Assignment #1

Data Warehouse Implementation Project

ITMD-526

ILLINOIS TECH

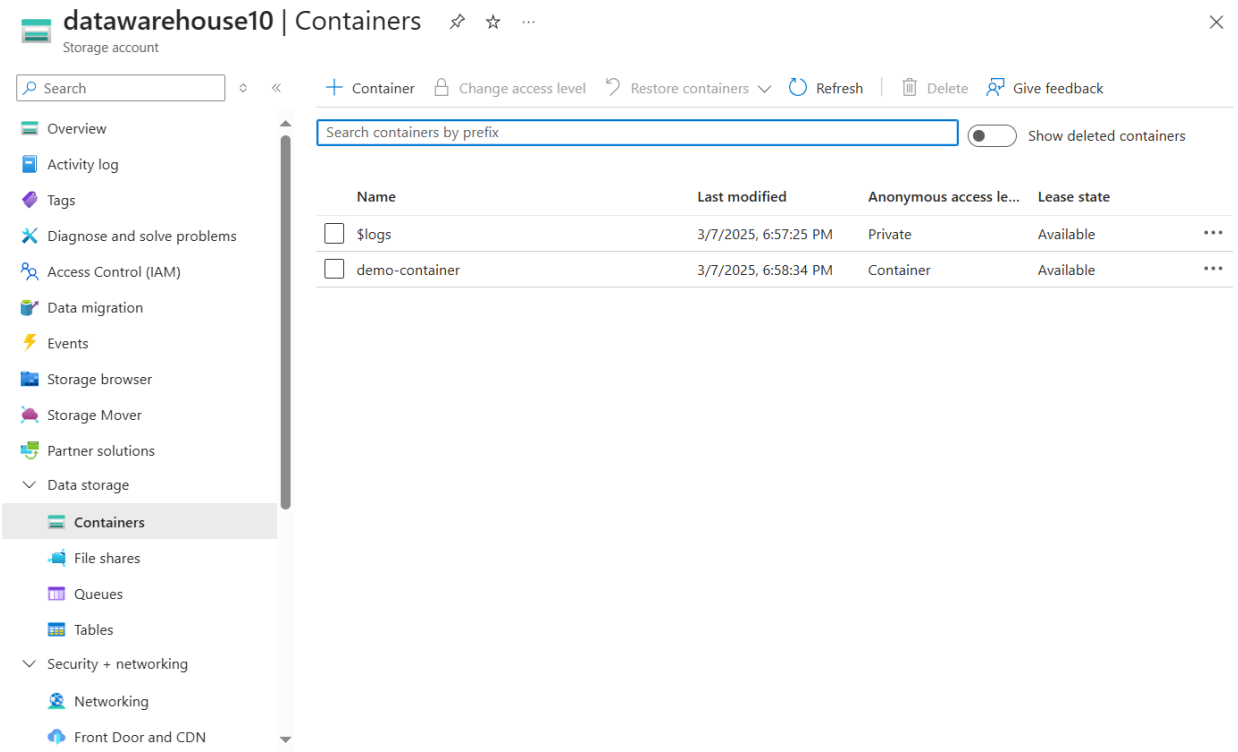
ILLINOIS INSTITUTE OF TECHNOLOGY

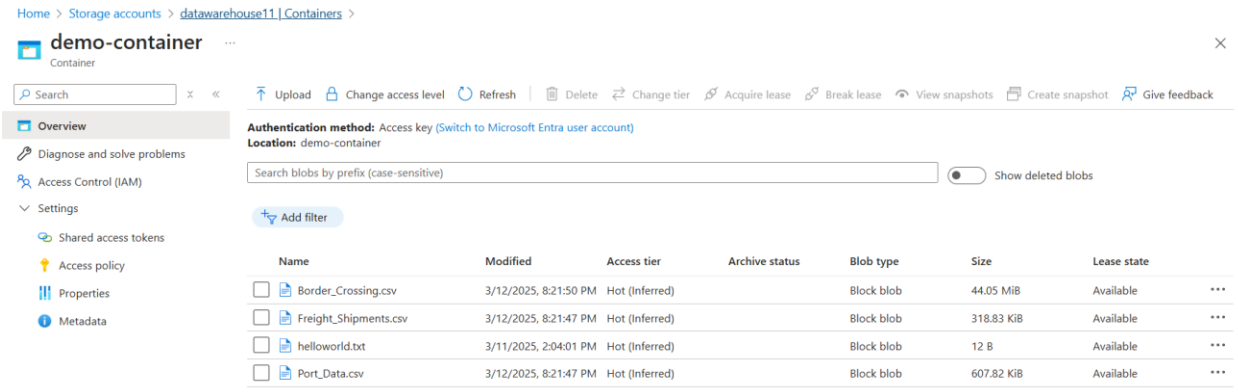
Ignacio Hidalgo Power

Summary of the project

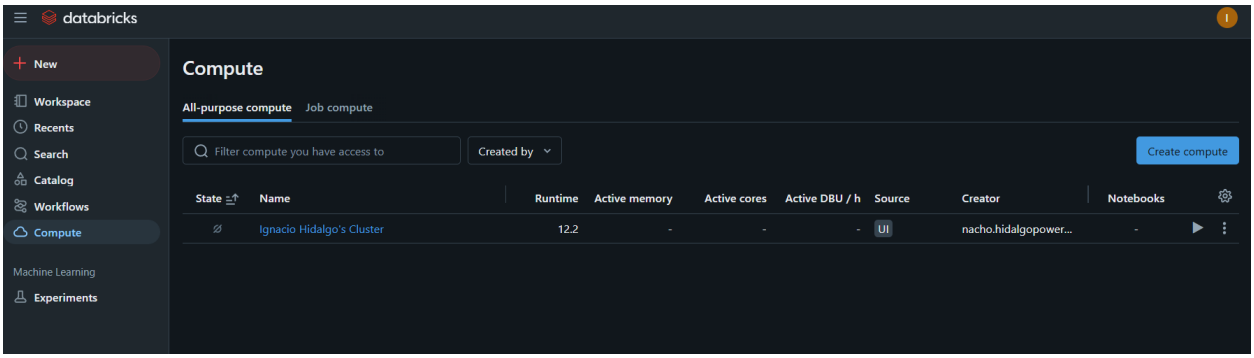
The objective of the project was to implement a data warehouse that could connect to an external, azure based, storage unit that held the data sources.

The first step was gathering the data. In order to have reliable, up to date, real world data we used the Bureau of Transport Statistics from the US. Once we had the data and we stored it into a container on the Azure platform we implemented through Databricks an ETL that connected to the Container through a SAS.





Once the connection was through, we stored the CSV files onto a Spark Data Frame and preprocessed the data on the Spark Cluster to make use of the parallelization that is supported by Spark Data Frames, instead of the pandas one.



When the preprocessing was done, we normalized the data. Our approach, to saving resources and make use of the optimization that databricks allows, consisted on doing the preprocessing in the data frame and then convert it into a SQL Table, stored in our Logistics_DB. This made use of the functionality that Spark offers of transforming the data frame into a table and not requiring a DDL script.

```
from pyspark.sql.functions import col, regexp_extract

df_shipment_filtered = df_shipment.select(
    col("Mode"),
    col("Statistic"),
    col("Year"),
    col("Value"),
    col("Units"),
    col("Statistic Short Name").alias("StatisticShortName")).filter(
        (col("Mode")=="Water")  |
        (col("Mode")=="Rail") |
        (col("Mode")=="Air and truck-air") |
        (col("Mode")=="Air") |
        (col("Mode")=="Pipeline") |
        (col("Mode")=="Highway"))
```

Once we had the tables stored we made the Dimensions and Facts tables. One dimension table for each of the different sources, Ports, Shipments and Borders, and fact tables for the Ports that are at each border and another for trades.

Challenges:

There were many challenges throughout the project. The first was the Azure Blob Storage that we wanted to implement. We first had in mind to use Azure Databricks, thanks to the ease of use, rich resources online, and ease of integration with the storage. However, we were unable to use it due to the nature of our Azure subscription and the scope that was limited to us.

When the Azure Databricks were unable for us, we moved onto Databricks. To improve performance and security we tried to launch Databricks locally to increase the number of resources we could use and to make use of the key vaults that can be used and give extra security to our cluster, but because we were using the community edition, we were also unable to do this.

Thankfully Databricks community edition supports a VM of 15GB of memory and 2 cores for free, so we could use that for our cluster. However, one thing that this VM had was that the way to connect to the Azure storage was through the web using the SAS key anonymously. But because of our lack of experience with Azure we had set the security requirements to enter the container too strict and for some days we were stuck being unable to connect. To fix this we had to create another storage unit with another container which had a less strict security requirement.

Our next challenge came with preprocessing. The data we gathered was quite dirty and dispersed, so we were forced to filter with the data frames functionality all of data, putting limits and taking away rows that shouldn't be there.

Our last challenge was that when deploying the normalized data we had to delete and create a new compute cluster due to the Databricks Community policy. When this happened all the memory databases and tables were removed, but the cache stored the names of the tables. To be able to restart and re run the DDL scripts that generated our tables we made use of the SQL syntaxis 'CREATE OR REPLACE TABLE', that way when the memory saw that the name was already in use it would just replace it.

Analysis:

After seeing the implementation of the dimensional model, there are possible improvements to do. For future development we should be able to create more facts tables and further increase the number of dimensions tables.

A goal is to generate tables that specialize in certain queries that can be used for analysis. Along with using those new tables we would also like to improve the optimization of the table through the use of SQL techniques, like indexes and temp tables.

Future Improvements:

Our main goals for future improvements are:

- Improve security connection: We want to improve the security of our cloud storage, so that if it were to store sensitive data it would comply with strict security metrics.
- DB Schema: We feel that the schema design can be improved as if to allow for better analysis, faster queries and more robust data quality.
- Report Generation: We want to implement a report generation that automatically gathers the data from the source if there has been an update, preprocess it, analyses the data, generates the report and stores it in a Azure Container in a different storage account.