AASD 4012 - Final Project

Group 1:

Cheuk Yin (Natalie) Li (101386432)
Chuen Kei (Ken) Ho (101410183)
Chun Cheong (Isaac) Mak (101409987)
Hei Yuet (Jessica) Lee (101409639)
Ka Ho (Foster) Cheung (101422288)
Ka Tsun (AI) Chan (101420274)
Man Him (Thomas) Lam (101411299)
Shui Hei (Salina) Yung (101409756)
Sik Yin (Samuels) Sun (101409665)
Siu Fung (Todd) Chan (101427740)
Zifeng (Philip) Chen (101411338)

Background

The TTC has a major problem with constant service disruption

- Line 3 Scarborough SRT early closure on 23 Feb

However, with the above unsatisfactory experiences, TTC decided to make some unexpected changes after 26 March

- TTC increasing fare to $3.35.
- Longer wait times for most lines

Problem statement

How does the weather affect the TTC service of subway? If not, what is the factor affecting TTC service the most?

Data

From 5th of January, 2014 to 14th of November, 2018

Toronto_weather.csv
(Structured)

- Rain and snow data

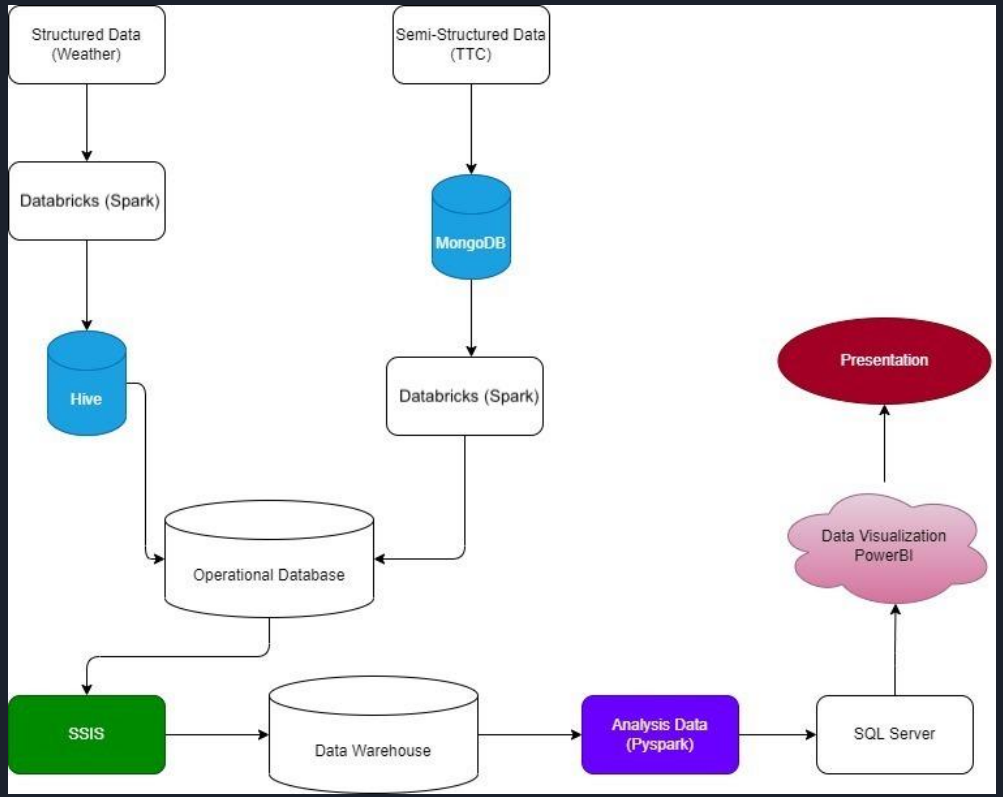
toronto_subway.json
(Semi-structured)

- Service delay data
- Location data

Tools

- Hive (Structured data)
- MongoDB (Semi-structured data)
- Spark (Big data operator)
- DataBricks (Data Sharing/Deploying)
- SSIS (Data integration)
- PySpark (Spark applications using Python, Data Analysis)
- PowerBi (Data visualization)

Overview of the Project



Hive & Databricks for Structured Data

```
# File location and type
file_location = "/FileStore/tables/Toronto_weather.csv"
file_type = "csv"

# CSV options
infer_schema = "false"
first_row_is_header = "false"
delimiter = ","

# The applied options are for CSV files. For other file types, these will be ignored.
df = spark.read.format(file_type) \
  .option("inferSchema", infer_schema) \
  .option("header", first_row_is_header) \
  .option("sep", delimiter) \
  .load(file_location)
```

```
%sql
CREATE OR REPLACE TEMPORARY VIEW weather
USING CSV
OPTIONS (path "/FileStore/tables/Toronto_weather.csv", header "true", inferSchema "true")
```

- Import data into the DBFS of Databricks
- Create a temporary view (weather) storing the data by SQL

Hive & Databricks for Structured Data

```
%sql
CREATE TABLE hive_weather
USING hive
AS SELECT * FROM weather
```

```
table_name = "hive_weather"
output_path = "/FileStore/tables/hive_weather"

df_h = spark.table(table_name)
df_h.write.option("header", "true").csv(output_path)
```

- Create a table (hive_weather) using hive by selecting all data from view (weather)
- Write the hive_weather as a csv file and store it in the Databricks

Hive & Databricks for Structured Data

```scala
%scala

import java.util.Properties

val jdbcHostName = "aasd4012group1projectserver.database.windows.net"
val jdbcPort  =1433
val jdbcdbName = "group1project"
val myproperties = new Properties()

myproperties.put("user", "azureuser")
myproperties.put("password", "IamfromHK1997")

val url = s"jdbc:sqlserver://${jdbcHostName}:${jdbcPort};database=${jdbcdbName}"
val driverClass = "com.microsoft.sqlserver.jdbc.SQLServerDriver"
myproperties.setProperty("Driver", driverClass)
```

- Use jdbc to connect the databricks to the operational database
- Send the hive_weather.csv to the operational database by jdbc (Scala)

```scala
%scala
val mydf = spark.read.format("csv").option("header","true").option("inferSchema", "true").load("/FileStore/tables/hive_weather/part-00000-tid-1608839276936960033-fbc64133-50e3-49ab-a36e-eef7a9393590-26-1-c000.csv")
display(mydf)
```

```scala
%scala
mydf.write.jdbc(url, "hive_weather", myproperties)
```

MongoDB & Databricks for Semi-structured Data


```
{
    "_id" : ObjectId("6407e7a982849c571ceb97a6"),
    "Date" : "2014/01/11",
    "Time" : "22:35",
    "Day" : "Saturday",
    "Station" : "BLOOR STATION",
    "Code" : "MUPR1",
    "Min Delay" : NumberInt(11),
    "Min Gap" : NumberInt(17),
    "Bound" : "N",
    "Line" : "YU",
    "Vehicle" : NumberInt(5288),
    "Vehicle Type" : "SUB",
    "CODE DESCRIPTION" : "Priority One - Train in Contact With Person"
}
```

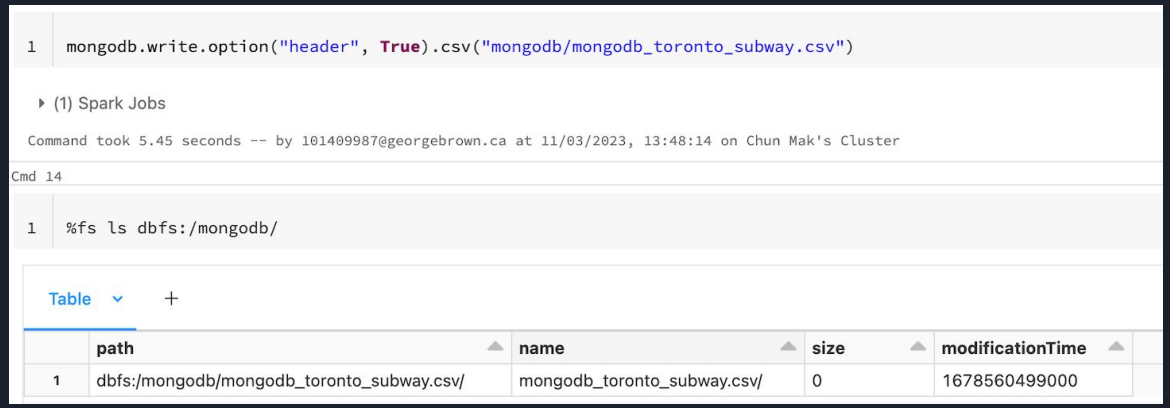

- Upload the data to MongoDB Atlas
- Connect the Databricks to the collection on the MongoDB Atlas

```
database = "big_data_project" #your database name
collection = "toronto_subway" #your collection name
connectionString = "mongodb+srv://philip11997:ab65495zxb@cluster0.hok2etk.mongodb.net/big_data_project?retryWrites=true&w=majority"
```

```
spark = SparkSession.builder.config('spark.mongodb.input.uri',connectionString).config('spark.mongodb.input.uri',
connectionString).config('spark.jars.packages', 'org.mongodb.spark:mongo-spark-connector_2.12:3.0.1').getOrCreate()
```

```
mongodb = spark.read.format("com.mongodb.spark.sql.DefaultSource").option("uri", connectionString).option("database", database).option("collection",
collection).load()
```

MongoDB & Databricks for Semi-structured Data



```
1  mongodb.write.option("header", True).csv("mongodb/mongodb_toronto_subway.csv")
```

▸ (1) Spark Jobs

Command took 5.45 seconds -- by 101409987@georgebrown.ca at 11/03/2023, 13:48:14 on Chun Mak's Cluster

Cmd 14

```
1  %fs ls dbfs:/mongodb/
```

Table +

	path	name	size	modificationTime
1	dbfs:/mongodb/mongodb_toronto_subway.csv/	mongodb_toronto_subway.csv/	0	1678560499000

- Preprocess the data in Databricks
- Write the data as a csv file (mongodb_toronto_subway.csv) and store it in the databrick

MongoDB & Databricks for Semi-structured Data

```scala
%scala

import java.util.Properties

val jdbcHostName = "aasd4012group1projectserver.database.windows.net"
val jdbcPort  =1433
val jdbcdbName = "group1project"
val myproperties = new Properties()

myproperties.put("user", "azureuser")
myproperties.put("password", "IamfromHK1997")

val url = s"jdbc:sqlserver://${jdbcHostName}:${jdbcPort};database=${jdbcdbName}"
val driverClass = "com.microsoft.sqlserver.jdbc.SQLServerDriver"
myproperties.setProperty("Driver", driverClass)
```

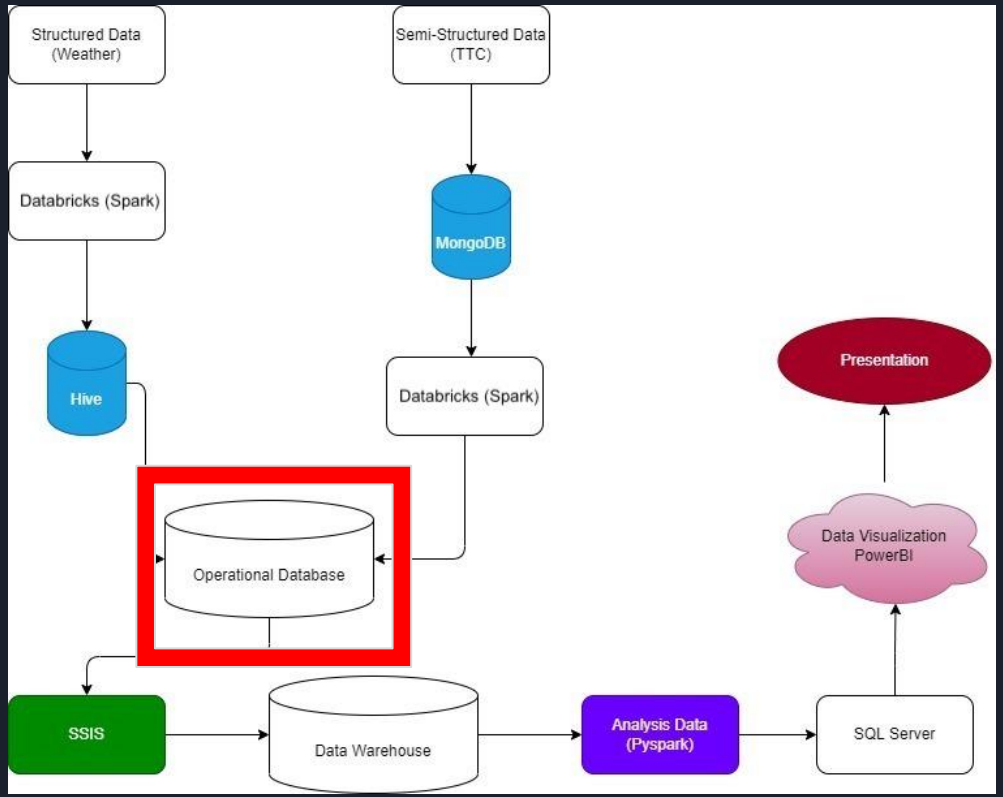
- Use jdbc to connect the databrick to the operational database
- Send the mongodb_toronto_subway.csv to the operational database by jdbc (Scala)

Scala

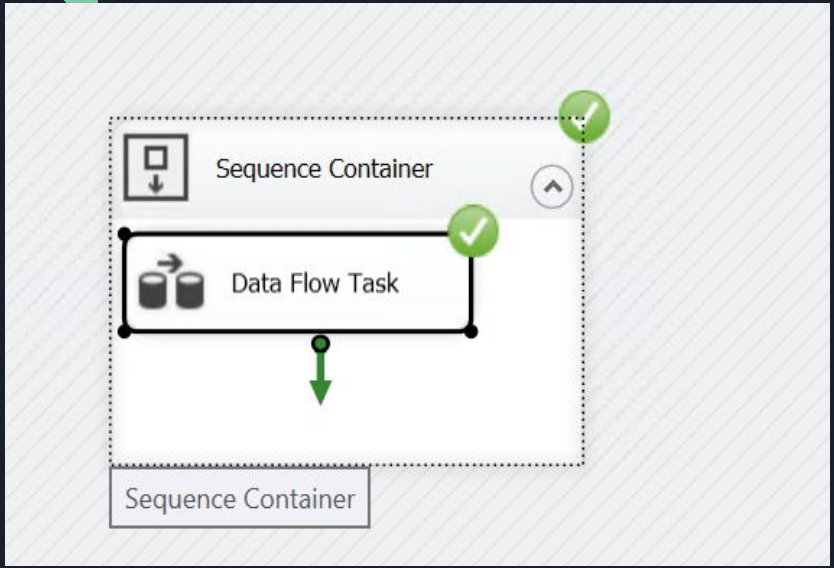
```scala
%scala
val mydf = spark.read.format("csv").option("header","true").option("inferSchema", "true").load("dbfs:/mongodb/mongodb_toronto_subway.csv")
display(mydf)
```

```scala
%scala
mydf.write.jdbc(url, "mongodb_toronto_subway", myproperties)
```

Overview of the Project (After Hive & MongoDB)

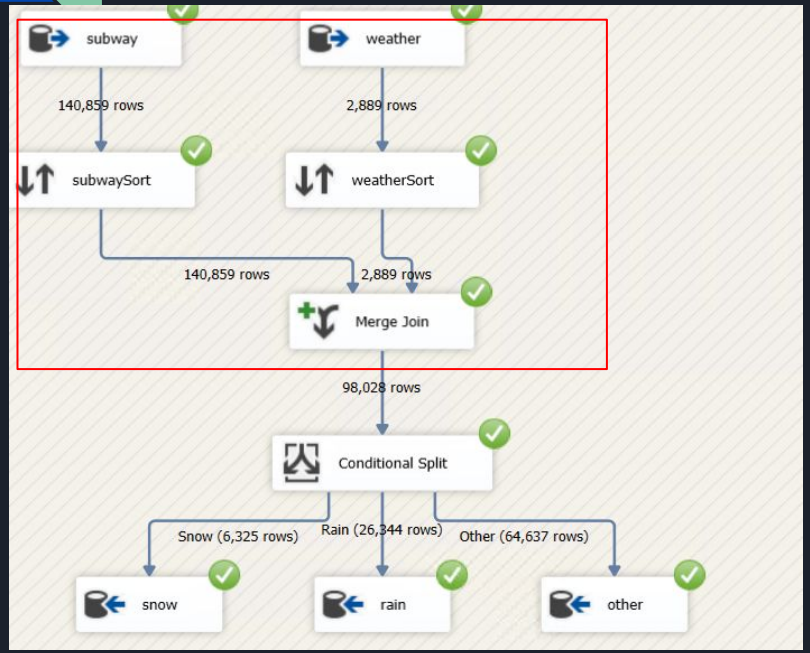


SSIS



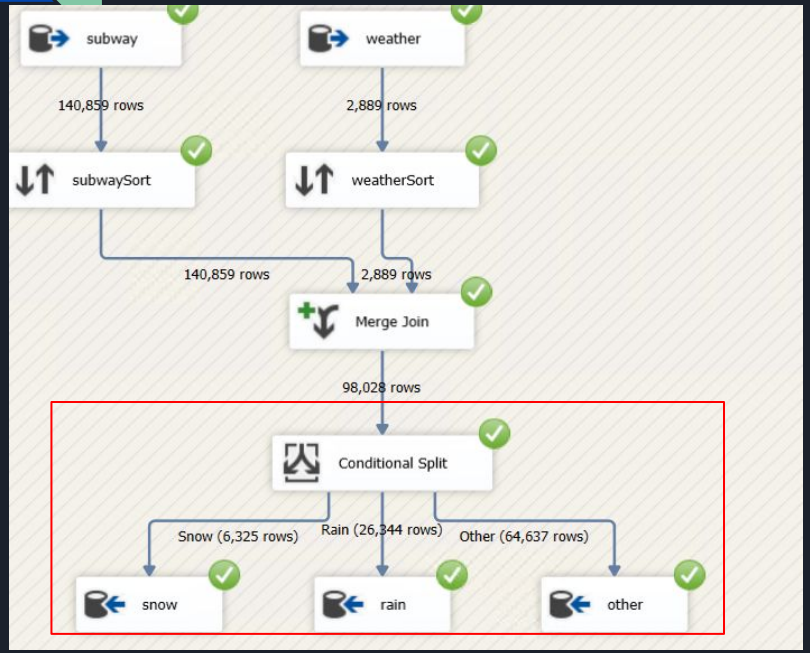
- Under control flow, create a sequence container to store data flow task

SSIS



- Connect to the operational database using ADO.NET connection manager
- Connect to tables (hive_weather & mongodb_toronto_subway)
- Sort data according to date and time in ascending order
- Apply merge(inner) join in weather table and subway table using the key of Date
- TTC delay and weather data in 2014 - 2018 remains

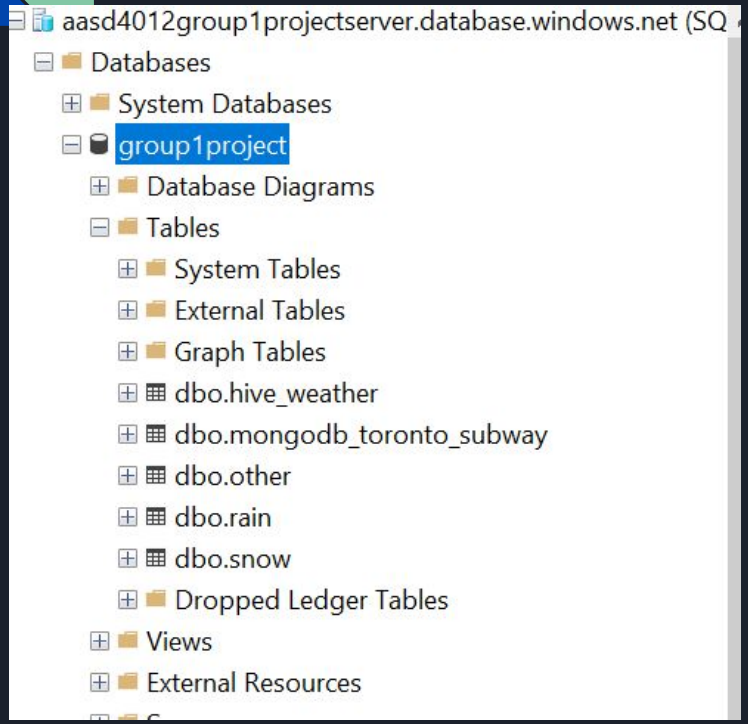
SSIS



- Split the datasets according to three conditions (snow/rain/other)

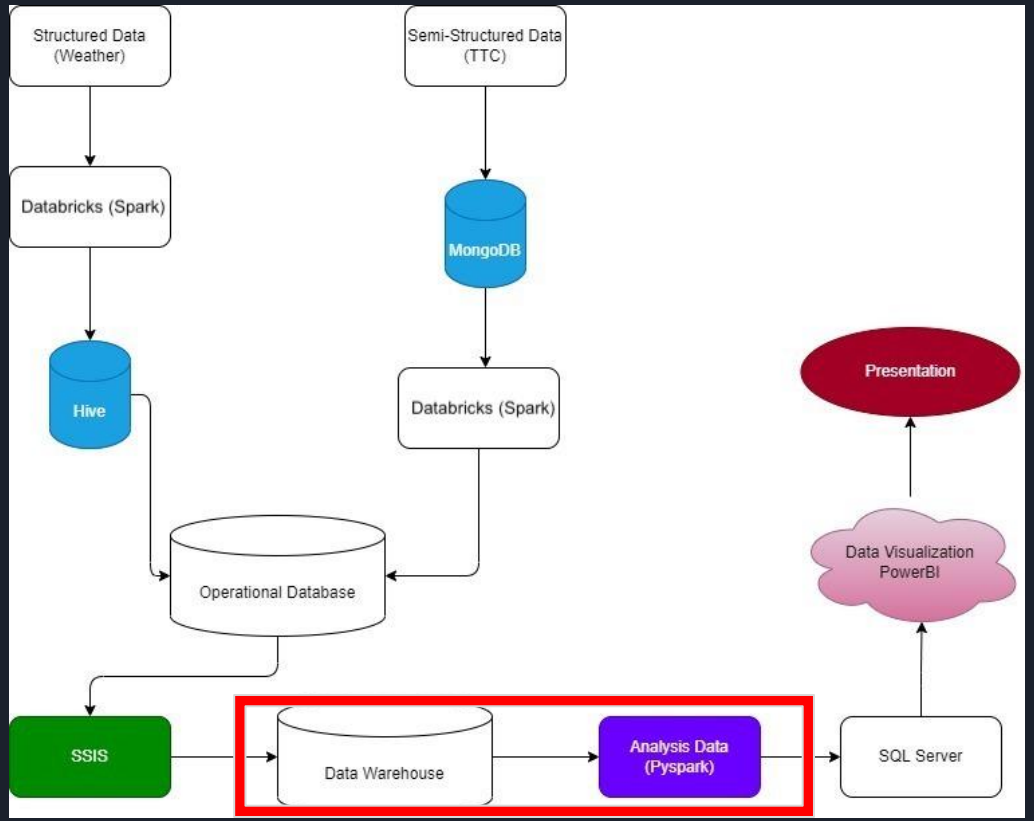
- Case 1: Snow > 0 and Snow not null
- Case 2: Rain > 0 and Rain not null
- Case 3: Snow not null and Rain not null and Snow equals to 0 and Rain equals to 0

SSIS

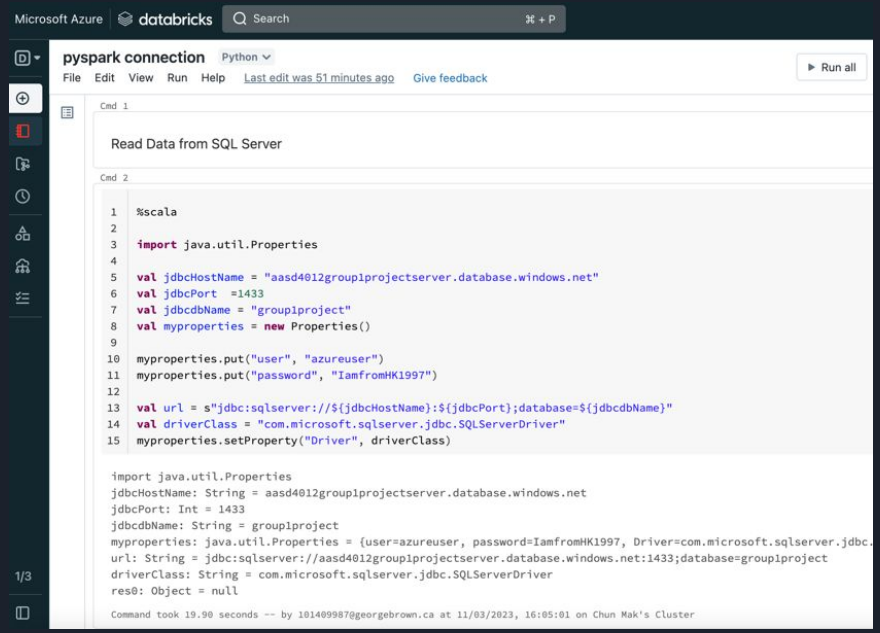


- Output the data (snow/rain/other) to corresponding table to the data warehouse using ADO.NET connection manager

PySpark & Databricks

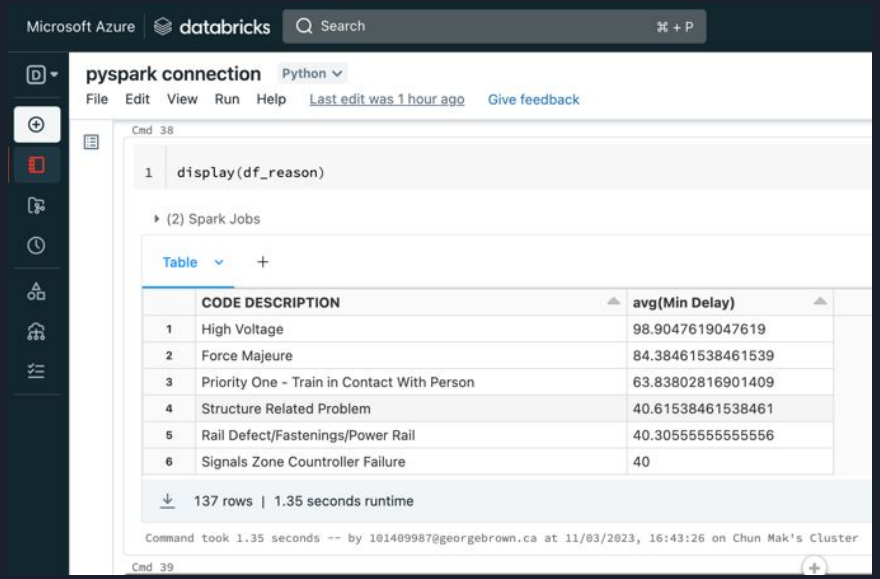


PySpark & Databricks



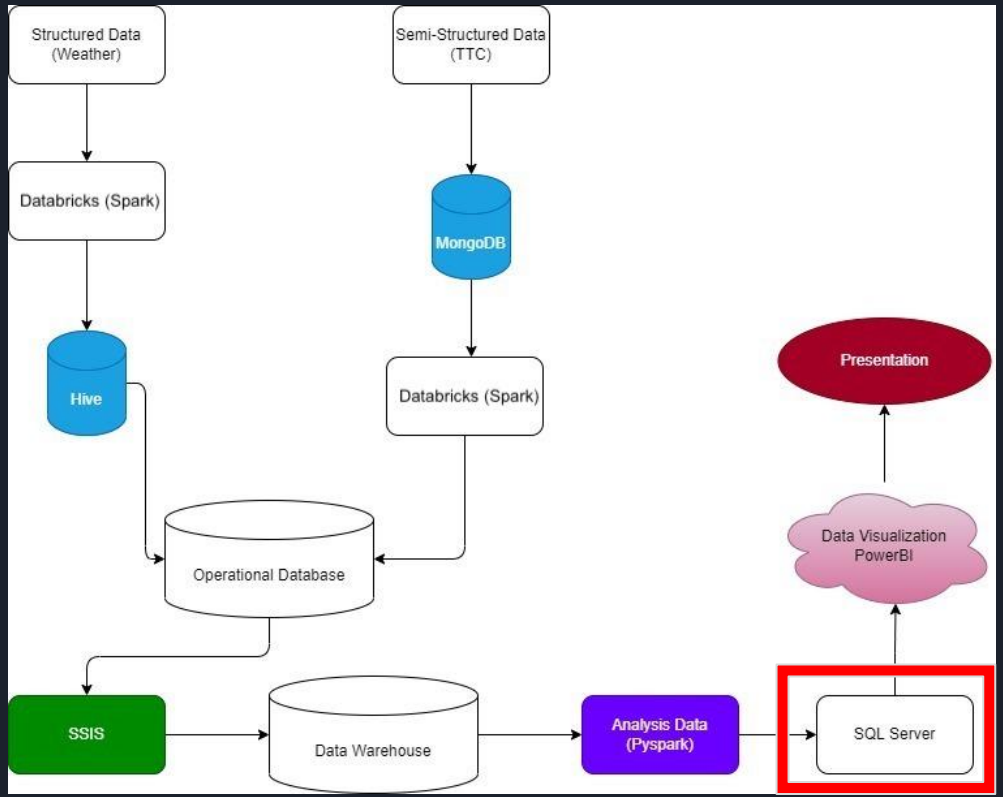
- Use jdbc to connect the databricks to the data warehouse where there are 3 tables generated in the previous steps
- Read 3 tables (snow, rain & other) in the database
- Apply groupby to calculate the average rate of rainfall/snowfall in each station and sort by descending order and by calendar month
- Use jdbc to connect the databrick to the database (group1project) in the SQL server (aasd4012group1projectserver)
- Output the result files in csv format to the database (group1project) by jdbc

PySpark & Databricks



- Initial thought: weather events do not cause significant delay
- More details to be discussed in the next section with PowerBI

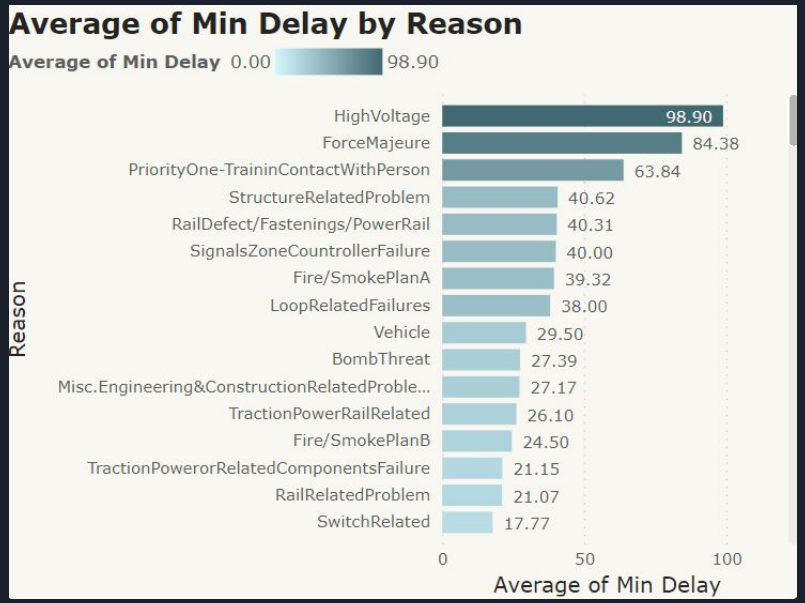
Overview of the Project (After PySpark & Databricks

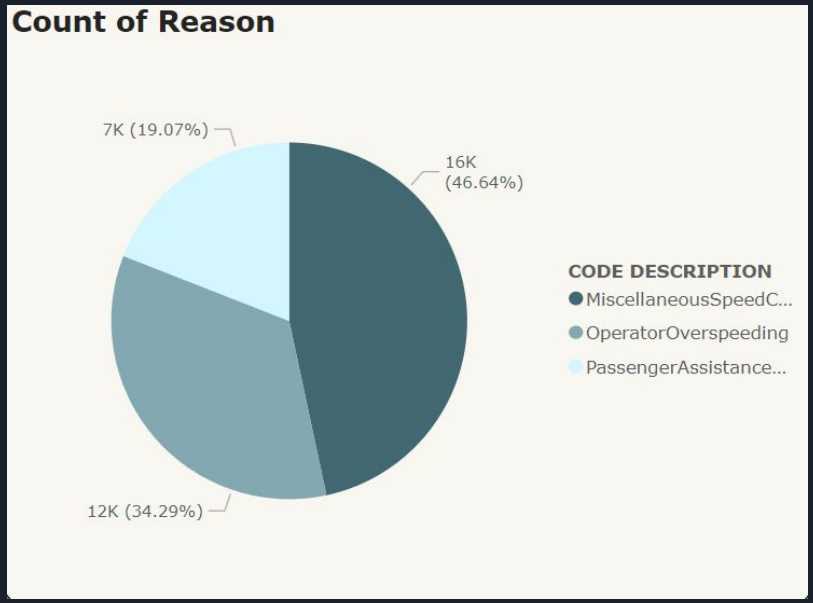


Power BI

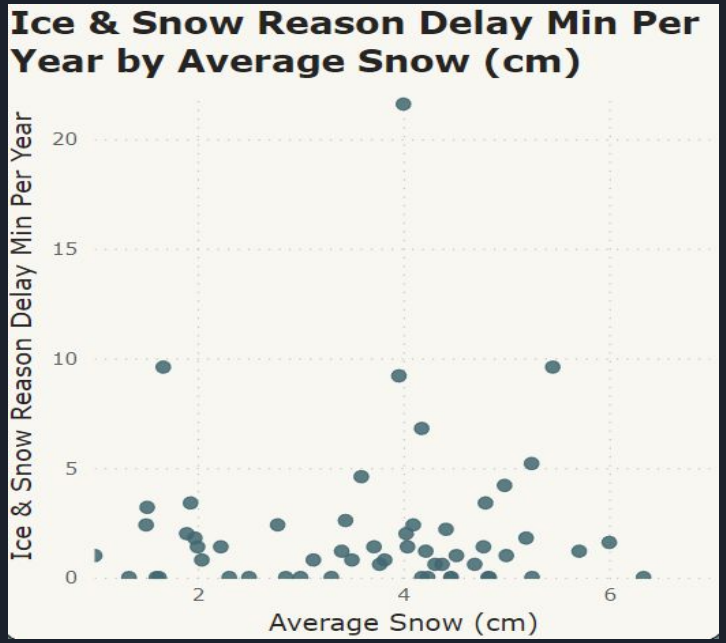
38.53K
Min Delay Per Year

122
Ice & Snow Min Delay Per Year



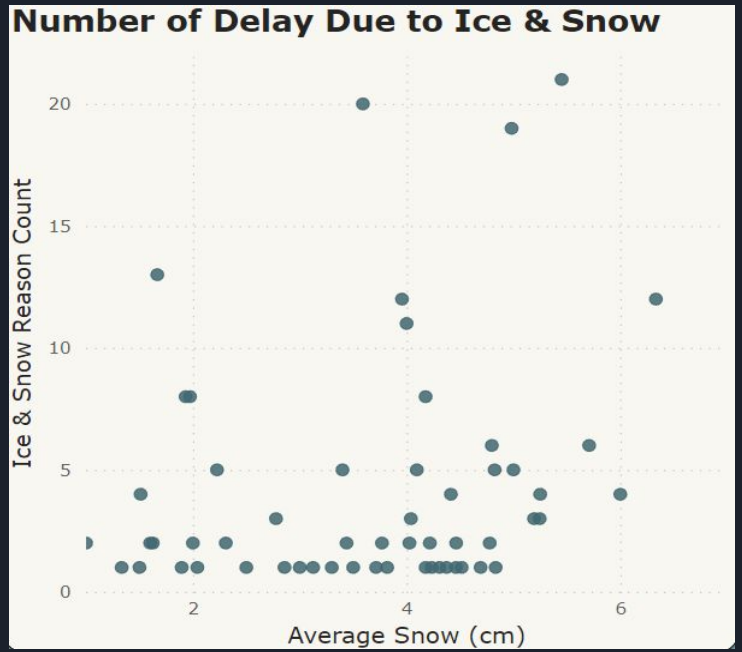


Correlation between Average Snow (cm) and Snow Reason Delay Min Per Year



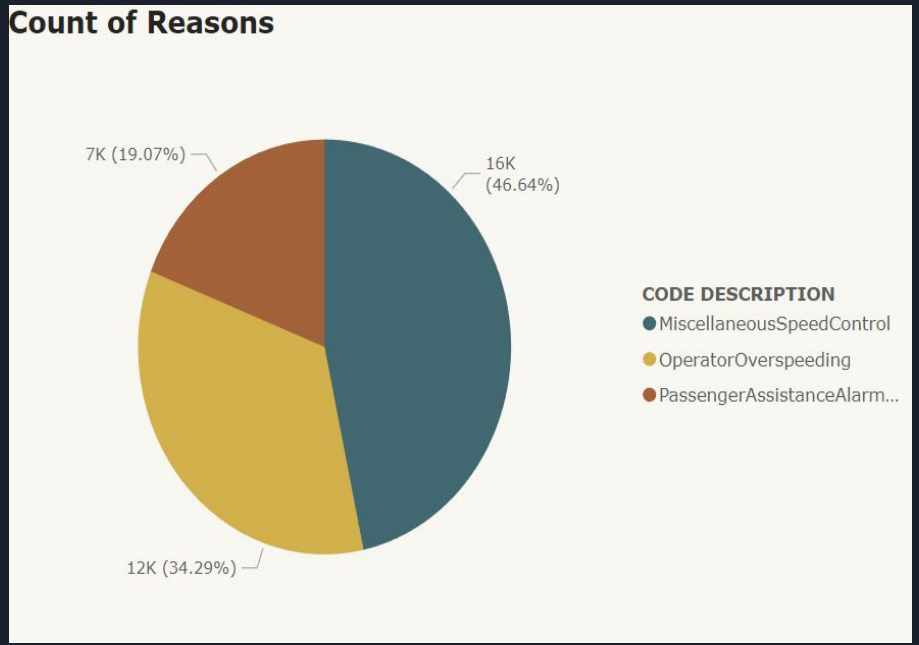
- Heavy snow at Eglinton west Station and Lawrence west Station No Ice & Snow delay within those station.
- As average snow increases, there are no clear pattern to show that Ice & Snow reason delay minute per year increases
- No clear positive correlation

Correlation between Average Snow (cm) and Snow Reason Count in 5 Years



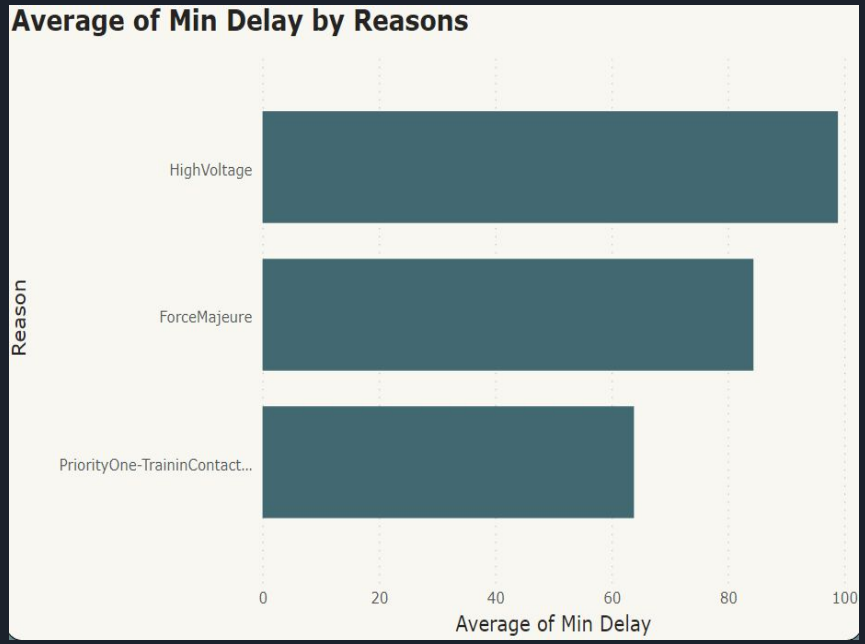
- Usually encounter heavy snow, but there are less Ice & Snow Reason Count
- As average snow increases, there are no clear pattern to show that Ice & Snow Reason Count increases
- No clear positive correlation

Top 3 Reasons Counted in 5 Years



- Count of Top 3 Reasons:
  Miscellaneous Speed Control
  Operator Overspeeding
  Passenger Assistance
- Top 3 Stations with those Reasons:
  Kipling Station
  Kennedy Station
  Keele Station

Top 3 Average of Min Delay by Reasons



- Top 3 Average of Min Delay:
  High Voltage
  Force Majeure (snow weather will not be consider)
  Train in Contract with Person
- Top 3 Stations with those Reasons:
  Kennedy Station
  Scarborough Centre Station
  Yorkdale Station

Conclusion

- Snow weather does not affect TTC services
- TTC services delay usually causes by human factors
- Services delay usually happens on Line 2
- TTC Should provide more training on Line 2 staffs and engineers