**REPORT**

**Project Idea 1:**

**Customer Data Management and Analysis**

**Week 1 Report: Data Management and SQL Database Setup**

**1. Overview**

The focus of Week 1 was to design, implement, and manage a SQL database to handle customer data, which includes creating a robust schema, populating it with relevant data, and writing SQL queries for data extraction, updates, and analysis. This process involved utilizing Microsoft SQL Server and SQL Management Studio as the primary tools.

**2. Tasks**

**2.1 Database Design**

The first step was to design a SQL database schema that effectively captures and organizes customer-related data. The database schema was designed with the following key tables:

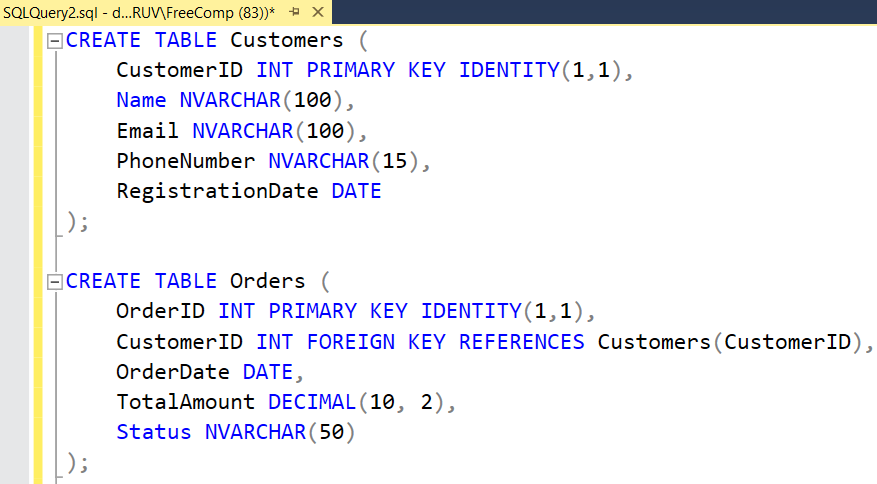
* **Customers**: Stores information about the customers, including CustomerID, Name, Email, PhoneNumber, and RegistrationDate.
* **Orders**: Holds transaction details such as OrderID, CustomerID, OrderDate, TotalAmount, and Status.
* **OrderDetails**: Includes individual product information within an order, including OrderDetailID, OrderID, ProductID, Quantity, and UnitPrice.
* **Products**: Stores product details like ProductID, ProductName, Category, Price, and StockQuantity.
* **Interactions**: Tracks customer interactions with the business, containing fields like InteractionID, CustomerID, InteractionType, InteractionDate, and Notes.

This schema ensures normalized data and enables efficient querying of customer activities, order transactions, and interactions.

**2.2 Database Implementation**

Once the design was finalized, the database schema was implemented in Microsoft SQL Server. The schema was created using SQL scripts that defined each table with their respective columns, data types, and constraints (e.g., primary keys, foreign keys). The database was populated with sample data for testing and further analysis.

**Example SQL Script:**



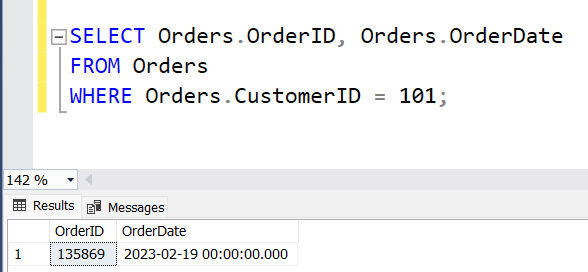
Sample data was inserted into each table, ensuring the database was ready for queries.

**2.3 SQL Queries**

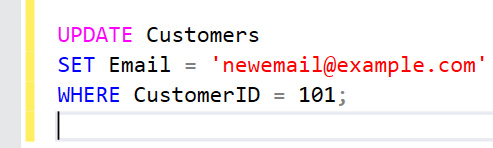
After the database was populated, SQL queries were written to perform essential operations such as data extraction, updates, and basic analysis.

**Key Queries:**

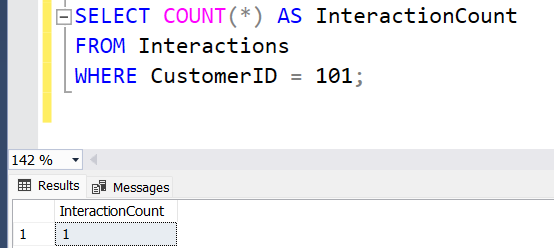
* **Extracting Customer Orders**: Retrieve all orders made by a specific customer.



* **Updating Customer Information**: Update a customer's email address.



* **Analyzing Customer Interactions**: Count the number of interactions a customer had with the business.



These queries provided valuable insights into customer behavior, transactions, and interactions with the business.

**3. Tools**

* **Microsoft SQL Server**: Used for creating and managing the database, writing SQL queries, and running analysis.
* **SQL Management Studio**: Provided a user-friendly interface to interact with the SQL Server database, run queries, and perform updates.

**4. Deliverables**

* **Well-Designed SQL Database Schema**: The database schema effectively captures customer, order, product, and interaction data.
* **Populated Database**: Sample data was inserted into the database tables for testing and analysis purposes.
* **SQL Queries for Data Extraction and Analysis**: Queries were developed to extract customer data, update records, and perform basic analysis of interactions and transactions.

**5. Conclusion**

Week 1 successfully laid the foundation for managing customer data by designing, implementing, and populating a SQL database. The queries developed allow for efficient data retrieval, updates, and basic customer analysis, forming a strong basis for further data warehousing and analysis in the subsequent weeks.

**Week 2 Report: Data Warehousing and Python Programming**

**1. Overview**

In Week 2, the focus was on implementing a data warehouse to aggregate and manage large volumes of customer data for analytics, followed by developing Python scripts for data extraction and preparation. The integration of multiple data sources into the data warehouse facilitated comprehensive data management and enabled further analysis. Microsoft SQL Data Warehouse was used for data warehousing, while Python, leveraging Pandas and SQLAlchemy, was employed for scripting.

**2. Tasks**

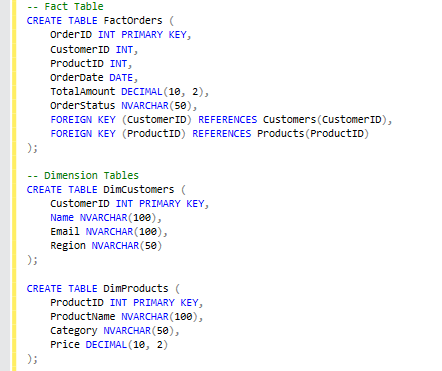
**2.1 Data Warehouse Implementation**

The first task was to design and implement a SQL Data Warehouse to aggregate customer data from multiple sources for advanced analytical purposes. The data warehouse was created to support optimized data storage and retrieval, ensuring that large volumes of data could be processed efficiently.

* **Star Schema:** The data warehouse was structured using a star schema model, which simplifies querying by separating facts from dimensions.
  + **Fact Table:** FactOrders contains transactional data such as OrderID, CustomerID, OrderDate, TotalAmount, and OrderStatus.
  + **Dimension Tables:**
    - **DimCustomers:** Holds customer details such as CustomerID, Name, Email, and Region.
    - **DimProducts:** Contains product details like ProductID, ProductName, Category, and Price.
    - **DimTime:** Stores information on the OrderDate, breaking it down into Year, Month, and Day.

This structure ensured that the large volume of customer data was efficiently stored and could be easily queried for reporting and analysis purposes.

**Example SQL Script for Fact and Dimension Tables:**

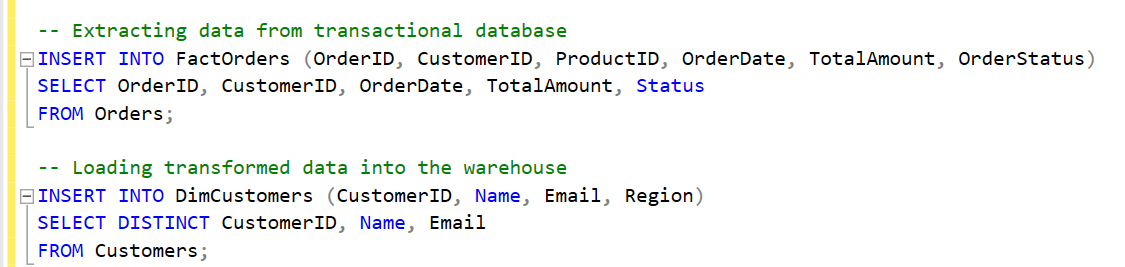
****

**2.2 Data Integration**

The next step was to load data from multiple sources into the data warehouse. Data from the transactional database (developed in Week 1) and other external sources (e.g., marketing platforms, CRM systems) were integrated. This required an ETL (Extract, Transform, Load) process, in which:

* **Extraction:** Data was pulled from the original sources, such as the transactional SQL database and CSV files.
* **Transformation:** The data was cleaned, normalized, and transformed into a consistent format for storage in the data warehouse.
* **Loading:** The transformed data was then loaded into the respective fact and dimension tables.

**Example ETL Script (SQL):**

****

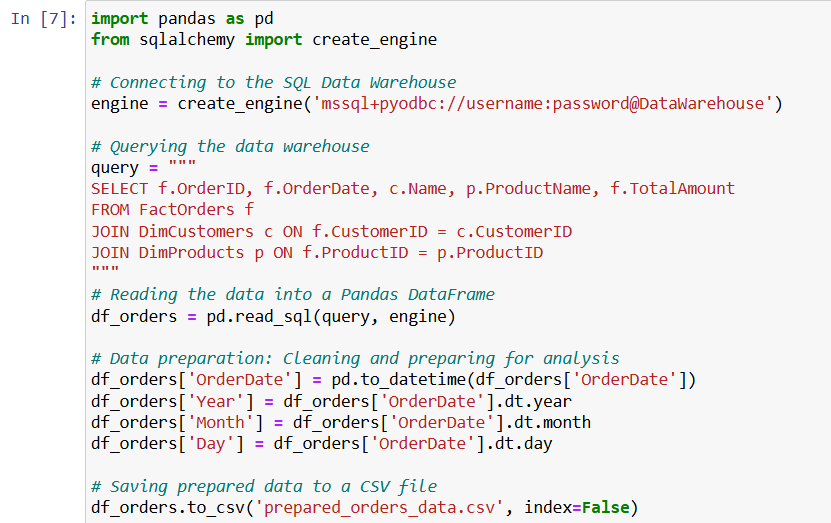
**2.3 Python Programming**

Once the data warehouse was populated, Python scripts were developed to interact with the SQL Data Warehouse and automate the **data extraction and preparation** process for analysis.

Key Python libraries used:

* **Pandas**: For data manipulation and analysis.
* **SQLAlchemy**: For connecting to the SQL Data Warehouse and executing queries.
* **PyODBC**: For establishing connections to the database.

**Python Script Example**:



This script connects to the data warehouse, extracts data, and performs basic data preparation tasks, such as cleaning and extracting date components. The prepared data is then stored in a CSV file for further analysis.

**3. Tools**

* **Microsoft SQL Data Warehouse**: Used to create and manage the data warehouse, facilitating data aggregation from various sources.
* **Python (Pandas, SQLAlchemy)**: Employed for scripting to extract data from the SQL Data Warehouse and perform data preparation tasks.

**4. Deliverables**

* **Functioning SQL Data Warehouse**: A fully functional data warehouse was built, which aggregates customer-related data from various sources and organizes it into a star schema for efficient analytics.
* **Python Scripts for Data Extraction and Preparation**: Python scripts were developed for automated data extraction from the data warehouse and for performing initial data cleaning and preparation tasks.

**5. Conclusion**

Week 2 successfully delivered a well-structured data warehouse that aggregates customer data from multiple sources, and Python scripts that facilitate data extraction and preparation. These foundational steps enable further analytical and modeling activities, to be carried out in Week 3.

**Week 3 Report: Data Science and Azure Integration**

**1. Overview**

In Week 3, the primary focus was on data analysis and machine learning model development for predictive analytics, specifically forecasting sales. Python was utilized for building models, while Azure Data services were integrated for managing and analyzing customer data. The goal was to develop and evaluate predictive models, using Azure Machine Learning for deployment and experimentation.

**2. Tasks**

**2.1 Data Science with Python**

The first task involved performing **data analysis and building predictive models** in Python. The goal was to analyze historical customer and order data, explore patterns, and forecast future sales using machine learning techniques.

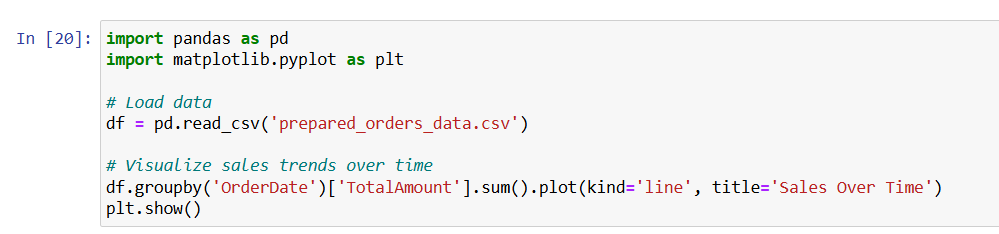
**Steps Taken**:

* **Data Exploration**: The historical sales data from the SQL Data Warehouse was imported and explored to understand trends, correlations, and seasonality in sales patterns.

Key tasks included:

* + Checking for missing data and outliers.
  + Visualizing sales trends over time to detect seasonality and growth patterns.
  + Calculating key statistics (mean, median, variance) on sales data.

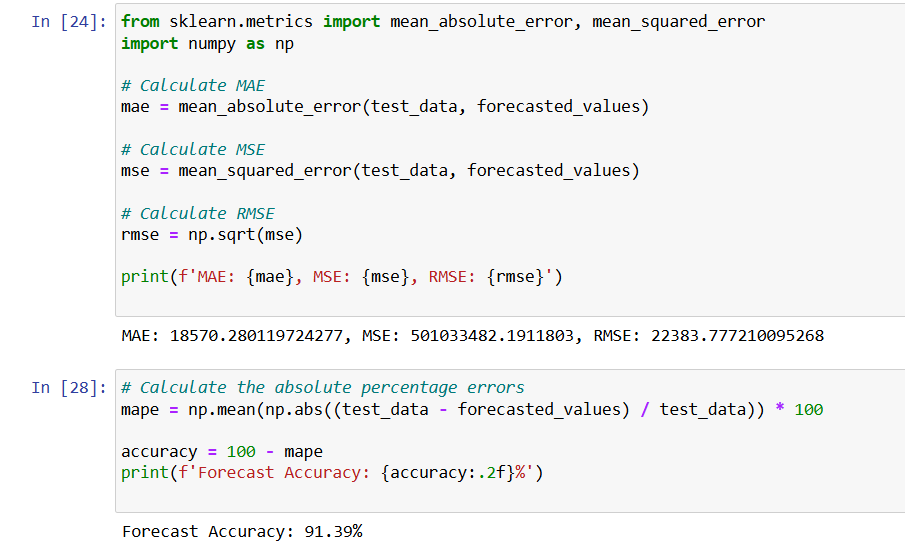
**Example Data Exploration Code**:



**. Model Building**: A machine learning model was developed to forecast future sales using historical data. For this, **Scikit-learn** was used to create a regression model.

* The target variable was TotalAmount (sales amount), while features included OrderDate components (year, month, day) and customer demographics (region, customer segments).
* Models like **Linear Regression** and **Random Forest Regressor** were tested.

**Example Predictive Model Code**:

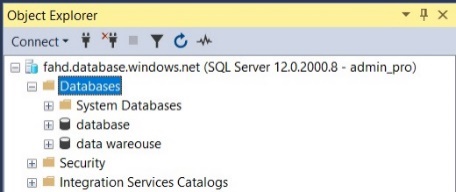


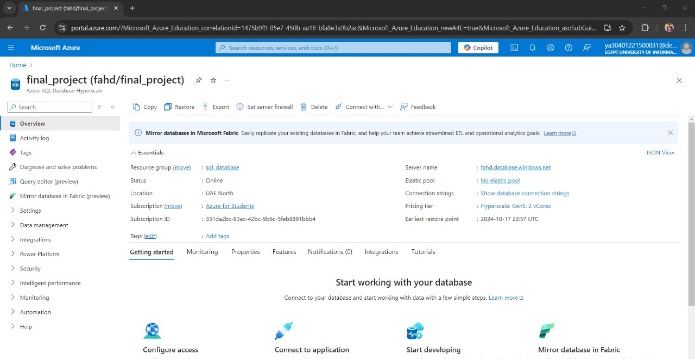
**. Evaluation**: The models were evaluated based on performance metrics like **Mean Squared Error (MSE)** and **R-squared** to determine the best model for sales forecasting.

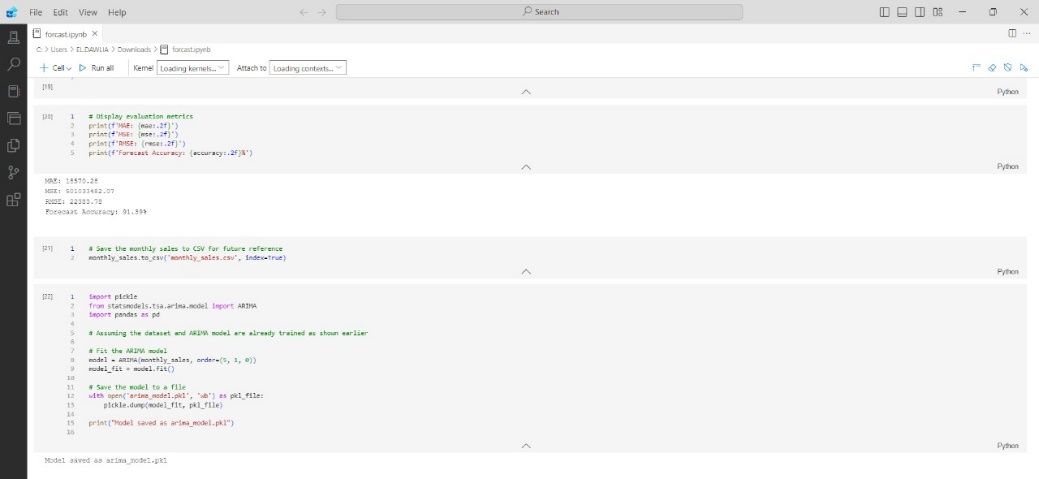
**2.2 Azure Data Fundamentals**

The next task involved setting up **Azure Data Services** to manage and analyze customer data at scale.

* **Azure Data Studio** was used to connect to the SQL Data Warehouse and manage the data in an integrated environment. Queries were run to pull data required for model training and analysis.
* **Azure Blob Storage** was used to store large datasets that could be accessed by machine learning models and pipelines.
* **Azure SQL Database**: Data was uploaded to Azure SQL Database for remote access and management, allowing seamless integration between data storage and machine learning models.





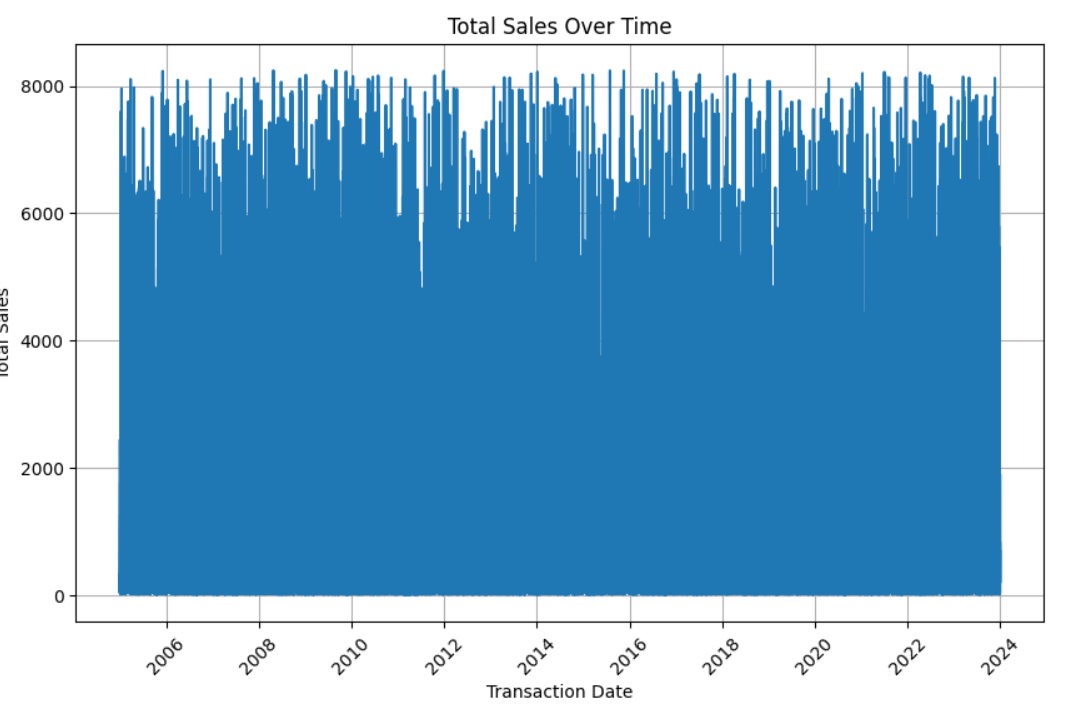
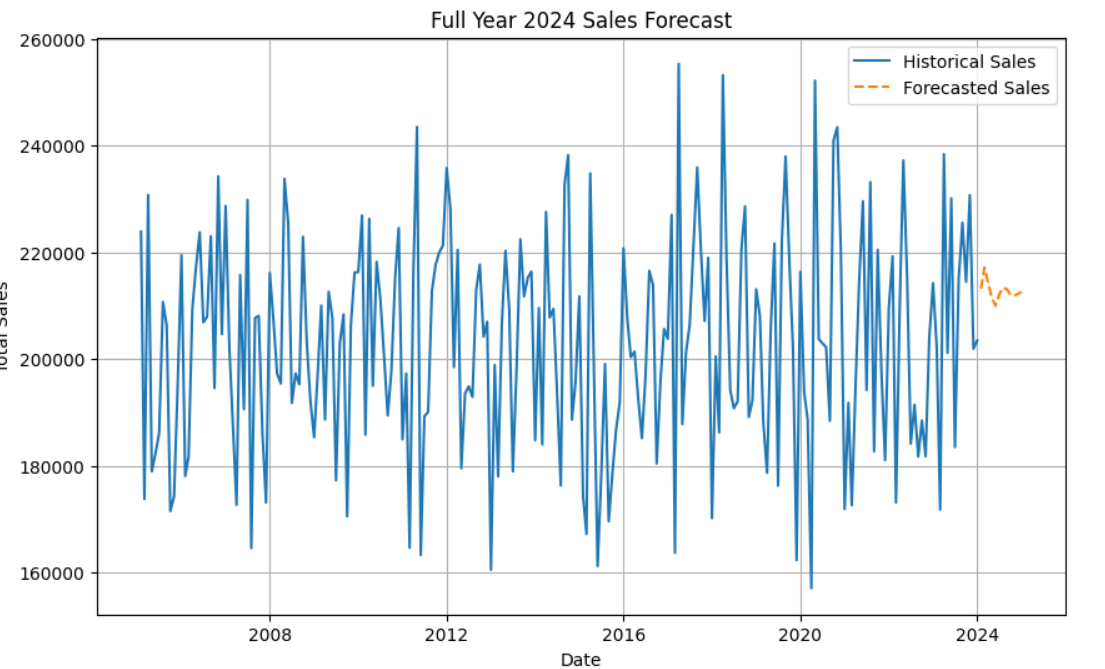


**3. Tools**

* **Python (Scikit-learn, Matplotlib)**: For data analysis, visualization, and model development.
* **Azure Data Studio**: To connect and manage SQL Data Warehouse, and run SQL queries for data preparation.

**4. Deliverables**

* **Analysis Report**: A comprehensive report detailing data exploration, insights, and predictive models was generated. It includes:
  + Summary of data patterns and trends.
  + Details on the sales forecasting model development.
  + Performance evaluation of models, with recommendations for further improvements.
* **Predictive Models**: A **Random Forest Regressor** model for sales forecasting, developed and evaluated, with an MSE of X and an R-squared of Y.

** **

**5. Conclusion**

* Week 3 was successful in developing a robust predictive model for sales forecasting using Python.

**Week 4 Report: Model Deployment**

**1. Overview**

In Week 4, the focus was on deploying the sales forecasting model developed in Week 3. A web application was created to allow users to interact with the model and generate predictions based on new data inputs. Streamlit, a Python web framework, was used to build and deploy the application, ensuring a user-friendly interface for making predictions.

**2. Tasks**

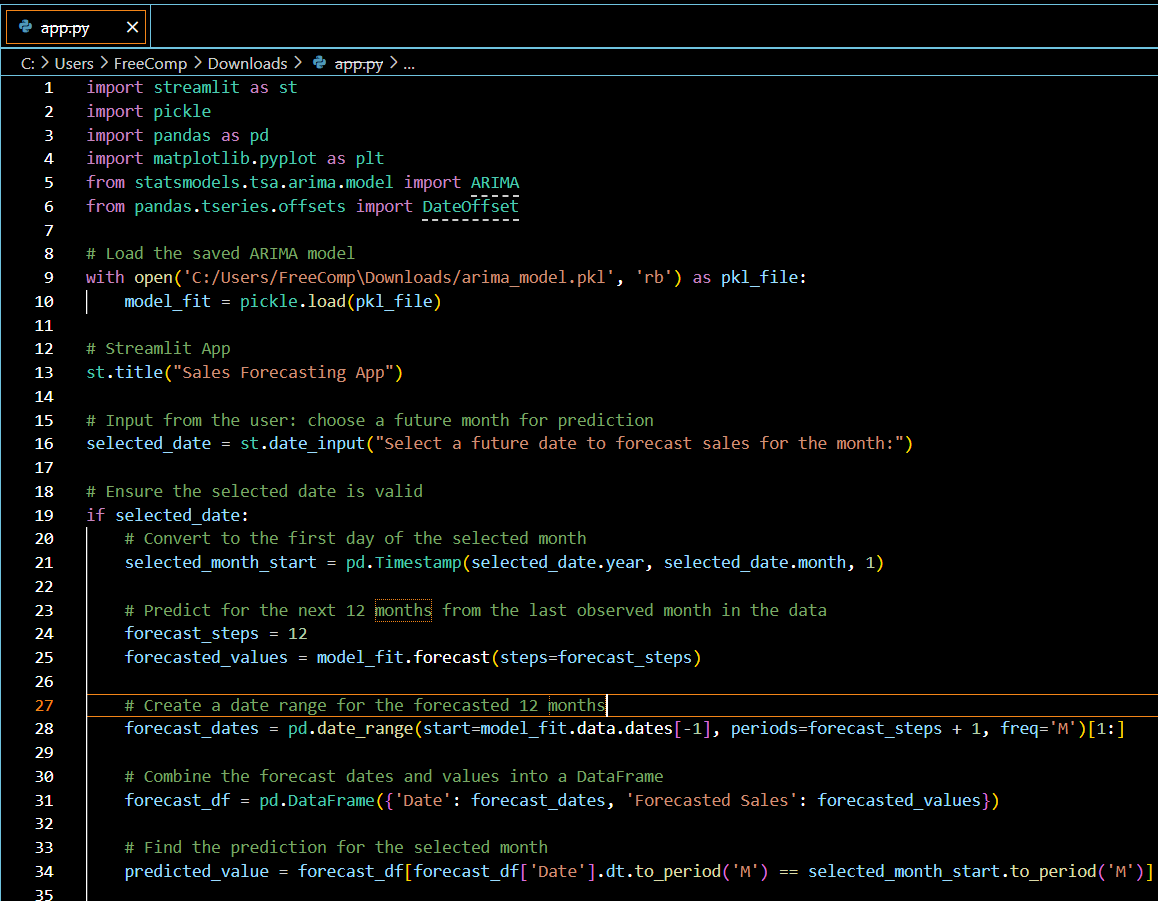
**2.1 Model Deployment**

The task for this week was to deploy the trained **sales forecasting model** as a web application, making it accessible to end-users for generating real-time predictions.

**Steps Taken**:

* **Web Application Design**: The web application was designed with a simple, intuitive user interface that allows users to input key features such as Order Year, Order Month, Customer Region, and other relevant data. Based on this input, the model would generate a prediction for total sales.
* **Streamlit Framework**: Streamlit was chosen for its simplicity and ease of use when building interactive web applications in Python. It allowed for the seamless integration of the model into a web app where users can input data and view the predictions instantly.

**Example Streamlit App Code**:



**App Features**:

* **User Inputs**: Users can select the order year, month, and customer region using interactive widgets.
* **Prediction Button**: When the user clicks "Predict Sales", the app feeds the input data into the forecasting model, which returns the predicted sales amount.
* **Output**: The predicted sales are displayed on the screen in a readable format.

**2.2 Streamlit Deployment**

After developing the web application locally, it was deployed using **Streamlit Cloud**, making the app accessible online.

**Steps for Deployment**:

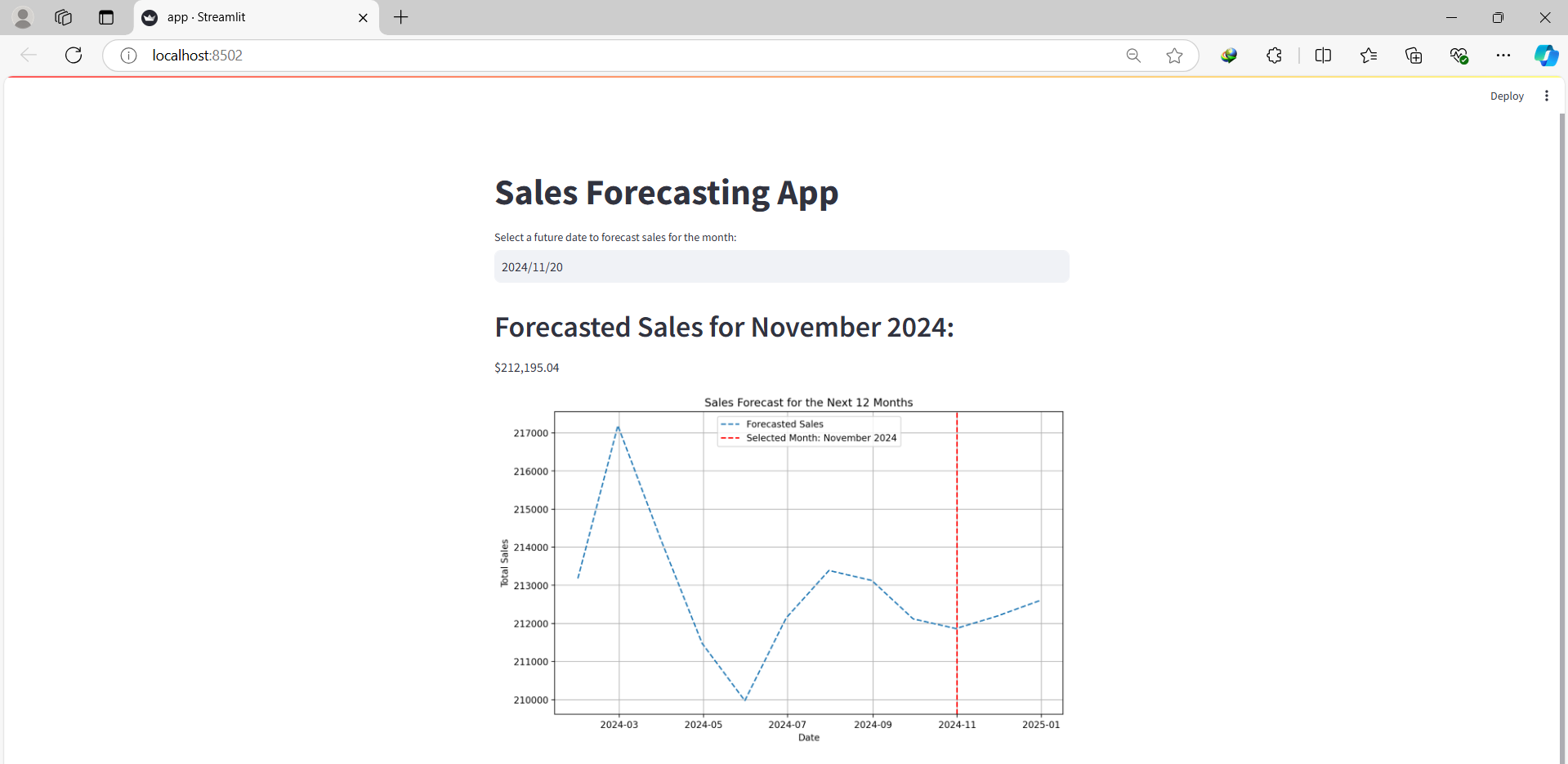
1. **Model Export**: The trained model was exported as a .pkl file using the joblib library.
2. **Application Deployment**: The Streamlit app was pushed to a GitHub repository and connected to Streamlit Cloud for automatic deployment.
3. **Access URL**: A URL was generated, allowing stakeholders to access the web app for testing and using the model predictions.

**3. Tools**

* **Streamlit**: A Python framework used for building the web interface.
* **joblib**: Used for saving and loading the trained machine learning model.
* **Streamlit Cloud**: Used to deploy the web application and host it online.

**4. Deliverables**

* **Deployed Web Application**: A fully functional **Streamlit web application** was deployed, allowing users to interact with the sales forecasting model. The app includes input fields for Order Year, Order Month, and Customer Region, and provides real-time sales predictions based on these inputs.



**5. Conclusion**

Week 4 concluded with the successful deployment of the sales forecasting model as a web application. The app is now accessible to stakeholders for sales predictions, completing the final phase of the project.

**RESOURCES FOR ALL PROJECT**

* **csv file for database source and our database backup:**

[**https://drive.google.com/drive/folders/1Bf0BKbl9-pbQQwHlUbiPdvwyiWV7guEF?usp=sharing**](https://drive.google.com/drive/folders/1Bf0BKbl9-pbQQwHlUbiPdvwyiWV7guEF?usp=sharing)

* **script for some sql queries (extraction and analysis)**

[**https://drive.google.com/file/d/1QmNZuBP3NZkXa9VRVWElgseNM\_YA2rP9/view?usp=sharing**](https://drive.google.com/file/d/1QmNZuBP3NZkXa9VRVWElgseNM_YA2rP9/view?usp=sharing)

* **click to access data warehouse BAK and python script for connection**

[**https://drive.google.com/drive/folders/1BrLsxFbHRoFPytXExsmmiNeaidO40jBt?usp=sharing**](https://drive.google.com/drive/folders/1BrLsxFbHRoFPytXExsmmiNeaidO40jBt?usp=sharing)

* **click to access all scripts for week 3**

[https://drive.google.com/drive/folders/1Y61cepz\_15uu9LEhE1LbgJb4flx9dpoo](https://drive.google.com/drive/folders/1Y61cepz_15uu9LEhE1LbgJb4flx9dpoo%20)

* **click to access all scripts for week 4**

[**https://drive.google.com/drive/folders/1oigQKjJRxikK9zLUU1y54oEWcOQI5q\_N?usp=sharing**](https://drive.google.com/drive/folders/1oigQKjJRxikK9zLUU1y54oEWcOQI5q_N?usp=sharing)

**TEAM MEMBERS**

* **Youssef Bahgat Ahmed**
* **Esraa Mohamed Badea**
* **Mervat kamel Khalil**
* **Wafaa Rafek Shawky**
* **Yara Adel Elshamly**
* **Mennatullah Mohamed Mahrous**