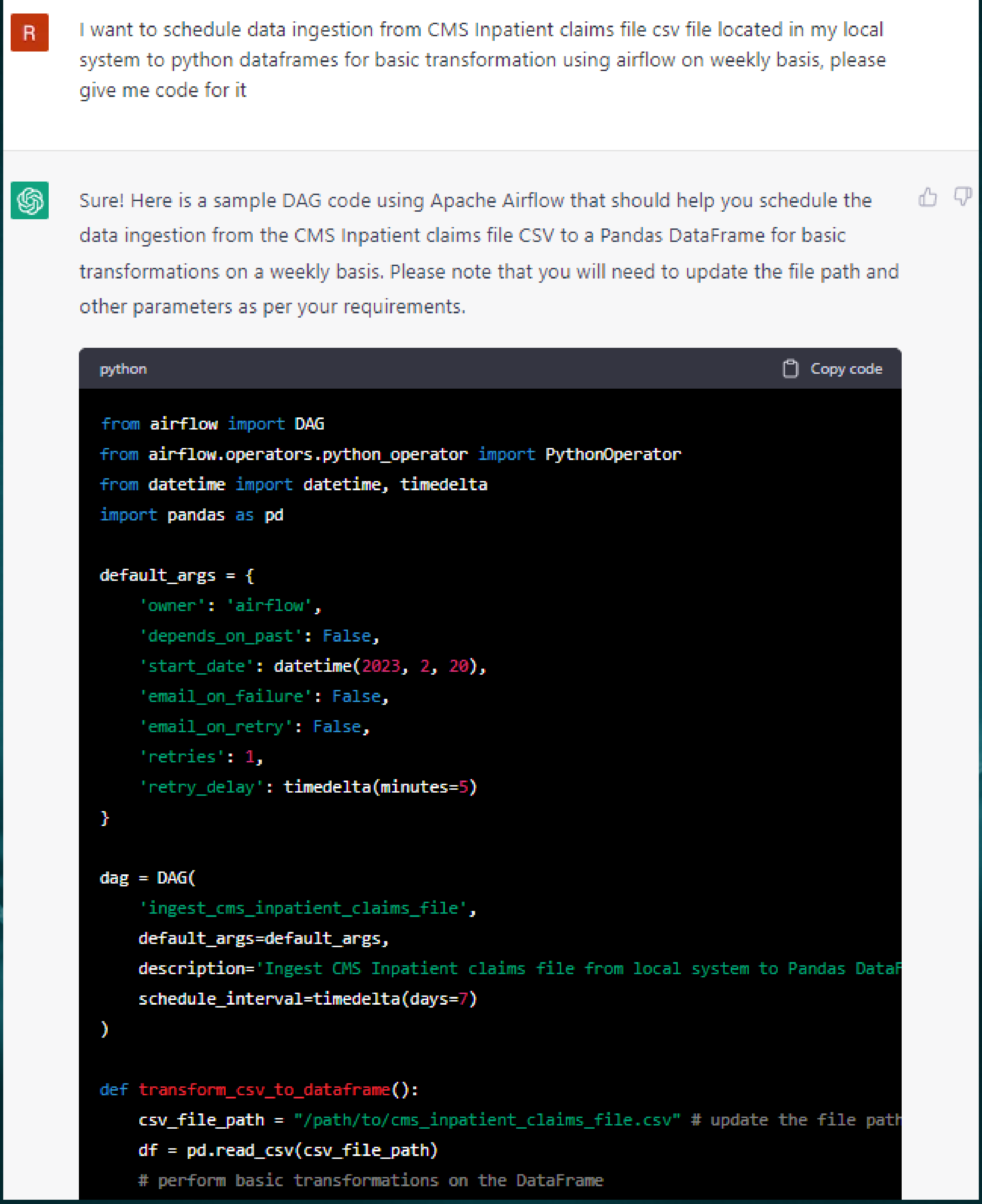
HEALTHARK

Automated Scheduling of Healthcare Data Ingestion with the help of Airflow and ChatGPT

Step: 1.1

Problem Statement: There is a requirement to automate ingesting data from Inpatient Claims file, but with the help of airflow we need to automate this process on weekly basis and store the data in our local database without any manual intervention.

1. Asking ChatGPT to provide code logic for the ingestion of data on weekly basis.



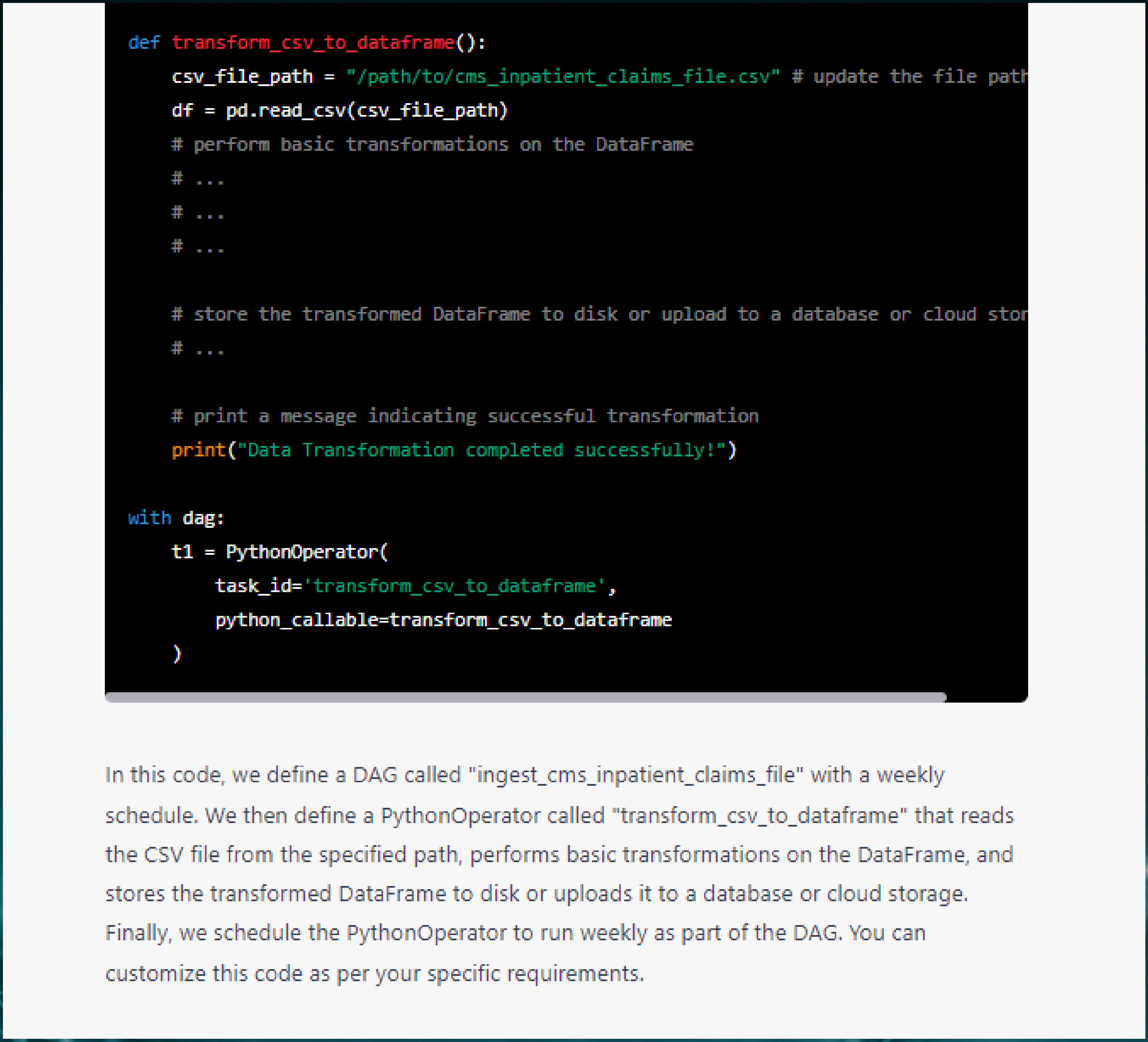
I want to schedule data ingestion from CMS Inpatient claims file csv file located in my local system to python dataframes for basic transformation using airflow on weekly basis, please give me code for it

Sure! Here is a sample DAG code using Apache Airflow that should help you schedule the data ingestion from the CMS Inpatient claims file CSV to a Pandas DataFrame for basic transformations on a weekly basis. Please note that you will need to update the file path and other parameters as per your requirements.

```python
from airflow import DAG
from airflow.operators.python_operator import PythonOperator
from datetime import datetime, timedelta
import pandas as pd

default_args = {
    'owner': 'airflow',
    'depends_on_past': False,
    'start_date': datetime(2023, 2, 20),
    'email_on_failure': False,
    'email_on_retry': False,
    'retries': 1,
    'retry_delay': timedelta(minutes=5)
}

dag = DAG(
    'ingest_cms_inpatient_claims_file',
    default_args=default_args,
    description='Ingest CMS Inpatient claims file from local system to Pandas DataF
    schedule_interval=timedelta(days=7)
)

def transform_csv_to_dataframe():
    csv_file_path = "/path/to/cms_inpatient_claims_file.csv" # update the file path
    df = pd.read_csv(csv_file_path)
    # perform basic transformations on the DataFrame
```

Swipe

Step: 1.2

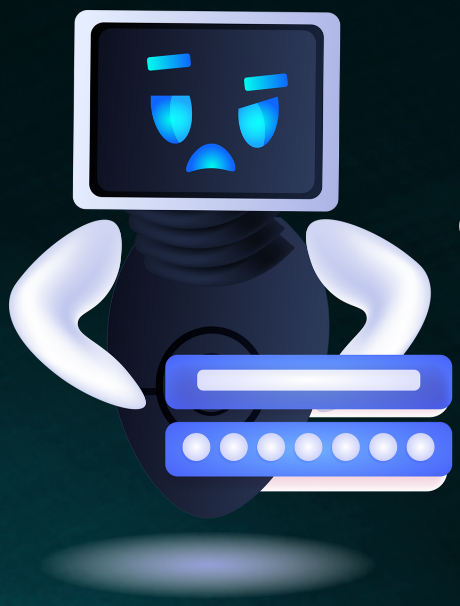


```
def transform_csv_to_dataframe():
    csv_file_path = "/path/to/cms_inpatient_claims_file.csv" # update the file path
    df = pd.read_csv(csv_file_path)
    # perform basic transformations on the DataFrame
    # ...
    # ...
    # ...

    # store the transformed DataFrame to disk or upload to a database or cloud stor
    # ...

    # print a message indicating successful transformation
    print("Data Transformation completed successfully!")

with dag:
    t1 = PythonOperator(
        task_id='transform_csv_to_dataframe',
        python_callable=transform_csv_to_dataframe
    )
```

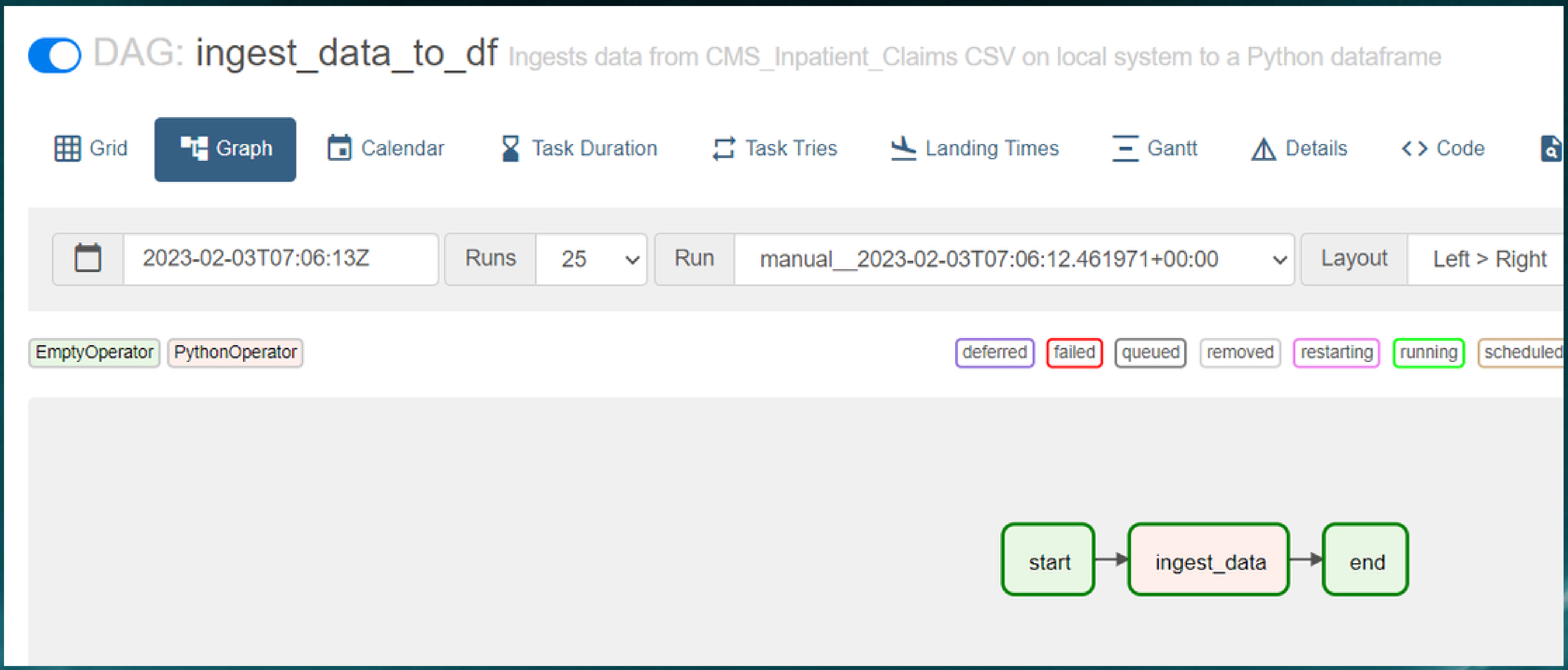
In this code, we define a DAG called "ingest_cms_inpatient_claims_file" with a weekly schedule. We then define a PythonOperator called "transform_csv_to_dataframe" that reads the CSV file from the specified path, performs basic transformations on the DataFrame, and stores the transformed DataFrame to disk or uploads it to a database or cloud storage. Finally, we schedule the PythonOperator to run weekly as part of the DAG. You can customize this code as per your specific requirements.

Swipe

Step: 1.3



Step Implementation Review: The Code Provided by Chat GPT was very accurate except a small omission of datetime library. The flow was running smoothly



DAG: ingest_data_to_df Ingests data from CMS_Inpatient_Claims CSV on local system to a Python dataframe

2023-02-03T07:06:13Z

Runs 25

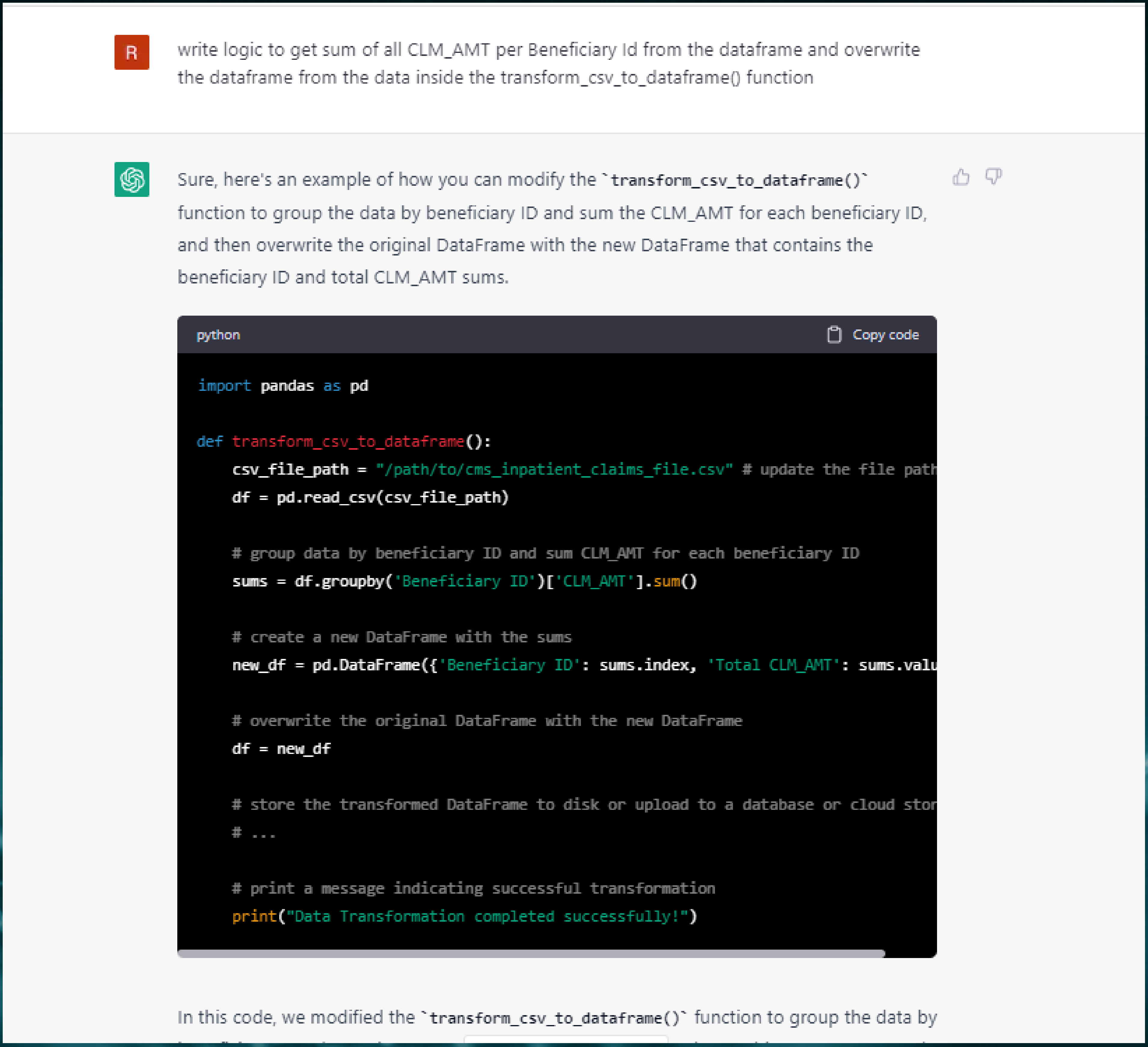
Run manual__2023-02-03T07:06:12.461971+00:00

Layout Left > Right

HEALTHARK

Step: 2.1

2. Asking transformation logic to obtain sum of all claims done by a beneficiary



write logic to get sum of all CLM_AMT per Beneficiary Id from the dataframe and overwrite the dataframe from the data inside the transform_csv_to_dataframe() function

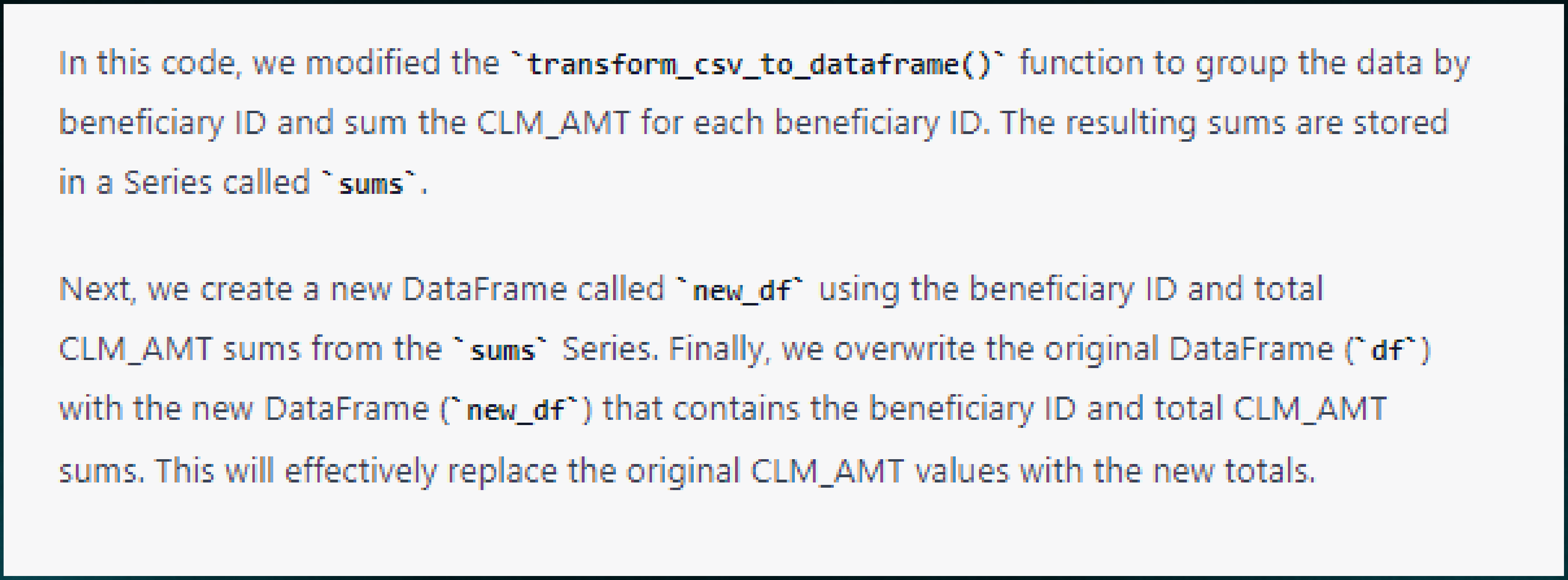
Sure, here's an example of how you can modify the `transform_csv_to_dataframe()` function to group the data by beneficiary ID and sum the CLM_AMT for each beneficiary ID, and then overwrite the original DataFrame with the new DataFrame that contains the beneficiary ID and total CLM_AMT sums.

```python
import pandas as pd

def transform_csv_to_dataframe():
    csv_file_path = "/path/to/cms_inpatient_claims_file.csv" # update the file path
    df = pd.read_csv(csv_file_path)

    # group data by beneficiary ID and sum CLM_AMT for each beneficiary ID
    sums = df.groupby('Beneficiary ID')['CLM_AMT'].sum()

    # create a new DataFrame with the sums
    new_df = pd.DataFrame({'Beneficiary ID': sums.index, 'Total CLM_AMT': sums.valu

    # overwrite the original DataFrame with the new DataFrame
    df = new_df

    # store the transformed DataFrame to disk or upload to a database or cloud stor
    # ...

    # print a message indicating successful transformation
    print("Data Transformation completed successfully!")
```

In this code, we modified the `transform_csv_to_dataframe()` function to group the data by

Swipe

Step: 2.2



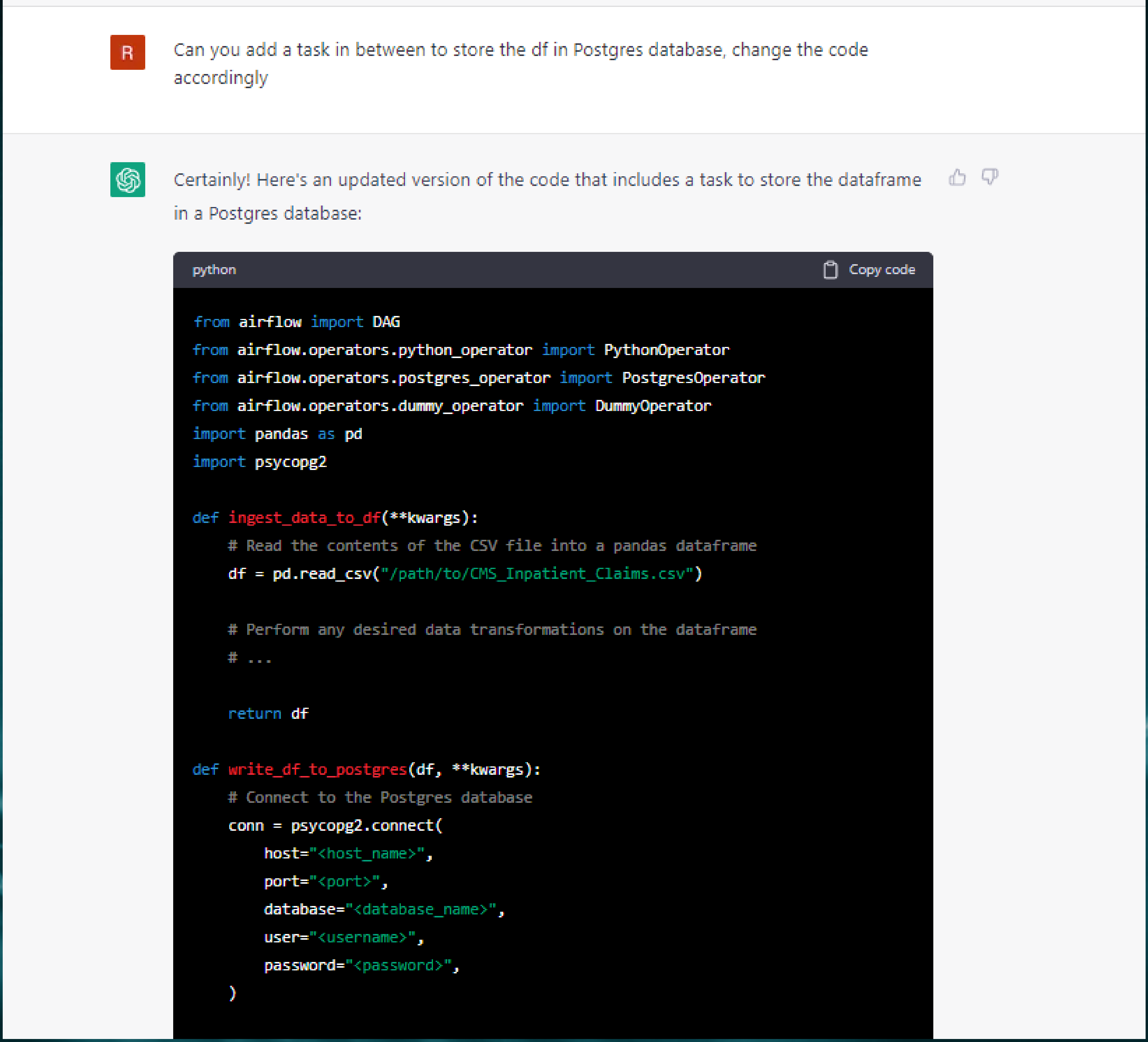
In this code, we modified the `` `transform_csv_to_dataframe()` `` function to group the data by beneficiary ID and sum the CLM_AMT for each beneficiary ID. The resulting sums are stored in a Series called `` `sums` ``.

Next, we create a new DataFrame called `` `new_df` `` using the beneficiary ID and total CLM_AMT sums from the `` `sums` `` Series. Finally, we overwrite the original DataFrame (`` `df` ``) with the new DataFrame (`` `new_df` ``) that contains the beneficiary ID and total CLM_AMT sums. This will effectively replace the original CLM_AMT values with the new totals.

Step Implementation Review: The transformation logic provided was working correctly, we were able to obtain a solid skeleton to implement the requirement

Step: 3.1

3. Now the next step is to get this transformed data in database, let's ask Chat GPT if it can include a task for it in between.



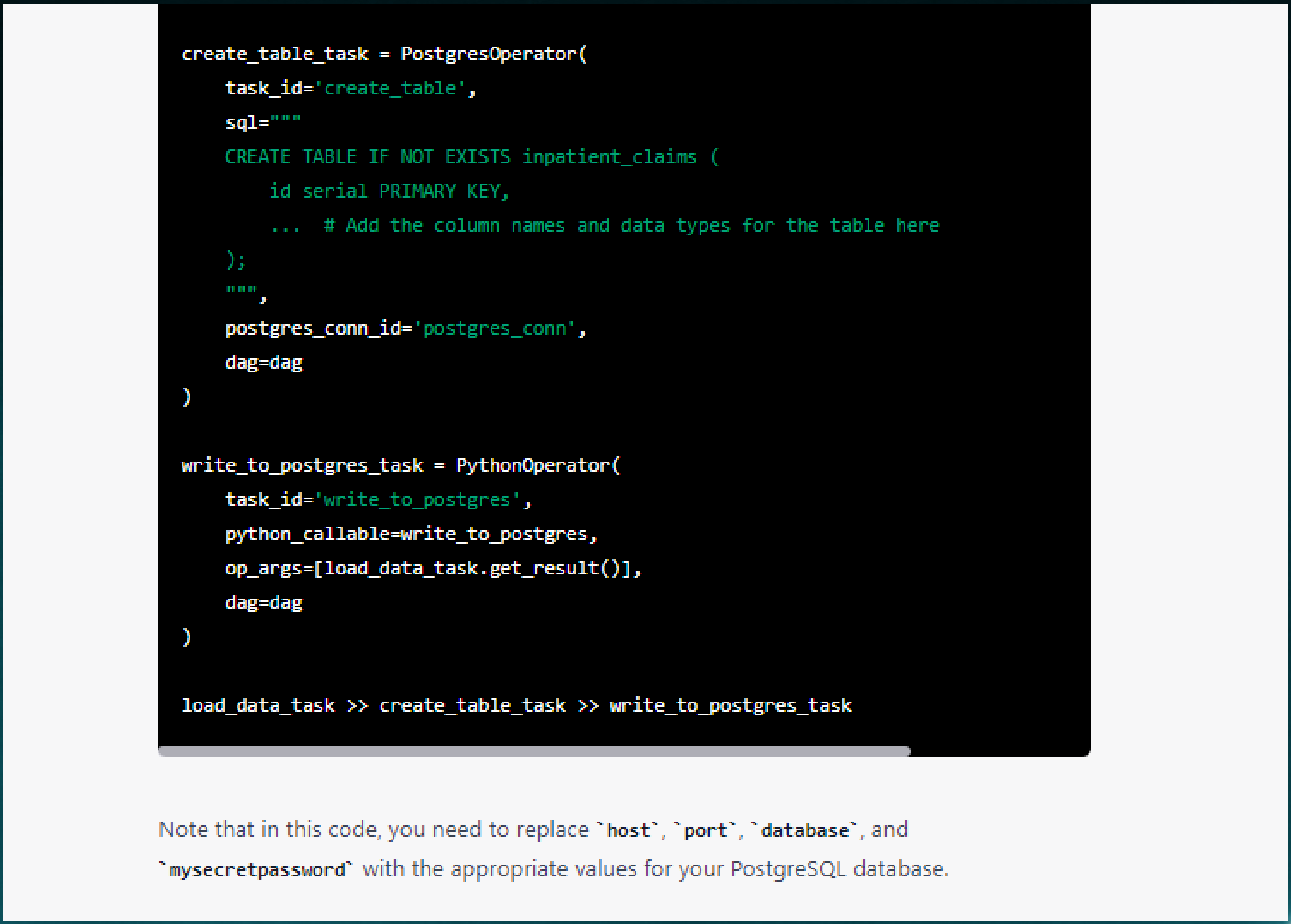
R

Can you add a task in between to store the df in Postgres database, change the code accordingly

Certainly! Here's an updated version of the code that includes a task to store the dataframe in a Postgres database:

```python
from airflow import DAG
from airflow.operators.python_operator import PythonOperator
from airflow.operators.postgres_operator import PostgresOperator
from airflow.operators.dummy_operator import DummyOperator
import pandas as pd
import psycopg2

def ingest_data_to_df(**kwargs):
    # Read the contents of the CSV file into a pandas dataframe
    df = pd.read_csv("/path/to/CMS_Inpatient_Claims.csv")

    # Perform any desired data transformations on the dataframe
    # ...

    return df

def write_df_to_postgres(df, **kwargs):
    # Connect to the Postgres database
    conn = psycopg2.connect(
        host="<host_name>",
        port="<port>",
        database="<database_name>",
        user="<username>",
        password="<password>",
    )
```

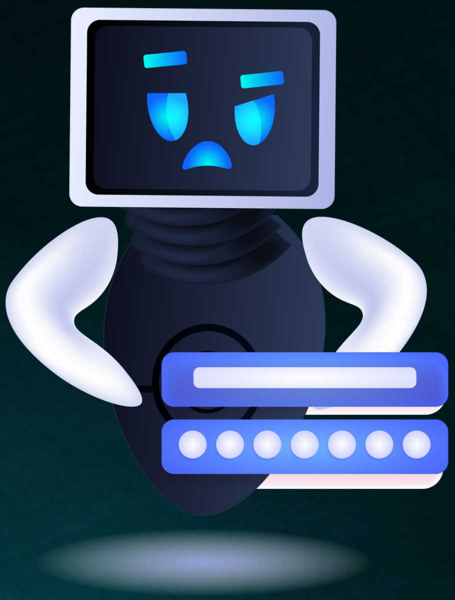
Step: 3.2



```
create_table_task = PostgresOperator(
    task_id='create_table',
    sql="""
    CREATE TABLE IF NOT EXISTS inpatient_claims (
        id serial PRIMARY KEY,
        ...  # Add the column names and data types for the table here
    );
    """,
    postgres_conn_id='postgres_conn',
    dag=dag
)

write_to_postgres_task = PythonOperator(
    task_id='write_to_postgres',
    python_callable=write_to_postgres,
    op_args=[load_data_task.get_result()],
    dag=dag
)

load_data_task >> create_table_task >> write_to_postgres_task
```

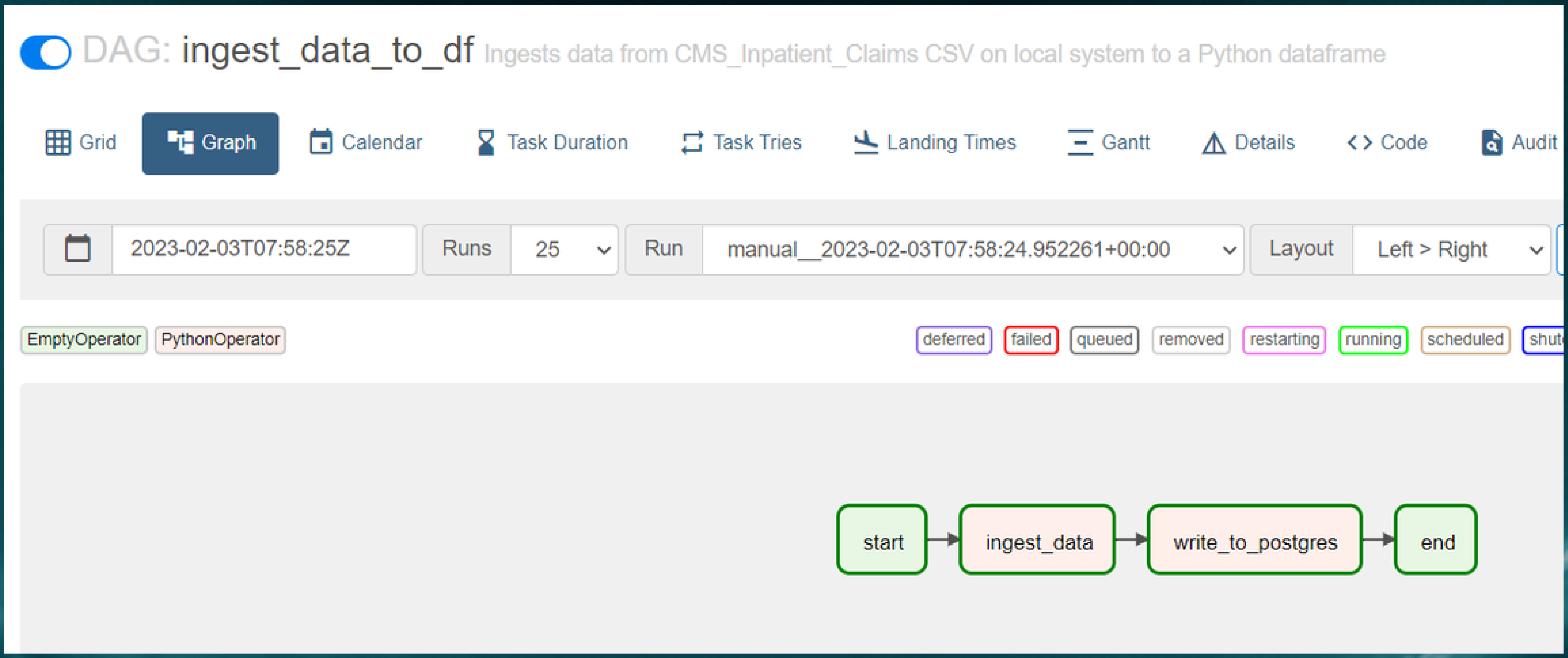
Note that in this code, you need to replace `` `host` ``, `` `port` ``, `` `database` ``, and `` `mysecretpassword` `` with the appropriate values for your PostgreSQL database.

Step: 3.3



Step Implementation Review: ChatGPT was logically able to incorporate the change but it struggled to create a working connection between the database and the code.

After fixing the connection manually the other sections of code worked fine.



DAG: ingest_data_to_df Ingests data from CMS_Inpatient_Claims CSV on local system to a Python dataframe

2023-02-03T07:58:25Z

Runs 25

Run manual__2023-02-03T07:58:24.952261+00:00

Layout Left > Right

Swipe

Conclusion:

With Chat GPT we were able to get a decent head start and save an ample amount of time in writing simple airflow initiation, scheduling and dependency logic.

Apart from some complex requirements Chat GPT was able to deliver the required solution.

ChatGPT efficiency in Airflow:

Efficient

- Scheduling and adding a dependency on tasks
- Initiating and creating DAGs
- Basic transformation logic
- Simple python functions

Partially Efficient

- Creating technical infrastructure links
- Complex transformations
- Transferring data to local database
- Structuring code blocks

Swipe

Authors

Ani Kannal
Head of Data Science and Analytics
Healthark Insights

R M Saivighnesh
Consultant
Healthark Insights

CONTACT US
821, Sun Avenue One
Manekbag, Ambawadi,
Amdedabad, India
contact@healtharkinsights.com
healtharkinsights.com

Copyright © 2022 HEALTHARK. All rights reserved. 02.2023