Data Engineer Project Report

**Team members**:

1. Ahmed Gharib
2. Pierre Mousa
3. Hager Foda
4. Ahmed Hossam
5. Aya Abdelsamad
6. Esraa Zakaria

**Instructor**: Amera Youssef

**Group code**: CAI1\_AIS10d

**Idea 3**: Customer Feedback Analysis and Improvement

**Table of Contents**

* Introduction
* Project Overview
* Tools and Technologies
* Week-by-Week Breakdown
* Challenges and Solutions
* Results and Findings
* Future Work

Introduction

This project focuses on developing a complete solution for customer feedback management and analysis for McDonalds. The system is designed to collect, store, process, and analyze customer feedback data to provide actionable insights. The core components include a SQL database for data storage, a data warehouse for aggregation, and a sentiment analysis model for classifying feedback as positive, neutral, or negative.

The final output is a deployed of an exe file that uses the model to predict the feedback of the customers.

Project Overview

This project is structured into four key phases:

1. **SQL Database Setup and Data Collection**: A relational database was designed using Microsoft SQL Server to manage customer feedback, including tables for feedback forms, customer profiles, and categories. Historical feedback data was imported, and SQL queries were created for data extraction and summarization.
2. **Data Warehouse and Python Processing**: A data warehouse was implemented to aggregate feedback data for analysis. Python was used for cleaning and preprocessing the data, with an emphasis on text processing for sentiment analysis.
3. **Sentiment Analysis and Azure Integration**: Machine learning models were developed using Python to perform sentiment analysis, categorizing feedback as positive, neutral, or negative. Microsoft Azure services were integrated to provide enhanced data storage and computational resources.
4. **Deployment**: The final model was deployed via a desktop application built with python GUI library, using the trained model to classify new feedbacks to gain insights from feedback trends.

**Tools and Technologies**

The following tools and technologies were utilized throughout the project:

* **Database Management**:
  + Microsoft SQL Server
  + SQL Management Studio
* **Data Processing and Analysis**:
  + Python:
    - Pandas
    - NLTK (Natural Language Toolkit)
    - SpaCy
* **Data Warehousing**:
  + Microsoft SQL Data Warehouse
  + Visual Studio
* **Cloud Services**:
  + Azure Data Studio
  + Azure Services
* **Machine learning model**:
  + NumPy
  + Matplotlib
  + SpaCy
  + NTLK
  + Scikt-learn
  + Joblib
* **MLOps**:
  + MLflow
* **Desktop Application Development**:
  + Tkintr(Python GUI library)

**Week 1: SQL Database Setup and Data Collection**

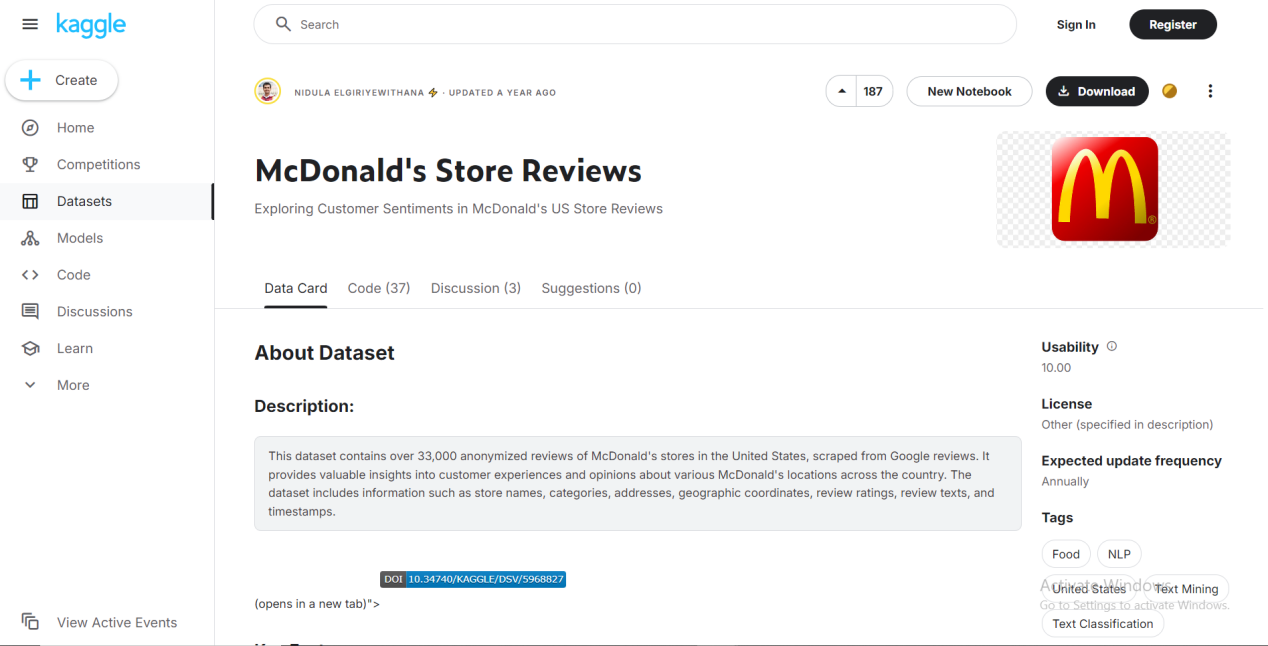
**Team Members Involved:**

* Ahmed Gharib
  + **Responsibilities:**
    - Database Creation
    - Dataset Assembling
* Pierre Mousa
  + **Responsibilities:**
    - Schema Design
    - Data Analysis

**Week Overview**

In the first week of the Customer Feedback Analysis and Improvement project, our team focused on setting up the SQL database to manage customer feedback. We utilized a dataset obtained from Kaggle, specifically the McDonald's Store Reviews dataset ( https://www.kaggle.com/datasets/nelgiriyewithana/mcdonalds-store-reviews). This dataset provided historical customer feedback data, which was crucial for our analysis.

Our activities involved designing the database schema, importing the historical customer feedback data from the Kaggle dataset, and writing SQL queries to extract and summarize the data.



**Database Schema**

The following tables were created to structure the database:

**Customer Table**

**Columns:**

* id (INT, Primary Key)
* names (VARCHAR)
* states (VARCHAR)

**Store Table**

**Columns:**

* id (INT, Primary Key)
* latitude (FLOAT)
* longitude (FLOAT)
* addresses (VARCHAR)

**Product Table**

**Columns:**

* id (INT, Primary Key)
* names (VARCHAR)
* price (FLOAT)

**Orders Table**

**Columns:**

* id (INT, Primary Key)
* dates (DATETIME)
* customer\_id (INT, Foreign Key referencing Customer)
* store\_id (INT, Foreign Key referencing Store)

**OrderDetails Table**

**Columns:**

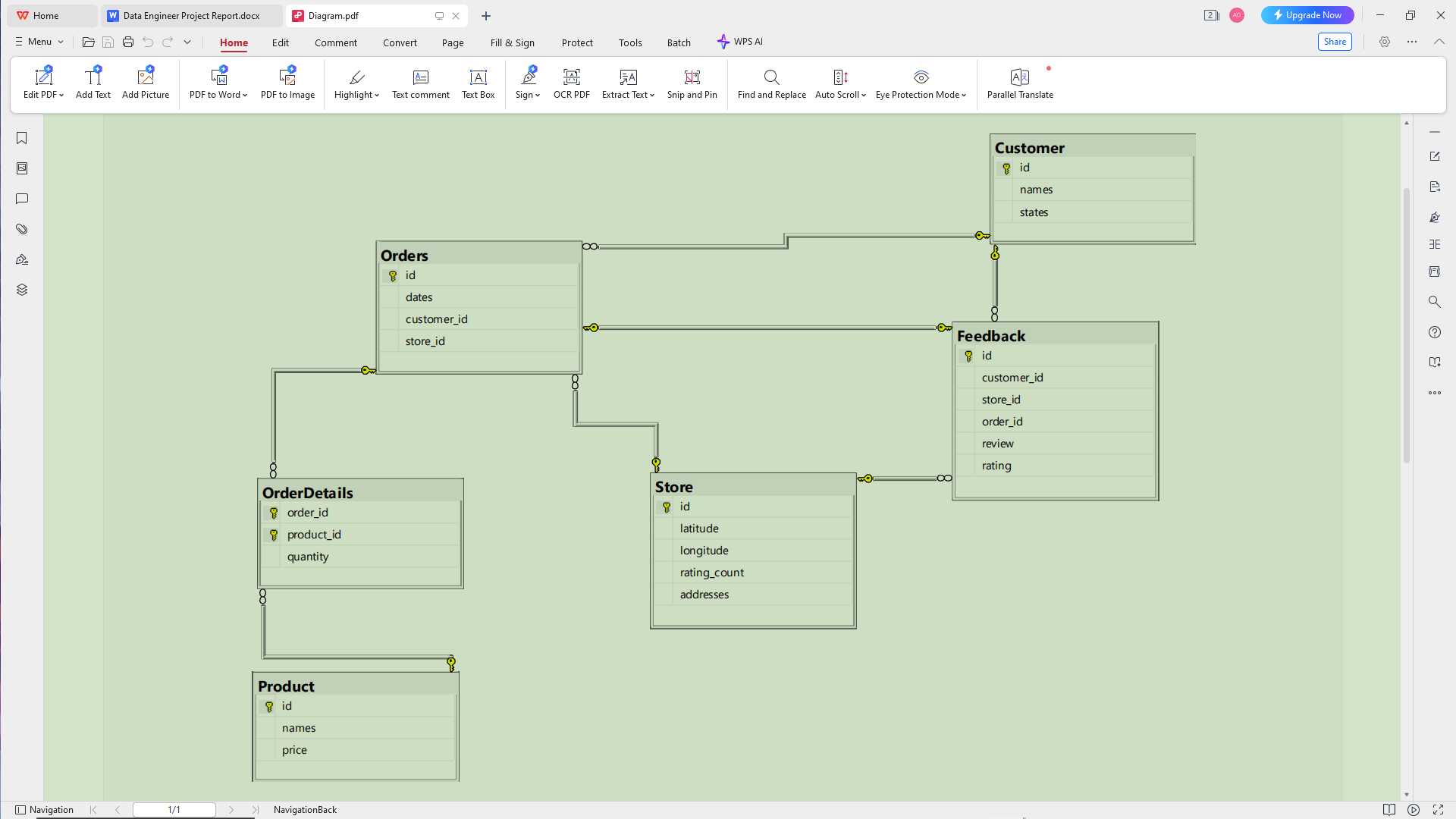
* order\_id (INT, Foreign Key referencing Orders)
* product\_id (INT, Foreign Key referencing Product)
* quantity (INT, Primary Key composite with order\_id)

**Feedback Table**

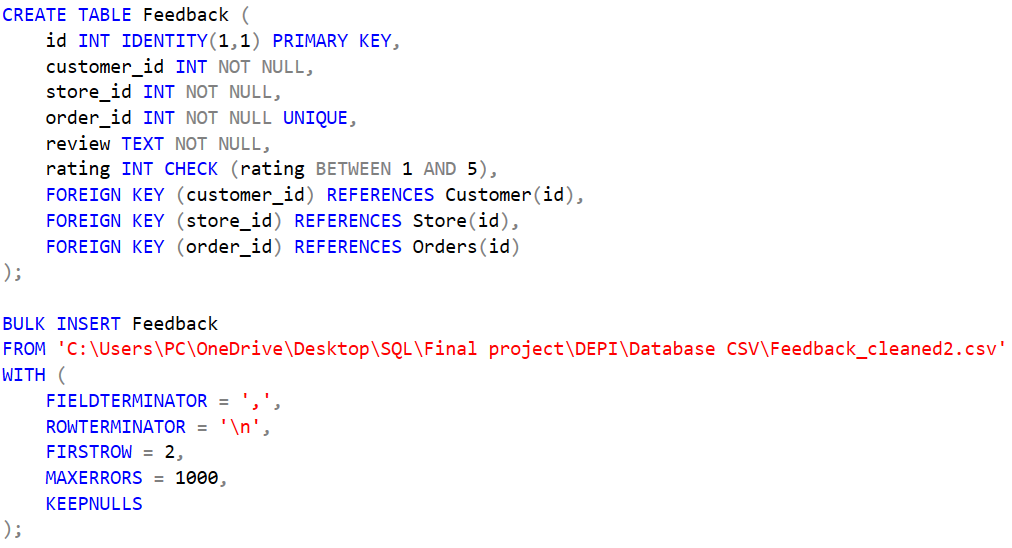
**Columns:**

* id (INT, Primary Key)
* customer\_id (INT, Foreign Key referencing Customer)
* store\_id (INT, Foreign Key referencing Store)
* order\_id (INT, Unique, Foreign Key referencing Orders)
* review (TEXT)
* rating (INT, CHECK between 1 and 5)

**The Relationships:**



* Orders and product: Many to Many (using OrderDetails)
* Orders and Feedback: One to One
* Orders and Customer: One to Many
* Orders and Store: One to Many
* Store and Feedback: One to Many
* Customer and Feedback: One to Many



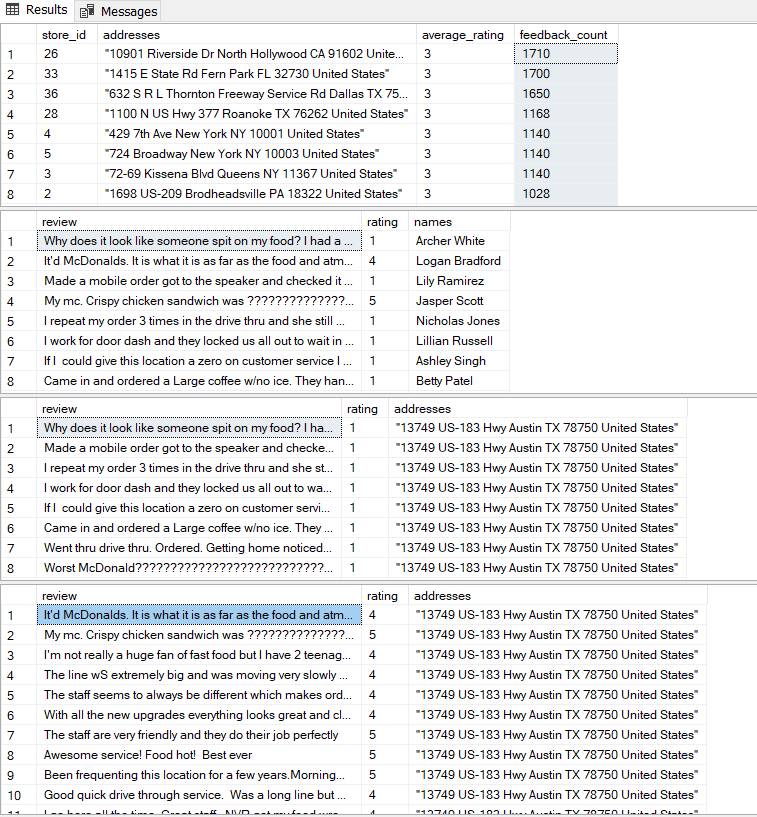
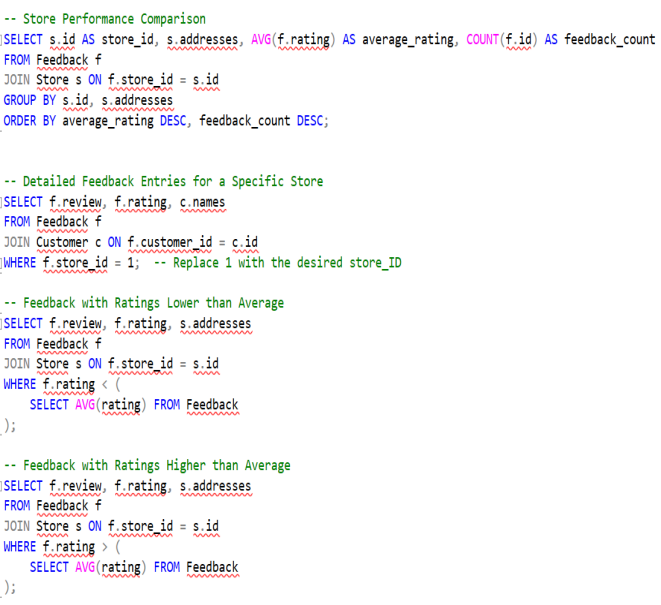
**Data Import**

Historical customer feedback data was imported into the database using the BULK INSERT command for each table, ensuring efficient data entry from CSV files.

**SQL Queries for Data Extraction and Analysis**

To analyze the feedback data, several SQL queries were written:

* **Total Number of Feedback Entries**
* **Total Feedback Count by Store**
* **Rating of Each Customer**
* **Monthly Feedback Count Over Time**
* **Average Rating Over Time**
* **Feedback Distribution by Rating**
* **Top Products by Average Rating**
* **Feedback Sentiment Analysis**
* **Average Rating for All Customers**
* **Store Performance Comparison**
* **Detailed Feedback Entries for a Specific Store**
* **Feedback with Ratings Higher than Average**
* **Feedback with Ratings Lower than Average**
* **Combine Feedback and Total Order Amount**



These queries provided insights into customer feedback patterns, helping identify trends and areas for improvement.

**Week conclusion:** The first week established a strong foundation for the project with a well-structured SQL database and initial data analysis capabilities. Moving forward, the focus will shift to data warehousing and processing in the following weeks.

### **Week 2: Data Warehouse and Python Data Processing**

**Team Members Involved**

* Ahmed Hossam
* Hager Foda

**Responsibilities:**

**Hager Foda:**

* + Data warehouse setup and data integration.

**Ahmed Hossam:**

* + Data cleaning and preprocessing using Python.
  + Data visualization and exploration.

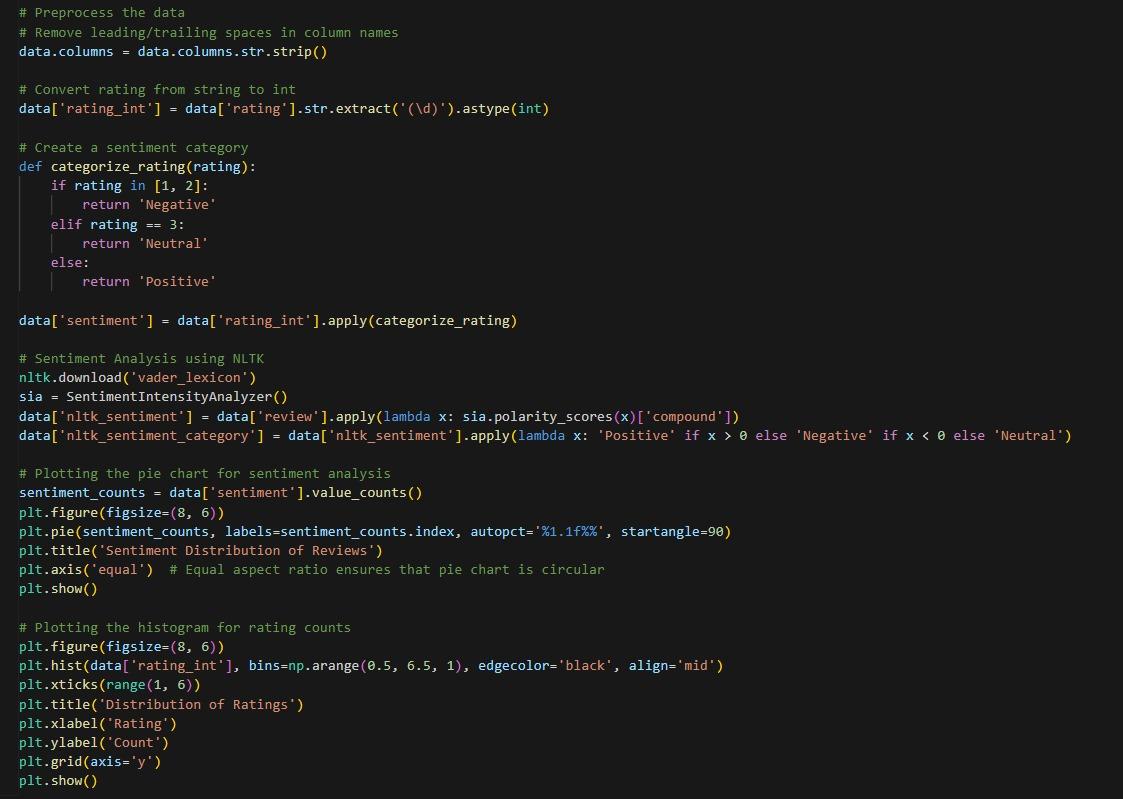
**Week Overview**

During the second week of the **Customer Feedback Analysis and Improvement** project, our focus was on implementing a **data warehouse** for aggregating customer feedback data and performing Python-based **data processing** to clean and prepare the data for sentiment analysis.

**Key Activities:**

**Data Cleaning and Preprocessing with Python:**

* Python libraries such as **Pandas** and **NLTK** were used to preprocess feedback data.
* Key tasks included handling missing data, removing duplicates, and text processing for sentiment analysis.
* Outliers were detected using Z-score and IQR methods and removed to improve data quality.



**Handling Missing Data:**

* + For numerical columns, missing values were replaced with either the mean or median.
  + For categorical columns, the mode was used to fill missing values.
  + A threshold was applied to drop columns with excessive missing data.

**Outlier Detection and Removal:**

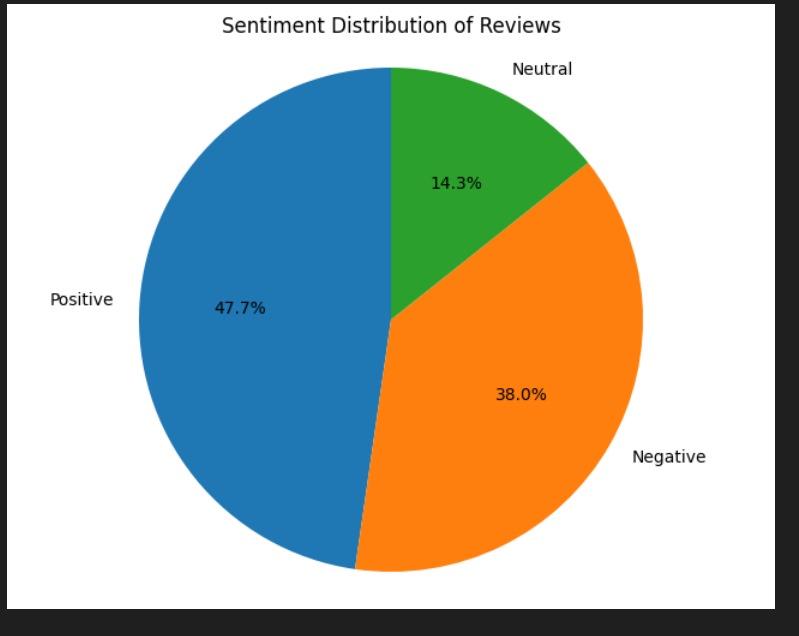
* + Outliers were detected using **Z-score** (threshold = 3) and **IQR (Interquartile Range)**, and they were removed to improve data integrity.
  + Outlier removal was visualized through boxplots.

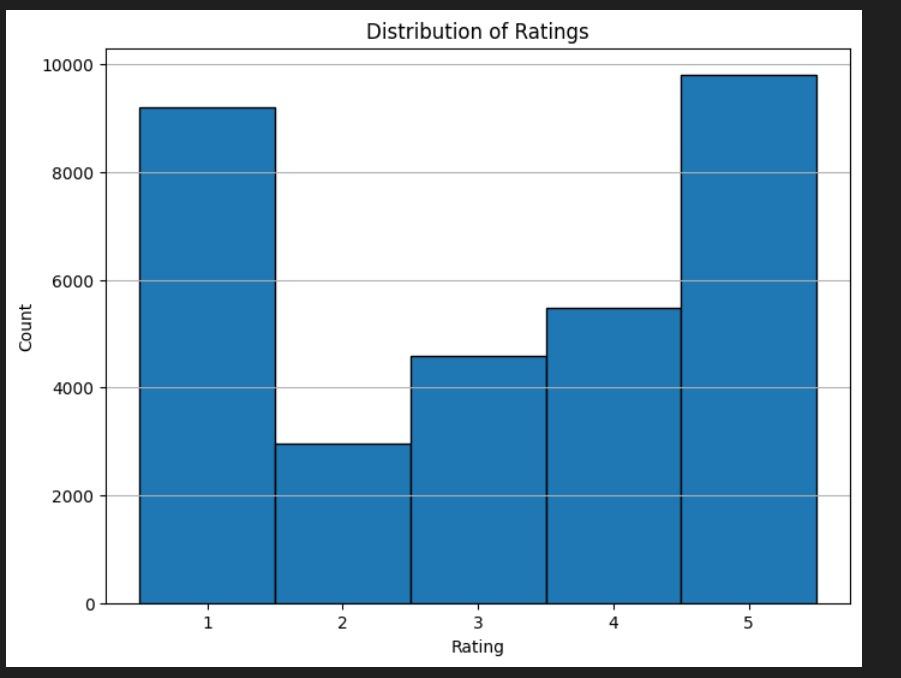
**Data Normalization and Encoding:**

* + Numerical columns were scaled using **StandardScaler** to ensure all features were on a similar scale.
  + Categorical variables were encoded using **LabelEncoder**, converting text categories into numeric codes.

**Visualizations for Data Exploration:**

* + Visual tools such as **histograms**, **boxplots**, and **correlation heatmaps** were employed to assess the distribution and relationships of the cleaned data.
  + Pair plots were generated to further investigate numerical feature interactions.



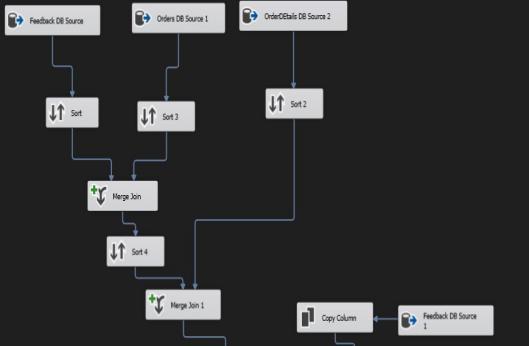
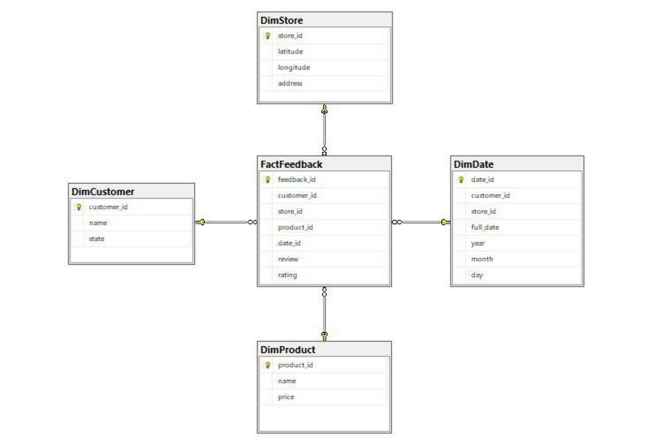


**Data Export:**

* + After the cleaning process, the cleaned datasets were exported as CSV files, which would later be used for data analysis and machine learning.

**Data Warehouse Setup:**

* We set up a **data warehouse** using **Microsoft SQL Data Warehouse** to store and manage large-scale feedback data.
* The feedback data was integrated into the warehouse from various tables, including customer profiles, orders, and feedback categories, to facilitate further analysis.



**Week Conclusion:** By the end of Week 2, the data was successfully loaded into the data warehouse and cleaned using Python. This provided a solid base for future analysis, especially for the upcoming sentiment analysis tasks.

**Week 3: Data Warehouse and Python Data Processing**

**Team Members Involved:**

* Aya Abdelsamad
* Esraa

**Responsibilities:**

Aya Abdelsamad**:**

* + data preprocessing, visualization
  + model architecture
  + MLflow integration

Esraa**:**

* + Azure-related tasks for model deployment and data management

**Week Overview:**

This week focused on setting up the data pipeline for sentiment analysis using a machine learning model, combined with Azure storage and resource management. Aya handled the data preprocessing, modeling, and integration with MLflow for model tracking, while Esraa managed the deployment of the data and model within Azure.

**Machine learning Model:**

* 1. **Preprocessing and Vizualization**
* **Data Import and Exploration:** Loaded the customer feedback dataset and explored its structure, including the first few rows and overall information.
* **Missing Values and Duplicates:** Identified and reported the number of missing values in each column and counted any duplicate rows.A screenshot of a computer code

  Description automatically generated
* **Statistical Summary:** Generated a statistical summary to understand the distribution of the dataset's numerical features.
* **Sentiment Labeling:** Created a new column to categorize feedback sentiments (Positive, Negative, Neutral) based on the rating values.
* **Label Encoding:** Transformed sentiment labels into numerical format using label encoding for easier analysis.
* **Feedback Segmentation:** Segmented feedback into three distinct categories (negative, neutral, positive) for focused analysis.
* **Word Cloud Visualization:** Generated word clouds for each sentiment category to visualize the most common words in the feedback. A black background with colorful words

  Description automatically generated

A close-up of words

Description automatically generatedA black background with words

Description automatically generated

* **Text Data Cleaning:** Preprocessed the review text by tokenizing, lemmatizing, and removing stopwords and punctuation to ensure clean and meaningful data for further analysis.

A computer screen shot of a code

Description automatically generated

**1.2 Model Architecture**

* **Input Features**: The review texts are used as input features for the model, while the sentiment labels are the target variable for classification.
* **Train-Test Split**: The data is divided into training and testing sets, with 80% for training and 20% for testing, ensuring the model is evaluated on unseen data.
* **TF-IDF Vectorization**: Text data is transformed into numerical format using TF-IDF (Term Frequency-Inverse Document Frequency), emphasizing important words while downplaying common ones.
* **LinearSVC Classifier**: A Linear Support Vector Classifier (LinearSVC) is used to classify sentiments based on the TF-IDF features, providing a robust solution for text classification.
* **Pipeline**: The model pipeline automates the process by combining the TF-IDF vectorizer and LinearSVC classifier into a single workflow for efficient training and prediction.A screenshot of a computer

  Description automatically generated

**1.3 MLflow Integration**

* **Parameter Logging**: Key model parameters, such as the type of model and TF-IDF settings, are logged in MLflow to track the model’s configuration.
* **Model Training**: The model is trained within an MLflow run, allowing the tracking of the training process and capturing relevant data.
* **Metric Logging**: Accuracy, as a key performance metric, is logged to MLflow for later analysis and comparison.
* **Model Logging**: The trained model is saved in MLflow, enabling future access for deployment or further experimentation.

A screenshot of a computer code

Description automatically generated

**1.4 Performance Metrics for Model Evaluation**

* **Accuracy**: The model achieved an accuracy of **83.1%**, meaning 83.1% of the predictions were correct.A close-up of a computer code

  Description automatically generated
* **Confusion Matrix**: Offers a detailed breakdown of correct and incorrect predictions for each sentiment class (Positive, Neutral, Negative), with a heatmap visualization to highlight areas of confusion.

A screenshot of a graph

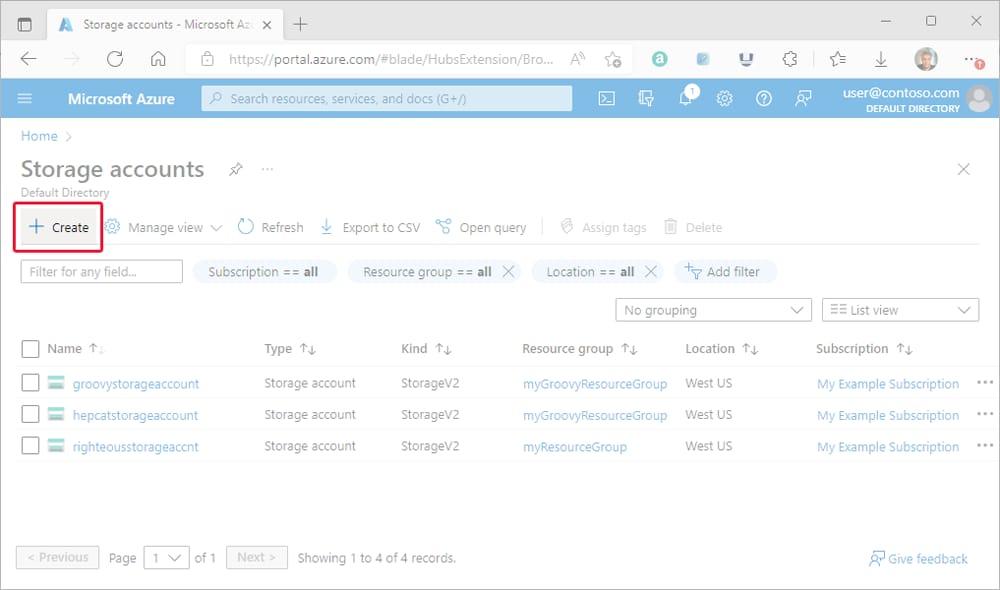
Description automatically generated

### **Azure Tasks**

#### Environment Setup

**Create Azure Storage Account:**

A storage account was created to contain Azure Storage data objects such as blobs, files, queues, and tables. This storage provides a unique namespace accessible globally over HTTP or HTTPS.



**Create Tables and File Shares:**

Tables were created with unique names, and database files were uploaded to Azure. The connection script was generated and executed in a virtual machine via PowerShell.

**Create Resource Group:**

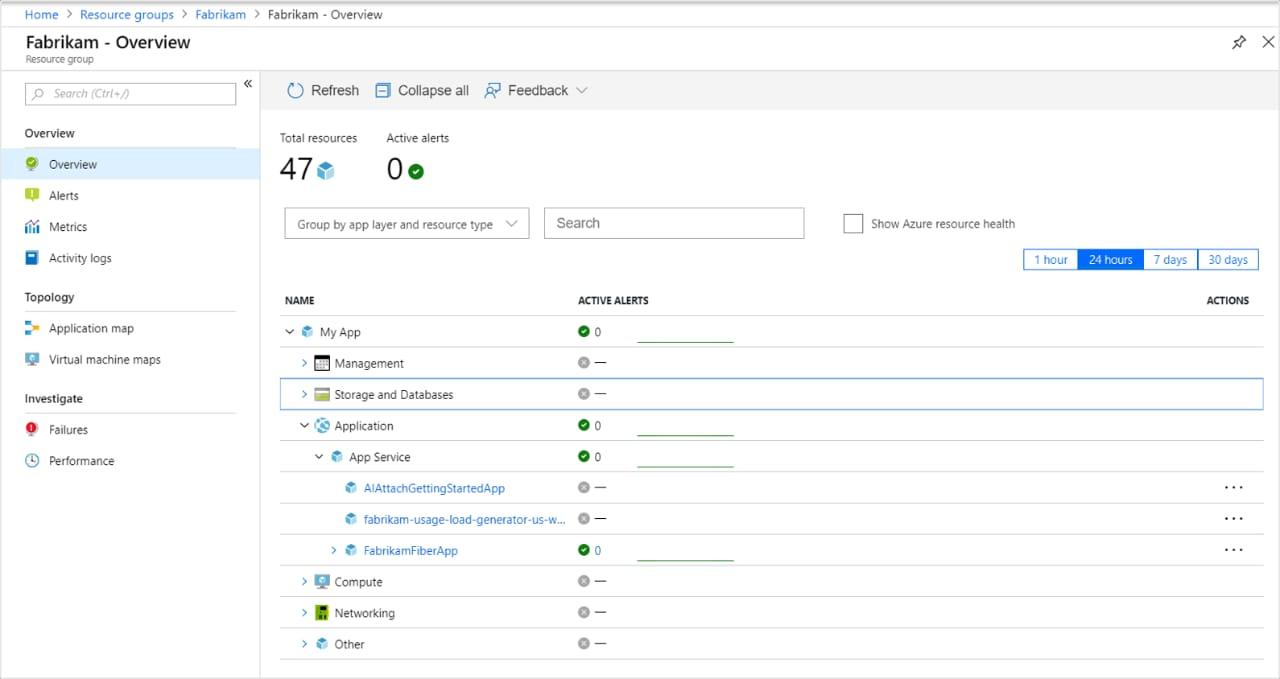
A resource group was created as a container to manage related resources for the Azure solution.

**Create Bing Search Resource:**

A Bing search resource was set up using Bing Search v7, providing access to Bing search APIs.

**Azure Data Factory Setup:**

A data pipeline was created using Azure Data Factory. This pipeline moves and transforms data from both on-premises and cloud sources to a centralized cloud data store for further analysis.



**Week conclusion:** This week marked significant progress in setting up the data preprocessing pipeline, model architecture, and Azure environment for model deployment and data management. Aya successfully built and integrated the sentiment analysis model with MLflow, while Esraa established the necessary Azure resources and infrastructure. The collaboration between Python data processing and Azure services will pave the way for efficient model deployment and real-time data management in the upcoming weeks.

### **Week 4: MLOps and Final Presentation**

**Team Members Involved:**

Ahmed Gharib

**Responsibilities:**

* + - Sentiment Analysis Application
    - GUI Development for Data Insertion and Feedback Analysis

Pierre Mousa

**Responsibilities:**

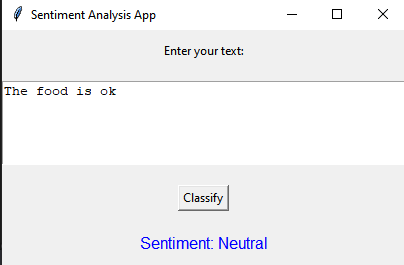
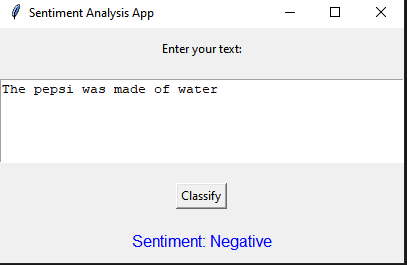
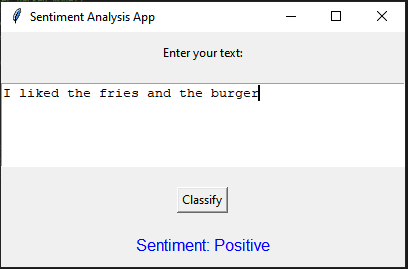
* + - Data Analysis GUI

**Week Overview:**

In the final week, we focused on developing the graphical user interface (GUI) to make the sentiment analysis model and data insertion accessible through a user-friendly application. The tasks for the week included:

1. **Sentiment Analysis Application:**

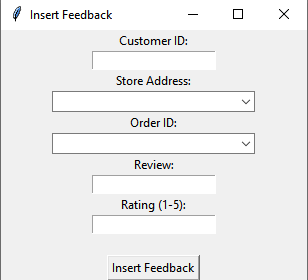
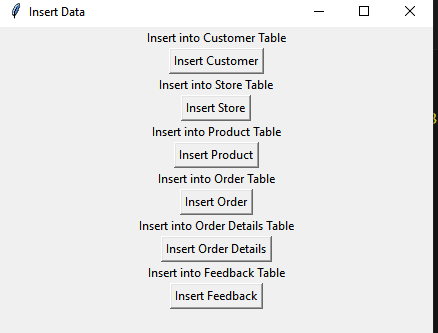
We developed a desktop application (.exe file that works on any windows no need for any additional installations) that takes user-inputted text (customer feedback) and classifies it as **positive**, **negative**, or **neutral** using the trained sentiment analysis model. This application provided real-time feedback to the user, offering insights into customer sentiment. (Refer to the screenshot for the result displayed as "Sentiment: Positive").



1. **GUI for Data Insertion:**

A GUI was built for inserting new records into the database, allowing the user to input information related to customers, stores, products, orders, order details, and feedback. This streamlined the process of updating and managing the database without requiring direct SQL commands.

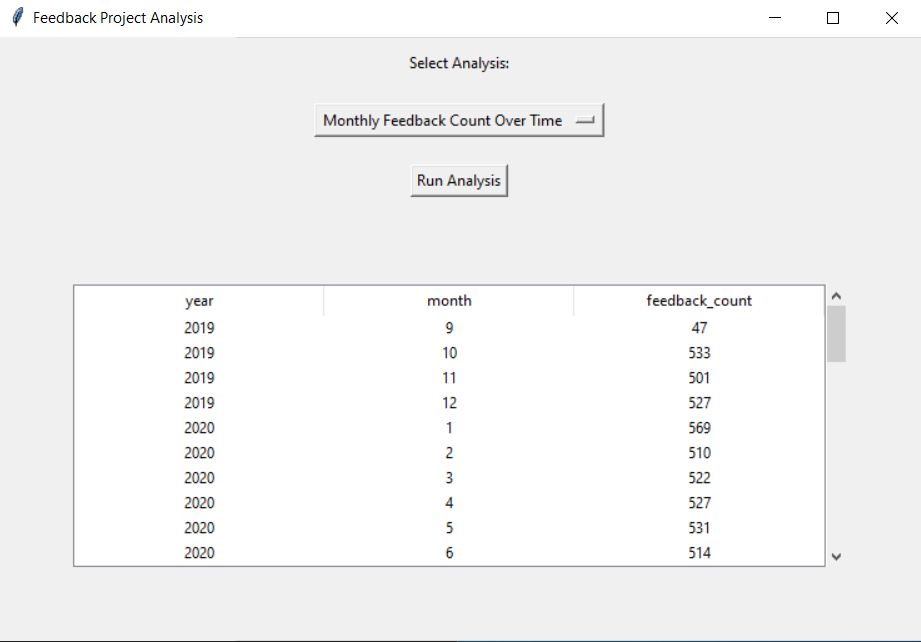
The user can interact with buttons for inserting data into different tables, making the application versatile for non-technical users.



1. **GUI for Data Analysis:**

The analysis GUI provided a selection of pre-defined queries that users can run to visualize feedback trends over time. For instance, one analysis presented the **Monthly Feedback Count Over Time**, displaying the feedback count per month in a table format.

This functionality allowed the team to gain insights into customer feedback patterns and helped in understanding store performance across different months.



**Results and Findings**

* The sentiment analysis application functioned as expected, accurately classifying customer feedback and offering real-time insights. This tool is beneficial for businesses to understand overall customer satisfaction.
* The data insertion GUI provided a seamless way to manage the database, ensuring that new records could be added easily and efficiently.
* The analysis GUI helped us visualize feedback trends and offered a comprehensive view of how feedback patterns changed over time, which is crucial for performance analysis.

**Week Conclusion:** The final week resulted in a fully operational desktop application with features for sentiment analysis, database management, and data visualization. This marks the successful deployment of the customer feedback system, which is now ready for use.

**Challenges and Solutions**

* **Challenge 1: Lack of Detailed and Reliable Data for the Database**
  + **Issue**: The only available data was a CSV file for machine learning, without a comprehensive dataset for building a detailed database.
  + **Solution**: We designed the database structure as if we had the full dataset and generated random data using Python (random, math, pandas). For each table, we created a corresponding CSV file and used bulk insert operations to populate the database.
* **Challenge 2: Converting a Python GUI into an Executable File**
  + **Issue**: The initial design used a Python-based GUI, but we wanted to convert it into an executable (.exe) file to make it portable for any Windows device.
  + **Solution**: After researching, we found a method to bundle all necessary data, tools, and dependencies into the executable file. We also saved the trained machine learning model and NLTK's en\_core within the executable, ensuring the end-user wouldn't need to install or configure anything. This made the software ready to run on any system without additional setup.

**Results and Findings**

**Database Insights**: The SQL database was successfully designed and populated with synthetic data. SQL queries were executed to extract information (as described in week 1 query analysis), this provide a solid foundation for analysis.

**Data Warehouse**: The integration of the SQL database with a data warehouse facilitated efficient data aggregation and storage, making it easier to perform large-scale analysis across different categories of feedback.

**Sentiment Analysis**: The sentiment analysis model, built using Scikit-learn and NLP techniques, classifying feedbacks into positive, neutral, and negative categories. The model performed well (as shown in the ML section in week 3). The feedback classification provided actionable insights into customer satisfaction levels.

**Azure Integration**: Azure services were successfully used to store and manage large-scale data, improving scalability and reliability in handling customer feedback data.

**Deployment**: The executable file (converted from the original Python GUI) provided a portable and easy-to-use solution, eliminating the need for additional software setup on the user’s device. The deployment also featured a smooth user experience.

**Future Work**

**Full restaurant Database**: instead of making a system for feedback only it is better to create a full restaurant system and for that creating a larger database is a must.

**Better Deployment V1**: Using a technology like Docker to make the exe file smaller (in size) and more practical.

**Better Deployment V2**: making a better App that is more practical for customer use to make full use of the system instead of the GUIs that we created.

**Better Sentiment Analysis Data**: the dataset that is used has a major problem which is the inconsistency of the classes, it has a lot less “Neutral” class than the other two classes.