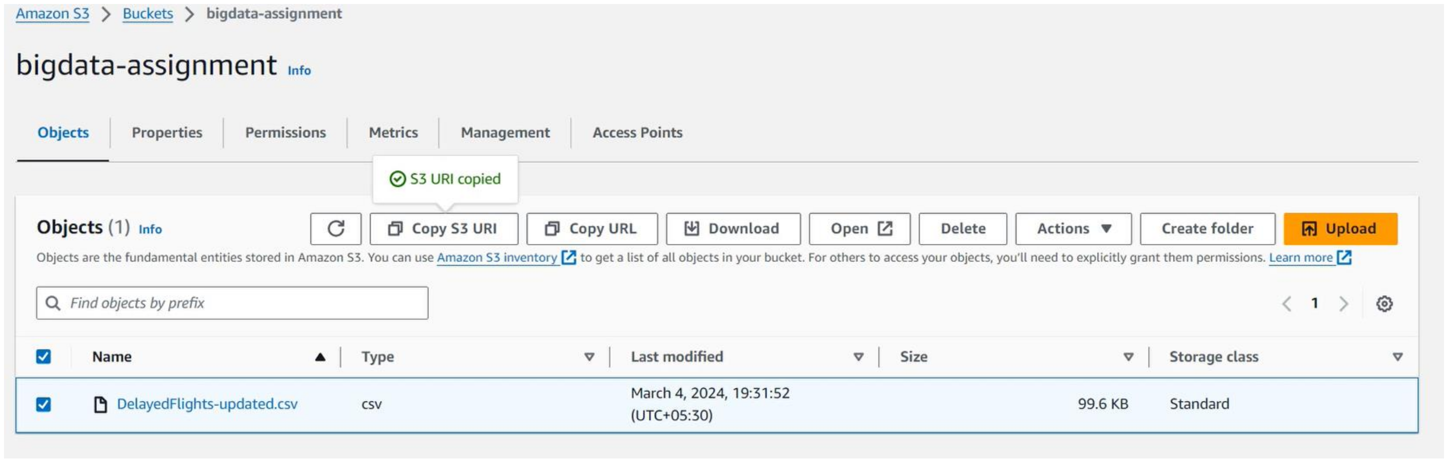
Big Data Video Presentation

Question d:

Uploading CSV in S3



Amazon S3 > Buckets > bigdata-assignment

bigdata-assignment Info

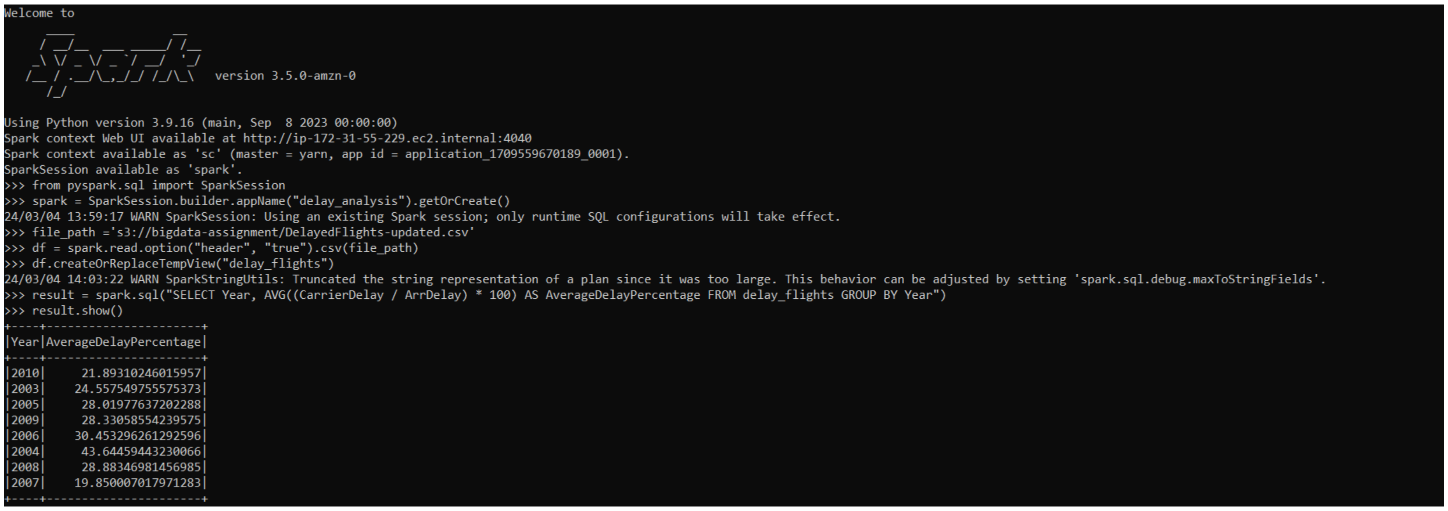
Objects (1) Info

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Find objects by prefix

Name	Type	Last modified	Size	Storage class
DelayedFlights-updated.csv	csv	March 4, 2024, 19:31:52 (UTC+05:30)	99.6 KB	Standard

Spark Initialization



```
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 3.5.0-amzn-0
      /_/

Using Python version 3.9.16 (main, Sep  8 2023 00:00:00)
Spark context Web UI available at http://ip-172-31-55-229.ec2.internal:4040
Spark context available as 'sc' (master = yarn, app id = application_1709559670189_0001).
SparkSession available as 'spark'.
>>> from pyspark.sql import SparkSession
>>> spark = SparkSession.builder.appName("delay_analysis").getOrCreate()
24/03/04 13:59:17 WARN SparkSession: Using an existing Spark session; only runtime SQL configurations will take effect.
>>> file_path ='s3://bigdata-assignment/DelayedFlights-updated.csv'
>>> df = spark.read.option("header", "true").csv(file_path)
>>> df.createOrReplaceTempView("delay_flights")
24/03/04 14:03:22 WARN SparkStringUtils: Truncated the string representation of a plan since it was too large. This behavior can be adjusted by setting 'spark.sql.debug.maxToStringFields'.
>>> result = spark.sql("SELECT Year, AVG((CarrierDelay / ArrDelay) * 100) AS AverageDelayPercentage FROM delay_flights GROUP BY Year")
>>> result.show()
+----+----------------------+
|Year|AverageDelayPercentage|
+----+----------------------+
|2010|     21.89310246015957|
|2003|    24.557549755575373|
|2005|     28.01977637202288|
|2009|     28.33058554239575|
|2006|    30.453296261292596|
|2004|     43.64459443230066|
|2008|     28.88346981456985|
|2007|    19.850007017971283|
+----+----------------------+
```

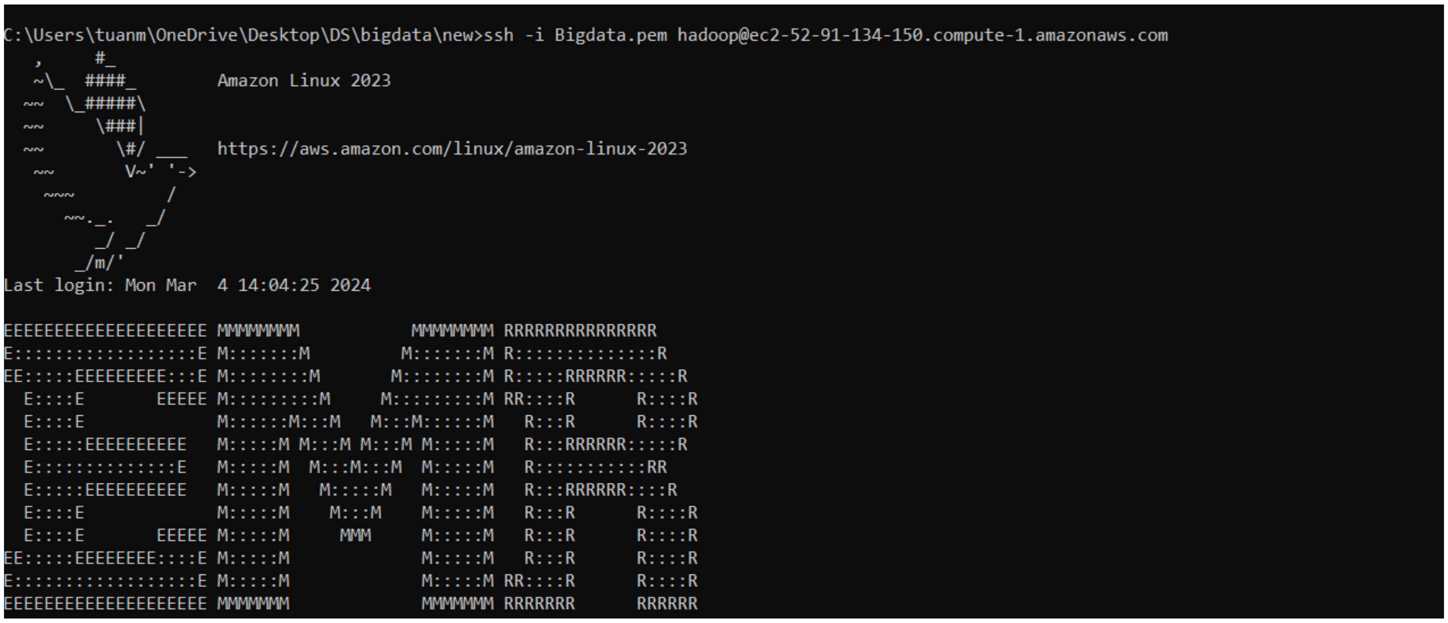
Spark Code

```
from pyspark.sql import SparkSession

# Create a Spark session

spark = SparkSession.builder.appName("DelayFlightsAnalysis").getOrCreate()


# Upload CSV and create DataFrame

csv_path = "path/to/DelayedFlights-updated.csv"

delay_flights_df = spark.read.csv(csv_path, header=True, inferSchema=True)


# Create a temporary table named delay_flights

delay_flights_df.createOrReplaceTempView("delay_flights")
```

```
# Run the first query using HiveQL
result_hiveql = spark.sql("""
    SELECT Year, AVG((CarrierDelay / ArrDelay) * 100) AS avg_delay_percentage
    FROM delay_flights
    GROUP BY Year""").show()


# Run the second query using Spark SQL
result_sparksql = spark.sql("""
    SELECT Year, AVG((CarrierDelay / ArrDelay) * 100) AS avg_delay_percentage
    FROM delay_flights
    GROUP BY Year""").show()
```

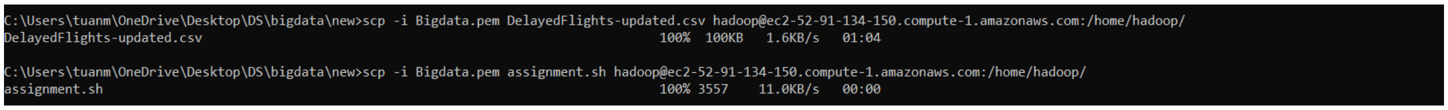
Question e:

EMR Cluster Initialization



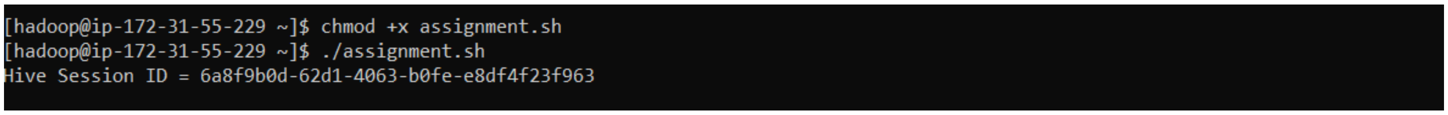
```
C:\Users\tuanm\OneDrive\Desktop\DS\bigdata\new>ssh -i Bigdata.pem hadoop@ec2-52-91-134-150.compute-1.amazonaws.com
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Mon Mar  4 14:04:25 2024

EEEEEEEEEEEEEEEEEEEE MMMMMMMM           MMMMMMMM RRRRRRRRRRRRRRR
E::::::::::::::::::E M:::::::M         M:::::::M R::::::::::::::R
EE:::::EEEEEEEEE:::E M::::::::M       M::::::::M R:::::RRRRRR:::::R
  E::::E       EEEEE M:::::::::M     M:::::::::M RR::::R      R::::R
  E::::E             M::::::M:::M   M:::M::::::M   R:::R      R::::R
  E:::::EEEEEEEEEE   M:::::M M:::M M:::M M:::::M   R:::RRRRRR:::::R
  E::::::::::::::E   M:::::M  M:::M:::M  M:::::M   R:::::::::::RR
  E:::::EEEEEEEEEE   M:::::M   M:::::M   M:::::M   R:::RRRRRR::::R
  E::::E             M:::::M    M:::M    M:::::M   R:::R      R::::R
  E::::E       EEEEE M:::::M     MMM     M:::::M   R:::R      R::::R
EE:::::EEEEEEEE::::E M:::::M             M:::::M   R:::R      R::::R
E::::::::::::::::::E M:::::M             M:::::M RR::::R      R::::R
EEEEEEEEEEEEEEEEEEEE MMMMMMM             MMMMMMM RRRRRRR      RRRRRR
```

Adding Shell Script and table in Hadoop directory



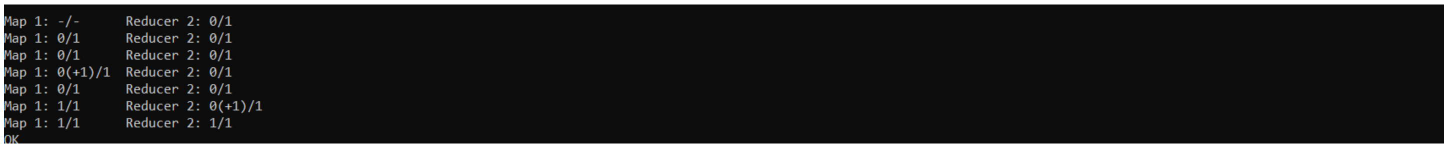
```
C:\Users\tuanm\OneDrive\Desktop\DS\bigdata\new>scp -i Bigdata.pem DelayedFlights-updated.csv hadoop@ec2-52-91-134-150.compute-1.amazonaws.com:/home/hadoop/
DelayedFlights-updated.csv                                                            100%  100KB   1.6KB/s   01:04

C:\Users\tuanm\OneDrive\Desktop\DS\bigdata\new>scp -i Bigdata.pem assignment.sh hadoop@ec2-52-91-134-150.compute-1.amazonaws.com:/home/hadoop/
assignment.sh                                                                         100% 3557    11.0KB/s   00:00
```

Grant permission to Shell Script and execute



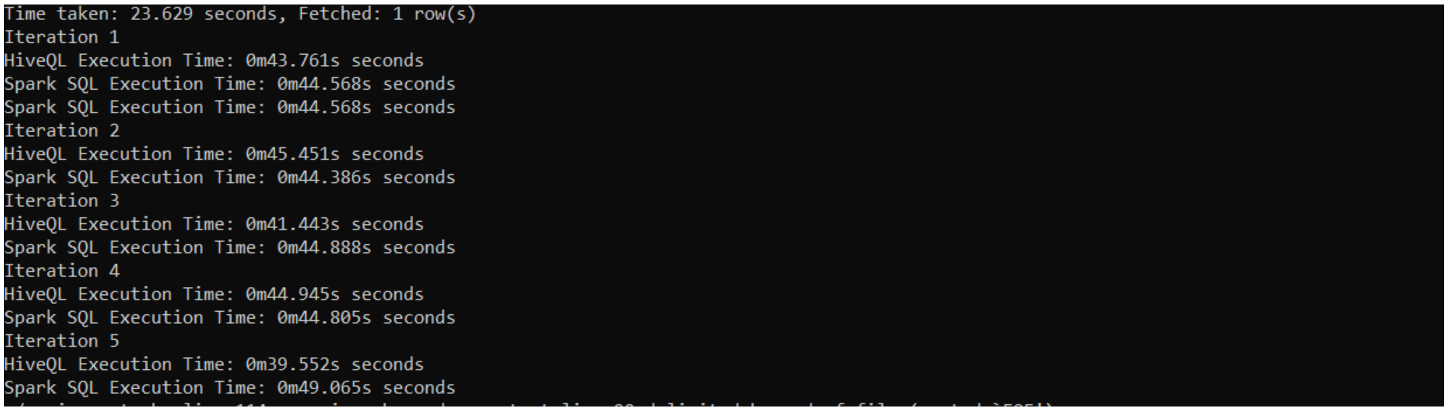
```
[hadoop@ip-172-31-55-229 ~]$ chmod +x assignment.sh
[hadoop@ip-172-31-55-229 ~]$ ./assignment.sh
Hive Session ID = 6a8f9b0d-62d1-4063-b0fe-e8df4f23f963
```

Map Reducer Worker distribution



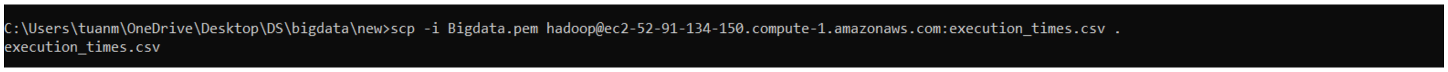
```
Map 1: -/-      Reducer 2: 0/1
Map 1: 0/1      Reducer 2: 0/1
Map 1: 0/1      Reducer 2: 0/1
Map 1: 0(+1)/1  Reducer 2: 0/1
Map 1: 0/1      Reducer 2: 0/1
Map 1: 1/1      Reducer 2: 0(+1)/1
Map 1: 1/1      Reducer 2: 1/1
OK
```

Run Time Calculation



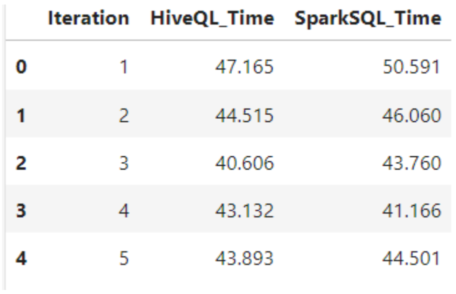
```
Time taken: 23.629 seconds, Fetched: 1 row(s)
Iteration 1
HiveQL Execution Time: 0m43.761s seconds
Spark SQL Execution Time: 0m44.568s seconds
Spark SQL Execution Time: 0m44.568s seconds
Iteration 2
HiveQL Execution Time: 0m45.451s seconds
Spark SQL Execution Time: 0m44.386s seconds
Iteration 3
HiveQL Execution Time: 0m41.443s seconds
Spark SQL Execution Time: 0m44.888s seconds
Iteration 4
HiveQL Execution Time: 0m44.945s seconds
Spark SQL Execution Time: 0m44.805s seconds
Iteration 5
HiveQL Execution Time: 0m39.552s seconds
Spark SQL Execution Time: 0m49.065s seconds
```

Download into local path

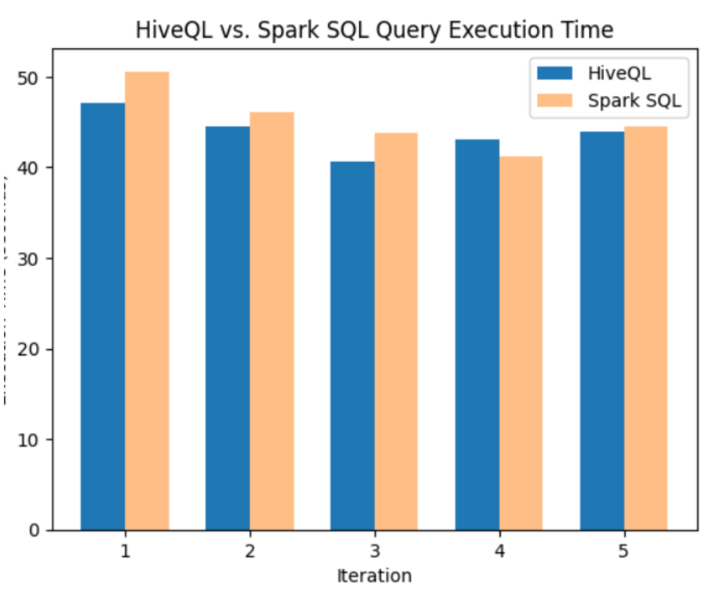


```
C:\Users\tuanm\OneDrive\Desktop\DS\bigdata\new>scp -i Bigdata.pem hadoop@ec2-52-91-134-150.compute-1.amazonaws.com:execution_times.csv .
execution_times.csv
```

Results



	Iteration	HiveQL_Time	SparkSQL_Time
0	1	47.165	50.591
1	2	44.515	46.060
2	3	40.606	43.760
3	4	43.132	41.166
4	5	43.893	44.501



Shell Script file is **assignment_for_5_iteration.sh**

Question f:

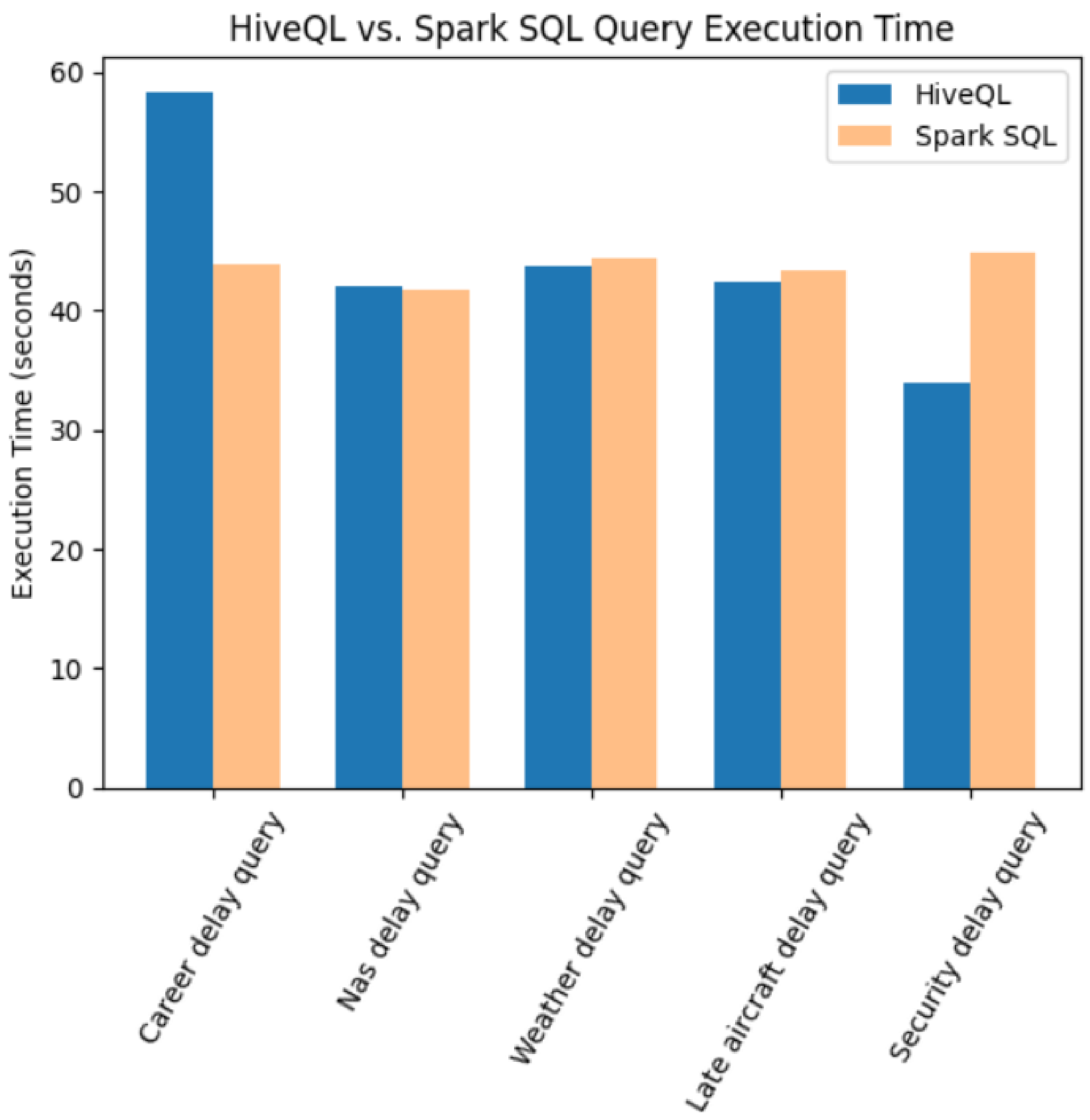
Same command approach as above. But Shell script is different. I have attached it in the same folder
assignment_for_5_queries.sh

Results

```
                    Mapping  HiveQL_Time  SparkSQL_Time
0         Career delay query       58.340         43.824
1            Nas delay query       42.040         41.719
2        Weather delay query       43.682         44.415
3  Late aircraft delay query       42.455         43.383
4       Security delay query       34.011         44.908
```

Question g:

Results


HiveQL vs. Spark SQL Query Execution Time
Execution Time (seconds)
HiveQL
Spark SQL
Career delay query
Nas delay query
Weather delay query
Late aircraft delay query
Security delay query