**Data Engineering 2**

**NYC Taxi Data Pipeline**

**Project Report**

**Team Members**

**Ashwith Anand Poojary(11038008)**

**Muskaan Chhabra(11038196)**

**Prof. Mohammad Abufouda**

**Prof. Frank Schulz**

**Applied Data Science and Analytics**

**Abstract**

This project focused on creating a data pipeline to collect, process, and visualize New York Taxi data using Microsoft Azure services like Data Factory, Azure Data Lake Storage Gen2, Azure Synapse, and Power BI. The goal was to automate data collection, clean and transform the data, and develop insightful visualizations to support forecasting and analysis. Although technical issues impeded the full execution of the pipeline, the project demonstrated the potential of using Azure, Synapse, and Power BI for effective data management and analysis. The culmination of this effort is a Power BI dashboard that showcases the insights derived from the cleaned and processed New York Taxi data, underscoring the potential for enhanced decision-making through robust data engineering practices.

**Table of Contents**

1. Introduction
2. Work
3. Dataset(source)
4. Pipeline
5. Pros and Cons
6. Roles
7. Conclusion
8. Reference

**Report**

**Topic 1: Introduction**

In the realm of data engineering and pipeline management, various challenges can hinder the efficiency and effectiveness of data operations. Notable issues include handling an overwhelming volume of data while extracting valuable insights, designing comprehensive end-to-end data pipelines, managing a variety of data sources like sensor outputs, logs, and social media feeds, and executing complex real-time analyses. Challenges can be faced-

1. **Data Integration**

Data integration involves combining data from different sources to create a unified view, making it accessible and useful across an organization. Key techniques include ETL (Extract, Transform, Load), which involves extracting data from various sources, transforming it to fit operational needs, and loading it into a target database or data warehouse.

1. **Data Quality**

The data quality challenge involves ensuring that data is accurate, complete, timely, and consistent across various systems. High data quality is critical for reliable analytics and decision-making. Ultimately, maintaining high data quality ensures that an organization can trust and effectively use its data assets.

1. **Scalability and Performance**

The scalability and performance challenge in data engineering involves designing systems that can handle increasing volumes of data efficiently without degrading user experience or system stability. Performance issues often arise as data volumes grow, necessitating optimizations in database design, query execution, and data processing strategies.

1. **Real time data processing**

Real-time data processing involves analyzing and acting upon data as it is generated, without noticeable delay, posing significant technical challenges. Ensuring data accuracy and consistency in real-time scenarios adds complexity, especially under varying network conditions and system loads.

1. **End to End pipeline**

Constructing an end-to-end data pipeline involves several crucial phases such as data gathering, cleansing, analysis, visualization, and management. Each phase demands meticulous planning and specialized knowledge. Effective pipeline design includes choosing the right technologies, designing data workflows, maintaining data quality and integrity, and implementing strong data governance.

1. **Fault Tolerance**

Developing resilient pipelines capable of withstanding failures, rectifying errors, and preserving data integrity is essential. Addressing challenges such as system malfunctions, network interruptions, or data loss incidents requires robust technical strategies.

1. **Data Compatibility**

Achieving compatibility across various systems, tools, and technologies within a pipeline presents significant challenges. Managing data across diverse formats, databases, or cloud platforms necessitates the use of suitable data connectors and effective integration techniques.

**Solutions:**

1. **Monitoring and Observability**

Implement comprehensive monitoring and observability tools to gain insights into the performance and health of data pipelines. Tools like Prometheus, Grafana, and Elastic Stack can help monitor data flows and alert on anomalies.

1. **Continuous Integration/Continuous Deployment (CI/CD)**

Adopt CI/CD practices for data pipelines to allow frequent updates and consistent deployments. This includes automated testing and deployment of pipeline changes.

1. **Scalability Planning**

Design pipelines to be inherently scalable using cloud services like AWS Auto Scaling, Google Cloud Autoscaler, or Kubernetes for orchestration, which can dynamically adjust resources based on the workload.

1. **Data Quality Solution**

Build a data lake using technologies like Apache Hadoop or cloud solutions like AWS S3, which can store data in its native format and then process it using transformation tools as needed.

1. **Data Profiling and Monitoring**

Regularly profile data to identify anomalies, and monitor data quality using dashboards that alert to quality degradation.

1. **Real-Time Data Processing Solutions**

Implement stream processing frameworks like Apache Kafka to handle real-time data flows and analytics.

1. **Data Pipeline Complexity Solution**

Utilize workflow orchestration tools like Apache Airflow to manage complex data pipelines. These tools help define, schedule, and monitor workflows and use containers (Docker) and orchestration systems (Kubernetes) to manage and scale individual components of data pipelines independently.

**Topic 2 – Work**

Data pipelines are essential in the data ecosystem, allowing organizations to efficiently process and analyze vast amounts of data from multiple sources. This literature review examines the methods and technologies utilized in data pipelines, emphasizing their importance and tracing their development over time.

A key technique is Extract, Transform, Load (ETL), where data is extracted from various sources, transformed to fit specific needs, and loaded into a designated system. Tools like Informatica, Talend, and Microsoft SSIS have been instrumental in optimizing this process, offering robust features for data handling.

As data grows in volume and speed, stream processing has become vital, processing data in real-time as it flows. Technologies such as Apache Kafka, Apache Flink, and Apache Spark Streaming have pioneered this area, providing scalable and resilient platforms for data pipelines.

Cloud computing has popularized cloud-based data pipelines due to their scalability and adaptability. Providers like AWS, Google Cloud, and Microsoft Azure offer managed services that ease pipeline development and management, also offering cost-efficient solutions for data handling.

Data governance and compliance have also become critical, prompting the use of tools for data cataloging, metadata management, and data lineage to maintain data integrity and comply with regulations such as GDPR and CCPA.

In conclusion, data pipeline methods and technologies have evolved significantly, supporting the processing, transformation, and analysis of extensive data sets. This evolution encompasses everything from traditional ETL processes and real-time streaming to cloud solutions, containerization, and DataOps, continuously advancing to meet the needs of data-centric organizations. Adopting these technologies and practices can significantly enhance data pipeline implementations, leading to better insights and decision-making.

**Topic 3- Dataset (Source)**

The NYC Taxi and Limousine Commission (TLC) Trip Record Data consists of extensive and detailed records from yellow and green taxis as well as For-Hire Vehicles (FHV). For yellow and green taxis, the data encompasses key information such as pick-up and drop-off dates and times, locations, trip distances, itemized fares, rate types, payment types, and the number of passengers as reported by drivers. This data is collected through technology solutions approved under the Taxicab & Livery Passenger Enhancement Programs (TPEP/LPEP), ensuring a rich dataset for analysis. The For-Hire Vehicle records include data such as dispatching base license number, and pick-up date, time, and taxi zone location ID, providing insights into the operations and reach of licensed bases. The TLC does not create this trip data but publishes it as provided by the bases, emphasizing that while the data undergoes routine reviews and validation, its accuracy and completeness are based on submissions by the bases and therefore, may not reflect all trips dispatched by all TLC-licensed entities. This dataset is critical for analyzing urban transportation patterns, economic factors influencing taxi usage, and assessing the effectiveness of the regulatory framework governing New York City's vast taxi and FHV operations. Analyzing such data helps in understanding market dynamics, planning city logistics, and enhancing transportation services to better meet the needs of the public.

**Date and Time Fields**: The TLC Trip Record Data contains critical temporal details such as pick-up and drop-off dates and times.

**Location Data**: The dataset includes detailed location fields capturing pick-up and drop-off points.

**Trip Distance**: The recorded trip distances provide insights into the travel behavior of passengers, the efficiency of routes taken by drivers, and the environmental impact of taxi operations.

**Fares and Payment Types**: The dataset breaks down fares into various components, including the base fare, taxes, tolls, and tips.

**Rate Types**: Rate type information indicates whether a trip is charged on a standard metered rate, negotiated price, or under a special rate category such as trips to and from airports.

**Passenger Counts**: The driver-reported count of passengers provides demographic insights and helps in understanding the utilization rates of taxi services.

**Topic 4-**

**Data Pipeline Architecture**

A diagram of a diagram

Description automatically generated

Application and Services used according to the flow:

1. Github raw URL
2. Data Factory
3. Azure Data Lake Storage Gen 2
4. Azure Databricks
5. Azure Synapse Analytics
6. Power BI

**Microsoft Azure:**

Microsoft Azure offers a comprehensive suite of services to build robust data pipelines, empowering users to ingest, store, process, and analyze massive amounts of data. Azure Data Factory orchestrates and automates data movement and transformation, while Azure Data Lake Storage Gen2 provides a highly scalable and secure storage solution. Azure Databricks enables powerful data processing capabilities with Apache Spark, seamlessly integrating with other Azure services. Azure Synapse Analytics merges big data and data warehousing technologies, facilitating complex analytical queries. Lastly, integration with Power BI allows for sophisticated data visualization and business intelligence capabilities, making insights accessible across organizations.

We will be using the subscription purchased as Free-Trail.

A close up of a word

Description automatically generated

This has $200 of credits provided by Microsoft for a month and so far this project has taken up $14.5 credits. So it is very much scalable.

A black text on a white background

Description automatically generated

A screenshot of a graph

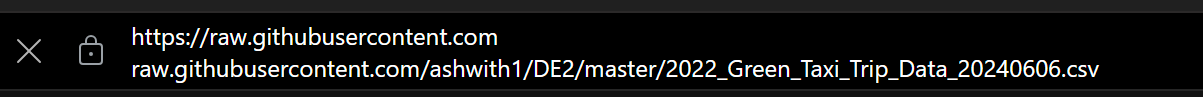
Description automatically generated

**Data Acquisition**:

The initial stage involves downloading the CSV files containing the TLC Trip Record Data. These files are typically large and detailed, requiring efficient handling right from the beginning. Once downloaded, the files are uploaded to GitHub, a platform used here not just for version control but also as an intermediate and accessible storage for raw data. This approach ensures that the raw data can be tracked and accessed at any stage of the pipeline, facilitating collaboration and transparency. From here we can receive it’s raw URL to act as a Sink.

A black rectangular object with white text

Description automatically generated



Keep the raw URL saved for now

**Azure Data Factory**:

Azure Data Factory is utilized to automate the data flow from GitHub to Azure Data Lake Storage Gen2. Data Factory acts as the orchestration layer in this pipeline, enabling the automation of data loading processes. It efficiently connects to GitHub, retrieves the latest data files, and securely transfers them to Azure Data Lake, ensuring that the data pipeline is scalable and maintainable.

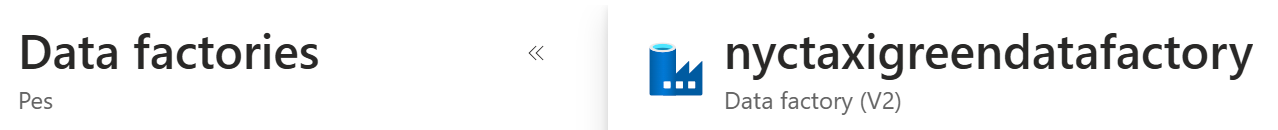
In Azure we first create a resource group and a storage account



A screenshot of a computer

Description automatically generated

Then we create and deploy Data Factory



In this we click on Launch Studio and it takes us to it’s page, here we open a new pipeline and use the CopyData function and drag it to the pipeline

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

You can name it and go to the source, we create a new source to Github raw url as Source and use the Storage Account in Azure as the sink. Here we do it to the raw-data folder in the storage container that was created inside the Storage Account named as below

After setting up the source and sink, we validate it and publish it to save it and then click on the Debug option to run it.

Here it has run successfully after a while.



To confirm, we can check the storage account and yes, it’s confirmed to have worked well.

A screenshot of a computer

Description automatically generated

**Azure Data Lake Storage Gen2**:

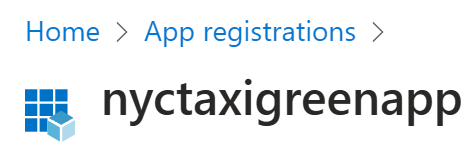
Azure Data Lake Storage Gen2 serves as the primary storage solution in this architecture. Initially, raw data from GitHub is stored here. After processing in Azure Databricks, the transformed data is stored back into Data Lake Gen2. This storage solution is highly scalable, secure, and optimized for big data operations, making it ideal for handling large volumes of taxi trip data.

**Azure Databricks**:

Azure Databricks is used for data processing and transformation. It integrates seamlessly with Azure Data Lake Storage Gen2, allowing direct access to the

stored data. In Databricks, data engineers and scientists can use Apache Spark to perform complex data transformations, clean-up operations, and feature engineering needed to prepare the data for analytical queries and reporting. In azure data bricks we have faced restrictions as mounting with Data Lake Storage Gen 2 is a complex task. This includes many steps:

We first create and deploy App Registrations in Azure. Here we receive the client id, tenant id and the client secret key.



This is used for receiving the data from the storage account. We then should provide access to this under the Storage Account container and provide this app as a role of Blob Contributor.

A screenshot of a computer

Description automatically generated

We then create and deploy Azure Databricks

A black text on a white background

Description automatically generated

After it’s done, we launch the workspace. After this it directs us to databricks, we must go to compute and create a cluster. After that we can create a notebook and work on the transformation.

A screenshot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

In this we work on the code,

This is the default way to mount the data, we provide the necessary values like client id, tenant id and the client secret key.

configs = {"fs.azure.account.auth.type": "OAuth",

"fs.azure.account.oauth.provider.type": "org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider",

"fs.azure.account.oauth2.client.id": "",

"fs.azure.account.oauth2.client.secret": '',

"fs.azure.account.oauth2.client.endpoint": "https://login.microsoftonline.com/tanent\_id/oauth2/token"}

dbutils.fs.mount(

source = "abfss://tokyo-olympic-data@tokyoolympicdata.dfs.core.windows.net", # contrainer@storageacc

mount\_point = "/mnt/tokyoolymic",

extra\_configs = configs)

Spark is already enabled so there’s no need to initialize it again.

In this we check for Null values and work on cleaning the data.

A screenshot of a computer

Description automatically generated

In this we check for duplicate values and work on cleaning the data.

A screen shot of a computer

Description automatically generated

Then we plot graphs to understand the data better and it helps us understand on how and why we have to run queries on it.

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

We then save it back to the Storage Account container under the folder of transform-data.

A screenshot of a computer

Description automatically generated

**I have then set this up as a pipeline in Data Factory to ensure it is part of the ETL pipeline.**

A screenshot of a computer

Description automatically generated

**This is then set up as a pipeline and connected it to CopyData.**

A green line on a black background

Description automatically generated

**Both are working when running it together as demonstrated.**

A screenshot of a computer

Description automatically generated

**Azure Synapse Analytics**:

After processing, the data is loaded into Azure Synapse Analytics. Synapse is a powerful analytics service that combines big data and data warehousing. It enables running big data analytics directly on the transformed data stored in Data Lake Gen2, allowing for the creation of scalable and efficient data models that are optimized for performance.

We first create and deploy the Azure Synapse Analytics.

A close up of a logo

Description automatically generated

Before we proceed, we need to provide access from Storage Account to Databases in Synapse. Hence, we go to Storage Account container, we then open the section Access Control (IAM), here we provide a role to the synapse services created.

A screenshot of a computer

Description automatically generated

We then open Synapse Studio, in this we go to Data section,

Here we created a Data Lake Database, after this do add table and connect from Storage Container to this database.

A screenshot of a computer

Description automatically generated

Here we create SQL script and work on the table according to the user stories.

1. User Story 1

As a Financial Analyst, I want to find the average distance of trips by weekday, so that I can analyze busiest days of a week.

A close-up of a computer screen

Description automatically generated A screenshot of a computer

Description automatically generated

1. User Story 2

As a financial analyst, I want to predict monthly fare amount trends based on data so that we can better forecast budget allocations and financial planning.

A computer screen shot of a number of words

Description automatically generated A screenshot of a computer

Description automatically generated

1. User Story 3

As a data analyst at a taxi company, I want to analyze top pickup locations to identify top locations so that we can optimize driver schedules and increase revenue.

A screen shot of a computer code

Description automatically generated A screenshot of a graph

Description automatically generated

1. User Story 4

As a business manager, I want to analyze how different types of trips contribute to our overall revenue. So that I can optimize our service pricing strategies to increase profit.

A screenshot of a computer code

Description automatically generated A screenshot of a graph

Description automatically generated

**Power BI Visualization**:

The final step involves visualizing the analyzed data using Power BI. Power BI is connected to Azure Synapse Analytics, where it can access the prepared datasets. Through Power BI, various dashboards and reports are created to provide insights into the data, such as travel patterns, fare statistics, and operational efficiency. These visualizations are crucial for stakeholders to make informed decisions and to easily comprehend complex data patterns.

**Visualization**

Here we use Power BI where we provide the query tables and the transform data table.

We have created the interactive dashboard for the following.

Below is the dashboard screenshot, which has designed based on business queries and representing the results of these queries.

A screenshot of a computer dashboard

Description automatically generated

We developed five user stories which are transformed into business queries that form the basis of our data analysis endeavours. By crafting specific queries that address the requirements outlined in the user stories, we ensure that our analysis remains focused and pertinent to the objectives of our project. These business queries guide our data exploration, helping us to identify critical variables, datasets, and metrics crucial for generating meaningful insights.

**Topic - 5**

**Pros and Cons**

1. **Azure Data Factory**

Pros:

* + Automation and Orchestration- Automates data flows, making it ideal for complex ETL and ELT workflows.
  + Integration- Connects seamlessly with various data sources and Azure services, facilitating a comprehensive data integration solution.
  + Scalability - Highly scalable, handling large volumes of data effectively without degradation in performance

Cons:

* + Cost - Can become expensive as data volumes and complexity of data workflows increase.

1. **Azure Data Lake Storage Gen2**

Pros:

* + Performance - Optimized for big data analytics with a hierarchical namespace, improving file management and system performance.
  + Security - Integrates with Azure Active Directory and supports access control lists (ACLs) for robust security and compliance.

Cons:

* + Complexity in Management - Managing large data sets and access can become complex, especially for users unfamiliar with cloud storage concepts.

1. **Azure Databricks**

Pros:

* + Powerful Processing - Offers a collaborative environment with Apache Spark, which is ideal for data processing, machine learning, and real-time analytics.
  + Integration - Native integration with other Azure services like ADLS Gen2, Cosmos DB, and Synapse Analytics.

Cons:

* + Complexity - Premium service with higher costs, especially for large scale or compute-intensive operations.

1. **Azure Synapse Analytics**

Pros:

* + Analytical Power - Combines big data analytics with data warehousing, providing a single service for all analytics needs.
  + Performance - Offers dedicated SQL pools for high-performance querying and data manipulation.

Cons:

* + Cost - The cost can be significant, especially when using on-demand query services or large dedicated SQL pools.

1. **Power BI**

Pros:

* + Visualization Capabilities - Robust visualization tools with real-time dashboard capabilities.
  + Integration - Strong integration with Azure services and other data sources, making it versatile for various use cases.
  + Easy to use - Relatively easy to use with a drag-and-drop interface and extensive documentation and community support.

Cons:

* + Data Volume Limits - Certain plans have data volume limits, which can be restrictive for large datasets.

**Topic- 6**

**Roles**

|  |  |  |  |
| --- | --- | --- | --- |
| Ashwith Anand Poojary | Data Pipeline Integration | Visualization | User Stories |
| Muskaan Chhabra | Data Transformation | Azure Synapse Analytics | User Stories |

**Topic- 7**

**Conclusion**

1. Developing an Effective Data Pipeline with Microsoft Azure
2. Leveraging Tools Such as Databricks and Azure Synapse Analytics for Enhanced Data Management
3. Addressing Business Challenges Through Responsive User Query Resolution
4. Overcoming Obstacles with Success
5. Visualizing transform data using NYC Taxi Data Power BI Dashboard

**Topic- 8**

**References**

1. <https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page>
2. <https://medium.com/@Fehintiti/nyc-green-taxi-e159b796c0f3>
3. <https://github.com/toddwschneider/nyc-taxi-data>
4. https://www.linkedin.com/pulse/designing-effective-data-pipelines-azure-guide-ashfaq-pathan-/