***Assignment Document:***

1. Connected to mongoDB database using given link and credentials from the document.
2. Connected to PostgreSQL database using given link and credentials from the document.
3. Created Python script for data pipeline to extract data from mongodb to load it to Postgre db.  
   Visual studio code is used to create and deploy python script.

**Python Script:**

import pymongo

import psycopg2

from psycopg2.extras import execute\_values

# MongoDB connection details

MONGO\_URI = "mongodb://etl\_user:aP8fwfgftempRhkgGa9@3.251.75.195:27017/?authSource=sales"

MONGO\_DB = "sales"

MONGO\_COLLECTION = "sales"

# PostgreSQL connection details

POSTGRES\_HOST = "rds-module.cnc6gugkeq4f.eu-west-1.rds.amazonaws.com"

POSTGRES\_DB = "sales\_db3"

POSTGRES\_USER = "yousef"

POSTGRES\_PASSWORD = "yousef"

POSTGRES\_TABLE = "sales"

def fetch\_data\_from\_mongodb():

    """Fetch data from MongoDB collection."""

    client = pymongo.MongoClient(MONGO\_URI)

    db = client[MONGO\_DB]

    collection = db[MONGO\_COLLECTION]

    # Fetch all documents from the MongoDB collection

    data = list(collection.find())

    client.close()

    return data

def transform\_data(data):

    """Transform MongoDB data for PostgreSQL."""

    transformed\_data = []

    for doc in data:

        # Remove MongoDB-specific fields like '\_id' or convert as needed

        transformed\_doc = {

           "event\_time": doc.get("event\_time"),

            "order\_id": doc.get("order\_id"),

            "product\_id": doc.get("product\_id"),

            "category\_id": doc.get("category\_id"),

            "category\_code": doc.get("category\_code"),

            "brand": doc.get("brand"),

            "price": doc.get("price"),

            "user\_id": doc.get("user\_id"),

        }

        transformed\_data.append(tuple(transformed\_doc.values()))

    return transformed\_data

def insert\_data\_into\_postgresql(data):

    """Insert data into PostgreSQL table."""

    connection = psycopg2.connect(

        host=POSTGRES\_HOST,

        database=POSTGRES\_DB,

        user=POSTGRES\_USER,

        password=POSTGRES\_PASSWORD

    )

    cursor = connection.cursor()

    # Define the insert query

    insert\_query = f"""

    INSERT INTO {POSTGRES\_TABLE} (event\_time, order\_id, product\_id,category\_id,category\_code,brand,price,user\_id)

    VALUES %s

    """

#ON CONFLICT (primary\_key\_column) DO NOTHING; -- Adjust as needed

    # Use execute\_values for batch insertion

    execute\_values(cursor, insert\_query, data)

    connection.commit()

    cursor.close()

    connection.close()

def main():

    """Main function to orchestrate the data pipeline."""

    print("Fetching data from MongoDB...")

    mongodb\_data = fetch\_data\_from\_mongodb()

    print("Transforming data...")

    transformed\_data = transform\_data(mongodb\_data)

    print("Inserting data into PostgreSQL...")

    insert\_data\_into\_postgresql(transformed\_data)

    print("Data pipeline completed successfully!")

if \_\_name\_\_ == "\_\_main\_\_":

    main()

1. The extracted data inserted into a raw table.
2. Created the relational table from existing and given fields which insufficient fields but created the table in any way to normalize the database as below

**Raw Table with all fields:**

CREATE TABLE IF NOT EXISTS public.sales

( event\_time timestamp with time zone,

order\_id bigint,

product\_id bigint,

category\_id bigint,

category\_code character varying COLLATE pg\_catalog."default",

brand character varying COLLATE pg\_catalog."default",

price money, user\_id bigint )

**Normalized tables:**

**pSales table**CREATE TABLE public.psales

( sale\_id SERIAL PRIMARY KEY, -- Unique identifier for each sale

event\_time TIMESTAMPTZ NOT NULL,

order\_id BIGINT UNIQUE NOT NULL,

product\_id BIGINT NOT NULL REFERENCES products(product\_id),

user\_id BIGINT NOT NULL REFERENCES users(user\_id),

price MONEY NOT NULL );  
  
**product table:**

CREATE TABLE public.products

( product\_id BIGINT PRIMARY KEY,

category\_id BIGINT NOT NULL REFERENCES categories(category\_id),

brand VARCHAR NOT NULL );  
  
**categories table:**CREATE TABLE public.categories

( category\_id BIGINT PRIMARY KEY,

category\_code VARCHAR UNIQUE NOT NULL );  
  
  
**user table:**CREATE TABLE public.users

( user\_id BIGINT PRIMARY KEY );

1. The data inserted into new tables but there was lot of anomalies in the existing data which highly violate the relationship between tables so I removed few tables relation and primary keys to generate the reports.
2. Below are the queries to generate the final reports.

Query: Total Sales Within a Specified Period

SELECT

SUM(price) AS total\_sales,

COUNT(sale\_id) AS total\_transactions -- Optional: Count the number of sales

FROM

public.psales

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Query for sales per minute:

SELECT

p.brand,

DATE\_TRUNC('minute', s.event\_time) AS sales\_time,

SUM(s.price) AS total\_sales

FROM

public.psales s

JOIN

public.products p ON s.product\_id = p.product\_id

GROUP BY

p.brand, sales\_time

ORDER BY

sales\_time, p.brand;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

SELECT

p.brand,

DATE\_TRUNC('hour', s.event\_time) AS sales\_time,

SUM(s.price) AS total\_sales

FROM

public.psales s

JOIN

public.products p ON s.product\_id = p.product\_id

GROUP BY

p.brand, sales\_time

ORDER BY

sales\_time, p.brand;

1. ERD diagram generated but the relation is removed because of data. Saved in a file which is uploaded.