address". Using "customer_email", can you count the number of customers who use gmail, outlook, yahoo, or other providers?

Sure! Here's a SQL query you can use to count the number of customers who use different email providers:

```sql
SELECT
    CASE
        WHEN customer_email LIKE '%@gmail.com' THEN 'Gmail'
        WHEN customer_email LIKE '%@outlook.com' THEN 'Outlook'
        WHEN customer_email LIKE '%@yahoo.com' THEN 'Yahoo'
        ELSE 'Other'
    END AS email_provider,
    COUNT(*) AS num_customers
FROM customers
GROUP BY email_provider
ORDER BY num_customers DESC;
```

This query uses a CASE statement to categorize email addresses into four groups: "Gmail", "Outlook", "Yahoo", and "Other". It then counts the number of customers in each group and orders the results by the number of customers in descending order.

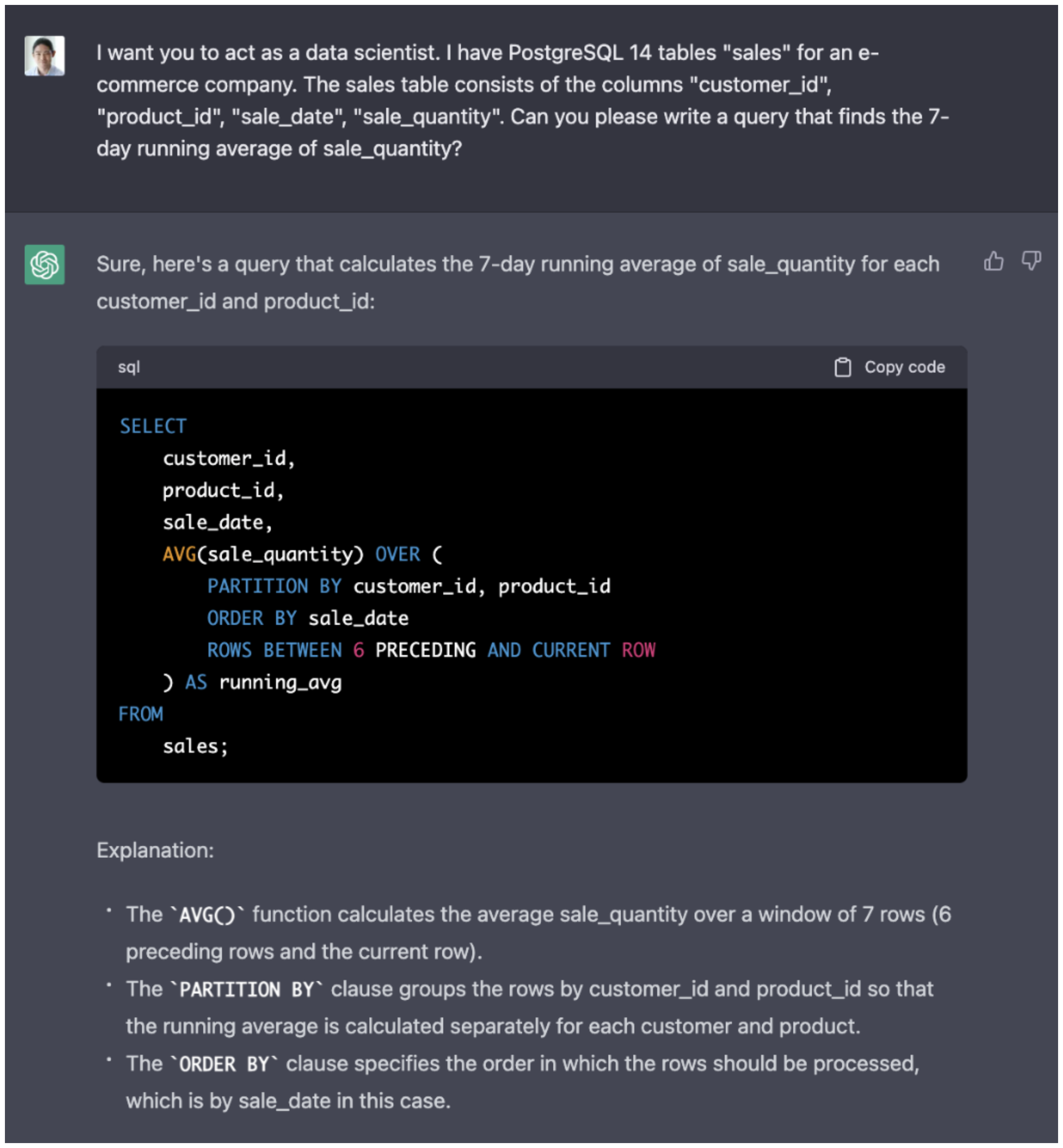
You can modify the email domain name (e.g., 'gmail.com') and add more email domains to this query to count the number of customers who use other email providers.

Write SQL queries from natural language

PROMPT STRUCTURE

I want you to act as a data scientist. I am running {PostgreSQL 14/MySQL 8/SQLite 3.4/other versions.}. I have the tables {table_name} which are {table description}. The sales table consists of the columns {column names}. Can you please write a query that finds the 7-day running average of {quantity}?

Example: 7 day running average in SQL



I want you to act as a data scientist. I have PostgreSQL 14 tables "sales" for an e-commerce company. The sales table consists of the columns "customer_id", "product_id", "sale_date", "sale_quantity". Can you please write a query that finds the 7-day running average of sale_quantity?

Sure, here's a query that calculates the 7-day running average of sale_quantity for each customer_id and product_id:

```sql
SELECT
    customer_id,
    product_id,
    sale_date,
    AVG(sale_quantity) OVER (
        PARTITION BY customer_id, product_id
        ORDER BY sale_date
        ROWS BETWEEN 6 PRECEDING AND CURRENT ROW
    ) AS running_avg
FROM
    sales;
```

Explanation:

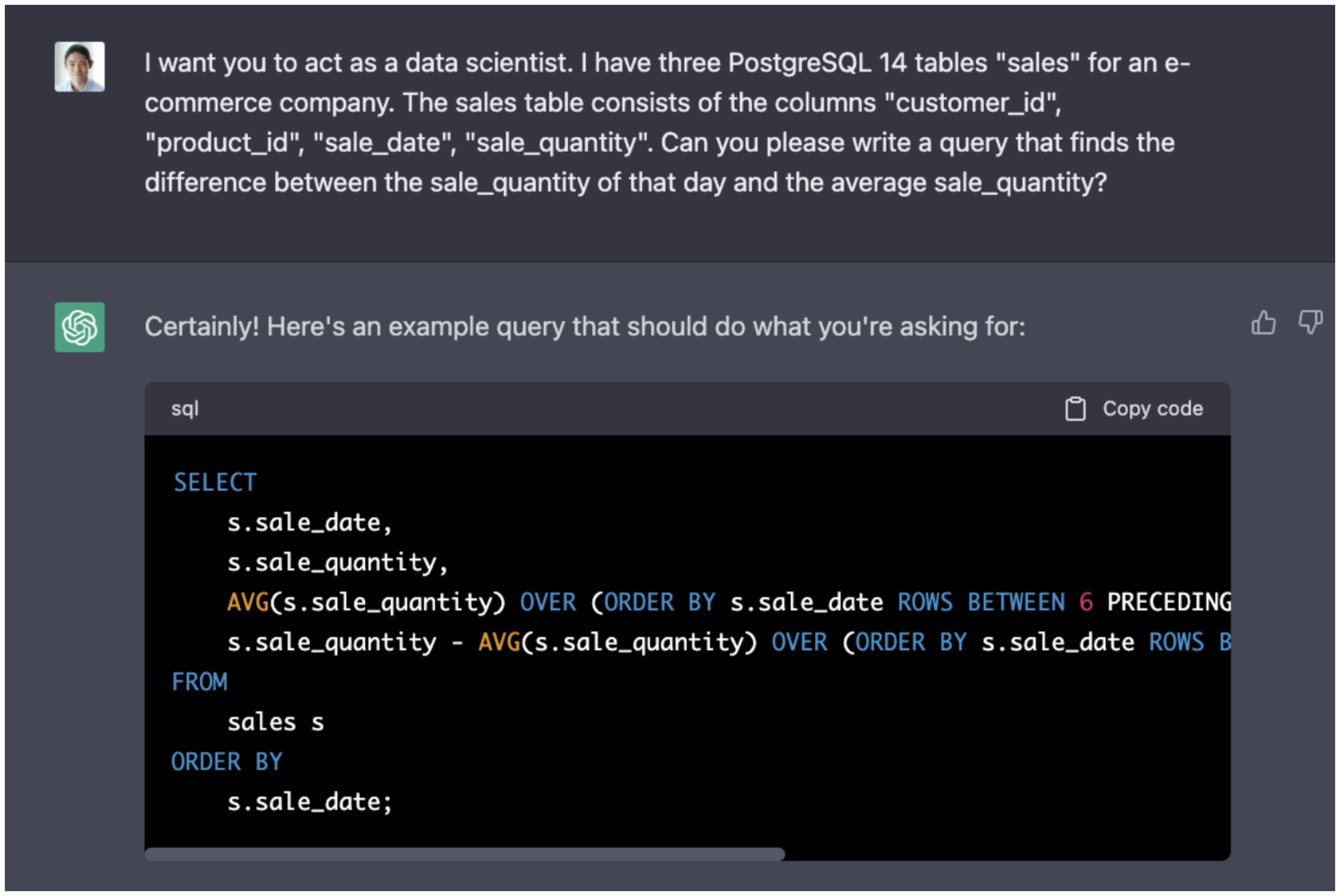
- The `AVG()` function calculates the average sale_quantity over a window of 7 rows (6 preceding rows and the current row).
- The `PARTITION BY` clause groups the rows by customer_id and product_id so that the running average is calculated separately for each customer and product.
- The `ORDER BY` clause specifies the order in which the rows should be processed, which is by sale_date in this case.

Write SQL queries from natural language

PROMPT STRUCTURE

I want you to act as a data scientist. I am running {PostgreSQL 14/MySQL 8/SQLite 3.4/other versions.}. I have the tables {table_name} which are {table description}. The sales table consists of the columns {column names}. Can you please write a query that finds {required window function}?

Example: Window functions in SQL



I want you to act as a data scientist. I have three PostgreSQL 14 tables "sales" for an e-commerce company. The sales table consists of the columns "customer_id", "product_id", "sale_date", "sale_quantity". Can you please write a query that finds the difference between the sale_quantity of that day and the average sale_quantity?

Certainly! Here's an example query that should do what you're asking for:

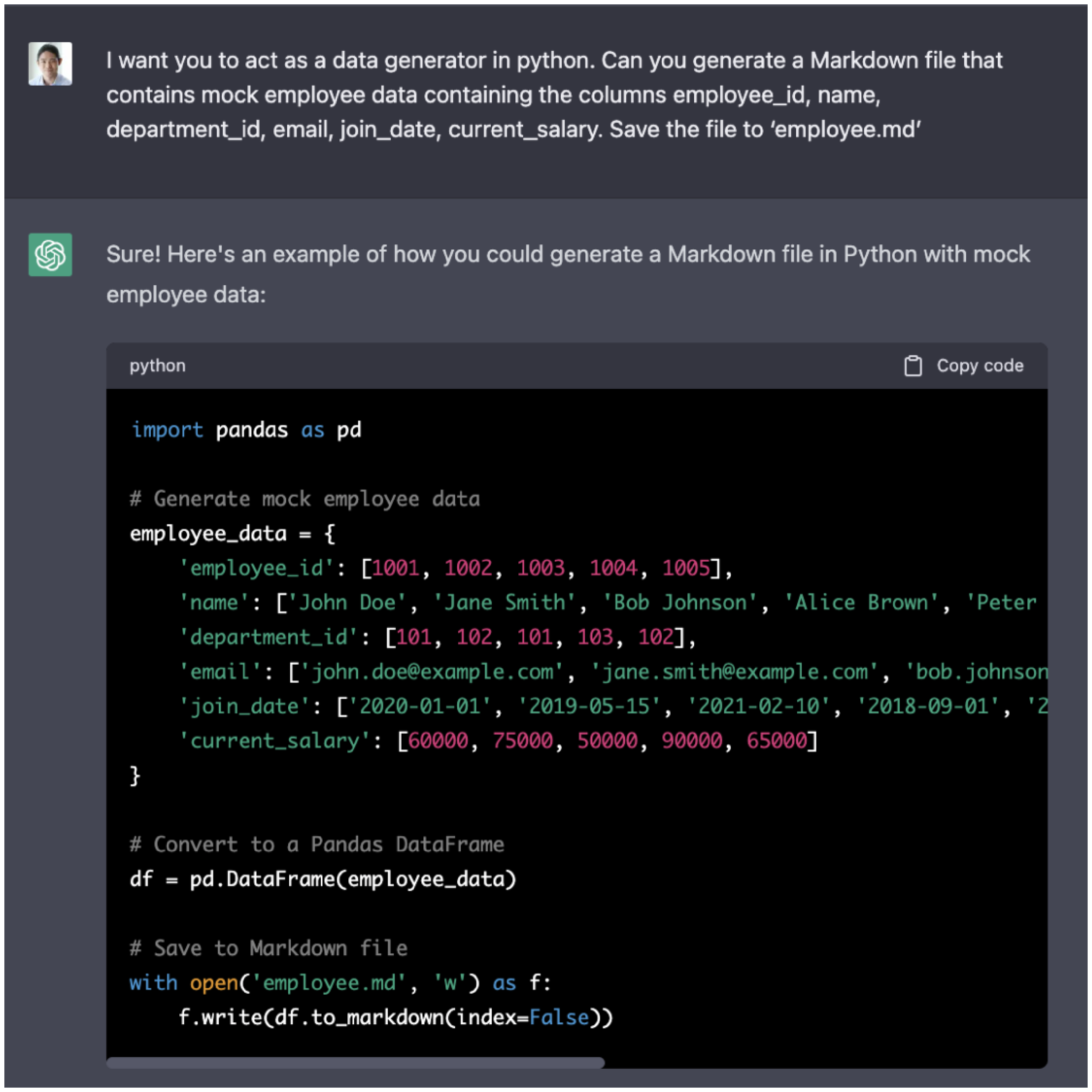
```sql
SELECT
    s.sale_date,
    s.sale_quantity,
    AVG(s.sale_quantity) OVER (ORDER BY s.sale_date ROWS BETWEEN 6 PRECEDING
    s.sale_quantity - AVG(s.sale_quantity) OVER (ORDER BY s.sale_date ROWS B
FROM
    sales s
ORDER BY
    s.sale_date;
```

Data generation workflow

PROMPT STRUCTURE

I want you to act as a data generator in Python. Can you generate a Markdown file that contains {data requirement}. Save the file to {filename}

Example: Generate Markdown



I want you to act as a data generator in python. Can you generate a Markdown file that contains mock employee data containing the columns employee_id, name, department_id, email, join_date, current_salary. Save the file to 'employee.md'

Sure! Here's an example of how you could generate a Markdown file in Python with mock employee data:

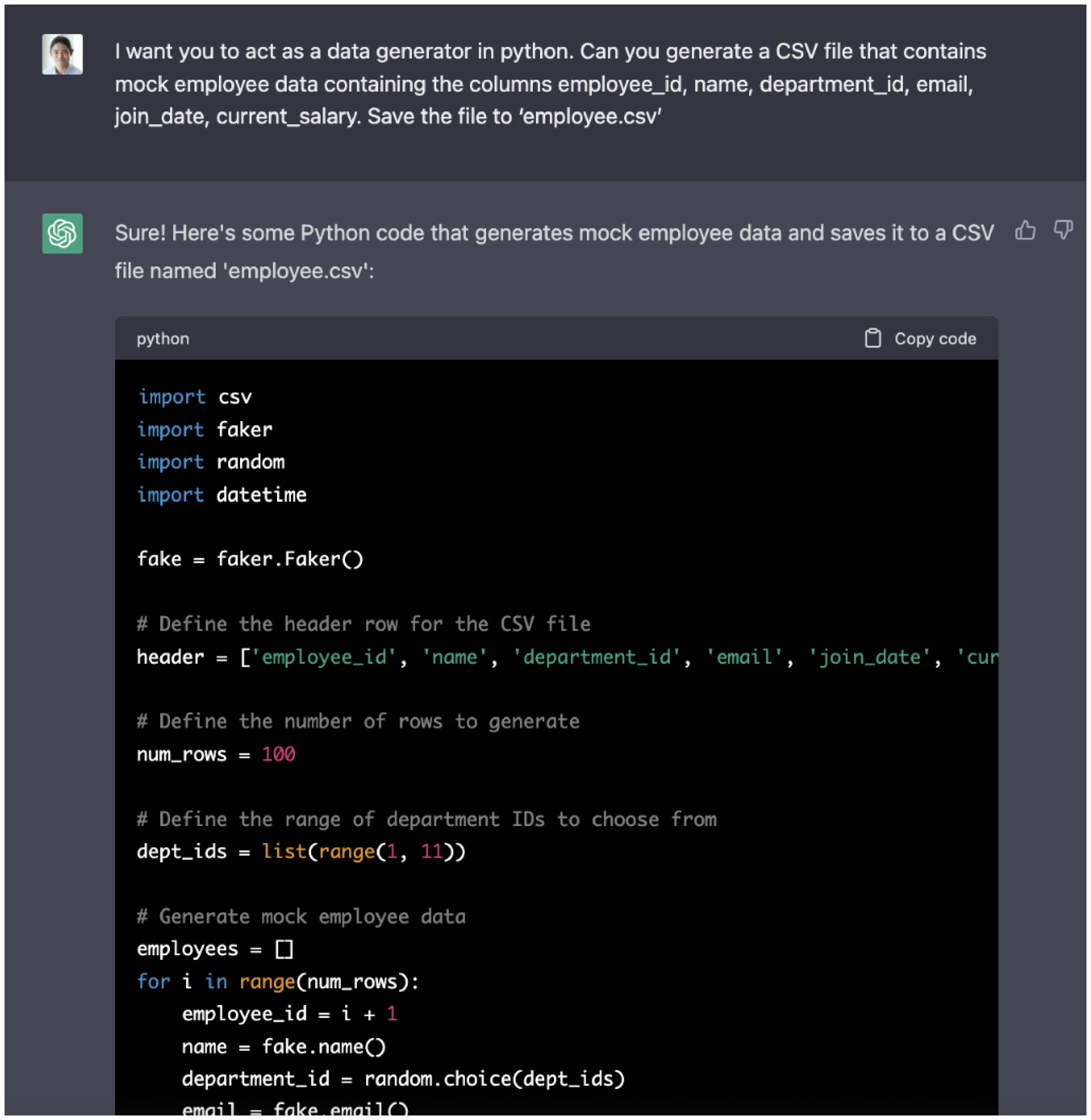
```python
import pandas as pd

# Generate mock employee data
employee_data = {
    'employee_id': [1001, 1002, 1003, 1004, 1005],
    'name': ['John Doe', 'Jane Smith', 'Bob Johnson', 'Alice Brown', 'Peter
    'department_id': [101, 102, 101, 103, 102],
    'email': ['john.doe@example.com', 'jane.smith@example.com', 'bob.johnson
    'join_date': ['2020-01-01', '2019-05-15', '2021-02-10', '2018-09-01', '2
    'current_salary': [60000, 75000, 50000, 90000, 65000]
}

# Convert to a Pandas DataFrame
df = pd.DataFrame(employee_data)

# Save to Markdown file
with open('employee.md', 'w') as f:
    f.write(df.to_markdown(index=False))
```

Data generation workflow

PROMPT STRUCTURE

I want you to act as a data generator in Python. Can you generate a CSV file that contains {data requirement}. Save the file to {filename}

Example: Generate CSV



I want you to act as a data generator in python. Can you generate a CSV file that contains mock employee data containing the columns employee_id, name, department_id, email, join_date, current_salary. Save the file to 'employee.csv'

Sure! Here's some Python code that generates mock employee data and saves it to a CSV file named 'employee.csv':

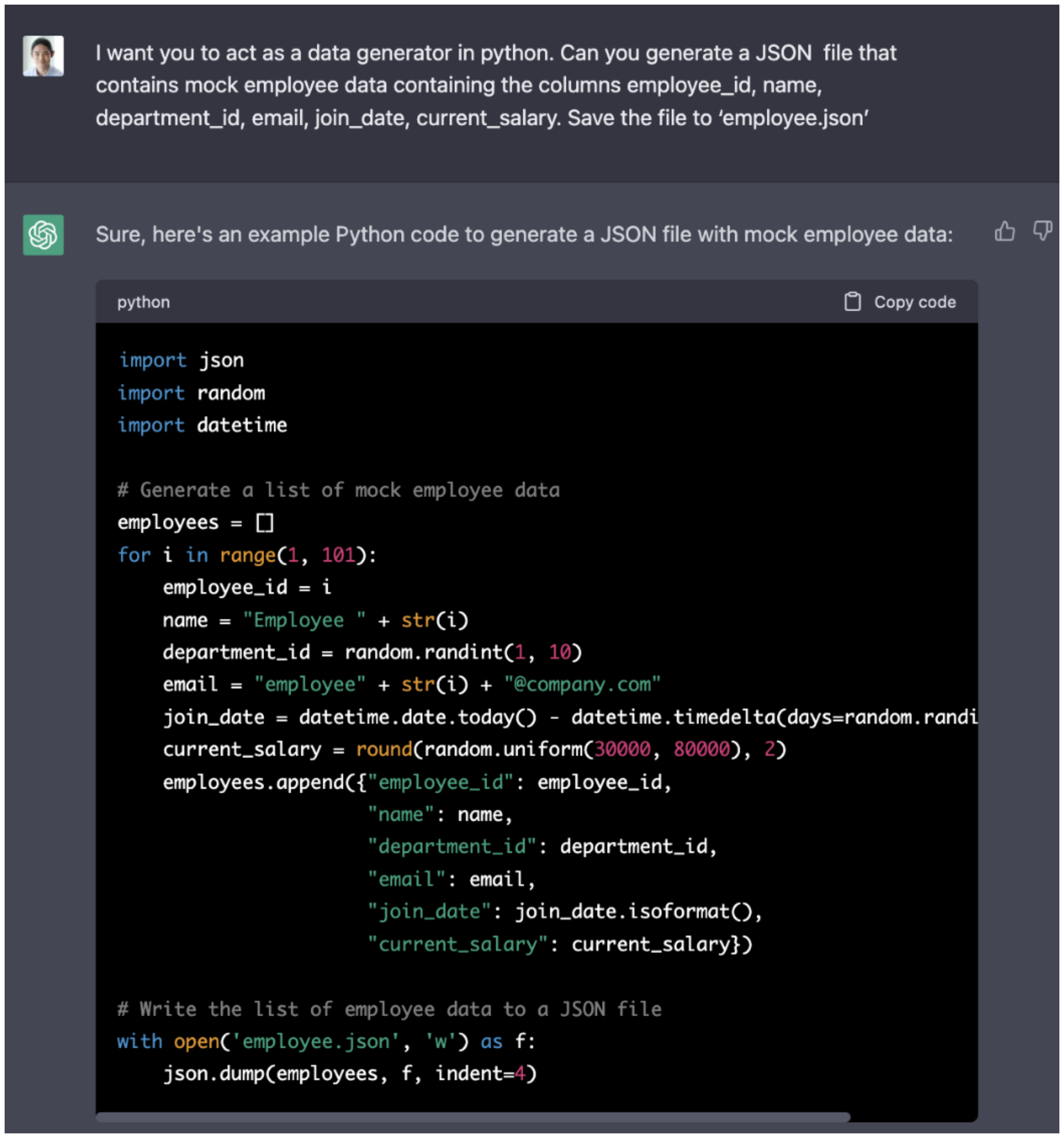
```python
import csv
import faker
import random
import datetime

fake = faker.Faker()

# Define the header row for the CSV file
header = ['employee_id', 'name', 'department_id', 'email', 'join_date', 'cur

# Define the number of rows to generate
num_rows = 100

# Define the range of department IDs to choose from
dept_ids = list(range(1, 11))

# Generate mock employee data
employees = []
for i in range(num_rows):
    employee_id = i + 1
    name = fake.name()
    department_id = random.choice(dept_ids)
    email = fake.email()
```

Data generation workflow

PROMPT STRUCTURE

I want you to act as a data generator in Python. Can you generate a JSON file that contains {data requirement}. Save the file to {filename}

Example: Generate JSON



I want you to act as a data generator in python. Can you generate a JSON file that contains mock employee data containing the columns employee_id, name, department_id, email, join_date, current_salary. Save the file to 'employee.json'

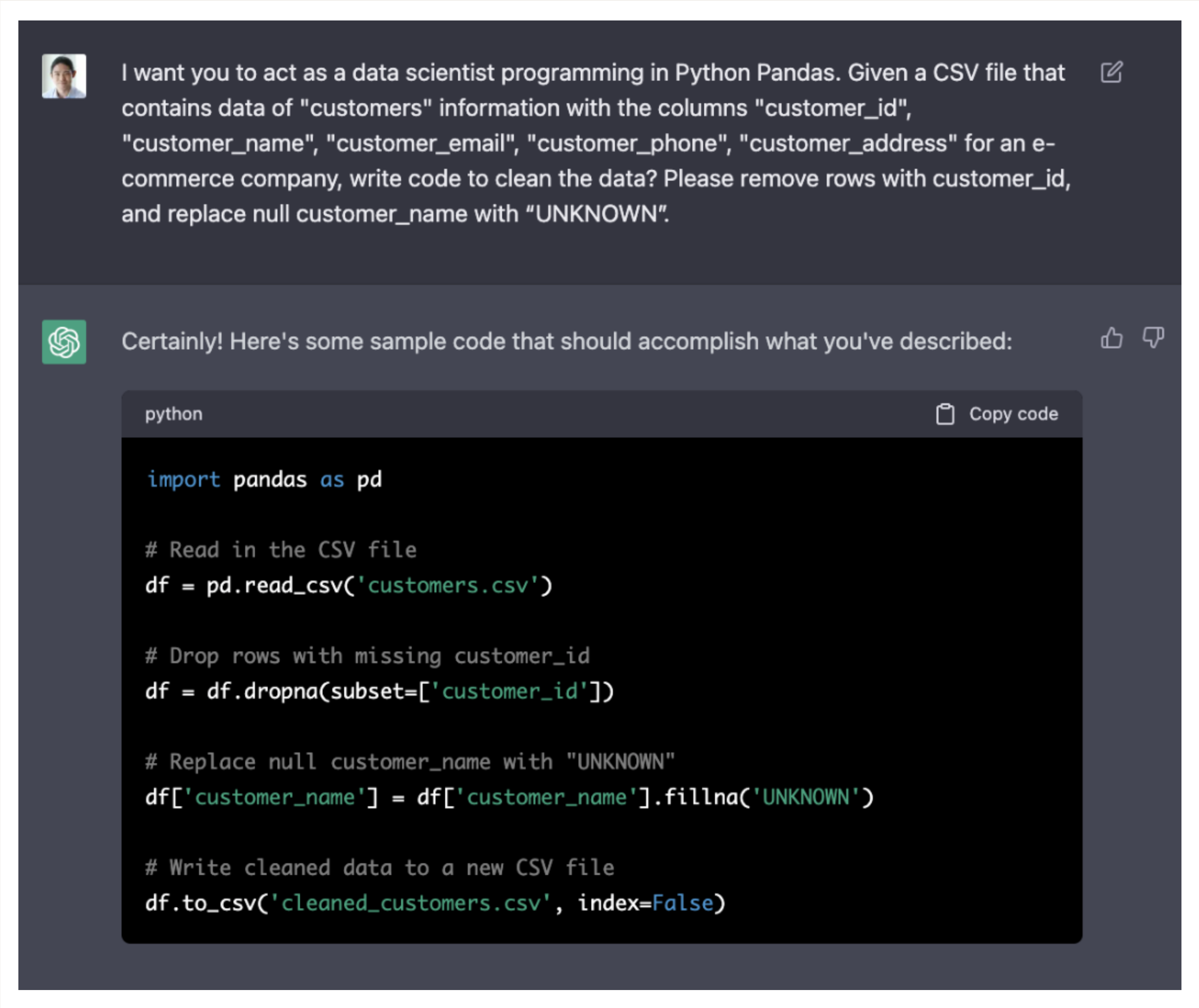
Sure, here's an example Python code to generate a JSON file with mock employee data:

```python
import json
import random
import datetime

# Generate a list of mock employee data
employees = []
for i in range(1, 101):
    employee_id = i
    name = "Employee " + str(i)
    department_id = random.randint(1, 10)
    email = "employee" + str(i) + "@company.com"
    join_date = datetime.date.today() - datetime.timedelta(days=random.randi
    current_salary = round(random.uniform(30000, 80000), 2)
    employees.append({"employee_id": employee_id,
                      "name": name,
                      "department_id": department_id,
                      "email": email,
                      "join_date": join_date.isoformat(),
                      "current_salary": current_salary})

# Write the list of employee data to a JSON file
with open('employee.json', 'w') as f:
    json.dump(employees, f, indent=4)
```

Data cleaning workflow

PROMPT STRUCTURE

I want you to act as a data scientist programming in Python Pandas. Given a CSV file that contains data of {dataframe name} with the columns {colum names} for {dataset context}, write code to clean the data? {Insert requirements for data}



I want you to act as a data scientist programming in Python Pandas. Given a CSV file that contains data of "customers" information with the columns "customer_id", "customer_name", "customer_email", "customer_phone", "customer_address" for an e-commerce company, write code to clean the data? Please remove rows with customer_id, and replace null customer_name with “UNKNOWN”.

Certainly! Here's some sample code that should accomplish what you've described:

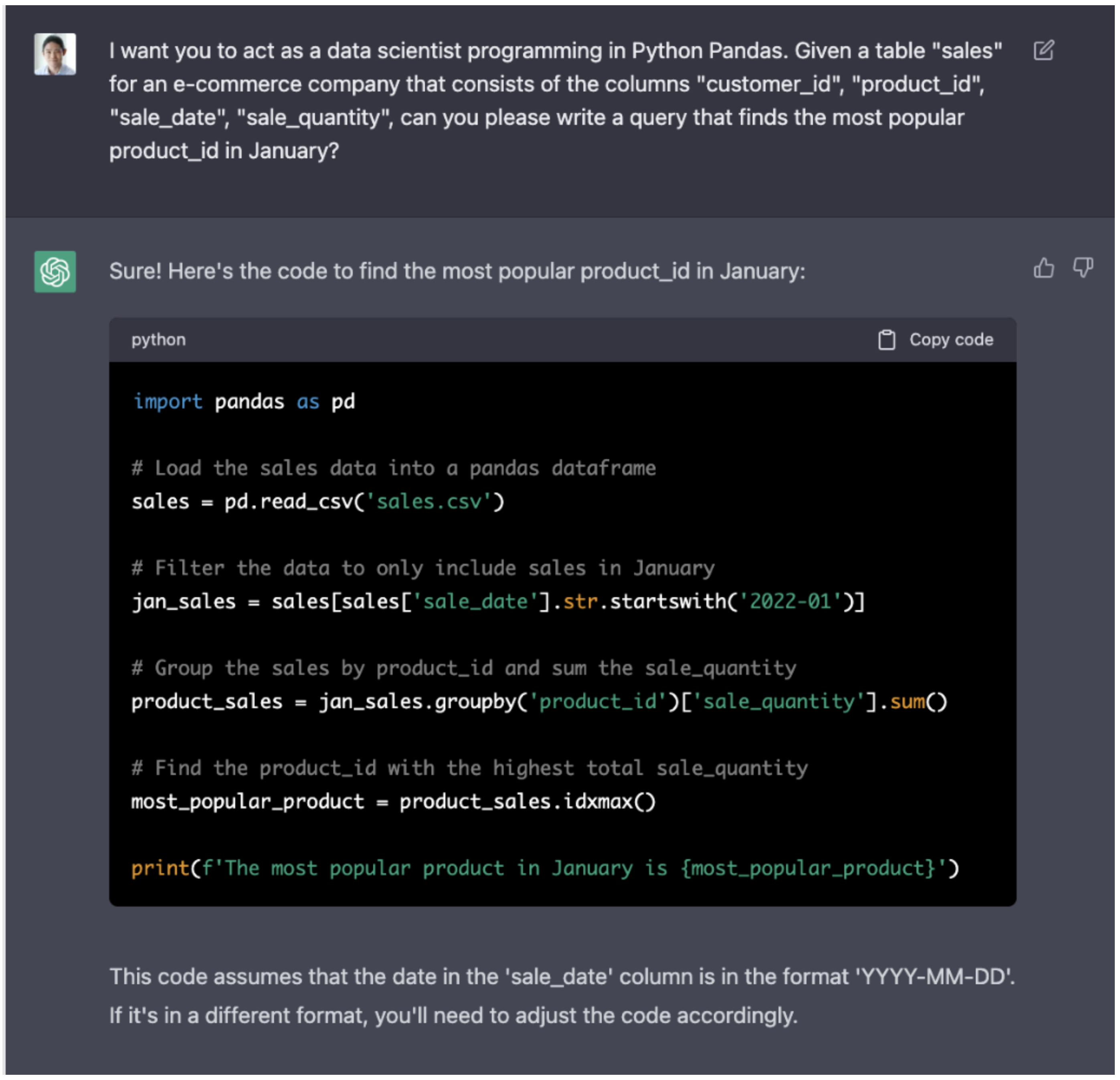
```python
import pandas as pd

# Read in the CSV file
df = pd.read_csv('customers.csv')

# Drop rows with missing customer_id
df = df.dropna(subset=['customer_id'])

# Replace null customer_name with "UNKNOWN"
df['customer_name'] = df['customer_name'].fillna('UNKNOWN')

# Write cleaned data to a new CSV file
df.to_csv('cleaned_customers.csv', index=False)
```

Data analysis workflow in pandas

PROMPT STRUCTURE

I want you to act as a data scientist programming in Python Pandas. Given a table {table name} that consists of the columns {column names} can you please write a query that finds {requirement}?

Example: Data Aggregation



I want you to act as a data scientist programming in Python Pandas. Given a table "sales" for an e-commerce company that consists of the columns "customer_id", "product_id", "sale_date", "sale_quantity", can you please write a query that finds the most popular product_id in January?

Sure! Here's the code to find the most popular product_id in January:

```python
import pandas as pd

# Load the sales data into a pandas dataframe
sales = pd.read_csv('sales.csv')

# Filter the data to only include sales in January
jan_sales = sales[sales['sale_date'].str.startswith('2022-01')]

# Group the sales by product_id and sum the sale_quantity
product_sales = jan_sales.groupby('product_id')['sale_quantity'].sum()

# Find the product_id with the highest total sale_quantity
most_popular_product = product_sales.idxmax()

print(f'The most popular product in January is {most_popular_product}')
```

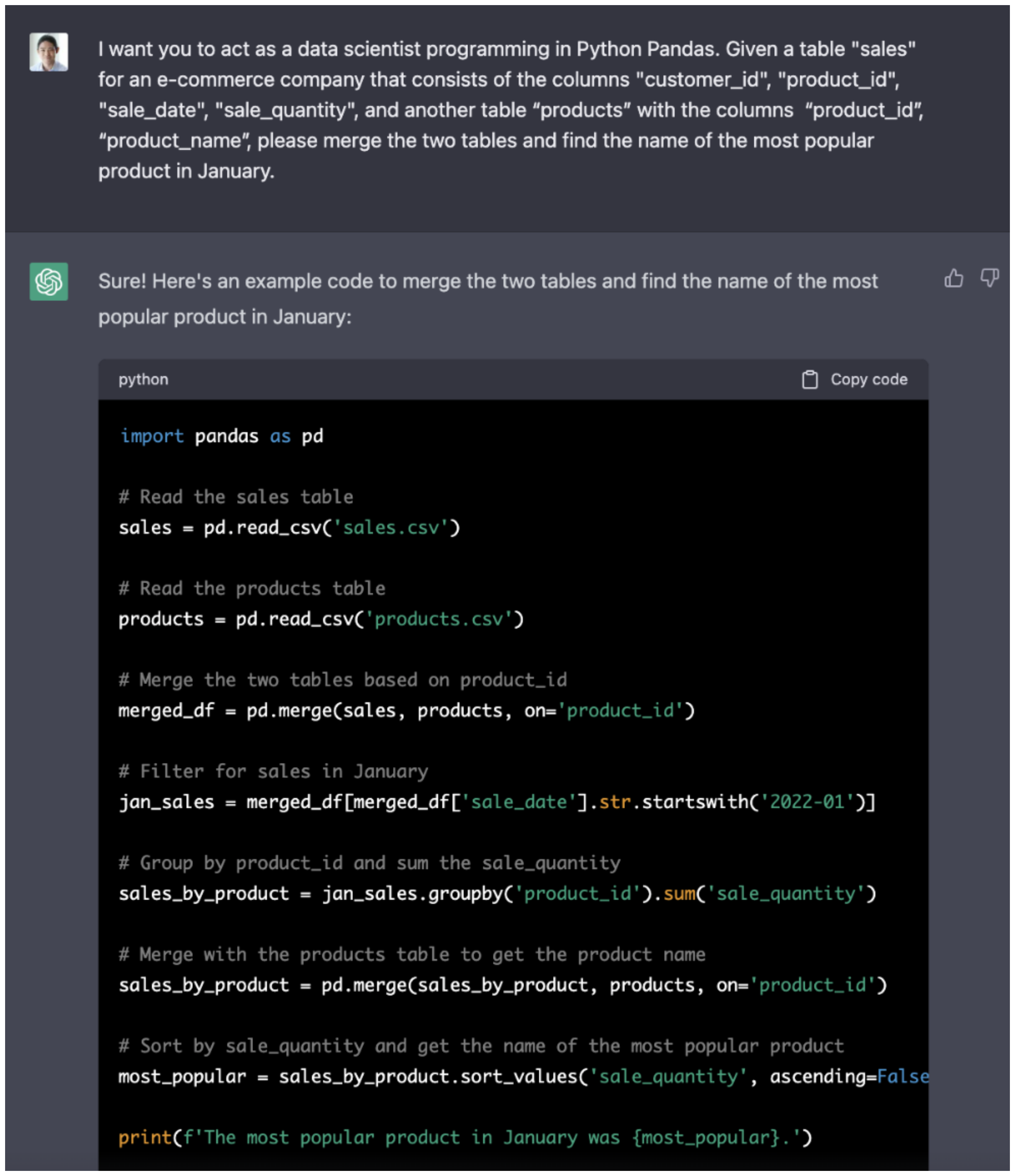
This code assumes that the date in the 'sale_date' column is in the format 'YYYY-MM-DD'. If it's in a different format, you'll need to adjust the code accordingly.

Data analysis workflow in pandas

PROMPT STRUCTURE

I want you to act as a data scientist programming in Python Pandas. Given a table {table 1 name} that consists of the columns {column names} and another table {table 2 name} with the columns {column names} , please merge the two tables. {Insert additional requirement, if any}

Example: Data Merging



I want you to act as a data scientist programming in Python Pandas. Given a table "sales" for an e-commerce company that consists of the columns "customer_id", "product_id", "sale_date", "sale_quantity", and another table “products” with the columns “product_id”, “product_name”, please merge the two tables and find the name of the most popular product in January.

Sure! Here's an example code to merge the two tables and find the name of the most popular product in January:

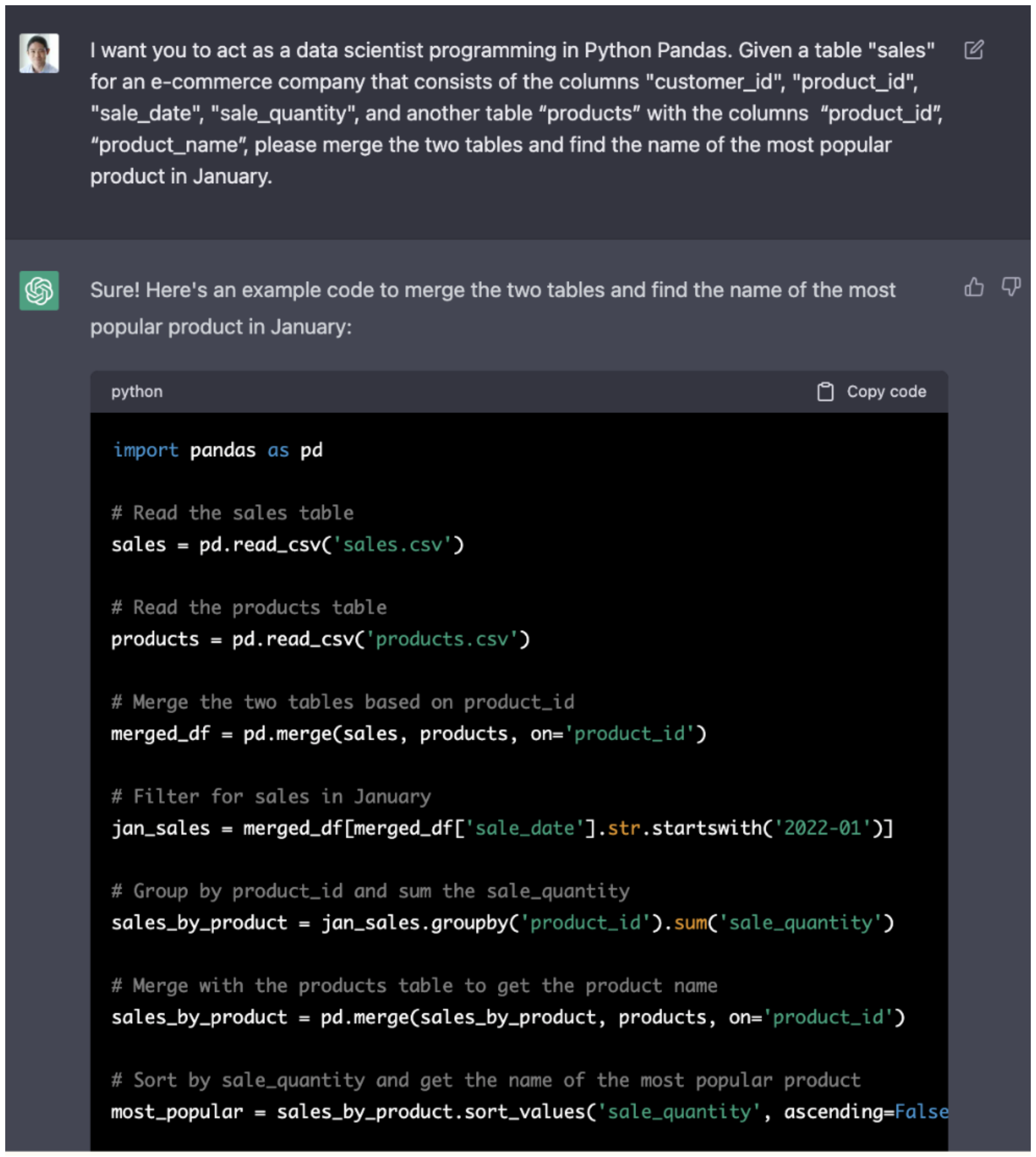
```python
import pandas as pd

# Read the sales table
sales = pd.read_csv('sales.csv')

# Read the products table
products = pd.read_csv('products.csv')

# Merge the two tables based on product_id
merged_df = pd.merge(sales, products, on='product_id')

# Filter for sales in January
jan_sales = merged_df[merged_df['sale_date'].str.startswith('2022-01')]

# Group by product_id and sum the sale_quantity
sales_by_product = jan_sales.groupby('product_id').sum('sale_quantity')

# Merge with the products table to get the product name
sales_by_product = pd.merge(sales_by_product, products, on='product_id')

# Sort by sale_quantity and get the name of the most popular product
most_popular = sales_by_product.sort_values('sale_quantity', ascending=False

print(f'The most popular product in January was {most_popular}.')
```

Data analysis workflow in pandas

PROMPT STRUCTURE

I want you to act as a data scientist programming in Python Pandas. Given a table {table name} that consists of the columns {column names} can you aggregate the {value} by {column} and convert it from long to wide format?

Example: Data Reshaping



I want you to act as a data scientist programming in Python Pandas. Given a table "sales" for an e-commerce company that consists of the columns "customer_id", "product_id", "sale_date", "sale_quantity", and another table "products" with the columns "product_id", "product_name", please merge the two tables and find the name of the most popular product in January.

Sure! Here's an example code to merge the two tables and find the name of the most popular product in January:

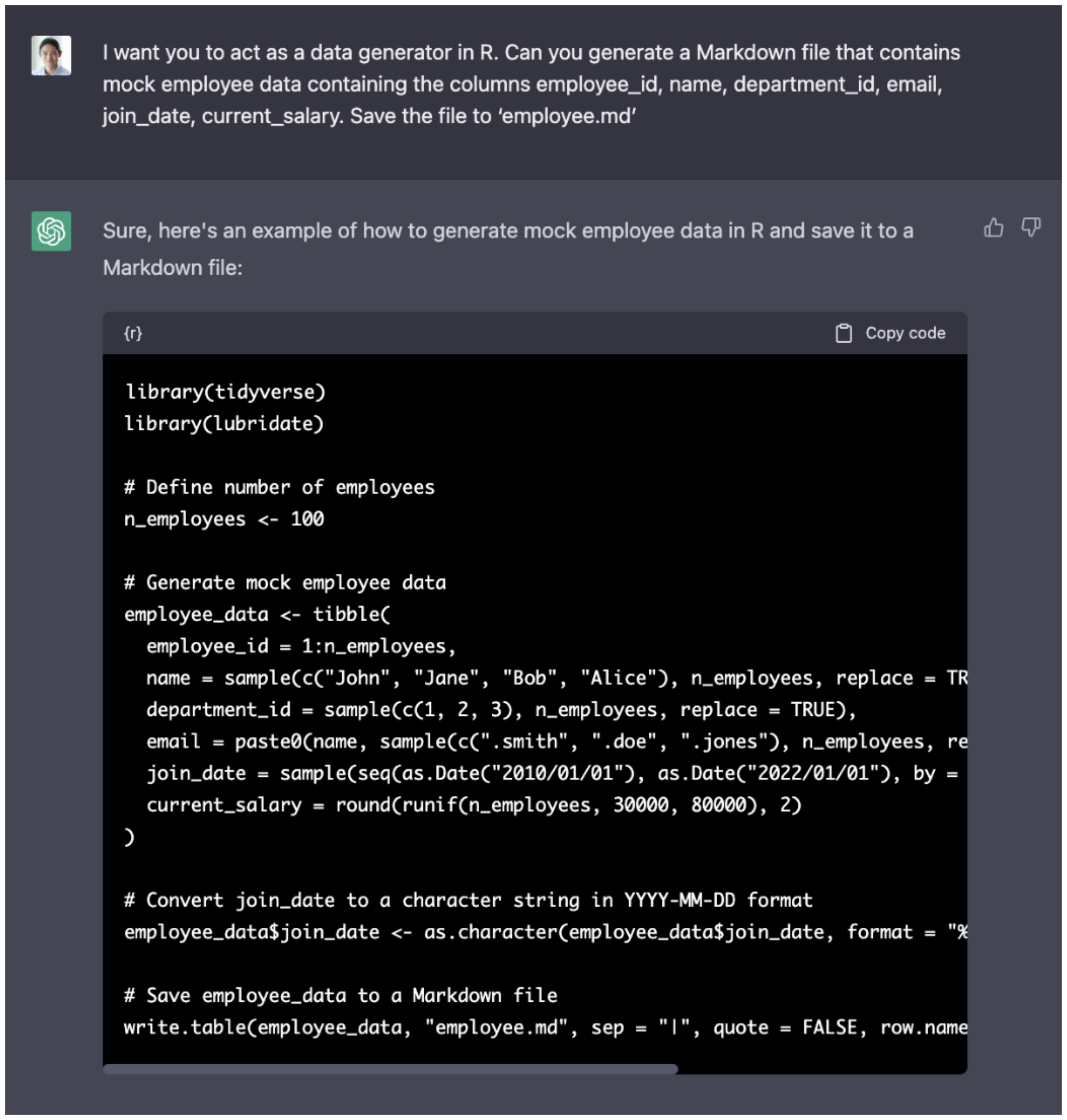
```python
import pandas as pd

# Read the sales table
sales = pd.read_csv('sales.csv')

# Read the products table
products = pd.read_csv('products.csv')

# Merge the two tables based on product_id
merged_df = pd.merge(sales, products, on='product_id')

# Filter for sales in January
jan_sales = merged_df[merged_df['sale_date'].str.startswith('2022-01')]

# Group by product_id and sum the sale_quantity
sales_by_product = jan_sales.groupby('product_id').sum('sale_quantity')

# Merge with the products table to get the product name
sales_by_product = pd.merge(sales_by_product, products, on='product_id')

# Sort by sale_quantity and get the name of the most popular product
most_popular = sales_by_product.sort_values('sale_quantity', ascending=False
```

Data generation workflow

PROMPT STRUCTURE

I want you to act as a data generator in R. Can you generate a Markdown file that contains {data requirement}. Save the file to {filename}

Example: Generate Markdown



I want you to act as a data generator in R. Can you generate a Markdown file that contains mock employee data containing the columns employee_id, name, department_id, email, join_date, current_salary. Save the file to ‘employee.md’

Sure, here's an example of how to generate mock employee data in R and save it to a Markdown file:

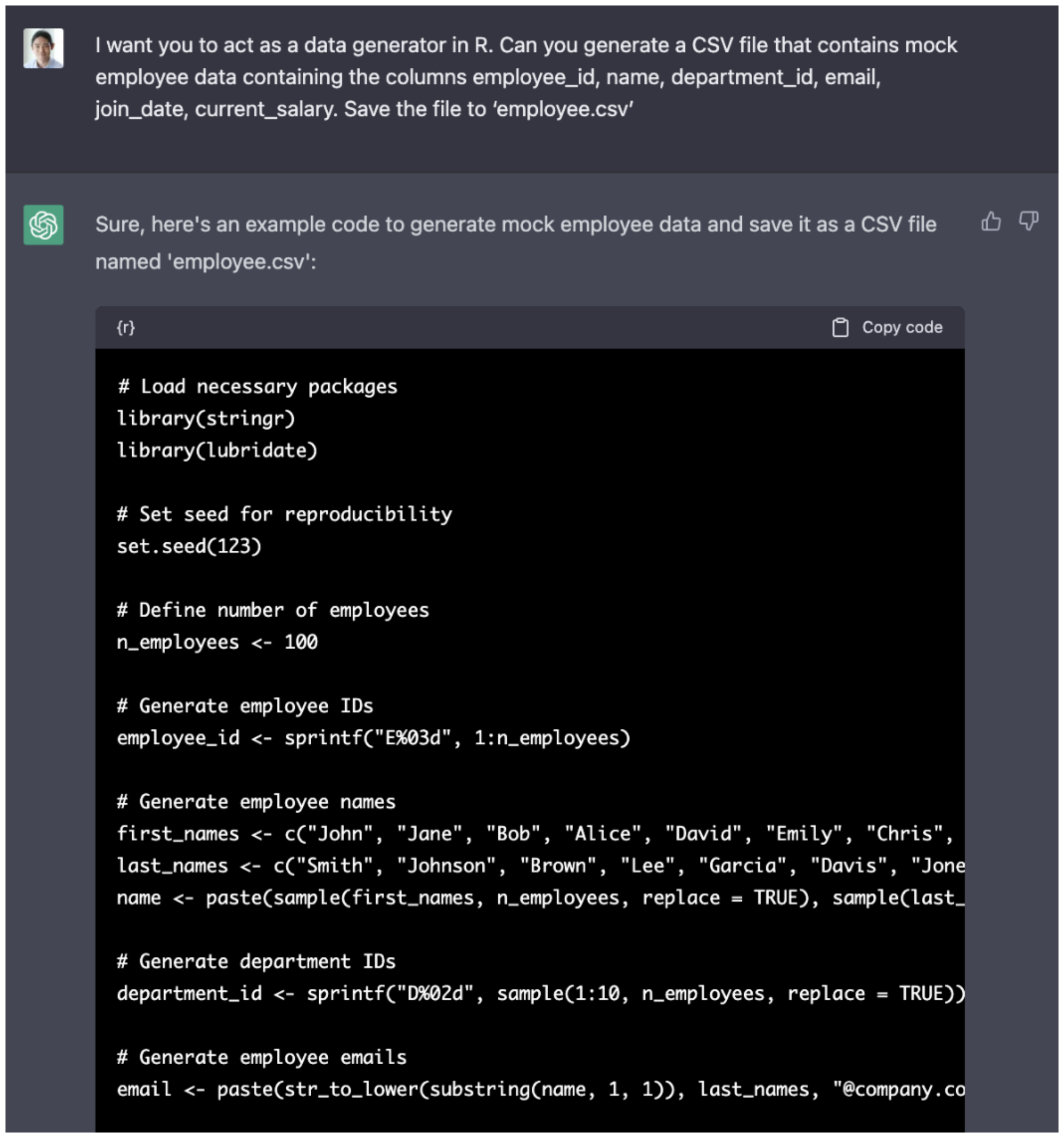
```
library(tidyverse)
library(lubridate)

# Define number of employees
n_employees <- 100

# Generate mock employee data
employee_data <- tibble(
  employee_id = 1:n_employees,
  name = sample(c("John", "Jane", "Bob", "Alice"), n_employees, replace = TR
  department_id = sample(c(1, 2, 3), n_employees, replace = TRUE),
  email = paste0(name, sample(c(".smith", ".doe", ".jones"), n_employees, re
  join_date = sample(seq(as.Date("2010/01/01"), as.Date("2022/01/01"), by =
  current_salary = round(runif(n_employees, 30000, 80000), 2)
)

# Convert join_date to a character string in YYYY-MM-DD format
employee_data$join_date <- as.character(employee_data$join_date, format = "%

# Save employee_data to a Markdown file
write.table(employee_data, "employee.md", sep = "|", quote = FALSE, row.name
```

Data generation workflow

PROMPT STRUCTURE

I want you to act as a data generator in R. Can you generate a CSV file that contains {data requirement}. Save the file to {filename}

Example: Generate CSV



I want you to act as a data generator in R. Can you generate a CSV file that contains mock employee data containing the columns employee_id, name, department_id, email, join_date, current_salary. Save the file to 'employee.csv'

Sure, here's an example code to generate mock employee data and save it as a CSV file named 'employee.csv':

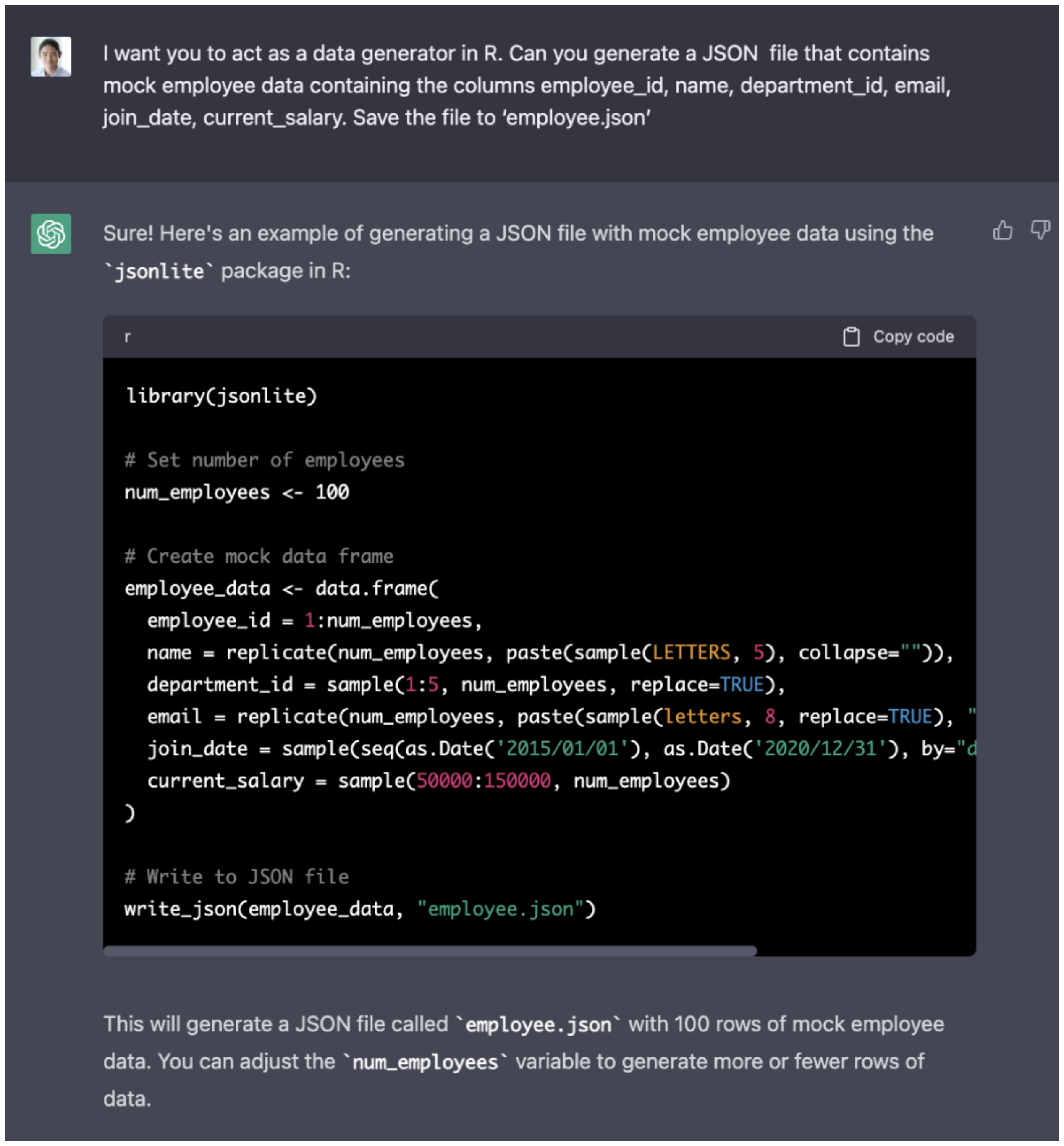
```
# Load necessary packages
library(stringr)
library(lubridate)

# Set seed for reproducibility
set.seed(123)

# Define number of employees
n_employees <- 100

# Generate employee IDs
employee_id <- sprintf("E%03d", 1:n_employees)

# Generate employee names
first_names <- c("John", "Jane", "Bob", "Alice", "David", "Emily", "Chris",
last_names <- c("Smith", "Johnson", "Brown", "Lee", "Garcia", "Davis", "Jone
name <- paste(sample(first_names, n_employees, replace = TRUE), sample(last_

# Generate department IDs
department_id <- sprintf("D%02d", sample(1:10, n_employees, replace = TRUE))

# Generate employee emails
email <- paste(str_to_lower(substring(name, 1, 1)), last_names, "@company.co
```

Data generation workflow

PROMPT STRUCTURE

I want you to act as a data generator in R. Can you generate a JSON file that contains {data requirement}. Save the file to {filename}

Example: Generate JSON



I want you to act as a data generator in R. Can you generate a JSON file that contains mock employee data containing the columns employee_id, name, department_id, email, join_date, current_salary. Save the file to ‘employee.json’

Sure! Here's an example of generating a JSON file with mock employee data using the `jsonlite` package in R:

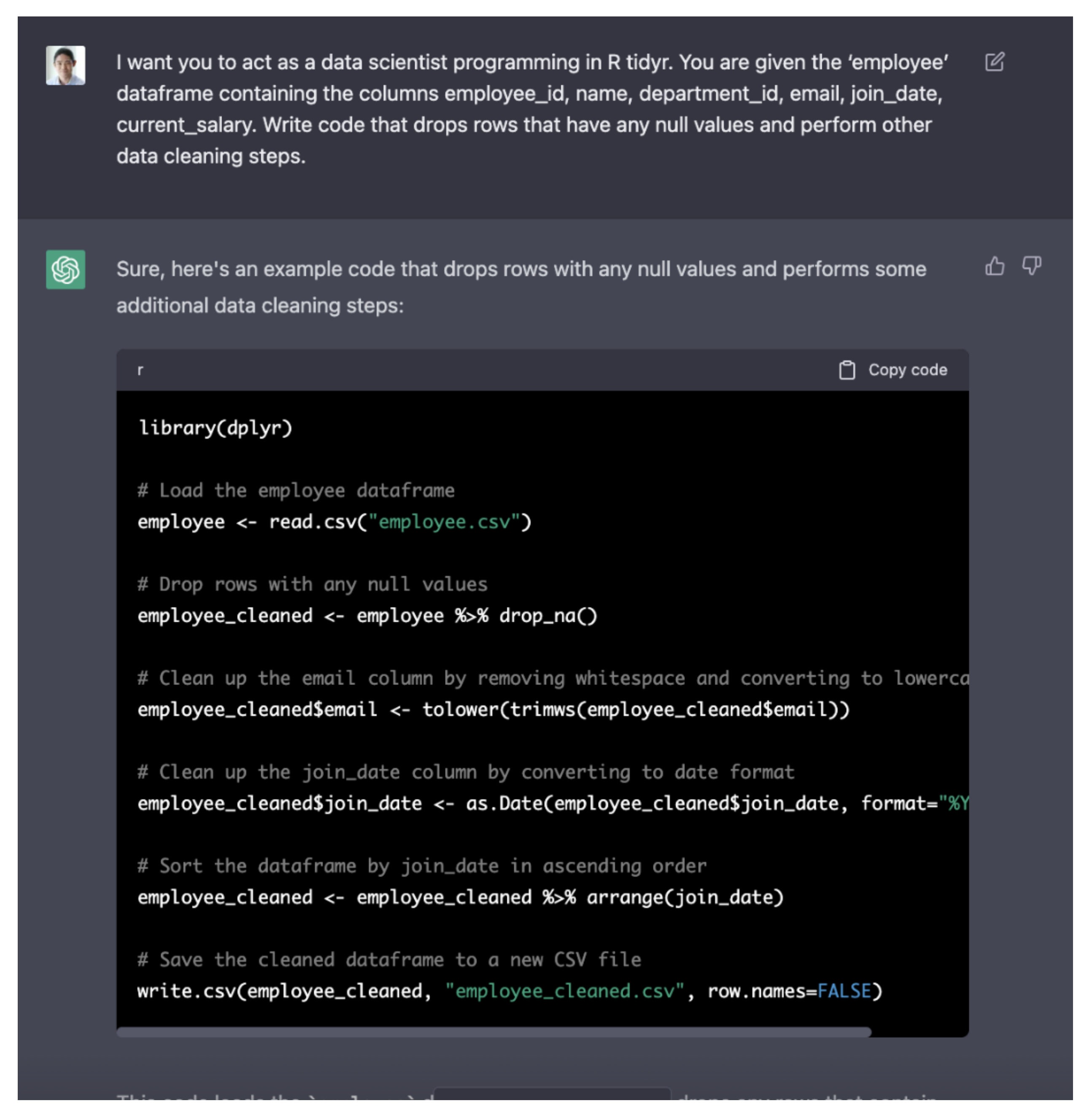
```r
library(jsonlite)

# Set number of employees
num_employees <- 100

# Create mock data frame
employee_data <- data.frame(
  employee_id = 1:num_employees,
  name = replicate(num_employees, paste(sample(LETTERS, 5), collapse="")),
  department_id = sample(1:5, num_employees, replace=TRUE),
  email = replicate(num_employees, paste(sample(letters, 8, replace=TRUE), "
  join_date = sample(seq(as.Date('2015/01/01'), as.Date('2020/12/31'), by="d
  current_salary = sample(50000:150000, num_employees)
)

# Write to JSON file
write_json(employee_data, "employee.json")
```

This will generate a JSON file called `employee.json` with 100 rows of mock employee data. You can adjust the `num_employees` variable to generate more or fewer rows of data.

Data cleaning workflow

PROMPT STRUCTURE

I want you to act as a data scientist programming in R tidyr. You are given the {dataframe name} dataframe containing the columns {column name}. {Insert requirement}



I want you to act as a data scientist programming in R tidyr. You are given the 'employee' dataframe containing the columns employee_id, name, department_id, email, join_date, current_salary. Write code that drops rows that have any null values and perform other data cleaning steps.

Sure, here's an example code that drops rows with any null values and performs some additional data cleaning steps:

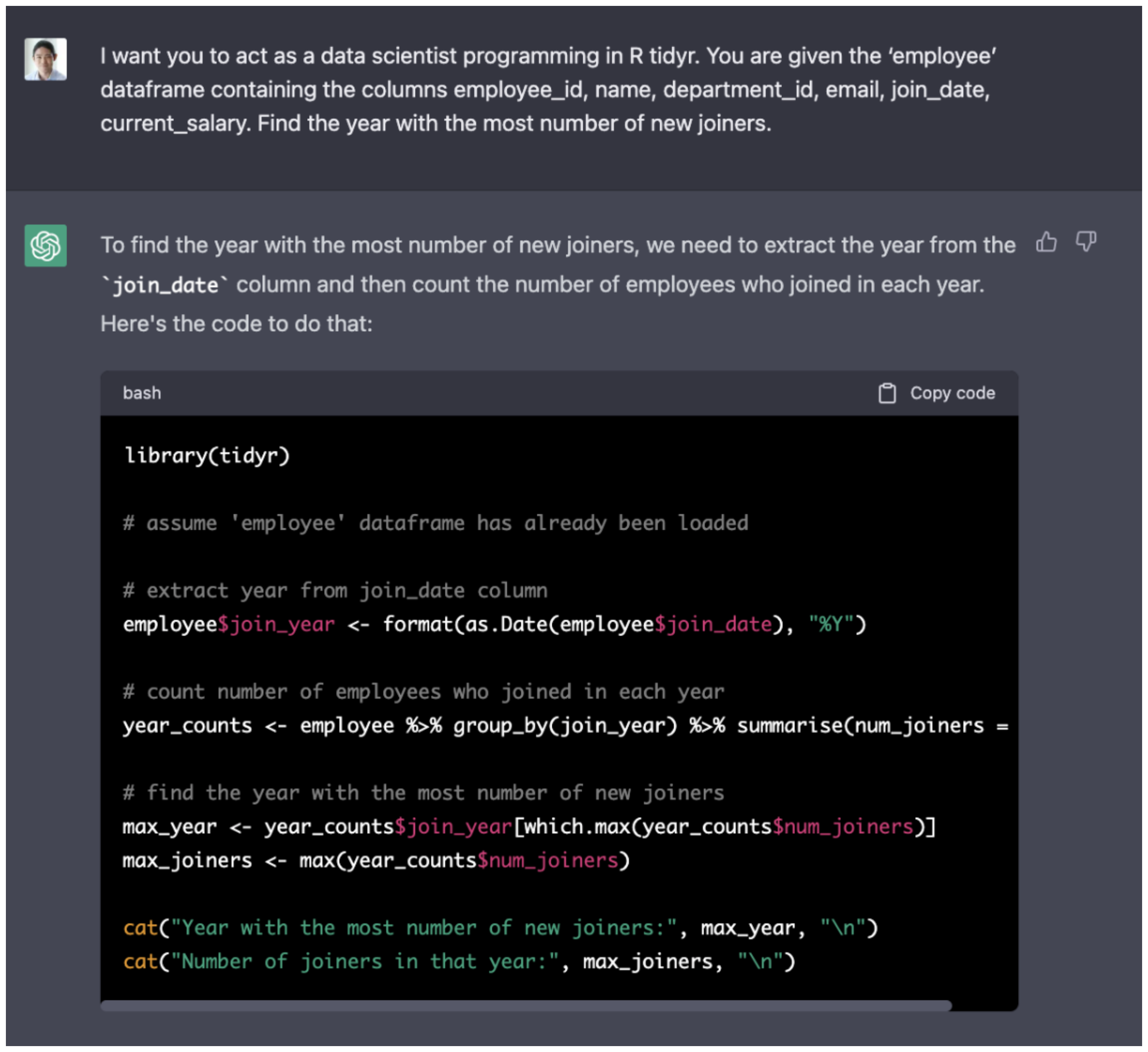
```r
library(dplyr)

# Load the employee dataframe
employee <- read.csv("employee.csv")

# Drop rows with any null values
employee_cleaned <- employee %>% drop_na()

# Clean up the email column by removing whitespace and converting to lowerca
employee_cleaned$email <- tolower(trimws(employee_cleaned$email))

# Clean up the join_date column by converting to date format
employee_cleaned$join_date <- as.Date(employee_cleaned$join_date, format="%Y

# Sort the dataframe by join_date in ascending order
employee_cleaned <- employee_cleaned %>% arrange(join_date)

# Save the cleaned dataframe to a new CSV file
write.csv(employee_cleaned, "employee_cleaned.csv", row.names=FALSE)
```

Data analysis workflow in tidyr

PROMPT STRUCTURE

I want you to act as a data scientist programming in R tidyr. You are given the {dataframe name} dataframe containing the columns {column name}. {Insert requirement}

Example: Data Aggregation



I want you to act as a data scientist programming in R tidyr. You are given the 'employee' dataframe containing the columns employee_id, name, department_id, email, join_date, current_salary. Find the year with the most number of new joiners.

To find the year with the most number of new joiners, we need to extract the year from the `join_date` column and then count the number of employees who joined in each year. Here's the code to do that:

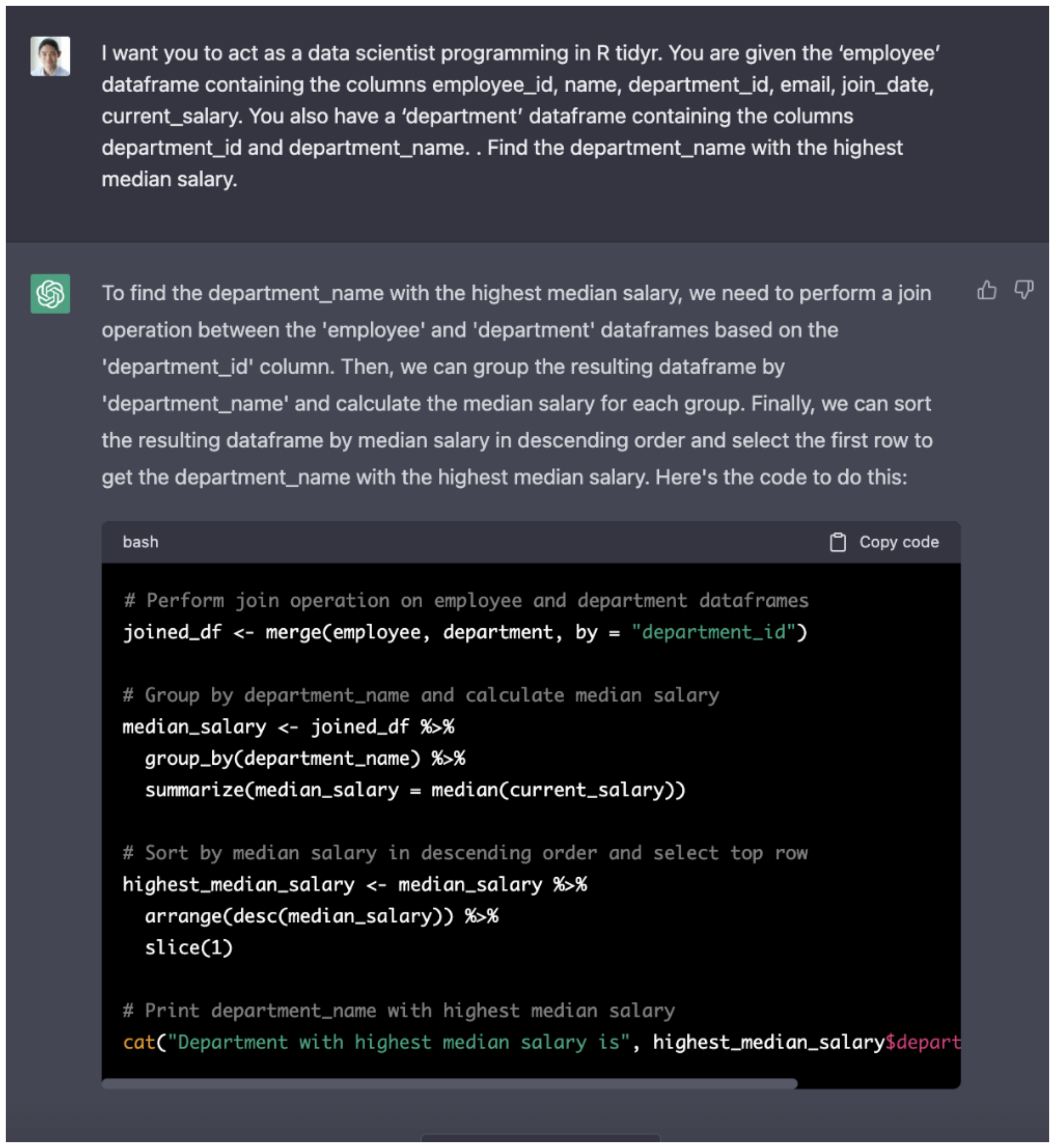
```
library(tidyr)

# assume 'employee' dataframe has already been loaded

# extract year from join_date column
employee$join_year <- format(as.Date(employee$join_date), "%Y")

# count number of employees who joined in each year
year_counts <- employee %>% group_by(join_year) %>% summarise(num_joiners =

# find the year with the most number of new joiners
max_year <- year_counts$join_year[which.max(year_counts$num_joiners)]
max_joiners <- max(year_counts$num_joiners)

cat("Year with the most number of new joiners:", max_year, "\n")
cat("Number of joiners in that year:", max_joiners, "\n")
```

Data analysis workflow in tidyr

PROMPT STRUCTURE

I want you to act as a data scientist programming in R tidyr. You are given the {dataframe 1 name} dataframe containing the columns {column name}. You also have a {dataframe 2 name} dataframe containing the columns {column name}. Find the {required output}

Example: Data Merging



I want you to act as a data scientist programming in R tidyr. You are given the 'employee' dataframe containing the columns employee_id, name, department_id, email, join_date, current_salary. You also have a 'department' dataframe containing the columns department_id and department_name. . Find the department_name with the highest median salary.

To find the department_name with the highest median salary, we need to perform a join operation between the 'employee' and 'department' dataframes based on the 'department_id' column. Then, we can group the resulting dataframe by 'department_name' and calculate the median salary for each group. Finally, we can sort the resulting dataframe by median salary in descending order and select the first row to get the department_name with the highest median salary. Here's the code to do this:

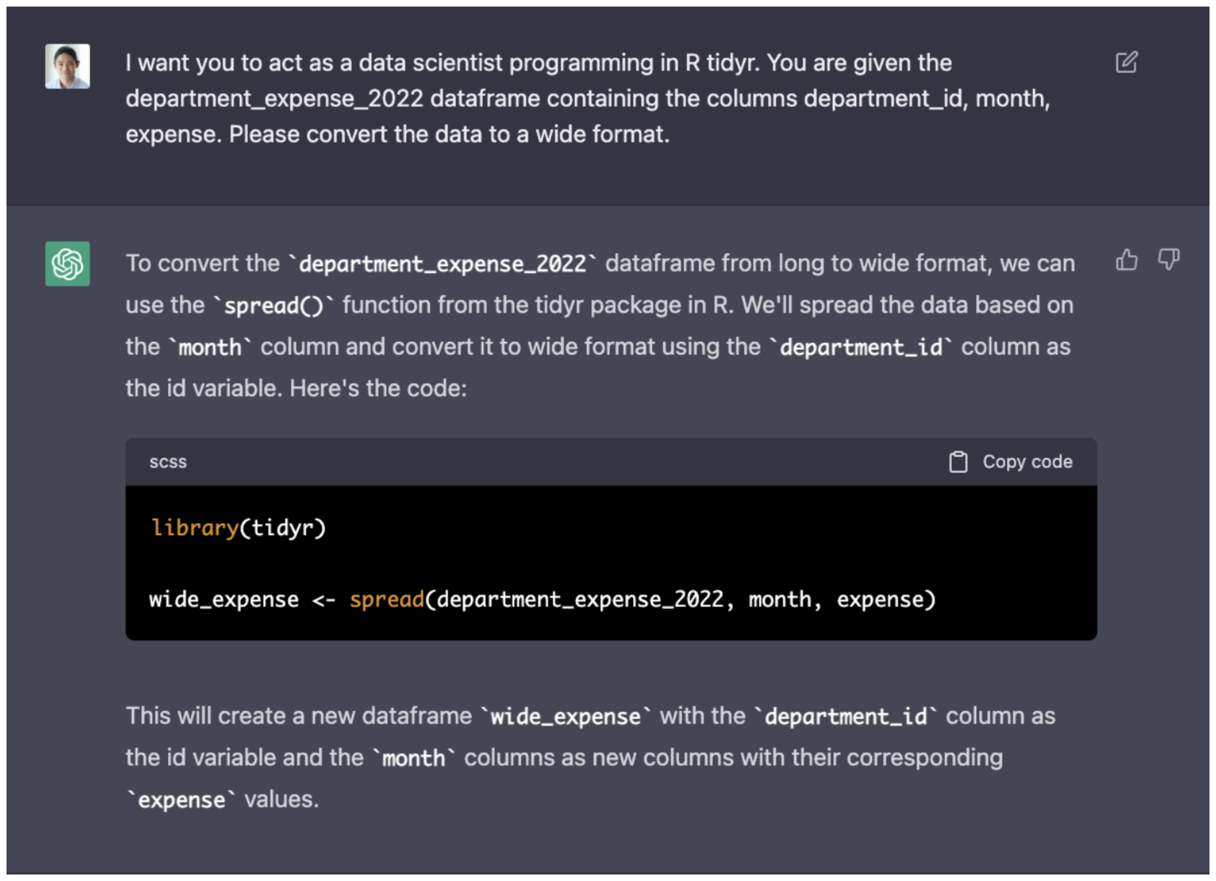
```
# Perform join operation on employee and department dataframes
joined_df <- merge(employee, department, by = "department_id")

# Group by department_name and calculate median salary
median_salary <- joined_df %>%
  group_by(department_name) %>%
  summarize(median_salary = median(current_salary))

# Sort by median salary in descending order and select top row
highest_median_salary <- median_salary %>%
  arrange(desc(median_salary)) %>%
  slice(1)

# Print department_name with highest median salary
cat("Department with highest median salary is", highest_median_salary$depart
```

Data analysis workflow in tidyr

PROMPT STRUCTURE

I want you to act as a data scientist programming in R tidyr. You are given the {dataframe name} dataframe containing the columns {column name}. Please convert the data to wide format.

Example: Data Reshaping (Long to Wide)



I want you to act as a data scientist programming in R tidyr. You are given the department_expense_2022 dataframe containing the columns department_id, month, expense. Please convert the data to a wide format.

To convert the `department_expense_2022` dataframe from long to wide format, we can use the `spread()` function from the tidyr package in R. We'll spread the data based on the `month` column and convert it to wide format using the `department_id` column as the id variable. Here's the code:

```
library(tidyr)

wide_expense <- spread(department_expense_2022, month, expense)
```

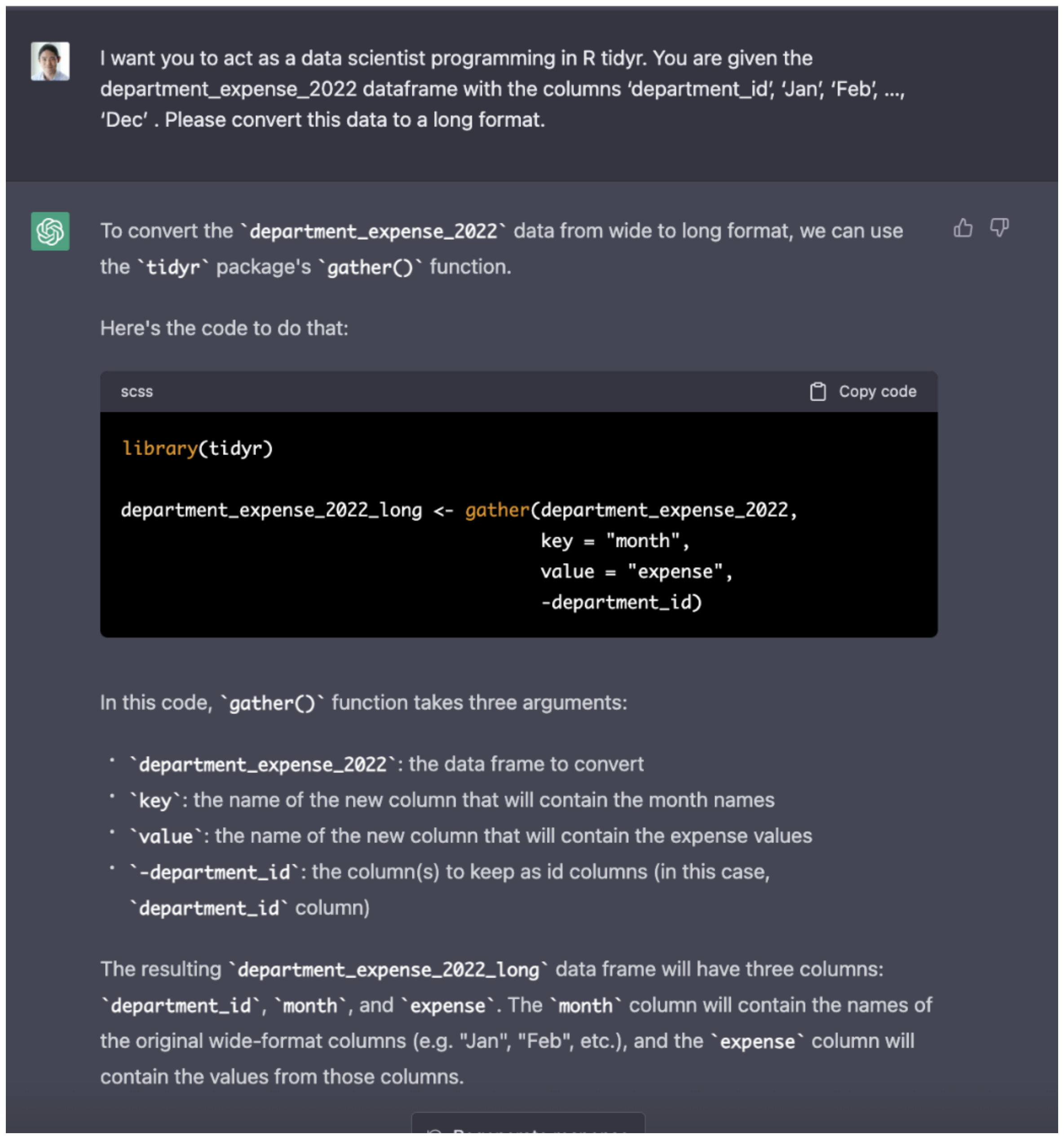
This will create a new dataframe `wide_expense` with the `department_id` column as the id variable and the `month` columns as new columns with their corresponding `expense` values.

Data analysis workflow in tidyr

PROMPT STRUCTURE

I want you to act as a data scientist programming in R tidyr. You are given the {dataframe name} dataframe containing the columns {column name}. Please convert the data to long format.

Example: Data Reshaping (Wide to Long)



I want you to act as a data scientist programming in R tidyr. You are given the department_expense_2022 dataframe with the columns 'department_id', 'Jan', 'Feb', ..., 'Dec' . Please convert this data to a long format.

To convert the `department_expense_2022` data from wide to long format, we can use the `tidyr` package's `gather()` function.

Here's the code to do that:

```
library(tidyr)

department_expense_2022_long <- gather(department_expense_2022,
                                       key = "month",
                                       value = "expense",
                                       -department_id)
```

In this code, `gather()` function takes three arguments:

- `department_expense_2022`: the data frame to convert
- `key`: the name of the new column that will contain the month names
- `value`: the name of the new column that will contain the expense values
- `-department_id`: the column(s) to keep as id columns (in this case, `department_id` column)

The resulting `department_expense_2022_long` data frame will have three columns: `department_id`, `month`, and `expense`. The `month` column will contain the names of the original wide-format columns (e.g. "Jan", "Feb", etc.), and the `expense` column will contain the values from those columns.

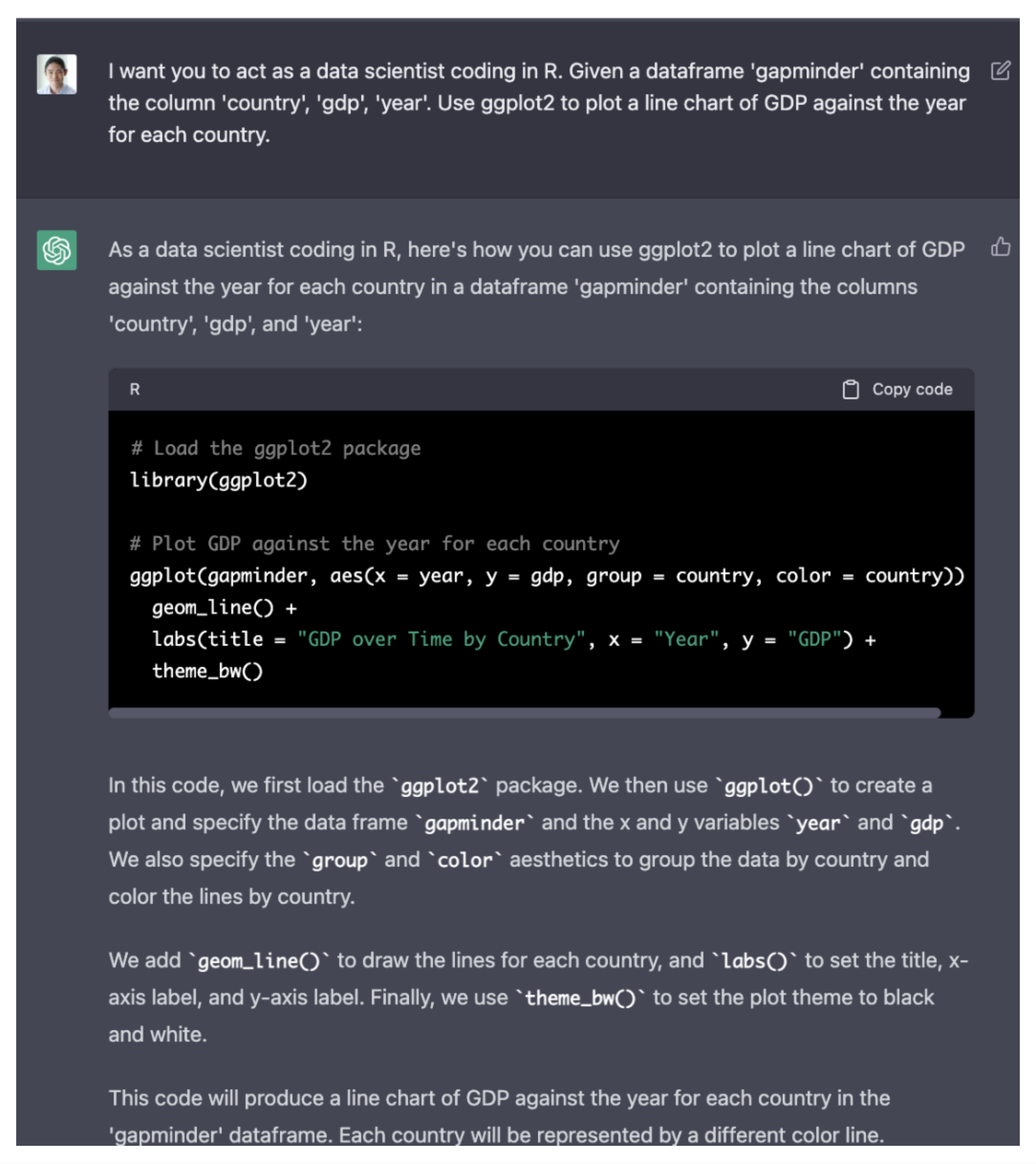
datacamp

Prompts for data visualization workflows

Creating plots in ggplot2

PROMPT STRUCTURE

I want you to act as a data scientist coding in R. Given a dataframe {dataframe name} containing the columns {column names} Use ggplot2 to plot a {chart type and requirement}.



I want you to act as a data scientist coding in R. Given a dataframe 'gapminder' containing the column 'country', 'gdp', 'year'. Use ggplot2 to plot a line chart of GDP against the year for each country.

As a data scientist coding in R, here's how you can use ggplot2 to plot a line chart of GDP against the year for each country in a dataframe 'gapminder' containing the columns 'country', 'gdp', and 'year':

```
# Load the ggplot2 package
library(ggplot2)

# Plot GDP against the year for each country
ggplot(gapminder, aes(x = year, y = gdp, group = country, color = country))
  geom_line() +
  labs(title = "GDP over Time by Country", x = "Year", y = "GDP") +
  theme_bw()
```

In this code, we first load the `ggplot2` package. We then use `ggplot()` to create a plot and specify the data frame `gapminder` and the x and y variables `year` and `gdp`. We also specify the `group` and `color` aesthetics to group the data by country and color the lines by country.

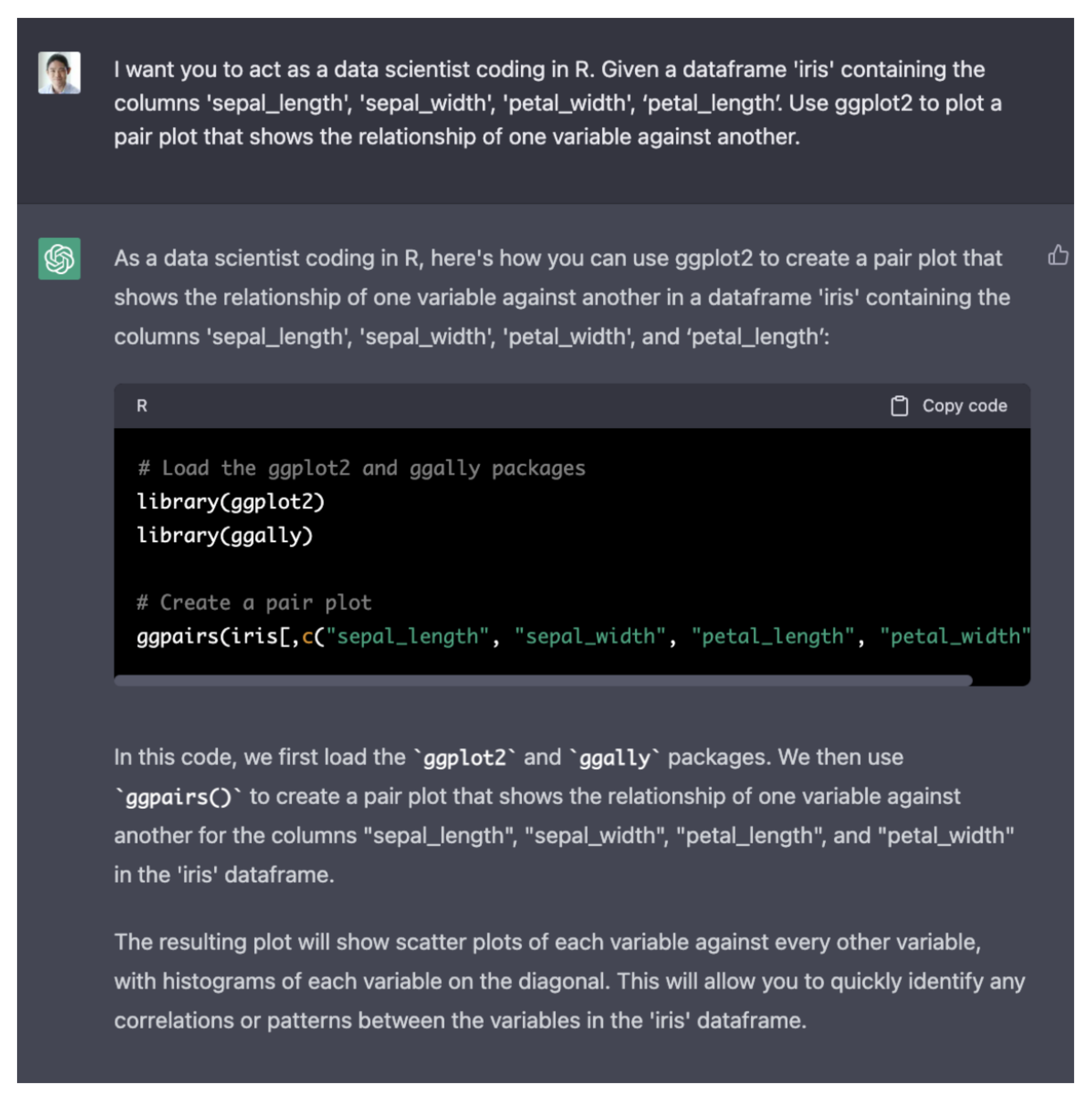
We add `geom_line()` to draw the lines for each country, and `labs()` to set the title, x-axis label, and y-axis label. Finally, we use `theme_bw()` to set the plot theme to black and white.

This code will produce a line chart of GDP against the year for each country in the 'gapminder' dataframe. Each country will be represented by a different color line.

Gridplot visualizations in ggplot2

PROMPT STRUCTURE

I want you to act as a data scientist coding in R. Given a dataframe {dataframe name} containing the columns {column names}. Use ggplot2 to plot a pair plot that shows the relationship of one variable against another.



I want you to act as a data scientist coding in R. Given a dataframe 'iris' containing the columns 'sepal_length', 'sepal_width', 'petal_width', 'petal_length'. Use ggplot2 to plot a pair plot that shows the relationship of one variable against another.

As a data scientist coding in R, here's how you can use ggplot2 to create a pair plot that shows the relationship of one variable against another in a dataframe 'iris' containing the columns 'sepal_length', 'sepal_width', 'petal_width', and 'petal_length':

```r
# Load the ggplot2 and ggally packages
library(ggplot2)
library(ggally)

# Create a pair plot
ggpairs(iris[,c("sepal_length", "sepal_width", "petal_length", "petal_width"
```

In this code, we first load the `ggplot2` and `ggally` packages. We then use `ggpairs()` to create a pair plot that shows the relationship of one variable against another for the columns "sepal_length", "sepal_width", "petal_length", and "petal_width" in the 'iris' dataframe.

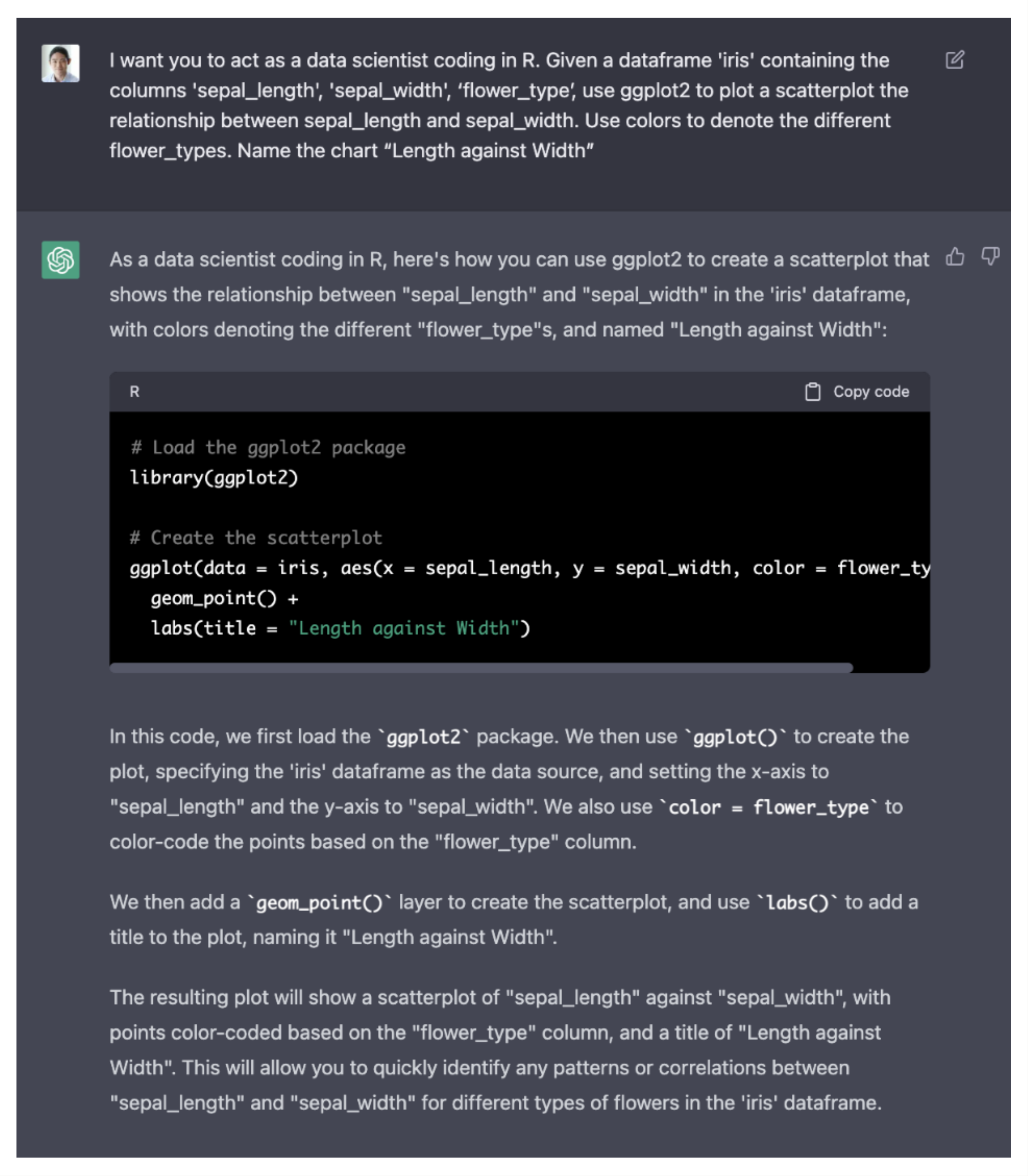
The resulting plot will show scatter plots of each variable against every other variable, with histograms of each variable on the diagonal. This will allow you to quickly identify any correlations or patterns between the variables in the 'iris' dataframe.

R data visualization workflows

Annotating and formatting plots

PROMPT STRUCTURE

I want you to act as a data scientist coding in R. Given a dataframe {dataframe name} containing the columns {column names}, use ggplot2 to plot a {chart type} the relationship between {variables}. {Insert annotation and formatting requirements}



I want you to act as a data scientist coding in R. Given a dataframe 'iris' containing the columns 'sepal_length', 'sepal_width', 'flower_type', use ggplot2 to plot a scatterplot the relationship between sepal_length and sepal_width. Use colors to denote the different flower_types. Name the chart "Length against Width"

As a data scientist coding in R, here's how you can use ggplot2 to create a scatterplot that shows the relationship between "sepal_length" and "sepal_width" in the 'iris' dataframe, with colors denoting the different "flower_type"s, and named "Length against Width":

```r
# Load the ggplot2 package
library(ggplot2)

# Create the scatterplot
ggplot(data = iris, aes(x = sepal_length, y = sepal_width, color = flower_ty
  geom_point() +
  labs(title = "Length against Width")
```

In this code, we first load the `ggplot2` package. We then use `ggplot()` to create the plot, specifying the 'iris' dataframe as the data source, and setting the x-axis to "sepal_length" and the y-axis to "sepal_width". We also use `color = flower_type` to color-code the points based on the "flower_type" column.

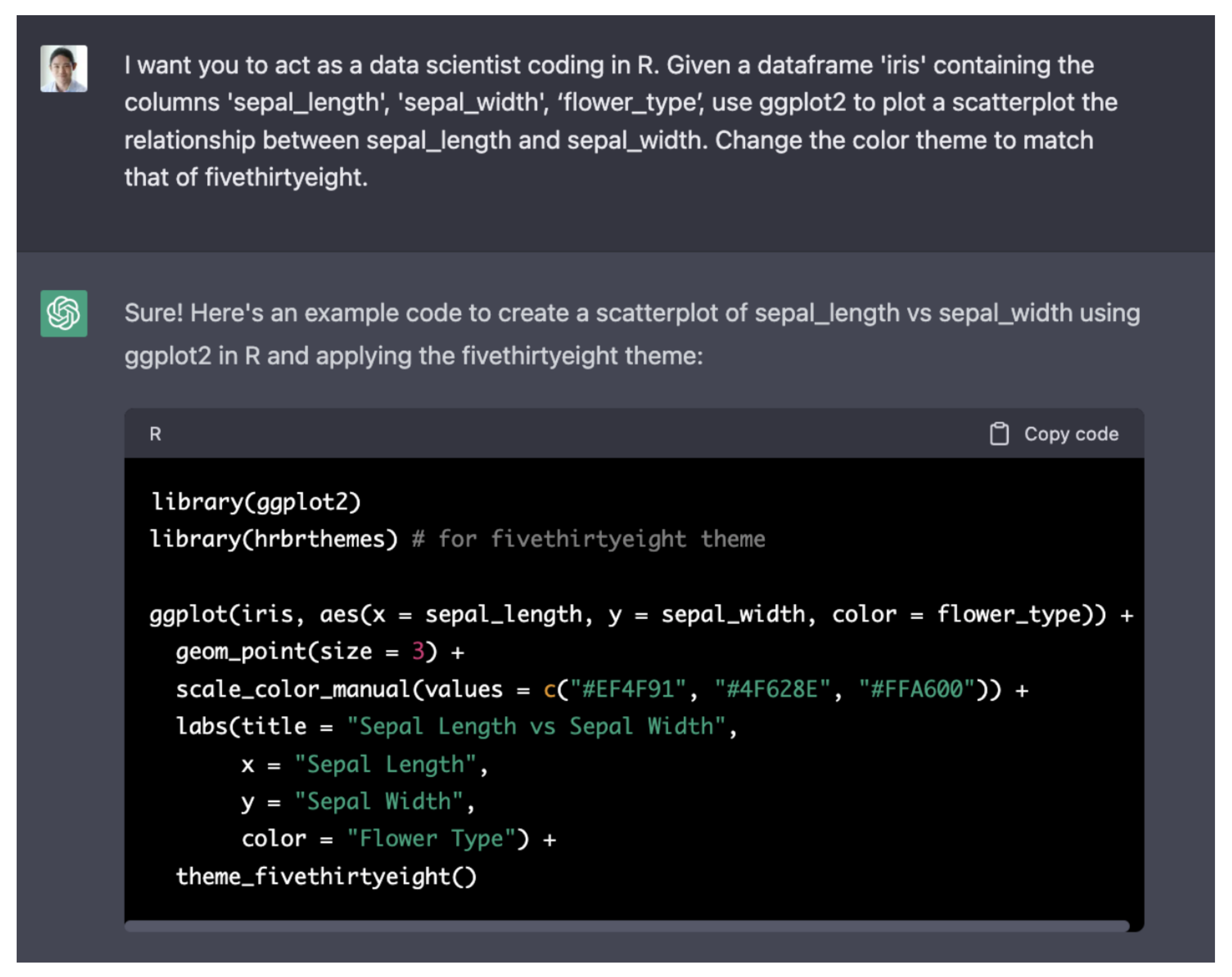
We then add a `geom_point()` layer to create the scatterplot, and use `labs()` to add a title to the plot, naming it "Length against Width".

The resulting plot will show a scatterplot of "sepal_length" against "sepal_width", with points color-coded based on the "flower_type" column, and a title of "Length against Width". This will allow you to quickly identify any patterns or correlations between "sepal_length" and "sepal_width" for different types of flowers in the 'iris' dataframe.

Changing plot themes in ggplot2

PROMPT STRUCTURE

I want you to act as a data scientist coding in R. Given a dataframe {dataframe name} containing the columns {column names}, use ggplot2 to to plot a {chart type} the relationship between {variables}. Change the color theme to match that of {theme}



I want you to act as a data scientist coding in R. Given a dataframe 'iris' containing the columns 'sepal_length', 'sepal_width', 'flower_type', use ggplot2 to plot a scatterplot the relationship between sepal_length and sepal_width. Change the color theme to match that of fivethirtyeight.

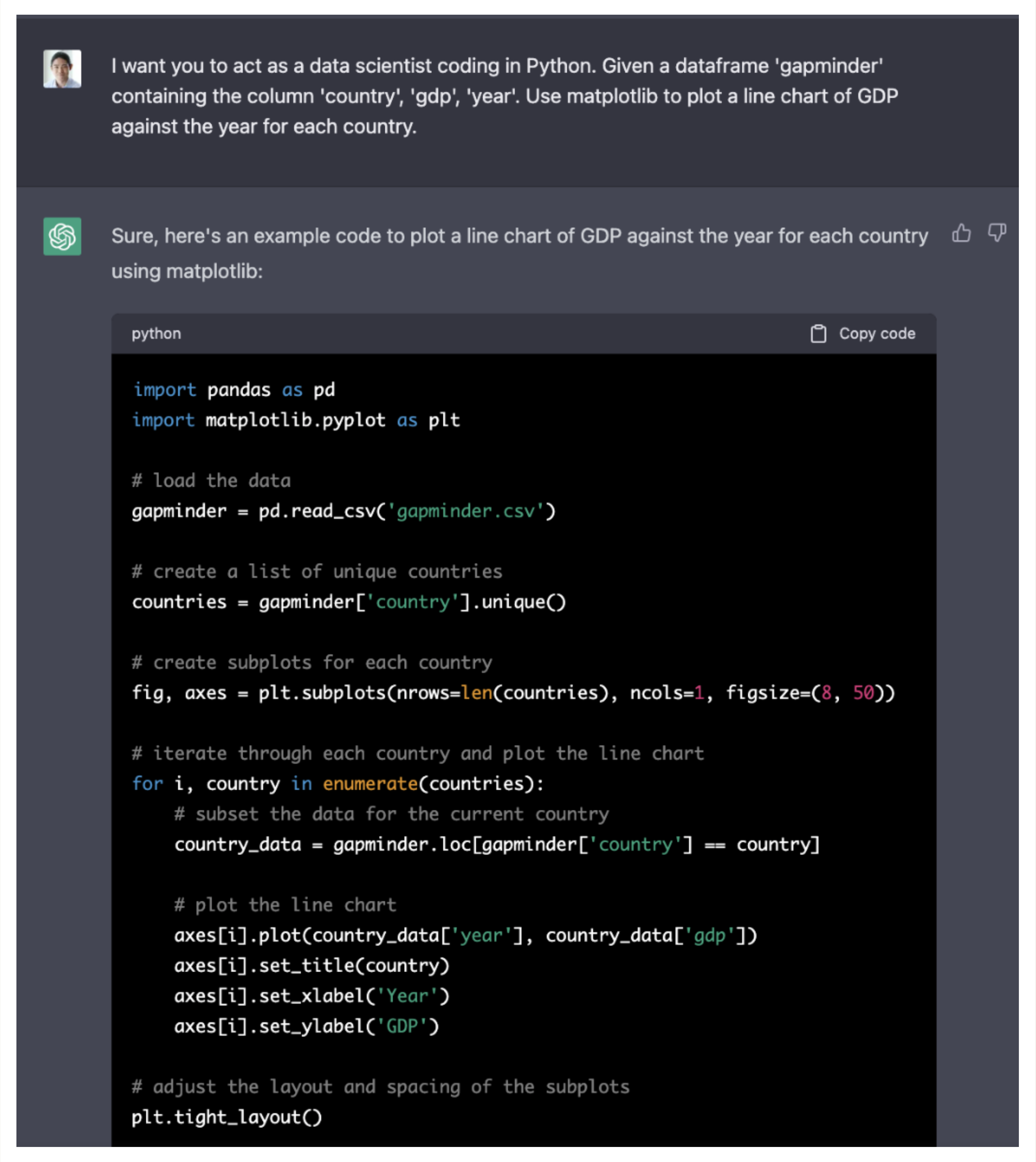
Sure! Here's an example code to create a scatterplot of sepal_length vs sepal_width using ggplot2 in R and applying the fivethirtyeight theme:

```r
library(ggplot2)
library(hrbrthemes) # for fivethirtyeight theme

ggplot(iris, aes(x = sepal_length, y = sepal_width, color = flower_type)) +
  geom_point(size = 3) +
  scale_color_manual(values = c("#EF4F91", "#4F628E", "#FFA600")) +
  labs(title = "Sepal Length vs Sepal Width",
       x = "Sepal Length",
       y = "Sepal Width",
       color = "Flower Type") +
  theme_fivethirtyeight()
```

Creating plots with matplotlib

PROMPT STRUCTURE

I want you to act as a data scientist coding in Python. Given a dataframe {dataframe name} containing the columns {column names} Use matplotlib to plot a {chart type and requirement}.



I want you to act as a data scientist coding in Python. Given a dataframe 'gapminder' containing the column 'country', 'gdp', 'year'. Use matplotlib to plot a line chart of GDP against the year for each country.

Sure, here's an example code to plot a line chart of GDP against the year for each country using matplotlib:

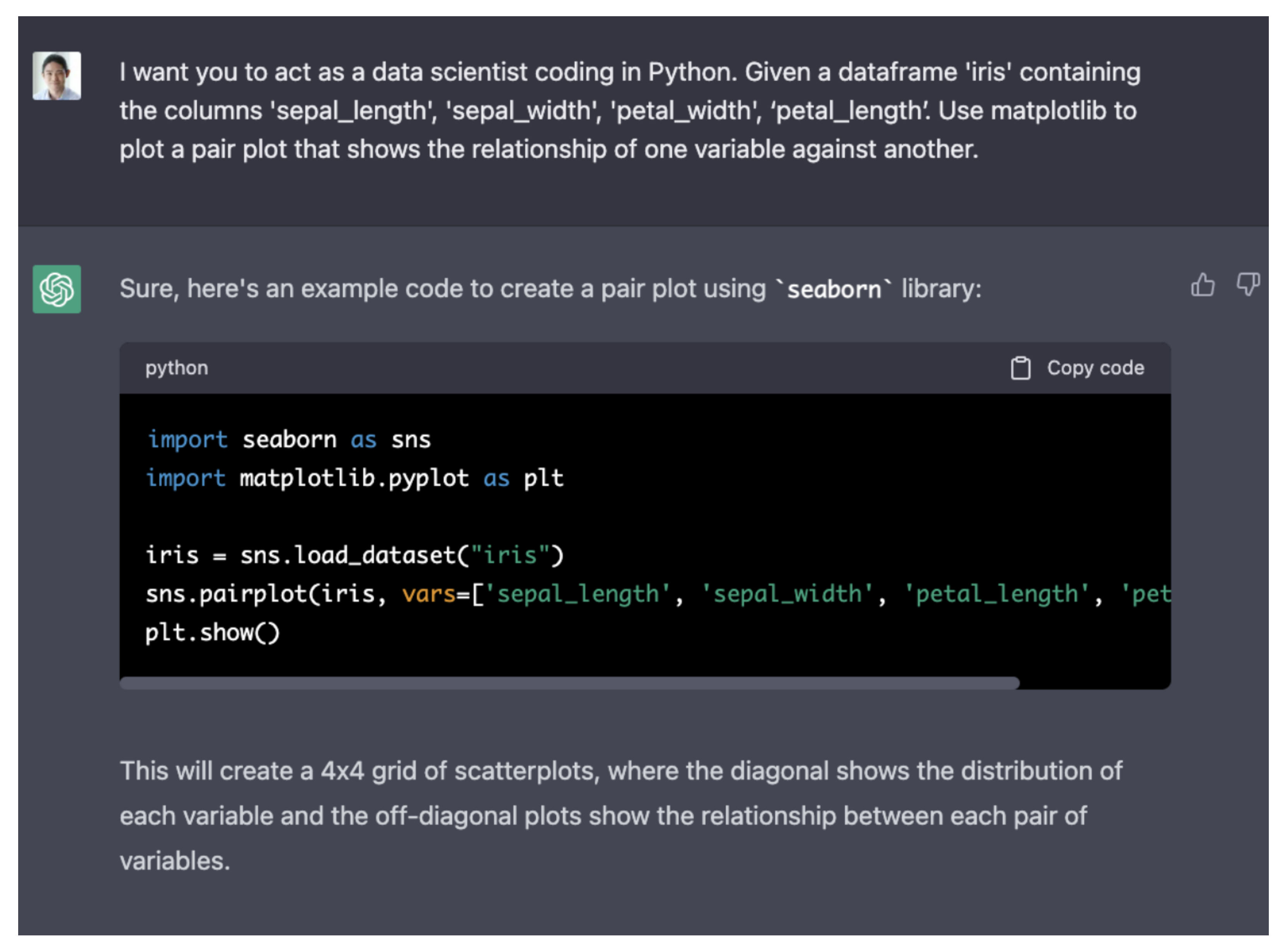
```python
import pandas as pd
import matplotlib.pyplot as plt

# load the data
gapminder = pd.read_csv('gapminder.csv')

# create a list of unique countries
countries = gapminder['country'].unique()

# create subplots for each country
fig, axes = plt.subplots(nrows=len(countries), ncols=1, figsize=(8, 50))

# iterate through each country and plot the line chart
for i, country in enumerate(countries):
    # subset the data for the current country
    country_data = gapminder.loc[gapminder['country'] == country]

    # plot the line chart
    axes[i].plot(country_data['year'], country_data['gdp'])
    axes[i].set_title(country)
    axes[i].set_xlabel('Year')
    axes[i].set_ylabel('GDP')

# adjust the layout and spacing of the subplots
plt.tight_layout()
```

datacamp

Crating pairplots with matplotlib

PROMPT STRUCTURE

I want you to act as a data scientist coding in Python. Given a dataframe {dataframe name} containing the columns {column names}. Use matplotlib to plot a pair plot that shows the relationship of one variable against another.



I want you to act as a data scientist coding in Python. Given a dataframe 'iris' containing the columns 'sepal_length', 'sepal_width', 'petal_width', 'petal_length'. Use matplotlib to plot a pair plot that shows the relationship of one variable against another.

Sure, here's an example code to create a pair plot using `seaborn` library:

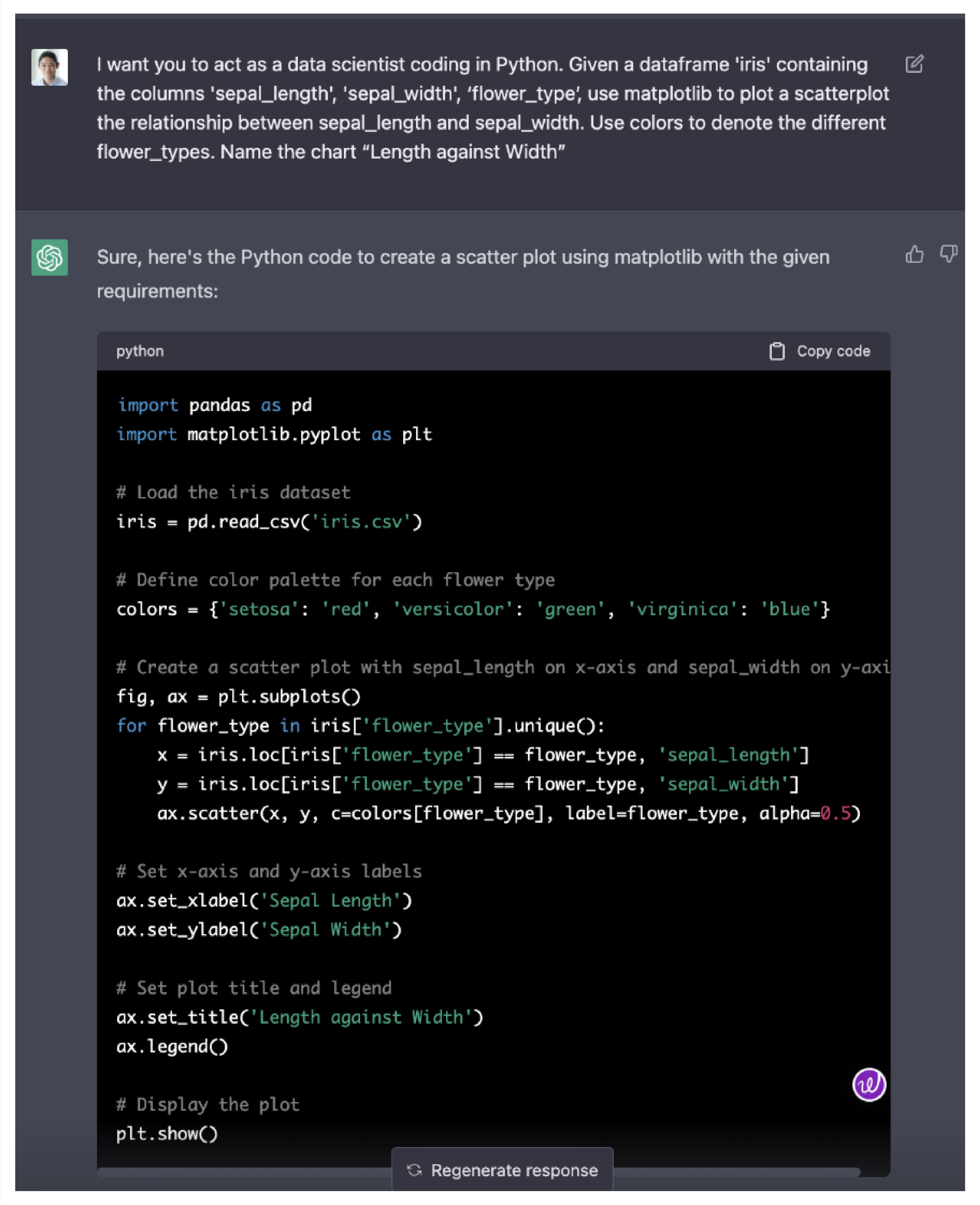
```python
import seaborn as sns
import matplotlib.pyplot as plt

iris = sns.load_dataset("iris")
sns.pairplot(iris, vars=['sepal_length', 'sepal_width', 'petal_length', 'pet
plt.show()
```

This will create a 4x4 grid of scatterplots, where the diagonal shows the distribution of each variable and the off-diagonal plots show the relationship between each pair of variables.

Annotating and formatting plots in matplotlib

PROMPT STRUCTURE

I want you to act as a data scientist coding in Python. Given a dataframe {dataframe name} containing the columns {column names}, use matplotlib to to plot a {chart type} the relationship between {variables}. {Insert annotation and formatting requirements}



I want you to act as a data scientist coding in Python. Given a dataframe 'iris' containing the columns 'sepal_length', 'sepal_width', ‘flower_type’, use matplotlib to plot a scatterplot the relationship between sepal_length and sepal_width. Use colors to denote the different flower_types. Name the chart “Length against Width”

Sure, here's the Python code to create a scatter plot using matplotlib with the given requirements:

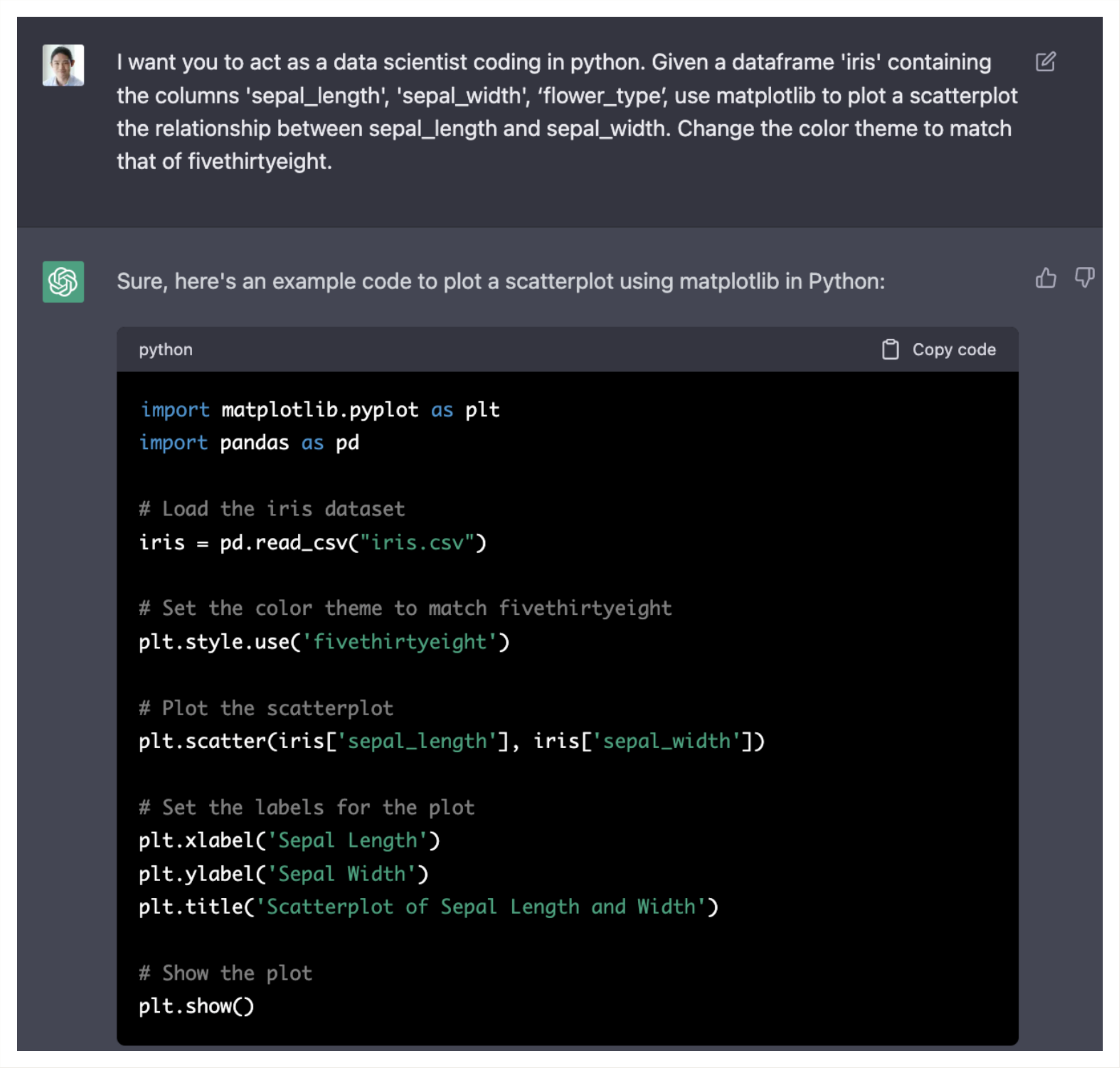
```python
import pandas as pd
import matplotlib.pyplot as plt

# Load the iris dataset
iris = pd.read_csv('iris.csv')

# Define color palette for each flower type
colors = {'setosa': 'red', 'versicolor': 'green', 'virginica': 'blue'}

# Create a scatter plot with sepal_length on x-axis and sepal_width on y-axi
fig, ax = plt.subplots()
for flower_type in iris['flower_type'].unique():
    x = iris.loc[iris['flower_type'] == flower_type, 'sepal_length']
    y = iris.loc[iris['flower_type'] == flower_type, 'sepal_width']
    ax.scatter(x, y, c=colors[flower_type], label=flower_type, alpha=0.5)

# Set x-axis and y-axis labels
ax.set_xlabel('Sepal Length')
ax.set_ylabel('Sepal Width')

# Set plot title and legend
ax.set_title('Length against Width')
ax.legend()

# Display the plot
plt.show()
```

datacamp

Changing plot themes in matplotlib

PROMPT STRUCTURE

I want you to act as a data scientist coding in Python. Given a dataframe {dataframe name} containing the columns {column names}, use matplotlib to to plot a {chart type} the relationship between {variables}. Change the color theme to match that of {theme}



I want you to act as a data scientist coding in python. Given a dataframe 'iris' containing the columns 'sepal_length', 'sepal_width', 'flower_type', use matplotlib to plot a scatterplot the relationship between sepal_length and sepal_width. Change the color theme to match that of fivethirtyeight.

Sure, here's an example code to plot a scatterplot using matplotlib in Python:

```python
import matplotlib.pyplot as plt
import pandas as pd

# Load the iris dataset
iris = pd.read_csv("iris.csv")

# Set the color theme to match fivethirtyeight
plt.style.use('fivethirtyeight')

# Plot the scatterplot
plt.scatter(iris['sepal_length'], iris['sepal_width'])

# Set the labels for the plot
plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')
plt.title('Scatterplot of Sepal Length and Width')

# Show the plot
plt.show()
```

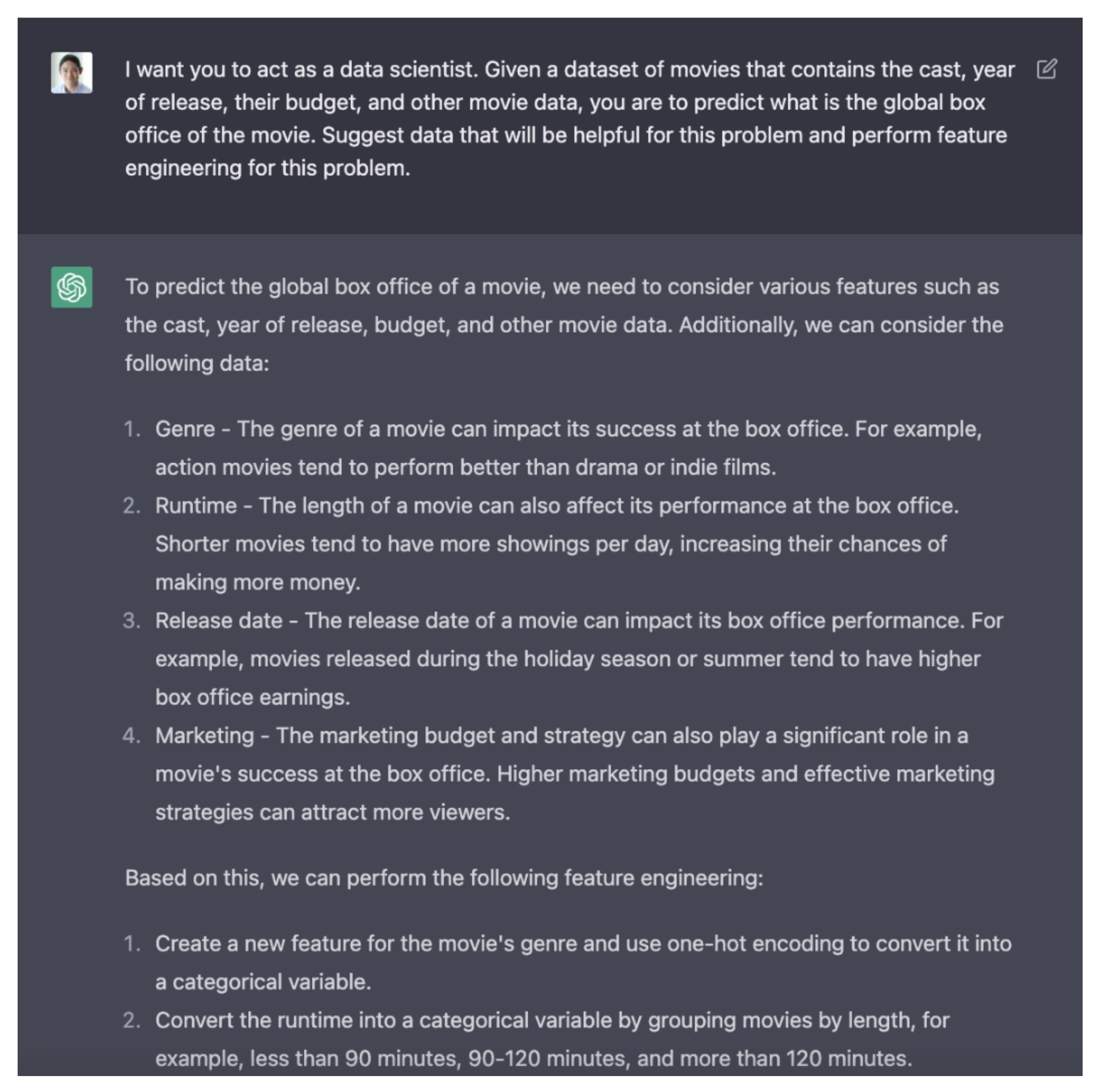
datacamp

Prompts for machine learning workflows

Feature engineering ideation

PROMPT STRUCTURE

I want you to act as a data scientist. Given a dataset of {dataset name} that contains the {columns}, you are to predict {predicted variable}. Suggest data that will be helpful for this problem and perform feature engineering for this problem.



I want you to act as a data scientist. Given a dataset of movies that contains the cast, year of release, their budget, and other movie data, you are to predict what is the global box office of the movie. Suggest data that will be helpful for this problem and perform feature engineering for this problem.

To predict the global box office of a movie, we need to consider various features such as the cast, year of release, budget, and other movie data. Additionally, we can consider the following data:

1. Genre - The genre of a movie can impact its success at the box office. For example, action movies tend to perform better than drama or indie films.
2. Runtime - The length of a movie can also affect its performance at the box office. Shorter movies tend to have more showings per day, increasing their chances of making more money.
3. Release date - The release date of a movie can impact its box office performance. For example, movies released during the holiday season or summer tend to have higher box office earnings.
4. Marketing - The marketing budget and strategy can also play a significant role in a movie's success at the box office. Higher marketing budgets and effective marketing strategies can attract more viewers.

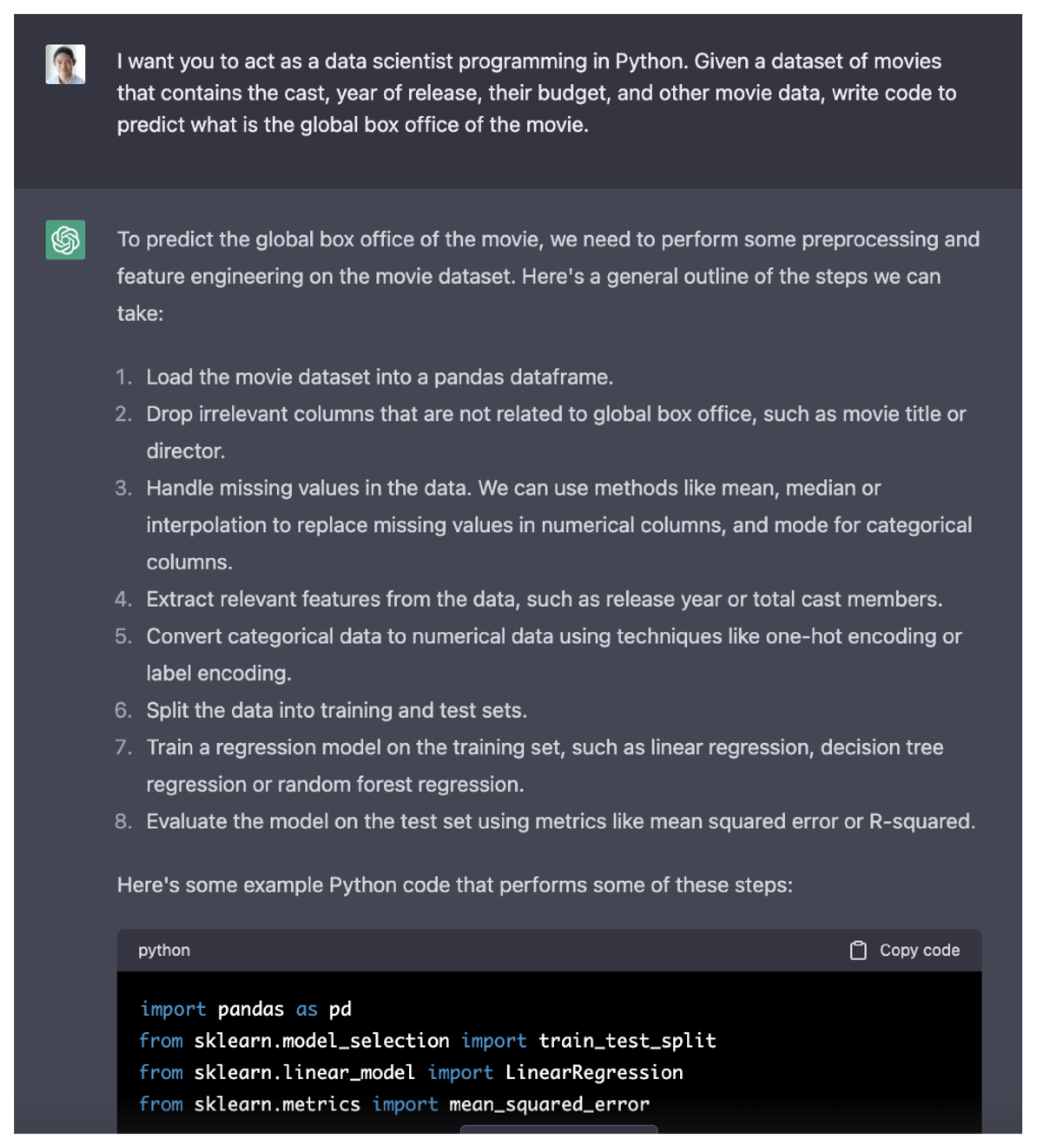
Based on this, we can perform the following feature engineering:

1. Create a new feature for the movie's genre and use one-hot encoding to convert it into a categorical variable.
2. Convert the runtime into a categorical variable by grouping movies by length, for example, less than 90 minutes, 90-120 minutes, and more than 120 minutes.

Model training workflow

PROMPT STRUCTURE

I want you to act as a data scientist programming in Python. Given a dataset of {dataframe name} that contains the {column name}, write code to predict {output variable}.



I want you to act as a data scientist programming in Python. Given a dataset of movies that contains the cast, year of release, their budget, and other movie data, write code to predict what is the global box office of the movie.

To predict the global box office of the movie, we need to perform some preprocessing and feature engineering on the movie dataset. Here's a general outline of the steps we can take:

1. Load the movie dataset into a pandas dataframe.
2. Drop irrelevant columns that are not related to global box office, such as movie title or director.
3. Handle missing values in the data. We can use methods like mean, median or interpolation to replace missing values in numerical columns, and mode for categorical columns.
4. Extract relevant features from the data, such as release year or total cast members.
5. Convert categorical data to numerical data using techniques like one-hot encoding or label encoding.
6. Split the data into training and test sets.
7. Train a regression model on the training set, such as linear regression, decision tree regression or random forest regression.
8. Evaluate the model on the test set using metrics like mean squared error or R-squared.

Here's some example Python code that performs some of these steps:

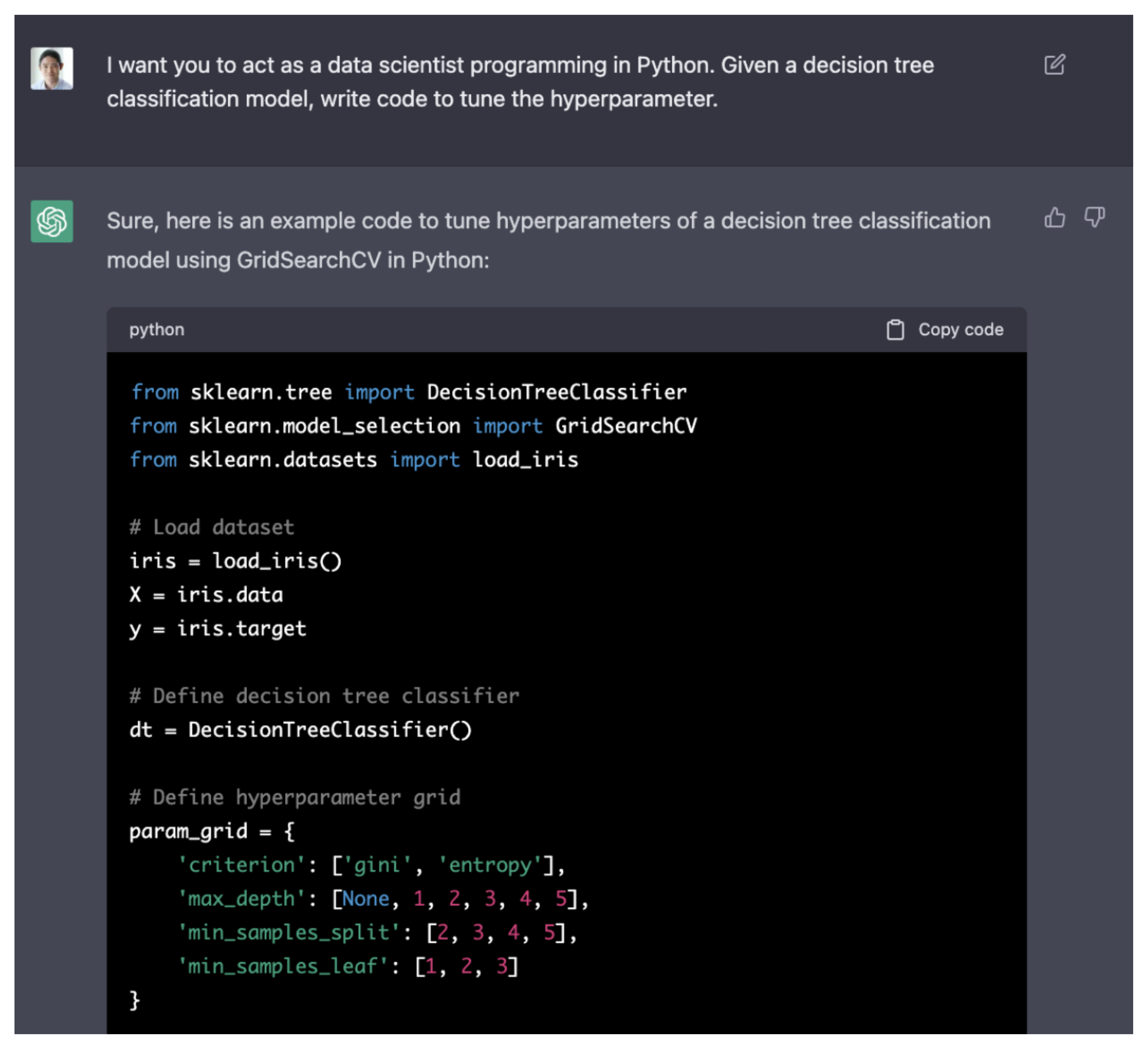
```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
```

Hyperparameter tuning workflow

PROMPT STRUCTURE

I want you to act as a data scientist programming in Python. Given a {type of model} model, write code to tune the hyperparameter.

Example:



I want you to act as a data scientist programming in Python. Given a decision tree classification model, write code to tune the hyperparameter.

Sure, here is an example code to tune hyperparameters of a decision tree classification model using GridSearchCV in Python:

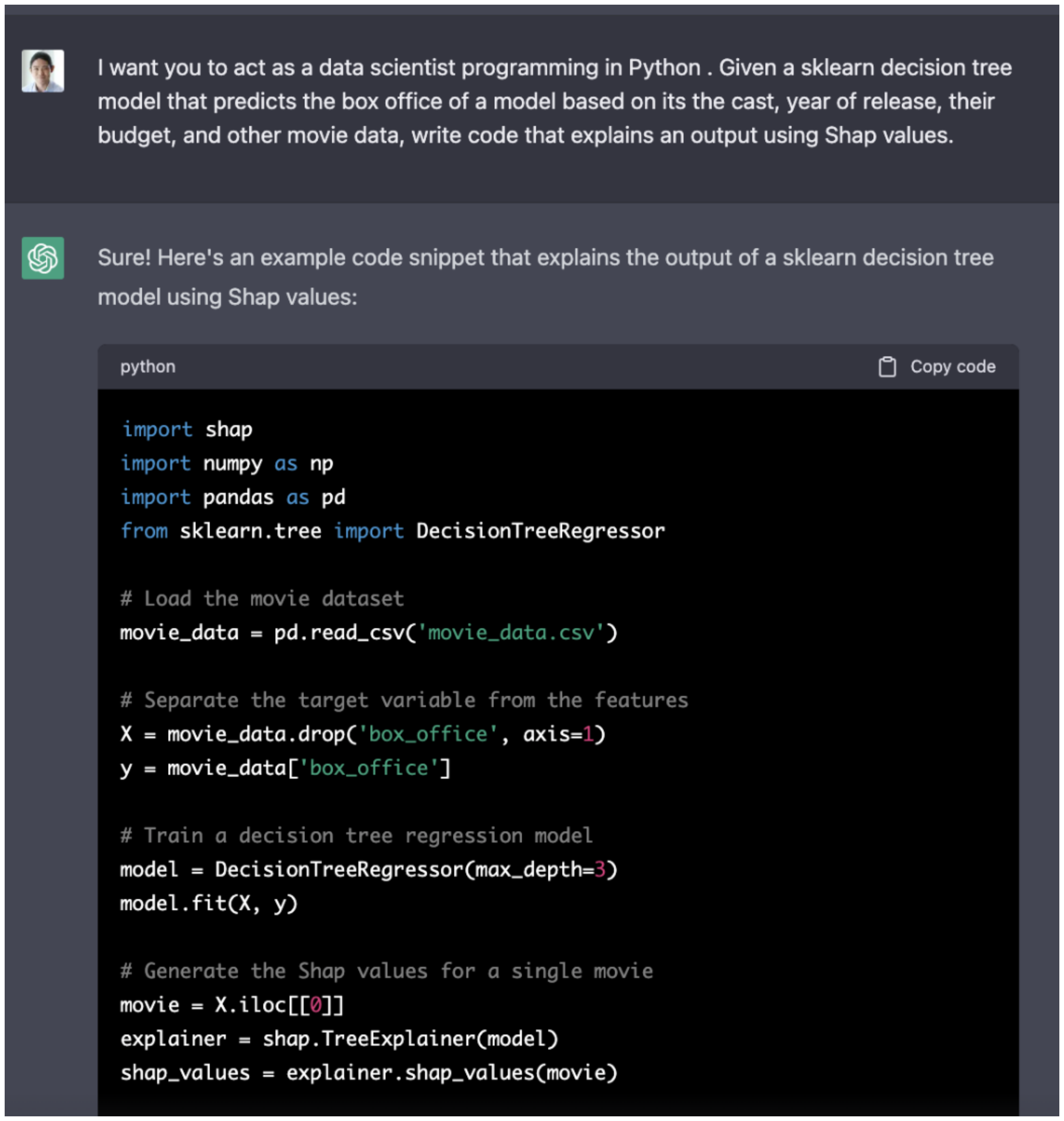
```python
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.datasets import load_iris

# Load dataset
iris = load_iris()
X = iris.data
y = iris.target

# Define decision tree classifier
dt = DecisionTreeClassifier()

# Define hyperparameter grid
param_grid = {
    'criterion': ['gini', 'entropy'],
    'max_depth': [None, 1, 2, 3, 4, 5],
    'min_samples_split': [2, 3, 4, 5],
    'min_samples_leaf': [1, 2, 3]
}
```

Model explainability workflow

PROMPT STRUCTURE

I want you to act as a data scientist programming in Python. Given a {type of model} that predicts the {predictor variable}, write code that explains an output using Shap values.

Example:



I want you to act as a data scientist programming in Python . Given a sklearn decision tree model that predicts the box office of a model based on its the cast, year of release, their budget, and other movie data, write code that explains an output using Shap values.

Sure! Here's an example code snippet that explains the output of a sklearn decision tree model using Shap values:

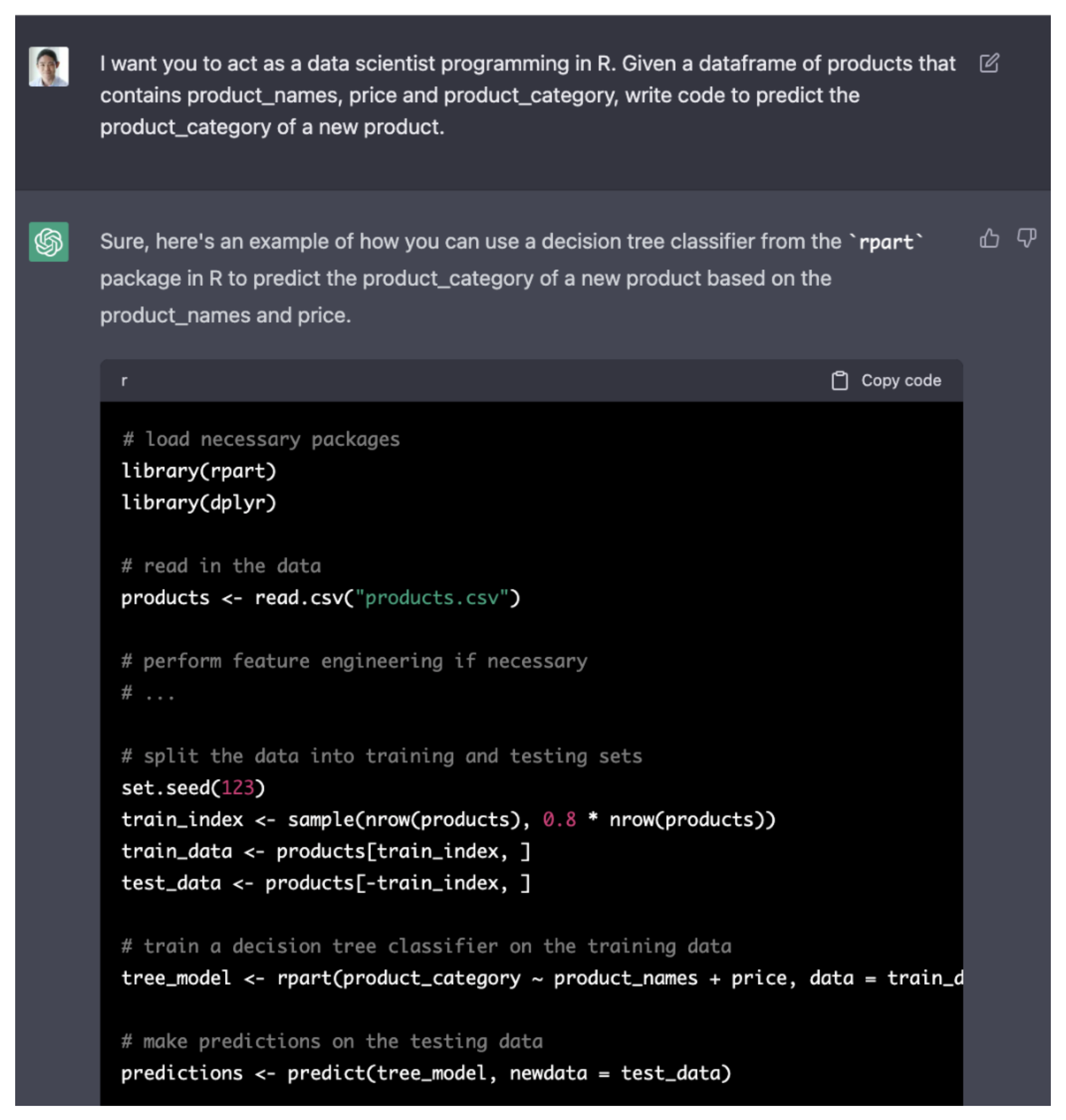
```python
import shap
import numpy as np
import pandas as pd
from sklearn.tree import DecisionTreeRegressor

# Load the movie dataset
movie_data = pd.read_csv('movie_data.csv')

# Separate the target variable from the features
X = movie_data.drop('box_office', axis=1)
y = movie_data['box_office']

# Train a decision tree regression model
model = DecisionTreeRegressor(max_depth=3)
model.fit(X, y)

# Generate the Shap values for a single movie
movie = X.iloc[[0]]
explainer = shap.TreeExplainer(model)
shap_values = explainer.shap_values(movie)
```

Model training workflow

PROMPT STRUCTURE

I want you to act as a data scientist programming in R. Given a dataframe of {dataframe name} that contains {column names}, write code to predict {output}.

Example: Train a classification model in R



I want you to act as a data scientist programming in R. Given a dataframe of products that contains product_names, price and product_category, write code to predict the product_category of a new product.

Sure, here's an example of how you can use a decision tree classifier from the `rpart` package in R to predict the product_category of a new product based on the product_names and price.

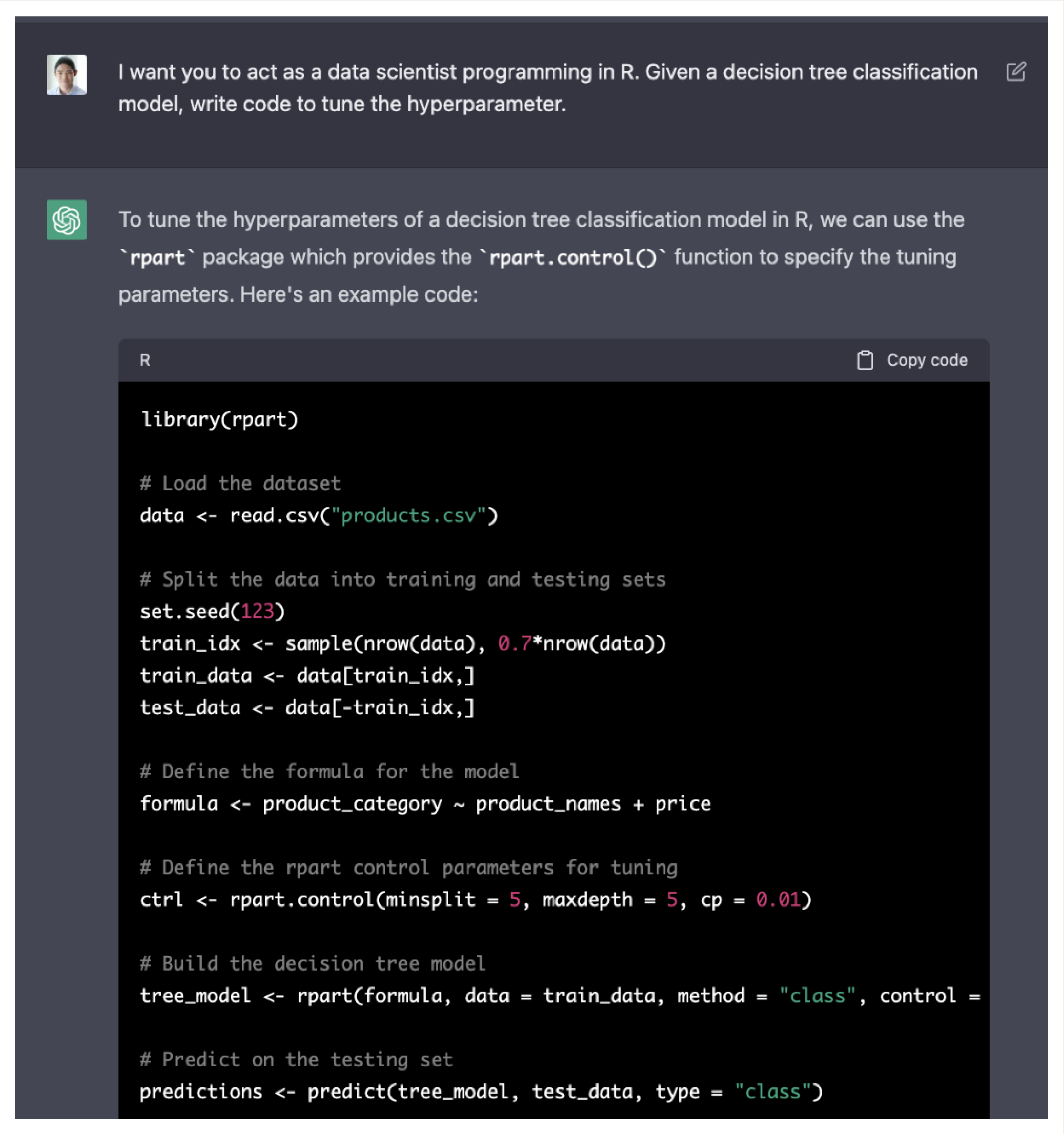
```r
# load necessary packages
library(rpart)
library(dplyr)

# read in the data
products <- read.csv("products.csv")

# perform feature engineering if necessary
# ...

# split the data into training and testing sets
set.seed(123)
train_index <- sample(nrow(products), 0.8 * nrow(products))
train_data <- products[train_index, ]
test_data <- products[-train_index, ]

# train a decision tree classifier on the training data
tree_model <- rpart(product_category ~ product_names + price, data = train_d

# make predictions on the testing data
predictions <- predict(tree_model, newdata = test_data)
```

Hyperparameter tuning workflow

PROMPT STRUCTURE

I want you to act as a data scientist programming in R. Given a {type of model} model, write code to tune the hyperparameter.

Example:



I want you to act as a data scientist programming in R. Given a decision tree classification model, write code to tune the hyperparameter.

To tune the hyperparameters of a decision tree classification model in R, we can use the `rpart` package which provides the `rpart.control()` function to specify the tuning parameters. Here's an example code:

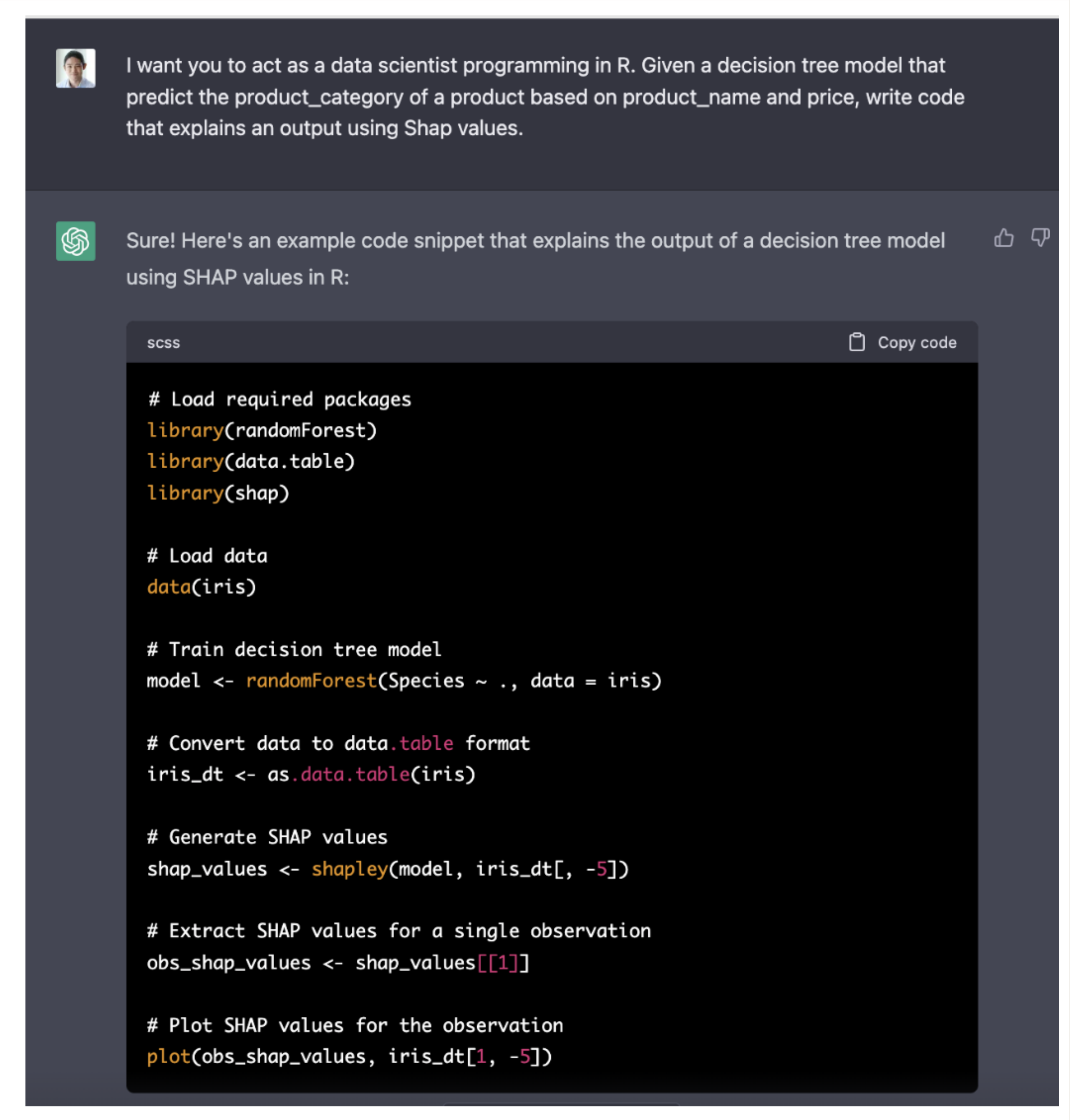
```r
library(rpart)

# Load the dataset
data <- read.csv("products.csv")

# Split the data into training and testing sets
set.seed(123)
train_idx <- sample(nrow(data), 0.7*nrow(data))
train_data <- data[train_idx,]
test_data <- data[-train_idx,]

# Define the formula for the model
formula <- product_category ~ product_names + price

# Define the rpart control parameters for tuning
ctrl <- rpart.control(minsplit = 5, maxdepth = 5, cp = 0.01)

# Build the decision tree model
tree_model <- rpart(formula, data = train_data, method = "class", control =

# Predict on the testing set
predictions <- predict(tree_model, test_data, type = "class")
```

Model explainability workflow

PROMPT STRUCTURE

I want you to act as a data scientist programming in R . Given a {type of model} that predicts the {predictor variable}, write code that explains an output using Shap values.

Example:



I want you to act as a data scientist programming in R. Given a decision tree model that predict the product_category of a product based on product_name and price, write code that explains an output using Shap values.

Sure! Here's an example code snippet that explains the output of a decision tree model using SHAP values in R:

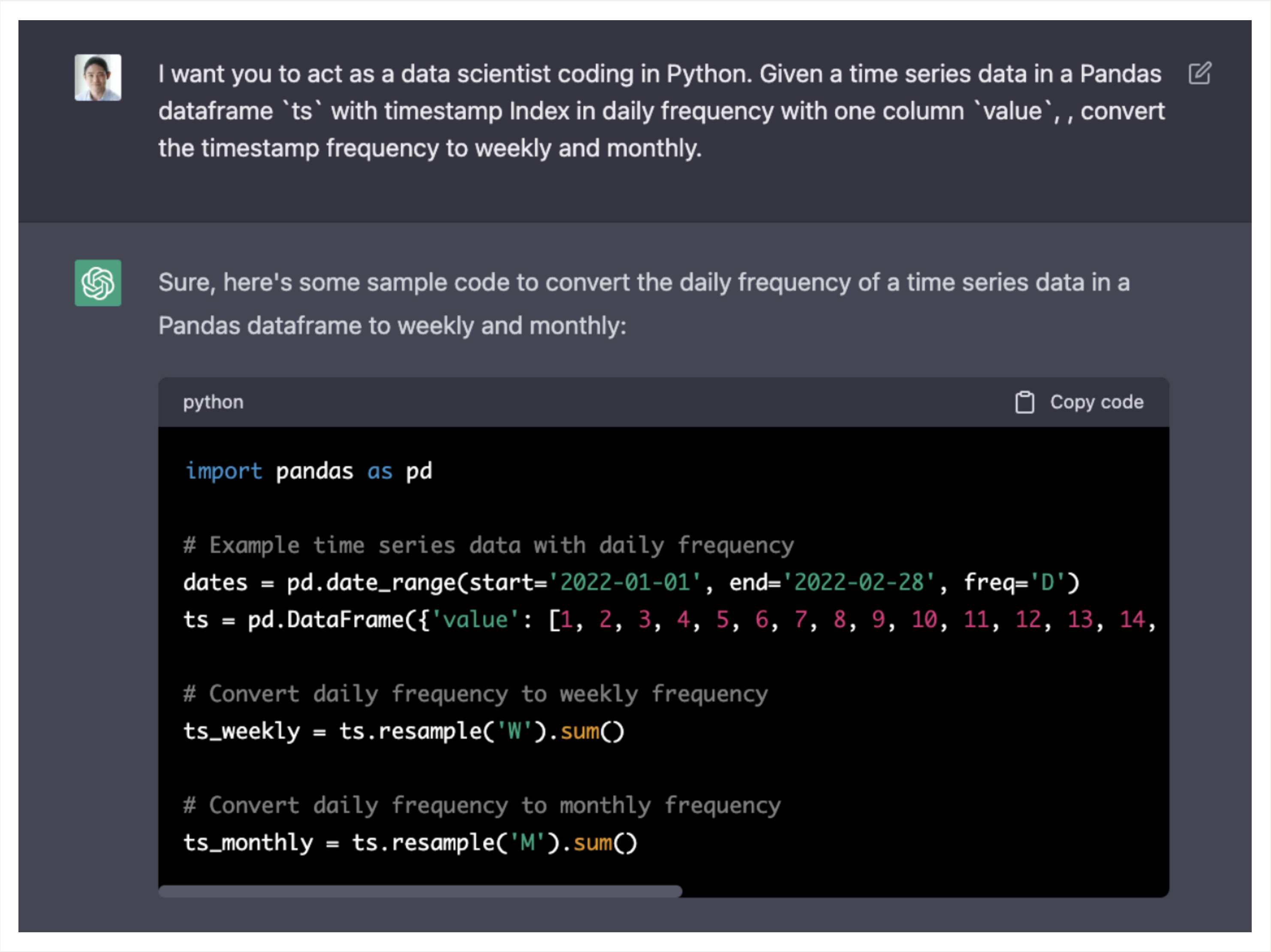
```
# Load required packages
library(randomForest)
library(data.table)
library(shap)

# Load data
data(iris)

# Train decision tree model
model <- randomForest(Species ~ ., data = iris)

# Convert data to data.table format
iris_dt <- as.data.table(iris)

# Generate SHAP values
shap_values <- shapley(model, iris_dt[, -5])

# Extract SHAP values for a single observation
obs_shap_values <- shap_values[[1]]

# Plot SHAP values for the observation
plot(obs_shap_values, iris_dt[1, -5])
```

Prompts for time series analysis workflows

Changing time horizons using pandas

PROMPT STRUCTURE

I want you to act as a data scientist coding in Python. Given a time series data in a Pandas dataframe {dataframe name} with timestamp Index in {original frequency} frequency with one column {column name}, convert the timestamp frequency to {desired frequency}.



I want you to act as a data scientist coding in Python. Given a time series data in a Pandas dataframe `ts` with timestamp Index in daily frequency with one column `value`, , convert the timestamp frequency to weekly and monthly.

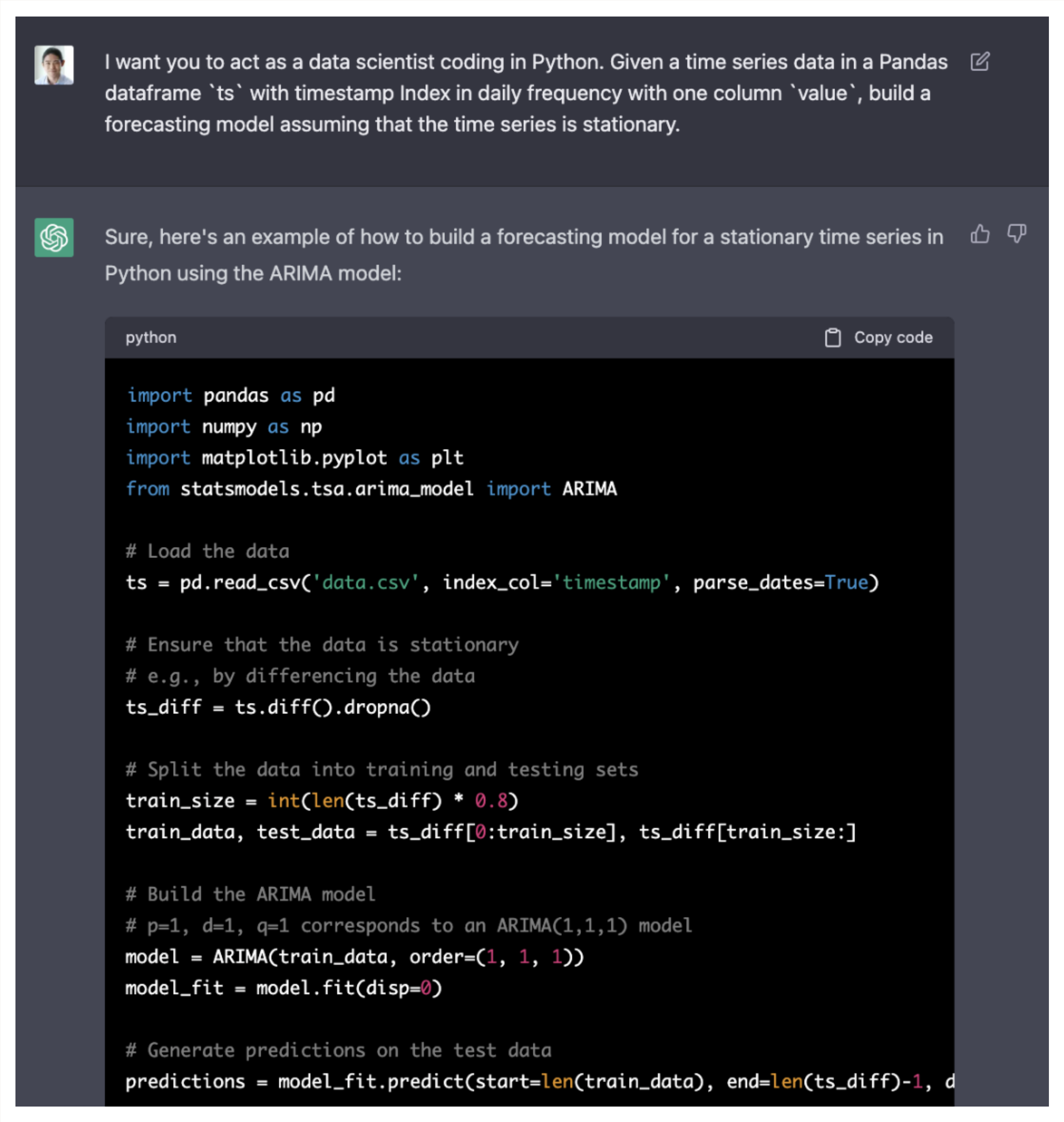
Sure, here's some sample code to convert the daily frequency of a time series data in a Pandas dataframe to weekly and monthly:

```python
import pandas as pd

# Example time series data with daily frequency
dates = pd.date_range(start='2022-01-01', end='2022-02-28', freq='D')
ts = pd.DataFrame({'value': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,

# Convert daily frequency to weekly frequency
ts_weekly = ts.resample('W').sum()

# Convert daily frequency to monthly frequency
ts_monthly = ts.resample('M').sum()
```

Build test series model

PROMPT STRUCTURE

I want you to act as a data scientist coding in Python. Given a time series data in a dataframe {dataframe name} with timestamp Index in {original frequency} frequency with one column {column name}, build a forecasting model, assuming data is stationary.



I want you to act as a data scientist coding in Python. Given a time series data in a Pandas dataframe `ts` with timestamp Index in daily frequency with one column `value`, build a forecasting model assuming that the time series is stationary.

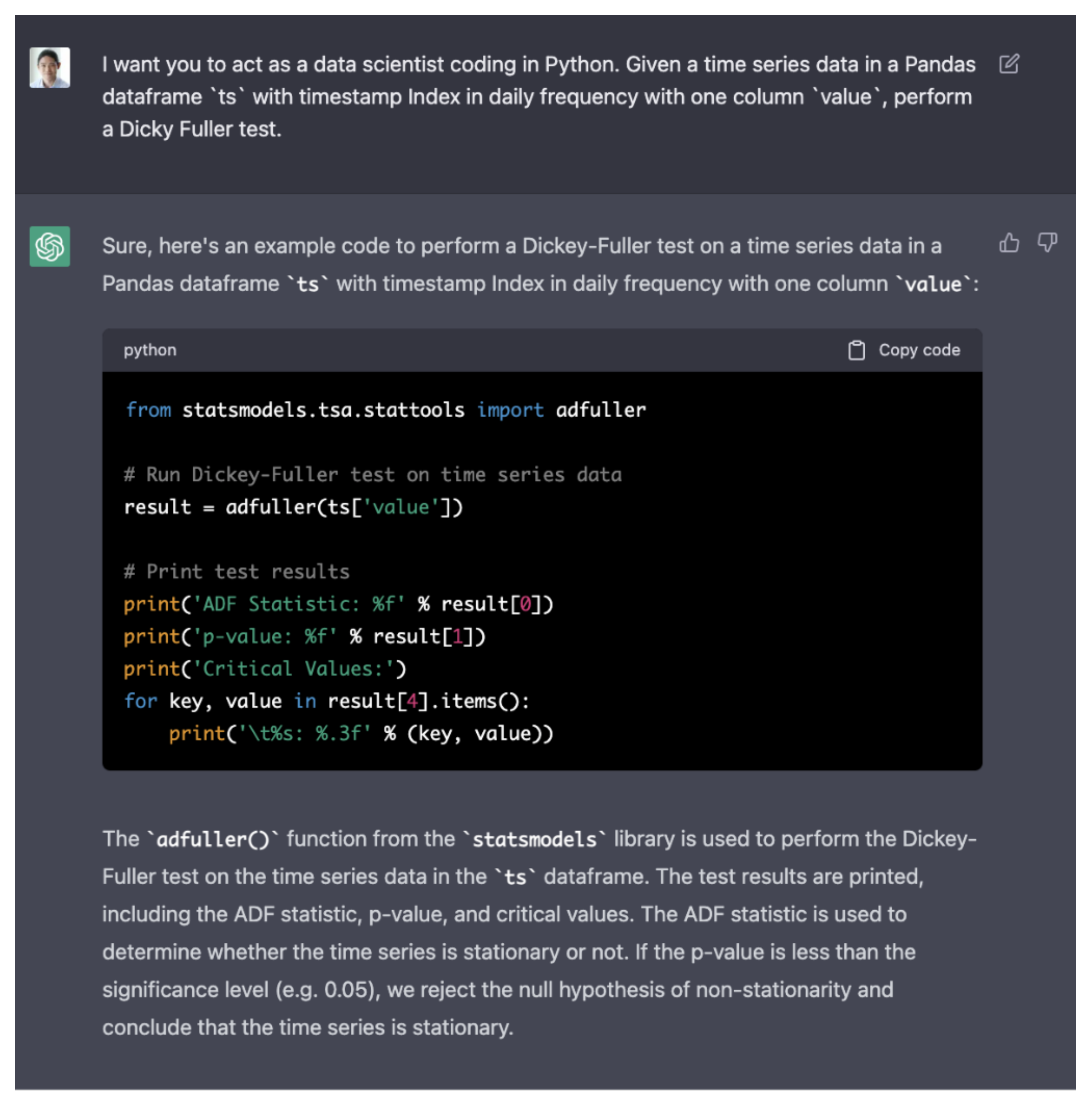
Sure, here's an example of how to build a forecasting model for a stationary time series in Python using the ARIMA model:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.arima_model import ARIMA

# Load the data
ts = pd.read_csv('data.csv', index_col='timestamp', parse_dates=True)

# Ensure that the data is stationary
# e.g., by differencing the data
ts_diff = ts.diff().dropna()

# Split the data into training and testing sets
train_size = int(len(ts_diff) * 0.8)
train_data, test_data = ts_diff[0:train_size], ts_diff[train_size:]

# Build the ARIMA model
# p=1, d=1, q=1 corresponds to an ARIMA(1,1,1) model
model = ARIMA(train_data, order=(1, 1, 1))
model_fit = model.fit(disp=0)

# Generate predictions on the test data
predictions = model_fit.predict(start=len(train_data), end=len(ts_diff)-1, d
```

Perform stationarity test

PROMPT STRUCTURE

I want you to act as a data scientist coding in Python. Given a time series data in a dataframe {dataframe name} with timestamp Index in {original frequency} frequency with one column {column name}, perform a Dicky Fuller test.



I want you to act as a data scientist coding in Python. Given a time series data in a Pandas dataframe `ts` with timestamp Index in daily frequency with one column `value`, perform a Dicky Fuller test.

Sure, here's an example code to perform a Dickey-Fuller test on a time series data in a Pandas dataframe `ts` with timestamp Index in daily frequency with one column `value`:

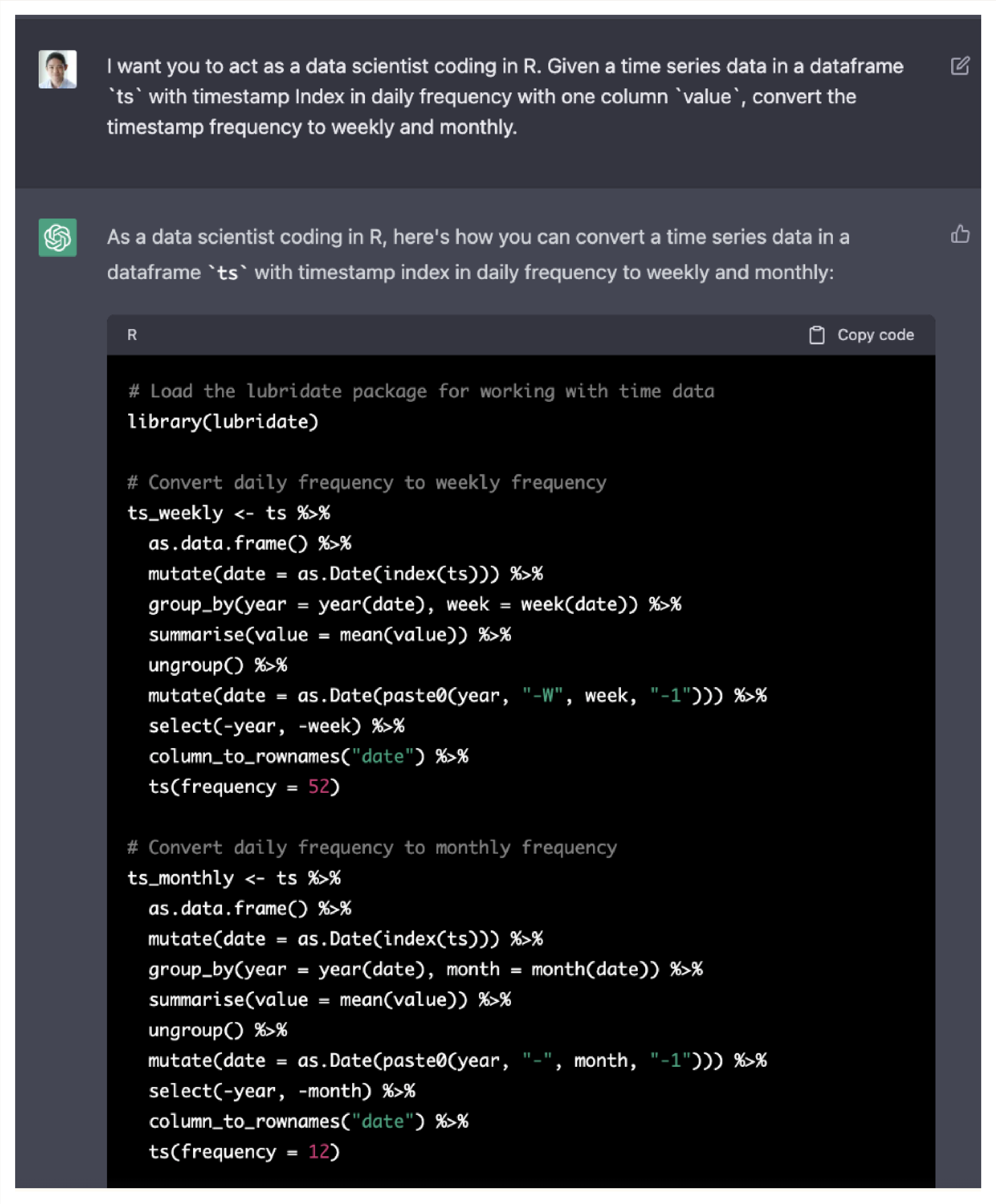
```python
from statsmodels.tsa.stattools import adfuller

# Run Dickey-Fuller test on time series data
result = adfuller(ts['value'])

# Print test results
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
    print('\t%s: %.3f' % (key, value))
```

The `adfuller()` function from the `statsmodels` library is used to perform the Dickey-Fuller test on the time series data in the `ts` dataframe. The test results are printed, including the ADF statistic, p-value, and critical values. The ADF statistic is used to determine whether the time series is stationary or not. If the p-value is less than the significance level (e.g. 0.05), we reject the null hypothesis of non-stationarity and conclude that the time series is stationary.

Changing time horizons

PROMPT STRUCTURE

I want you to act as a data scientist coding in R. Given a time series data in a dataframe {dataframe name} with timestamp Index in {original frequency} frequency with one column {column name}, convert the timestamp frequency to {desired frequency}



I want you to act as a data scientist coding in R. Given a time series data in a dataframe `ts` with timestamp Index in daily frequency with one column `value`, convert the timestamp frequency to weekly and monthly.

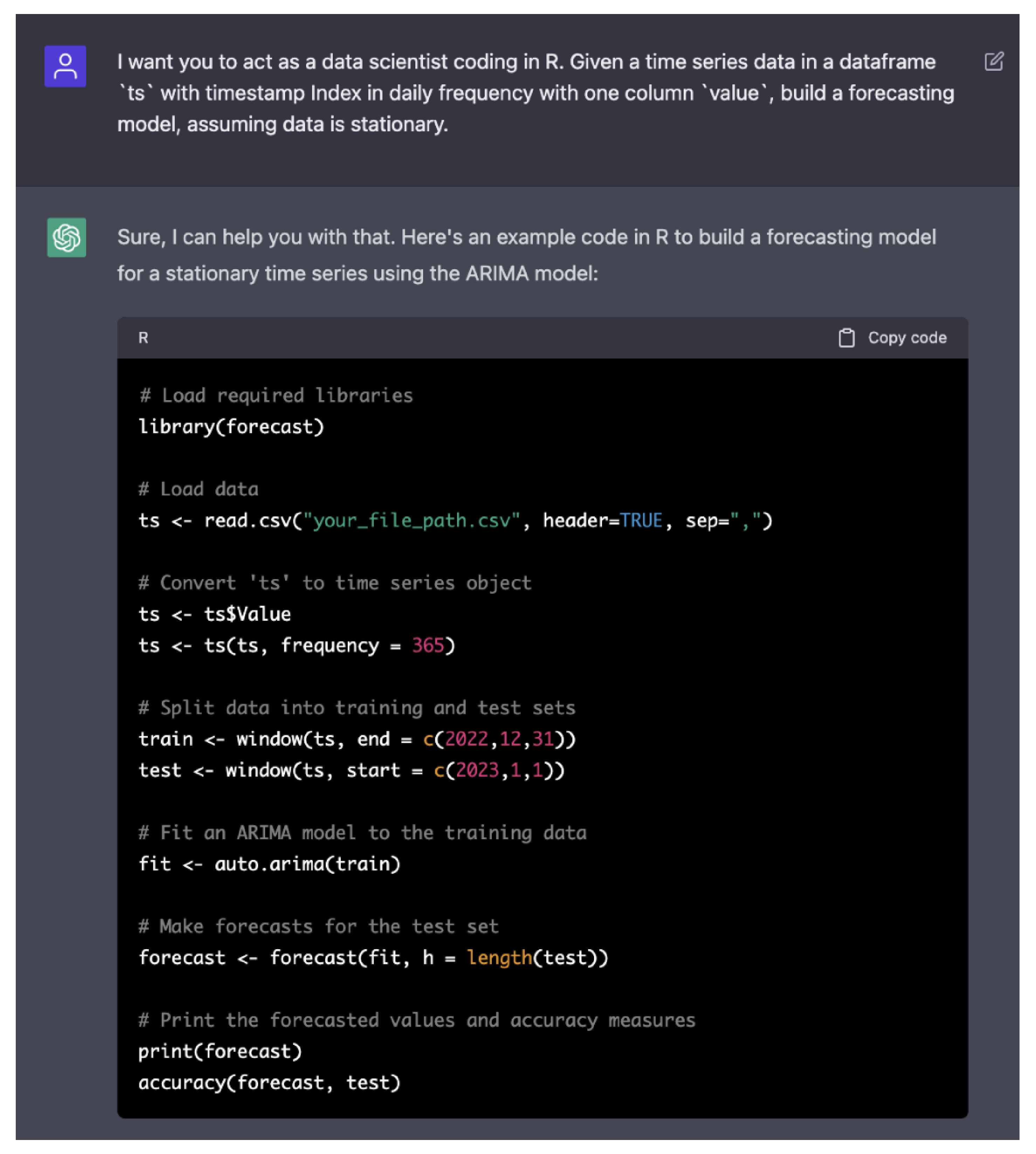
As a data scientist coding in R, here's how you can convert a time series data in a dataframe `ts` with timestamp index in daily frequency to weekly and monthly:

```r
# Load the lubridate package for working with time data
library(lubridate)

# Convert daily frequency to weekly frequency
ts_weekly <- ts %>%
  as.data.frame() %>%
  mutate(date = as.Date(index(ts))) %>%
  group_by(year = year(date), week = week(date)) %>%
  summarise(value = mean(value)) %>%
  ungroup() %>%
  mutate(date = as.Date(paste0(year, "-W", week, "-1"))) %>%
  select(-year, -week) %>%
  column_to_rownames("date") %>%
  ts(frequency = 52)

# Convert daily frequency to monthly frequency
ts_monthly <- ts %>%
  as.data.frame() %>%
  mutate(date = as.Date(index(ts))) %>%
  group_by(year = year(date), month = month(date)) %>%
  summarise(value = mean(value)) %>%
  ungroup() %>%
  mutate(date = as.Date(paste0(year, "-", month, "-1"))) %>%
  select(-year, -month) %>%
  column_to_rownames("date") %>%
  ts(frequency = 12)
```

Build test series model

PROMPT STRUCTURE

I want you to act as a data scientist coding in R. Given a time series data in a dataframe {dataframe name} with timestamp Index in {original frequency} frequency with one column {column name}, build a forecasting model, assuming data is stationary.



I want you to act as a data scientist coding in R. Given a time series data in a dataframe `ts` with timestamp Index in daily frequency with one column `value`, build a forecasting model, assuming data is stationary.

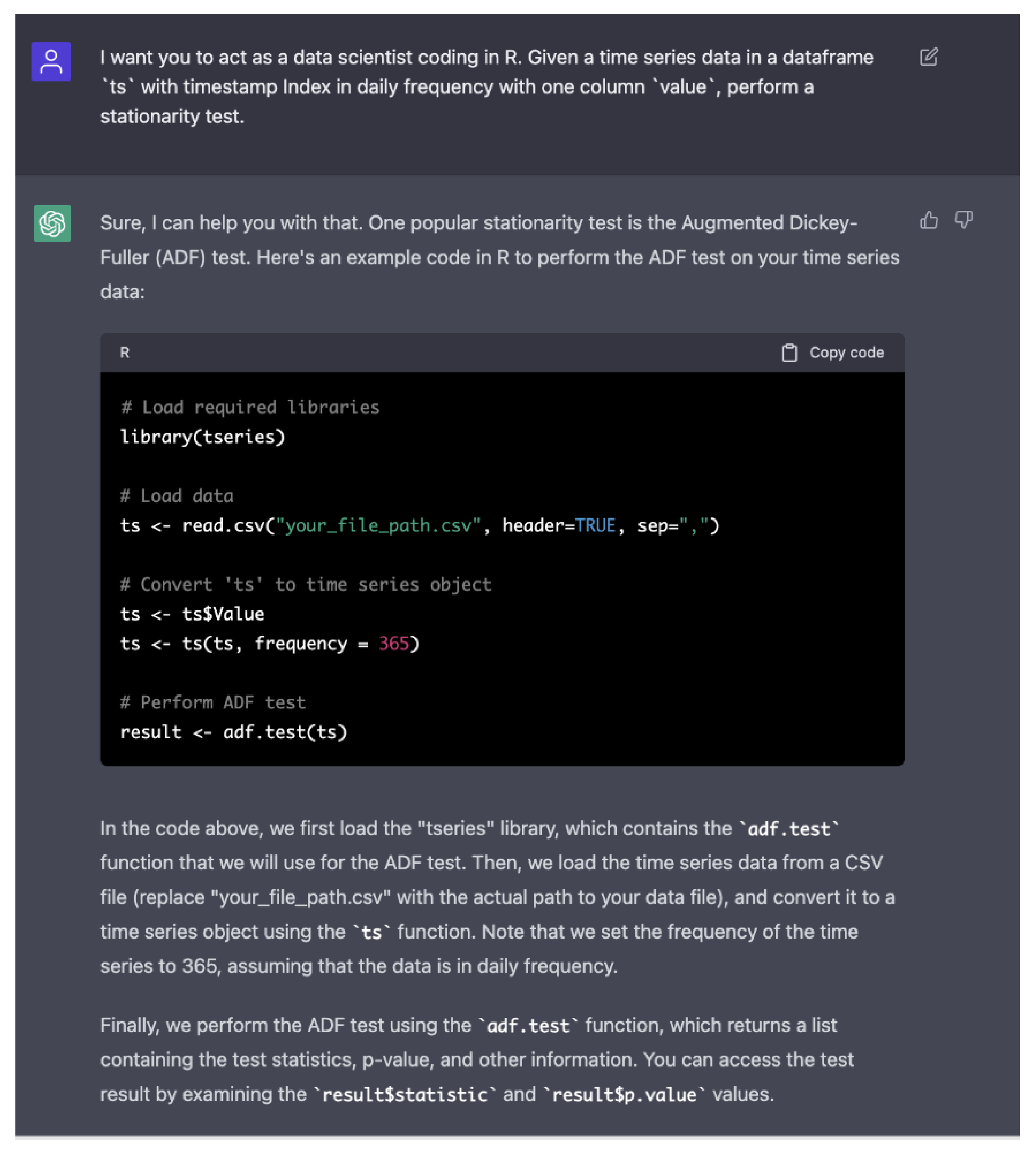
Sure, I can help you with that. Here's an example code in R to build a forecasting model for a stationary time series using the ARIMA model:

```r
# Load required libraries
library(forecast)

# Load data
ts <- read.csv("your_file_path.csv", header=TRUE, sep=",")

# Convert 'ts' to time series object
ts <- ts$Value
ts <- ts(ts, frequency = 365)

# Split data into training and test sets
train <- window(ts, end = c(2022,12,31))
test <- window(ts, start = c(2023,1,1))

# Fit an ARIMA model to the training data
fit <- auto.arima(train)

# Make forecasts for the test set
forecast <- forecast(fit, h = length(test))

# Print the forecasted values and accuracy measures
print(forecast)
accuracy(forecast, test)
```

Perform stationarity test

PROMPT STRUCTURE

I want you to act as a data scientist coding in R. Given a time series data in a dataframe {dataframe name} with timestamp Index in {original frequency} frequency with one column {column name}, perform a Dicky Fuller test.



I want you to act as a data scientist coding in R. Given a time series data in a dataframe `ts` with timestamp Index in daily frequency with one column `value`, perform a stationarity test.

Sure, I can help you with that. One popular stationarity test is the Augmented Dickey-Fuller (ADF) test. Here's an example code in R to perform the ADF test on your time series data:

```r
# Load required libraries
library(tseries)

# Load data
ts <- read.csv("your_file_path.csv", header=TRUE, sep=",")

# Convert 'ts' to time series object
ts <- ts$Value
ts <- ts(ts, frequency = 365)

# Perform ADF test
result <- adf.test(ts)
```

In the code above, we first load the "tseries" library, which contains the `adf.test` function that we will use for the ADF test. Then, we load the time series data from a CSV file (replace "your_file_path.csv" with the actual path to your data file), and convert it to a time series object using the `ts` function. Note that we set the frequency of the time series to 365, assuming that the data is in daily frequency.

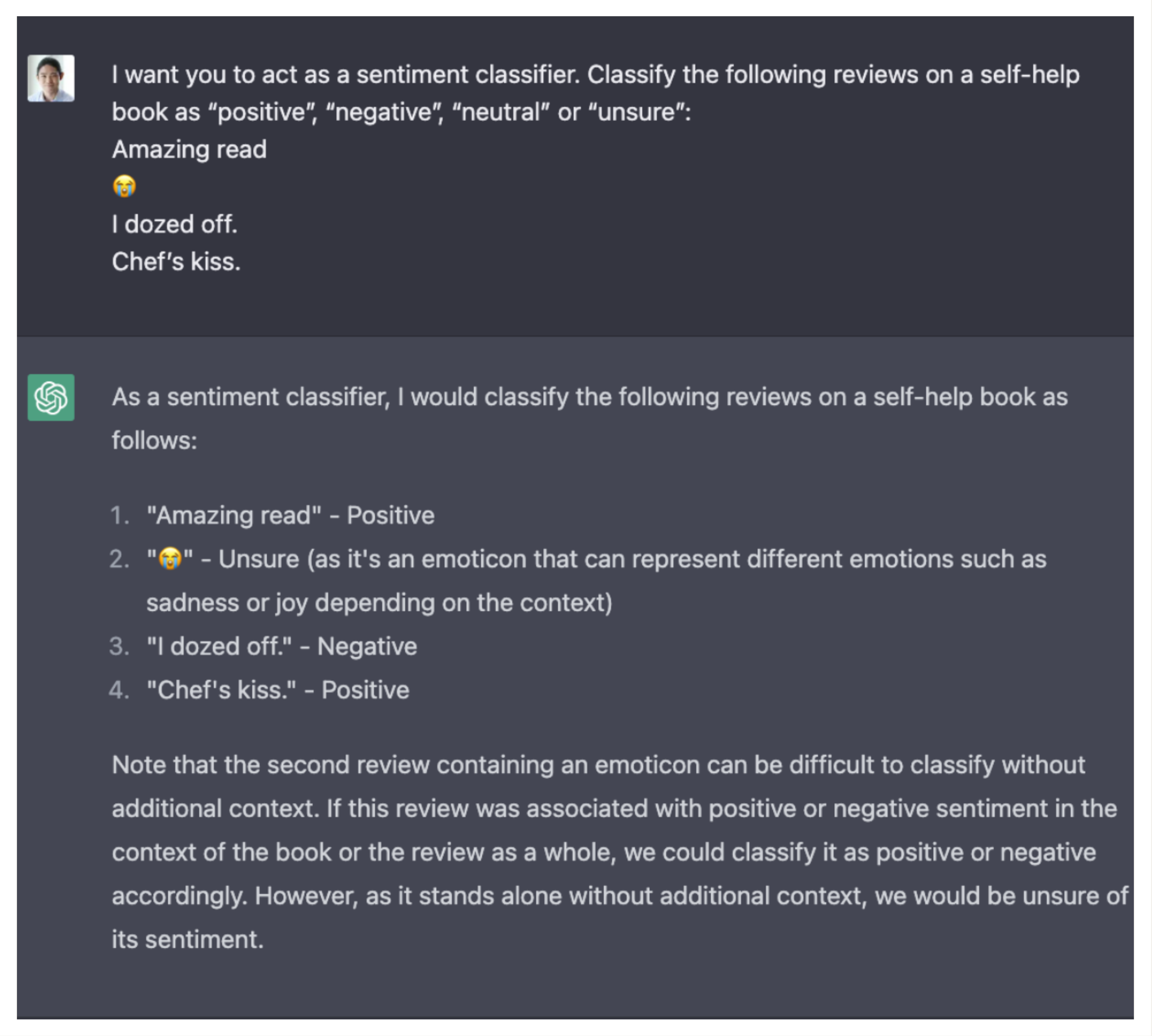
Finally, we perform the ADF test using the `adf.test` function, which returns a list containing the test statistics, p-value, and other information. You can access the test result by examining the `result$statistic` and `result$p.value` values.

Prompts for natural language processing workflows

Classify text sentiment

PROMPT STRUCTURE

I want you to act as a sentiment classifier. Classify the following text which came from {describe text origin} as “positive”, “negative”, “neutral” or “unsure”: {Insert text to be classifier}



I want you to act as a sentiment classifier. Classify the following reviews on a self-help book as “positive”, “negative”, “neutral” or “unsure”:
Amazing read
😭
I dozed off.
Chef’s kiss.

As a sentiment classifier, I would classify the following reviews on a self-help book as follows:

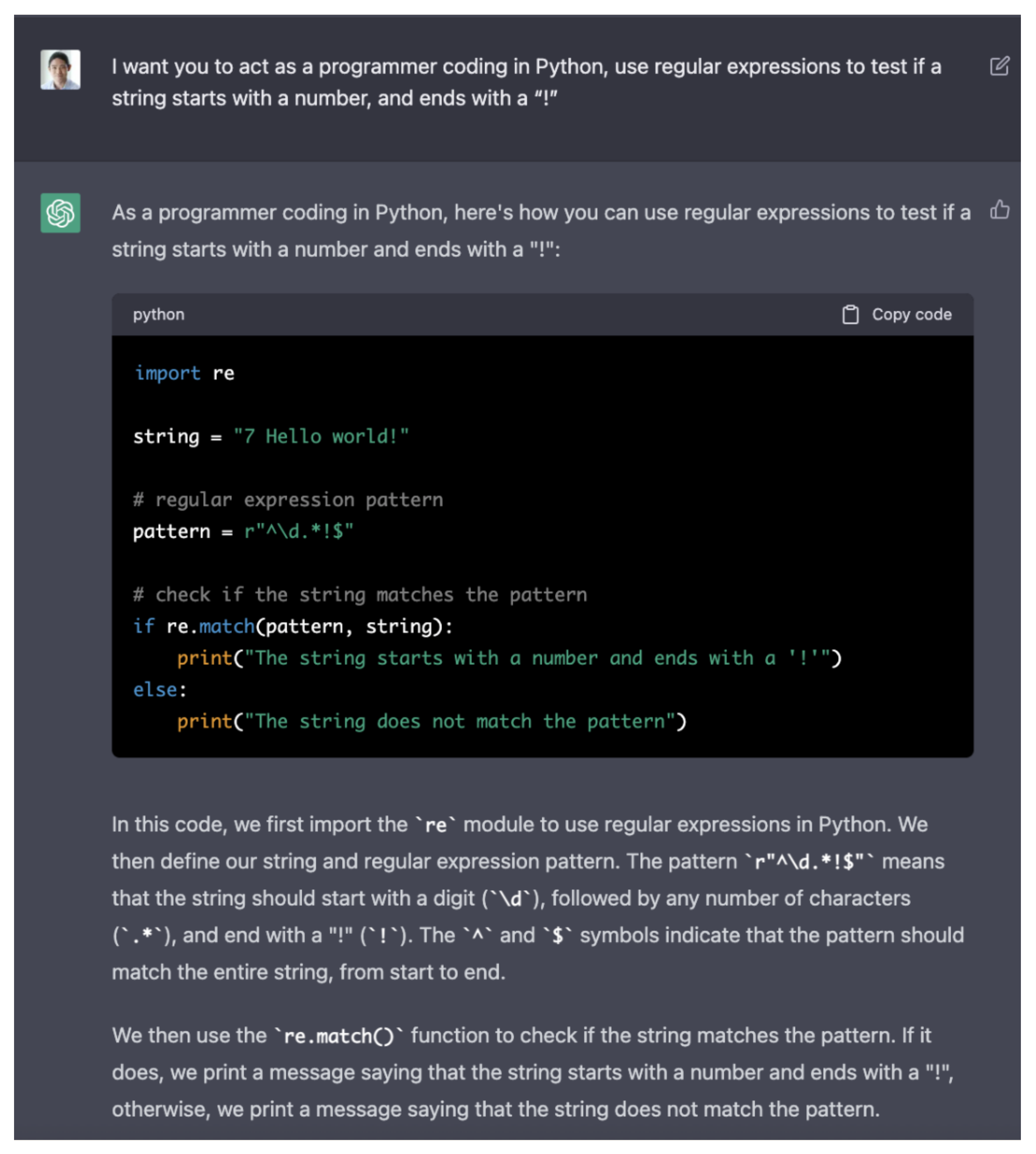
1. "Amazing read" - Positive
2. "😭" - Unsure (as it's an emoticon that can represent different emotions such as sadness or joy depending on the context)
3. "I dozed off." - Negative
4. "Chef's kiss." - Positive

Note that the second review containing an emoticon can be difficult to classify without additional context. If this review was associated with positive or negative sentiment in the context of the book or the review as a whole, we could classify it as positive or negative accordingly. However, as it stands alone without additional context, we would be unsure of its sentiment.

Create regular expressions

PROMPT STRUCTURE

I want you to act as a programmer coding in Python, use regular expressions to test if a string {insert requirements}



I want you to act as a programmer coding in Python, use regular expressions to test if a string starts with a number, and ends with a "!"

As a programmer coding in Python, here's how you can use regular expressions to test if a string starts with a number and ends with a "!":

```python
import re

string = "7 Hello world!"

# regular expression pattern
pattern = r"^\d.*!$"

# check if the string matches the pattern
if re.match(pattern, string):
    print("The string starts with a number and ends with a '!'")
else:
    print("The string does not match the pattern")
```

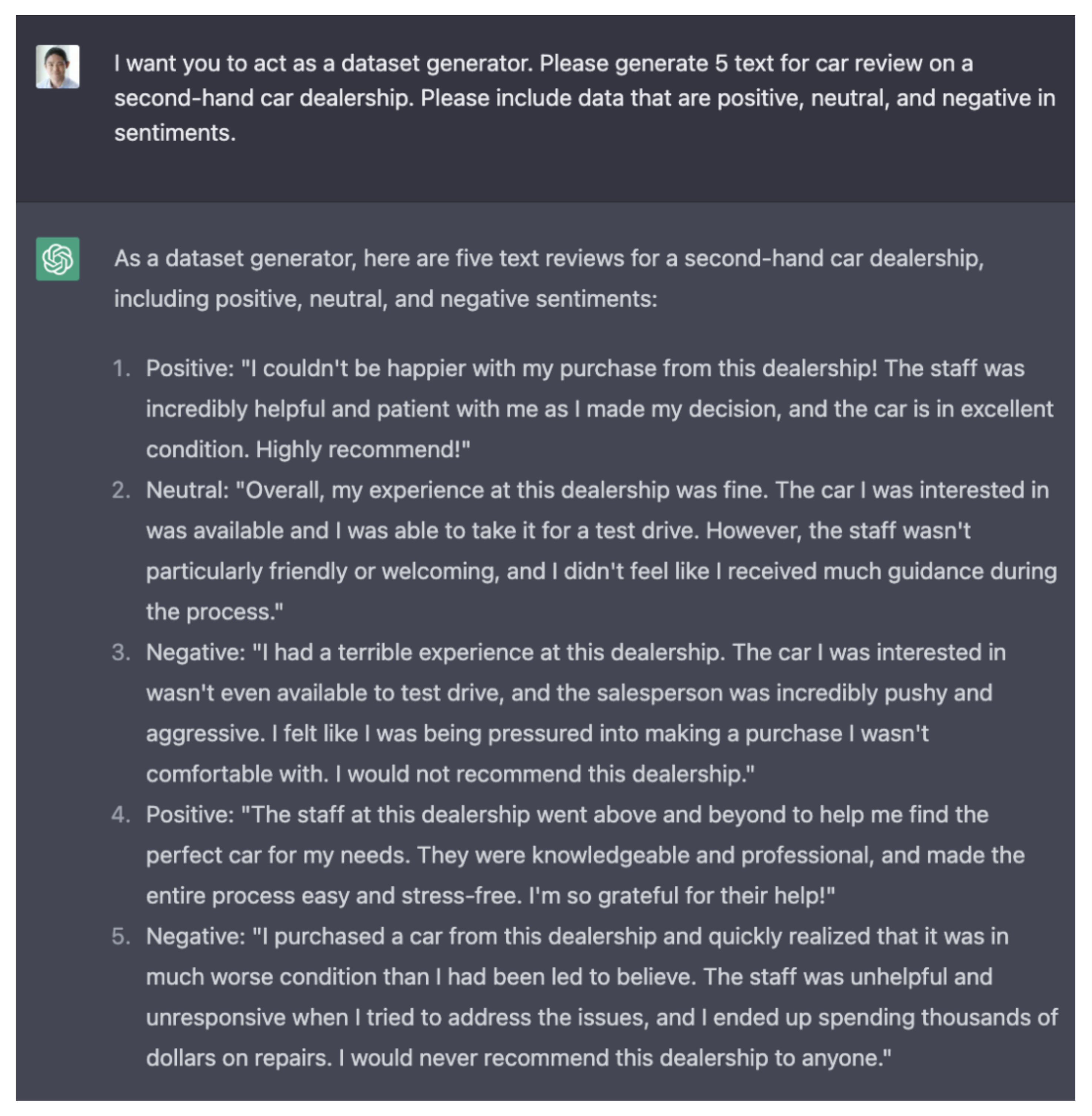
In this code, we first import the `re` module to use regular expressions in Python. We then define our string and regular expression pattern. The pattern `r"^\d.*!$"` means that the string should start with a digit (`\d`), followed by any number of characters (`.*`), and end with a "!" (`!`). The `^` and `$` symbols indicate that the pattern should match the entire string, from start to end.

We then use the `re.match()` function to check if the string matches the pattern. If it does, we print a message saying that the string starts with a number and ends with a "!", otherwise, we print a message saying that the string does not match the pattern.

Text dataset generation

PROMPT STRUCTURE

I want you to act as a dataset generator. Please generate {number of text} texts on {required text and the context}. {Insert additional requirements}



I want you to act as a dataset generator. Please generate 5 text for car review on a second-hand car dealership. Please include data that are positive, neutral, and negative in sentiments.

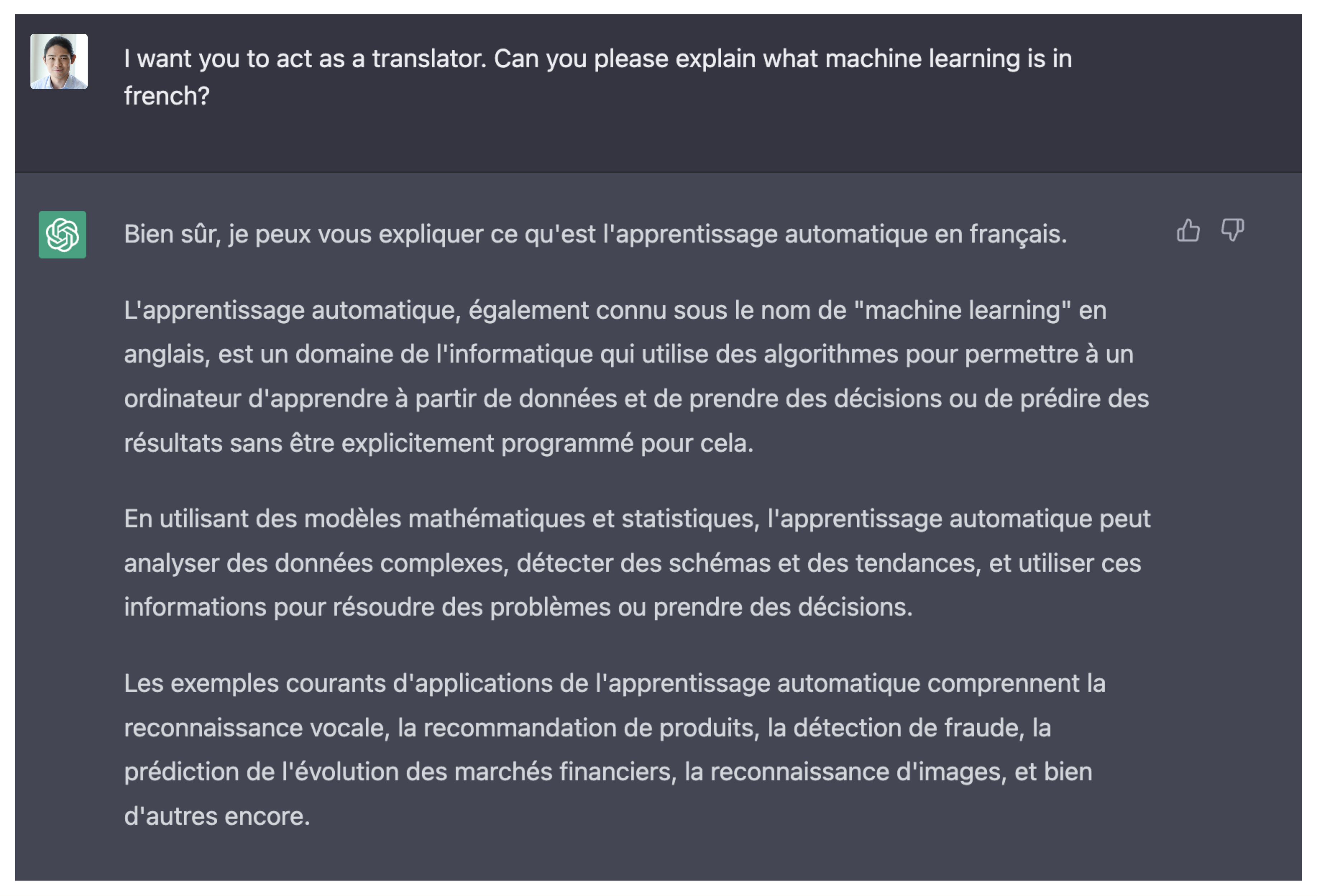
As a dataset generator, here are five text reviews for a second-hand car dealership, including positive, neutral, and negative sentiments:

1. Positive: "I couldn't be happier with my purchase from this dealership! The staff was incredibly helpful and patient with me as I made my decision, and the car is in excellent condition. Highly recommend!"
2. Neutral: "Overall, my experience at this dealership was fine. The car I was interested in was available and I was able to take it for a test drive. However, the staff wasn't particularly friendly or welcoming, and I didn't feel like I received much guidance during the process."
3. Negative: "I had a terrible experience at this dealership. The car I was interested in wasn't even available to test drive, and the salesperson was incredibly pushy and aggressive. I felt like I was being pressured into making a purchase I wasn't comfortable with. I would not recommend this dealership."
4. Positive: "The staff at this dealership went above and beyond to help me find the perfect car for my needs. They were knowledgeable and professional, and made the entire process easy and stress-free. I'm so grateful for their help!"
5. Negative: "I purchased a car from this dealership and quickly realized that it was in much worse condition than I had been led to believe. The staff was unhelpful and unresponsive when I tried to address the issues, and I ended up spending thousands of dollars on repairs. I would never recommend this dealership to anyone."

Machine translation

PROMPT STRUCTURE

I want you to act as a translator. Please translate {phrase} from {origin language} to {translated language}.



I want you to act as a translator. Can you please explain what machine learning is in french?

Bien sûr, je peux vous expliquer ce qu'est l'apprentissage automatique en français.

L'apprentissage automatique, également connu sous le nom de "machine learning" en anglais, est un domaine de l'informatique qui utilise des algorithmes pour permettre à un ordinateur d'apprendre à partir de données et de prendre des décisions ou de prédire des résultats sans être explicitement programmé pour cela.

En utilisant des modèles mathématiques et statistiques, l'apprentissage automatique peut analyser des données complexes, détecter des schémas et des tendances, et utiliser ces informations pour résoudre des problèmes ou prendre des décisions.

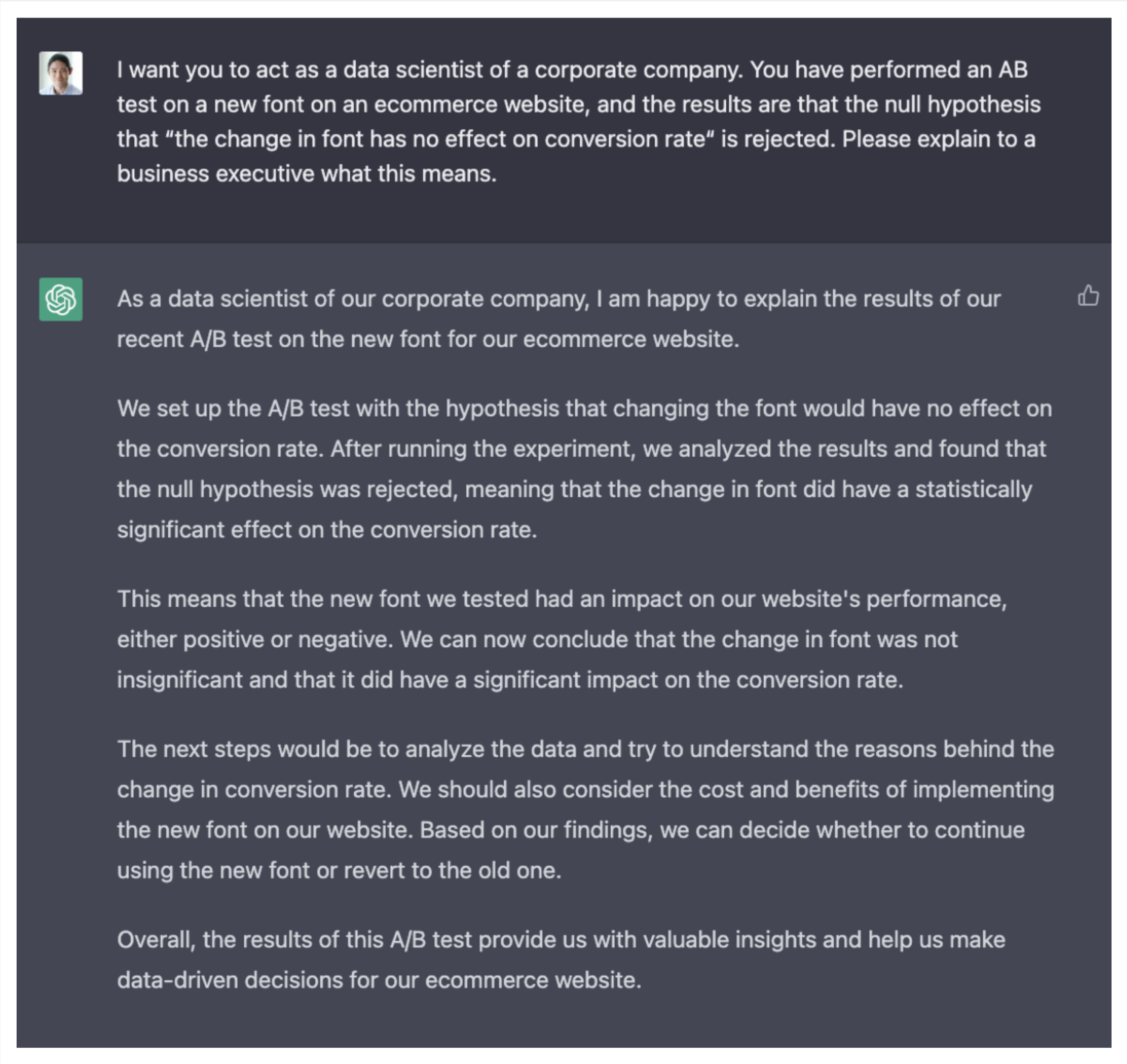
Les exemples courants d'applications de l'apprentissage automatique comprennent la reconnaissance vocale, la recommandation de produits, la détection de fraude, la prédiction de l'évolution des marchés financiers, la reconnaissance d'images, et bien d'autres encore.

Conceptual and career oriented prompts

Explain data concepts for business executives

PROMPT STRUCTURE

I want you to act as a data scientist of a corporate company. {Describe content in detail, if required} Please explain to a business executive what {concept} means.



I want you to act as a data scientist of a corporate company. You have performed an AB test on a new font on an ecommerce website, and the results are that the null hypothesis that “the change in font has no effect on conversion rate“ is rejected. Please explain to a business executive what this means.

As a data scientist of our corporate company, I am happy to explain the results of our recent A/B test on the new font for our ecommerce website.

We set up the A/B test with the hypothesis that changing the font would have no effect on the conversion rate. After running the experiment, we analyzed the results and found that the null hypothesis was rejected, meaning that the change in font did have a statistically significant effect on the conversion rate.

This means that the new font we tested had an impact on our website's performance, either positive or negative. We can now conclude that the change in font was not insignificant and that it did have a significant impact on the conversion rate.

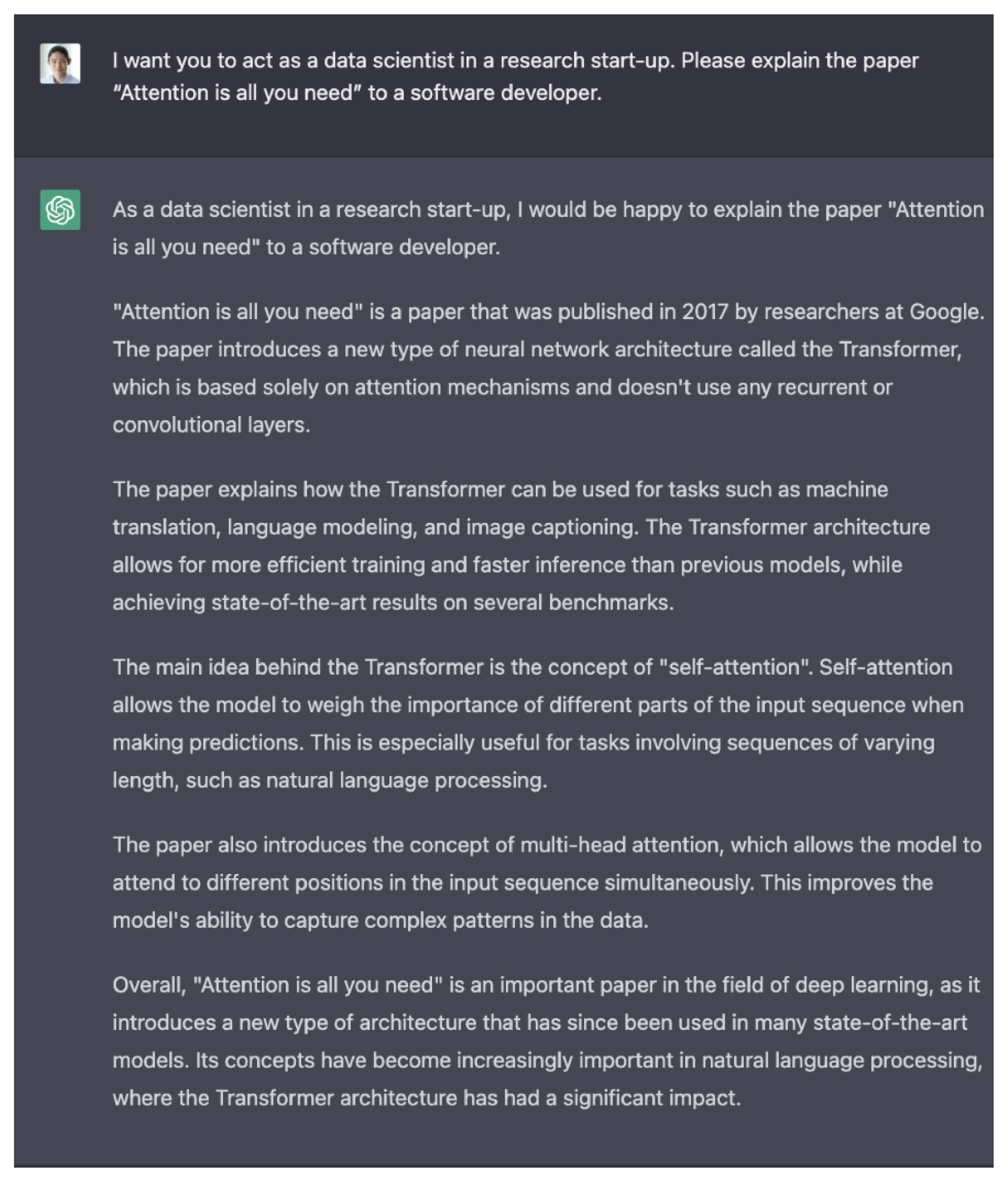
The next steps would be to analyze the data and try to understand the reasons behind the change in conversion rate. We should also consider the cost and benefits of implementing the new font on our website. Based on our findings, we can decide whether to continue using the new font or revert to the old one.

Overall, the results of this A/B test provide us with valuable insights and help us make data-driven decisions for our ecommerce website.

Summarize article/paper

PROMPT STRUCTURE

I want you to act as a data scientist in a research start-up. Please explain the paper {paper} to a {level of difficulty, e.g. software developer, five-year-old, business executive, professor}.



I want you to act as a data scientist in a research start-up. Please explain the paper "Attention is all you need" to a software developer.

As a data scientist in a research start-up, I would be happy to explain the paper "Attention is all you need" to a software developer.

"Attention is all you need" is a paper that was published in 2017 by researchers at Google. The paper introduces a new type of neural network architecture called the Transformer, which is based solely on attention mechanisms and doesn't use any recurrent or convolutional layers.

The paper explains how the Transformer can be used for tasks such as machine translation, language modeling, and image captioning. The Transformer architecture allows for more efficient training and faster inference than previous models, while achieving state-of-the-art results on several benchmarks.

The main idea behind the Transformer is the concept of "self-attention". Self-attention allows the model to weigh the importance of different parts of the input sequence when making predictions. This is especially useful for tasks involving sequences of varying length, such as natural language processing.

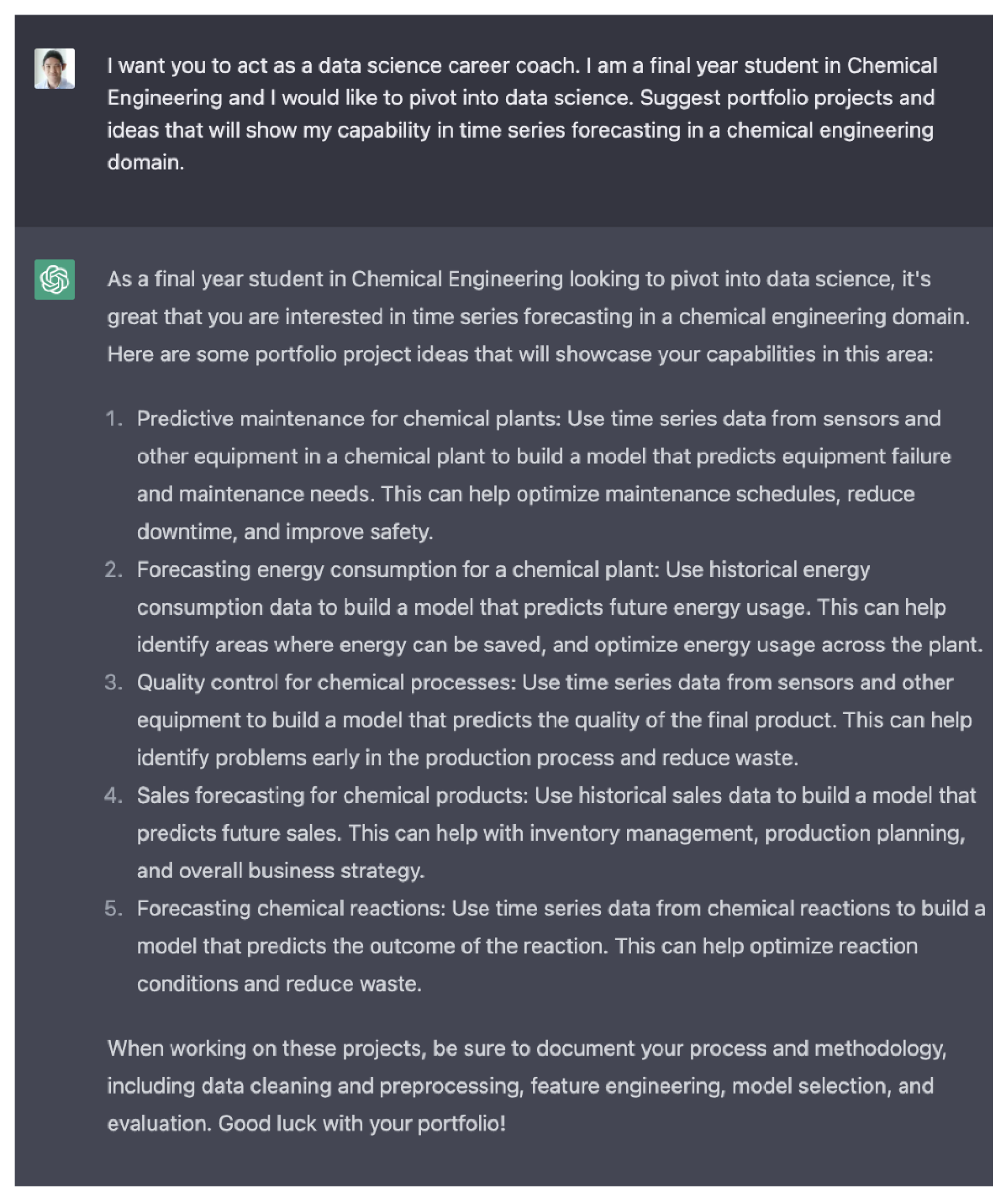
The paper also introduces the concept of multi-head attention, which allows the model to attend to different positions in the input sequence simultaneously. This improves the model's ability to capture complex patterns in the data.

Overall, "Attention is all you need" is an important paper in the field of deep learning, as it introduces a new type of architecture that has since been used in many state-of-the-art models. Its concepts have become increasingly important in natural language processing, where the Transformer architecture has had a significant impact.

Suggest portfolio projects and ideas

PROMPT STRUCTURE

I want you to act as a data science career coach. I am a {describe your background} and I would like to {describe career objective}. Suggest portfolio projects and ideas {describe objective of portfolio}



I want you to act as a data science career coach. I am a final year student in Chemical Engineering and I would like to pivot into data science. Suggest portfolio projects and ideas that will show my capability in time series forecasting in a chemical engineering domain.

As a final year student in Chemical Engineering looking to pivot into data science, it's great that you are interested in time series forecasting in a chemical engineering domain. Here are some portfolio project ideas that will showcase your capabilities in this area:

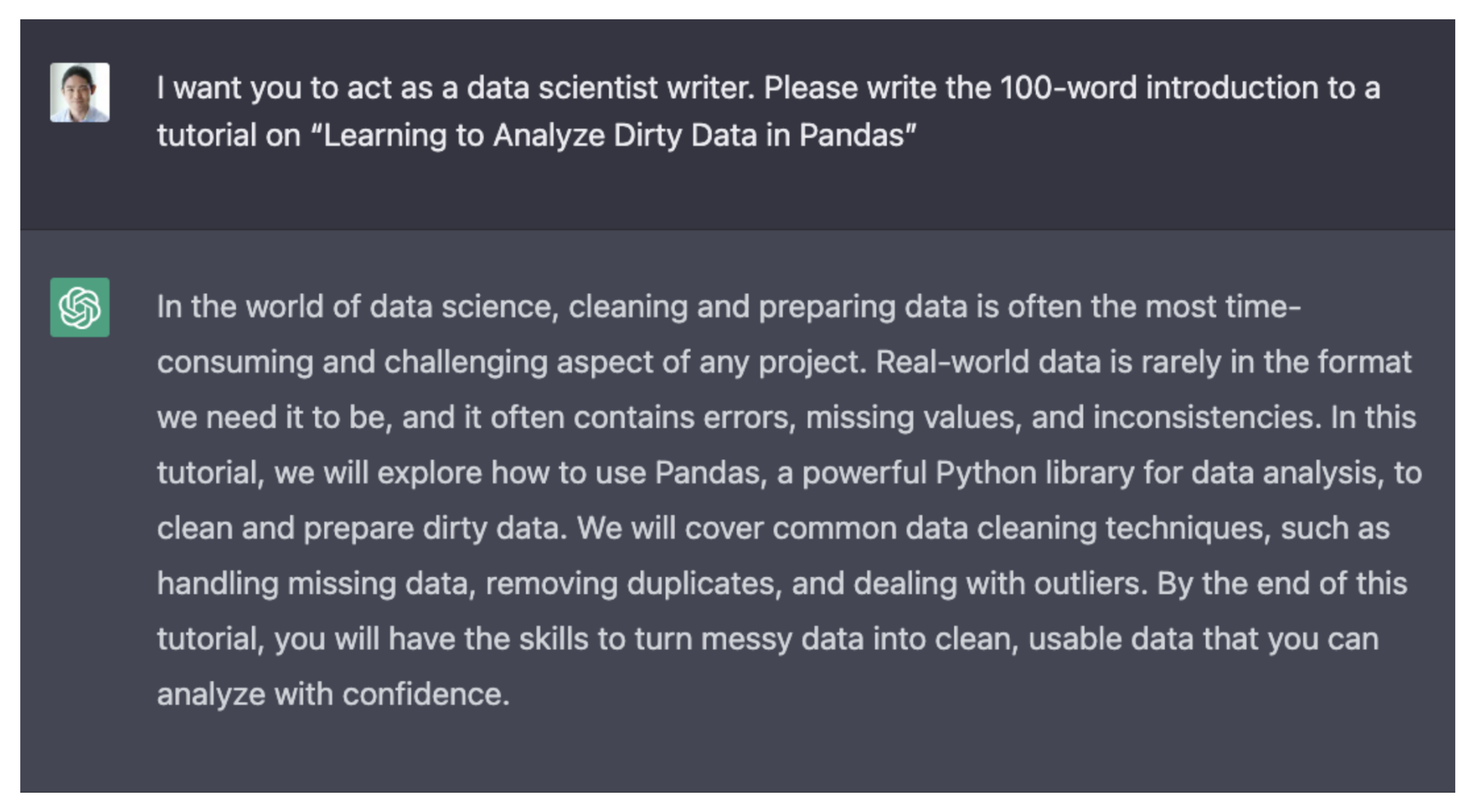
1. Predictive maintenance for chemical plants: Use time series data from sensors and other equipment in a chemical plant to build a model that predicts equipment failure and maintenance needs. This can help optimize maintenance schedules, reduce downtime, and improve safety.
2. Forecasting energy consumption for a chemical plant: Use historical energy consumption data to build a model that predicts future energy usage. This can help identify areas where energy can be saved, and optimize energy usage across the plant.
3. Quality control for chemical processes: Use time series data from sensors and other equipment to build a model that predicts the quality of the final product. This can help identify problems early in the production process and reduce waste.
4. Sales forecasting for chemical products: Use historical sales data to build a model that predicts future sales. This can help with inventory management, production planning, and overall business strategy.
5. Forecasting chemical reactions: Use time series data from chemical reactions to build a model that predicts the outcome of the reaction. This can help optimize reaction conditions and reduce waste.

When working on these projects, be sure to document your process and methodology, including data cleaning and preprocessing, feature engineering, model selection, and evaluation. Good luck with your portfolio!

Write tutorials

PROMPT STRUCTURE

I want you to act as a data scientist writer. Please write the {number-of-words}-word introduction to a tutorial on {title}. {Insert relevant key points}.



I want you to act as a data scientist writer. Please write the 100-word introduction to a tutorial on "Learning to Analyze Dirty Data in Pandas"

In the world of data science, cleaning and preparing data is often the most time-consuming and challenging aspect of any project. Real-world data is rarely in the format we need it to be, and it often contains errors, missing values, and inconsistencies. In this tutorial, we will explore how to use Pandas, a powerful Python library for data analysis, to clean and prepare dirty data. We will cover common data cleaning techniques, such as handling missing data, removing duplicates, and dealing with outliers. By the end of this tutorial, you will have the skills to turn messy data into clean, usable data that you can analyze with confidence.

About this cheat sheet

Author:

Travis Tang

Linkedin

Github

References:

60 ChatGPT prompts for data science

ChatGPT cheat sheet