**Project 1**

**Khaja Muneebuddin Adil**

1. **Finding and analysing data**

I got the dataset from Kaggle. It is a healthcare dataset and consists of 55,500 rows and 15 columns. There are some missing values and inconsistencies throughout the dataset.

**Technical Observations:**

Schema: The dataset has 15 columns as mentioned below

Name (StringType): The name of the patient.

Age (IntegerType): The age of the patient at the time of admission.

Gender (StringType): The gender of the patient.

Blood Type (StringType): The blood type of the patient.

Medical Condition (StringType): The diagnosed medical condition of the patient.

Date of Admission (StringType): The date the patient was admitted to the hospital.

Doctor (StringType): The attending doctor’s name.

Hospital (StringType): The name of the hospital where the patient was admitted.

Insurance Provider (StringType): The insurance provider covering the patient.

Billing Amount (FloatType): The total billing amount for the patient's treatment.

Room Number (IntegerType): The room number assigned to the patient during their stay.

Admission Type (StringType): The type of admission, such as urgent or elective.

Discharge Date (StringType): The date the patient was discharged from the hospital.

Medication (StringType): The prescribed medication for the patient.

Test Results (StringType): The results of any diagnostic tests.

A screenshot of a computer

Description automatically generated

**Data Quality:**

Some inconsistencies in case formatting (e.g., Name column contains both uppercase and lowercase letters inappropriately). Dates are not standardized, appearing in a dd-mm-yyyy format, which will require transformation.

**Non-Technical Observations:**

**Purpose and Insights of the Data:**

This dataset seems to capture healthcare-related information, including patient demographics, medical conditions, admission details, and billing information. The data can provide insights into hospital operations, patient demographics, average billing amounts, and patterns in medical conditions.

Analysing admission types (e.g., emergency vs. elective) and their corresponding billing can highlight cost trends. Exploring medical conditions by age group or gender could reveal trends in healthcare.

**Potential Improvements:**

Standardize the formatting of text data (e.g., proper capitalization of Name). Convert Date of Admission and Discharge Date columns to a proper date format for easier analysis and time-based computations.

1. **Architectural diagram**

A screenshot of a computer

Description automatically generated

1. **Data Pipeline Creation:**

**Data Ingestion:**

To ingest the data, I created an HTTP linked service in Azure Data Factory (ADF) to transfer data from GitHub to Azure Data Lake Storage (ADLS). I utilized a Copy Activity in ADF, using the GET request method. The data from GitHub was stored in a container named "raw."

**Data Storage:**

After the initial data ingestion, I linked my ADLS storage container to an Azure Databricks notebook for further processing. I used key-based authentication to connect Azure Data Lake Storage (ADLS) to Azure Databricks by configuring the storage account key within the Databricks environment. This setup provided a scalable and flexible storage solution, allowing me to manage various data formats efficiently.

A screenshot of a computer

Description automatically generated

**Data Transformation (PySpark):**

In the Databricks notebook, I executed a single PySpark script that contained both the data cleaning and transformation processes. Using the notebook function in ADF, I ran this code.

A screenshot of a computer

Description automatically generated

It included the following steps:

* I started by cleaning the dataset, which involved renaming several columns for consistency. For example, I changed "Medical Cc" to "Medical\_Condition" and "Room Number" to "Room\_Number."
* I also removed duplicate and null values from the dataset to enhance data quality.
* After cleaning, I stored the refined data in a new container called "cleaned."

A screenshot of a computer

Description automatically generated

Next, I utilized the cleaned data from the "cleaned" container as input for further transformations.

During the transformation in Databricks, I noticed that the dataset lacked a unique identifier for each record. Therefore, I added a new column called "ID."

The names of patients were inconsistently formatted, containing a mix of uppercase and lowercase letters. To standardize the names, I used the initcap function from PySpark, which capitalizes the first letter of each word in the name. This method was efficient compared to manually iterating through the names and changing each one individually. An alternative could have been using string functions to split the names and reformat them, but initcap streamlined the process significantly.

Finally, I stored the transformed data in a new container called "transformed" in ADLS, saving all the refined data in Parquet format for efficient storage.

A screenshot of a computer

Description automatically generated

1. **Transformation and Analytics:**

Once the data was transformed and stored in the "transformed" container in Azure Data Lake Storage Gen 2 (ADLS), I linked ADLS to Azure Synapse Analytics. I navigated to the transformed container, selected the dataset, and created an external table in Synapse Analytics for querying purposes.

A screenshot of a computer

Description automatically generated

**Synapse Analytics:**

With the external table created in Synapse, I executed several SQL queries to analyse the data like –

Counting the Number of Patients for Each Medical Condition

A screenshot of a computer

Description automatically generated

Calculating Total Billing by Insurance Provider

A screenshot of a computer

Description automatically generated

Analysing the Number of Patients by Gender for Each Medical Condition

A screenshot of a computer

Description automatically generated

Calculating Total Billing by Medical ConditionA screenshot of a computer

Description automatically generated

**Visualization and Power BI:**

After performing the SQL queries in Synapse, I connected Azure Synapse Analytics to Power BI using serverless SQL pool endpoints.

The visualizations I created included:

* **Clustered Bar Chart** and **Clustered Column Chart** to compare patient counts and billing amounts.
* **Line Chart** to track changes over time.
* **Pie Chart** to show the distribution of patients or billing amounts by insurance provider.
* **Slicer** to allow interactive filtering of data.

A screenshot of a computer

Description automatically generated

1. **Conclusion:**

**Advantages:**

* **Automated Data Processing:** The data pipeline I developed efficiently automates the process of data ingestion, cleaning, transformation, and storage. This automation not only reduces manual intervention but also minimizes the risk of human error, ensuring consistent and reliable data processing.
* **Improved Data Quality:** By performing cleaning tasks such as removing duplicates, handling missing values, and standardizing column names and patient names, the overall data quality has been significantly enhanced. This ensures that the data is accurate and ready for meaningful analysis.
* **Efficient Storage and Performance:** Storing the transformed data in Parquet format helped optimize the use of storage space in Azure Data Lake, lowering storage cost and query performance. It enables faster reading of specific columns during queries. This makes it easier and faster to retrieve and analyse large datasets in Synapse Analytics.
* **Scalability:** Leveraging Azure services like Azure Data Factory, Databricks, and Synapse Analytics ensures that the pipeline can scale with increasing data volumes, making it a highly adaptable solution for larger datasets in the future.
* **Actionable Insights:** The ability to perform complex SQL queries in Synapse Analytics has enabled the extraction of valuable insights from the dataset. For instance, queries such as total billing by insurance provider and patient counts by medical condition offer deep insights that can support data-driven decision-making.

**Improvements:**

* **Advanced Data Cleaning:** While the current data cleaning process efficiently handles duplicates and null values, further improvements can be made by incorporating advanced techniques for detecting outliers and imputing missing values.
* **Data Enrichment:** To provide deeper insights, the dataset could be further enriched with additional information, such as more detailed patient demographics or geographical data, which would offer a more comprehensive view for analysis.
* **Enhanced Visualizations:** Although the Power BI visualizations provide key insights, additional enhancements could be made to improve interactivity and user experience. For example, adding more filters and interactive elements to the Power BI dashboard can provide a more user-friendly interface for end-users.

**Benefits for the Organization:**

* **Improved Decision-Making:** By automating the data pipeline and ensuring the data is cleaned and transformed, the organization can gain accurate insights into patient demographics, medical conditions, and billing trends. This allows for more informed decision-making, leading to improved patient care and operational efficiency.
* **Cost Savings:** Automating the data pipeline and leveraging cloud-based services, such as Azure Data Lake Storage and Synapse Analytics, helps reduce costs associated with manual data handling, on-premises infrastructure, and inefficient data processing methods. This results in cost savings over time.
* **Data-Driven Operations:** With access to high-quality, actionable data, the organization can make better decisions related to resource allocation, service delivery, and financial management, leading to better operational efficiency and improved patient outcomes.