AZURE DATA BRICKS

TEAM 02

Vivek Rajendra Ugale
46310061

Sameer Shrikant Bhalerao
46310100

Mahamadwahid Patel
46310054

Vivek Uddhav Auchar
46310102

Ganesh Onkar More
46310099

Swaralee Rajkumar Maske
46312193

JUNE 19, 2023

CAPGEMINI
Airoli

Goal: To help you understand the capabilities and features of Spark SQL, including how to:

- Create tables from DataFrame
- Transforming DataFrames
- Aggregate data using built-in Spark functions
- Perform SQL*joins*
- Tune and optimize SQL queries for improved performance.
- Use the Spark UI to visualize the job processes and acquire performance insights.

Data: We will use the publicly available NYC Taxi Trip Record dataset.

Tasks: In this tutorial you will be performing the following tasks

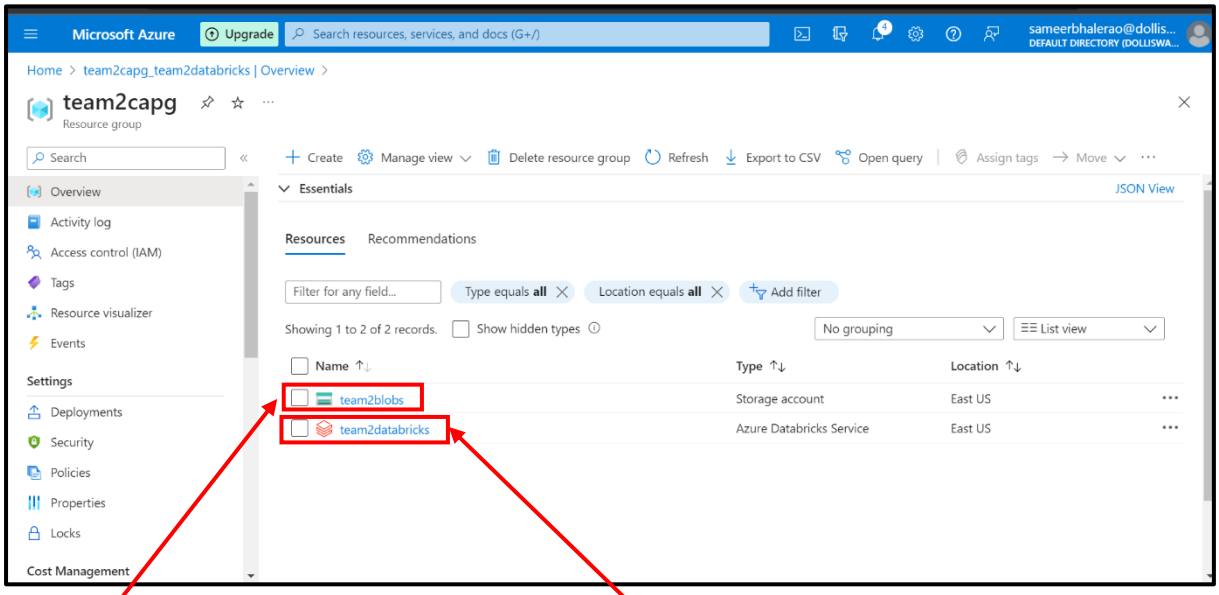
1. Convert DataFrames to tables using saveAsTable()
2. Basic Aggregation
    - Using the abs and round functions to perform calculations and transformations
3. Explore and implement various SQL *joins* to combine tables and DataFrames.
    - **Shuffle join** is the default join method in Spark SQL. Like every shuffle operation, it consists of moving data between executors.
    - **Broadcast join** uses broadcast variables to send DataFrames to join with other DataFrames as a broadcast variable (e.g. only once).
    - Query and perform joins on data directly from SQL Data Warehouse.
4. Implement various performance and optimization techniques
    - Caching interim partial results to be reused in subsequent stages.
    - Manipulating the size and number of the partitions that help parallelize distributed data processing executors.
5. Explore the features and Spark Databricks UI

Quizzes: The tutorial includes do-it-yourself quizzes. To solve the quizzes, follow these steps:

1. Write the code in the space provided.
2.
3. Run and test your code.
4.
5. You can validate the accuracy by running the provided validate function.
6.
7. If you want to see the answers, scroll to the bottom.

PREREQUISITES

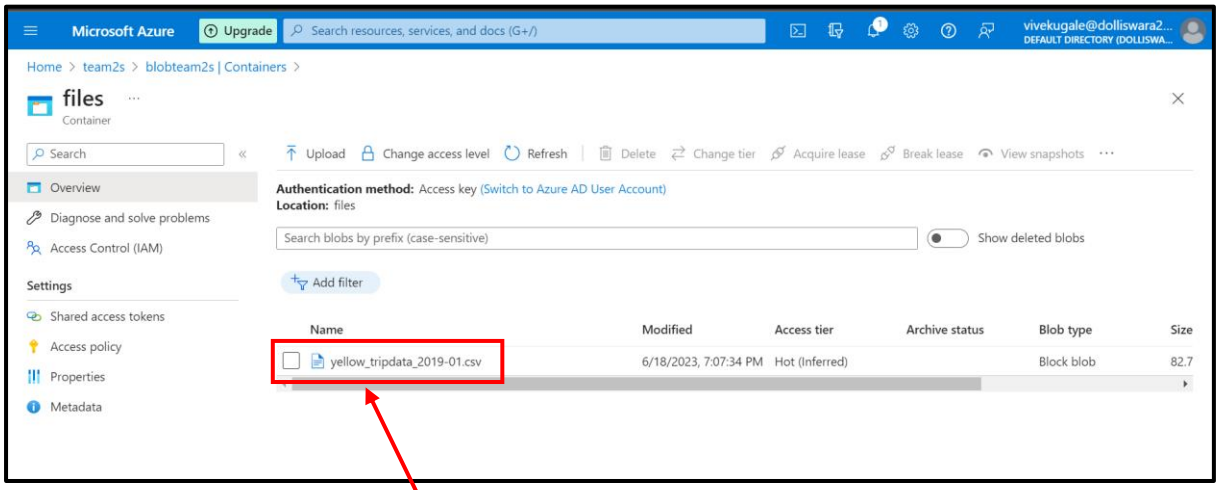
1. Resource Group



Blob Storage

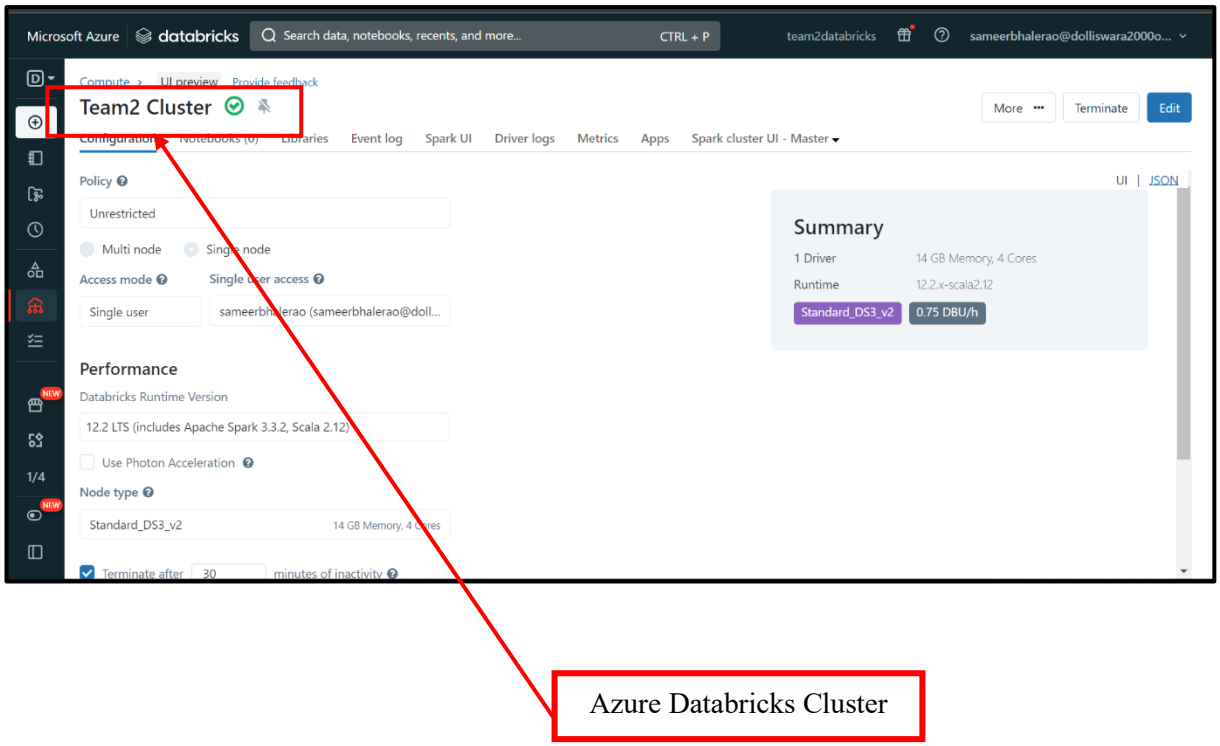
Azure Databricks

2. Azure Blob Storage



Azure blob storage => containers => Files

3. Azure Data Bricks Cluster.



Azure Databricks Cluster

4. Creating a notebook and connecting it to blob Storage file for further Tasks.

Cmd 3

```
storage_account_name = "blobteam2s"
storage_account_access_key = "+x45UCDY7umpZVqqgiK4Uz3kmfsOGTxXU5EDX1haIPl0rvYRiYkrIiWLRVkiUayG2N6hjNPDADG0+ASt7NgKNA=="
```

Command took 0.24 seconds -- by sameerbhalerao@dolliswara2000outlook.onmicrosoft.com at 6/19/2023, 9:37:35 AM on Team2 Cluster

Cmd 4

```
file_location = "wasbs://files@blobteam2s.blob.core.windows.net/"
file_type = "csv"
```

Command took 0.16 seconds -- by sameerbhalerao@dolliswara2000outlook.onmicrosoft.com at 6/19/2023, 9:37:35 AM on Team2 Cluster

Cmd 5

Python

```
spark.conf.set(
  "fs.azure.account.key."+storage_account_name+".blob.core.windows.net",
  storage_account_access_key)
```

Command took 0.19 seconds -- by sameerbhalerao@dolliswara2000outlook.onmicrosoft.com at 6/19/2023, 9:37:35 AM on Team2 Cluster

Task 1

Databases and Tables / Saving Dataset as Table

Use the Azure Databricks **Data** menu to view the NYC taxi trip dataset in a visual manner. The DataFrameWriter.saveAsTable method saves the content of a DataFrame as a specified table. use save mode("overwrite") api to write the tale more than once

PERFORMED BY : **Sameer Shrikant Bhalerao**.

Solution :-

```
# Task 1

team2 = spark.read.format(file_type).option("header", "true").load(file_location)
```

(1) Spark Jobs

team2: pyspark.sql.dataframe.DataFrame = [VendorID: string, tpep_pickup_datetime: string ... 16 more fields]

Command took 0.99 seconds -- by sameerbhalerao@dolliswara2000outlook.onmicrosoft.com at 6/17/2023, 8:06:18 PM on Team2 Cluster

Cmd 8

Initial Schemas

Python

```
team2.printSchema()
```

```
root
 |-- VendorID: string (nullable = true)
 |-- tpep_pickup_datetime: string (nullable = true)
 |-- tpep_dropoff_datetime: string (nullable = true)
 |-- passenger_count: string (nullable = true)
 |-- trip_distance: string (nullable = true)
 |-- RatecodeID: string (nullable = true)
 |-- store_and_fwd_flag: string (nullable = true)
 |-- PULocationID: string (nullable = true)
 |-- DOLocationID: string (nullable = true)
 |-- payment_type: string (nullable = true)
 |-- fare_amount: string (nullable = true)
 |-- extra: string (nullable = true)
 |-- mta_tax: string (nullable = true)
 |-- tip_amount: string (nullable = true)
 |-- tolls_amount: string (nullable = true)
 |-- improvement_surcharge: string (nullable = true)
 |-- total_amount: string (nullable = true)
 |-- congestion_surcharge: string (nullable = true)
```

Command took 0.13 seconds -- by sameerbhalerao@dolliswara2000outlook.onmicrosoft.com at 6/17/2023, 8:52:48 PM on Team2 Cluster

Changed Schemas

```python
team2 = team2.withColumn("passenger_count",team2.passenger_count.cast("Integer"))
team2 = team2.withColumn("trip_distance",team2.trip_distance.cast("float"))
team2 = team2.withColumn("RatecodeID",team2.RatecodeID.cast("Integer"))
team2 = team2.withColumn("PULocationID",team2.PULocationID.cast("Integer"))
team2 = team2.withColumn("DOLocationID",team2.DOLocationID.cast("Integer"))
team2 = team2.withColumn("payment_type",team2.payment_type.cast("Integer"))
team2 = team2.withColumn("extra",team2.extra.cast("float"))
team2 = team2.withColumn("mta_tax",team2.mta_tax.cast("float"))
team2 = team2.withColumn("tip_amount",team2.tip_amount.cast("float"))
team2 = team2.withColumn("tolls_amount",team2.tolls_amount.cast("float"))
team2 = team2.withColumn("improvement_surcharge",team2.improvement_surcharge.cast("float"))
team2 = team2.withColumn("total_amount",team2.total_amount.cast("float"))
team2 = team2.withColumn("congestion_surcharge",team2.congestion_surcharge.cast("Integer"))
```

```
team2: pyspark.sql.dataframe.DataFrame
  VendorID: string
  tpep_pickup_datetime: string
  tpep_dropoff_datetime: string
  passenger_count: integer
  trip_distance: float
  RatecodeID: integer
  store_and_fwd_flag: string
  PULocationID: integer
  DOLocationID: integer
  payment_type: integer
```

Python

```python
team2.write.mode("overwrite").format("csv").saveAsTable("Team2_Table")
```

(1) Spark Jobs

```
Command took 9.72 seconds -- by sameerbhalerao@dolliswara2000outlook.onmicrosoft.com at 6/17/2023, 8:07:36 PM on Team2 Cluster
```

Cmd 9

SQL

```sql
%sql

SELECT * FROM Team2_Table
```

(1) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [VendorID: string, tpep_pickup_datetime: string ... 16 more fields]

Output:-

SQL

```sql
%sql

SELECT * FROM Team2_Table
```

(1) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [VendorID: string, tpep_pickup_datetime: string ... 16 more fields]

Table +

	re_amount	extra	mta_tax	tip_amount	tolls_amount	improvement_surcharge	total_amount	congestion_surcharge
1	107.91	0	0	0	0	0	31107.91	null
2	130.71	0	0	0	0	0	30130.71	null
3	356.38	0	0	0	0	0	25356.38	null
4	477.08	0	0	0	0	0	18477.08	null
5	242.81	0	0	0	0	0	17242.81	null
6	66.65	0	0.5	0	0	0.3	6667.45	null
7	5.35	0	0.5	0	0	0.3	656.15	null

10,000 rows | Truncated data | 1.48 seconds runtime

Refreshed 4 minutes ago

Task 2

Basic aggregations

Once a table is created for our data, we can use Spark SQL to perform operations on our data. There are various ways to aggregate data using built-in Spark functions.

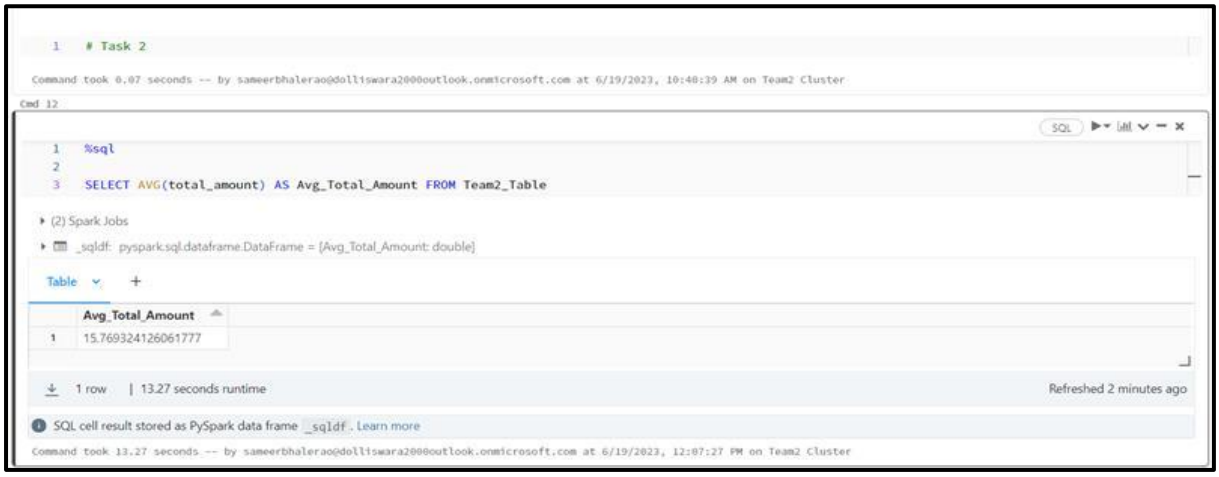
a. Give Average of total Amount

b. Give round value of average

c. Provide maximum and minimum fares

Performed By: - **Ganesh Onkar More**

Solution:-



Cmd 12

SQL

```
%sql

SELECT MIN(total_amount) as Minimum_totalamount, MAX(total_amount) as Maximum_totalamount FROM Team2_Table
```

(2) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [Minimum_totalamount: float, Maximum_totalamount: float]

Table +

	Minimum_totalamount	Maximum_totalamount
1	-300.3	31107.91

1 row | 1.73 seconds runtime Refreshed now

SQL cell result stored as PySpark data frame `_sqldf`. Learn more

Command took 1.73 seconds -- by sameerbhalerao@dolliswara2000outlook.onmicrosoft.com at 6/17/2023, 9:05:48 PM on Team2 Cluster

Cmd 13

Task 3

Note that some of the fares in the `faredata_table` are negative.

These figures indicate bad data. Convert all the negative amounts to positive ones and then sort the top 20 trips by their total fare.

Performed By:- **Vivek Rajendra Ugale**

Solution :-

Cmd 17

```
%sql
-- Task 3
create TEMPORARY VIEW FaresWithoutNegative as
SELECT VendorID, abs(fare_amount), abs(extra) , abs(mta_tax), abs(tip_amount), abs(tolls_amount), abs(improvement_surcharge),abs(total_amount)
AS total_fare
FROM Team2_Table
```

_sqldf: pyspark.sql.dataframe.DataFrame

OK

Command took 0.46 seconds -- by sameerbhalerao@dolliswara2000outlook.onmicrosoft.com at 6/19/2023, 9:58:51 AM on Team2 Cluster

Output:-

Cmd 18

```
%sql
SELECT * FROM Team2_Table ORDER BY total_amount DESC LIMIT 20
```

(1) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [VendorID: string, tpep_pickup_datetime: string ... 16 more fields]

Table

	nt_type	fare_amount	extra	mta_tax	tip_amount	tolls_amount	improvement_surcharge	total_amount	congestion_surcharge
1		12.645433125908939	0	0	0	0	0	31107.91	null
2		12.645433125908939	0	0	0	0	0	30130.71	null
3		12.645433125908939	0	0	0	0	0	25356.38	null
4		12.645433125908939	0	0	0	0	0	18477.08	null
5		12.645433125908939	0	0	0	0	0	17242.81	null
6		12.645433125908939	0	0.5	0	0	0.3	6667.45	null
7		12.645433125908939	0	0.5	0	0	0.3	656.15	null

20 rows | 2.21 seconds runtime

Refreshed 5 minutes ago

SQL cell result stored as PySpark data frame `_sqldf` . Learn more

Command took 2.21 seconds -- by sameerbhalerao@dolliswara2000outlook.onmicrosoft.com at 6/19/2023, 9:59:07 AM on Team2 Cluster

Task 4

1. Create a temporary view called `FaresWithoutNegative`, where all the negative fares have been converted to positive fares.
2. Starting with the table `FaresWithoutNegative`, create another view called `FaresWithoutNegativeSorted` where: 0. The data set has been reduced to the first 20 records. 0. The records are sorted by `total_amount` in descending order
3. `Display the contents of the temporary view by running a simple select query on it.`

Performed By:- **Swaralee Rajkumar Maske**

Solution:-

```
# Task 4
```

Cmd 20

```
%sql

CREATE TEMPORARY VIEW  FaresWithoutsNegative AS
SELECT * FROM Team2_Table
```

_sqldf: pyspark.sql.dataframe.DataFrame

OK

Command took 0.14 seconds -- by sameerbhalerao@dolliswara2000outlook.onmicrosoft.com at 6/17/2023, 10:53:45 PM on Team2 Cluster

Cmd 21

```
%sql

SELECT * FROM FaresWithoutsNegative
```

(1) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [VendorID: string, tpep_pickup_datetime: string ... 16 more fields]

Table

	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	RatecodeID	store_and_fwd_flag	PULocationID	DOLoc
1	1	03-01-2019 15:33	03-01-2019 15:44	2	1.9	1	N	125	186

Output:-

```
%sql

CREATE TEMPORARY VIEW FaresWithoutNegativeSorted AS
SELECT * FROM default.Team2_Table
ORDER BY total_amount DESC LIMIT 20
```

_sqldf: pyspark.sql.dataframe.DataFrame

OK

Command took 0.19 seconds -- by sameerbhalerao@dolliswara2000outlook.onmicrosoft.com at 6/17/2023, 10:50:34 PM on Team2 Cluster

Cmd 24

```
%sql

SELECT * FROM FaresWithoutNegativeSorted
```

(1) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [VendorID: string, tpep_pickup_datetime: string ... 16 more fields]

Table

	payment_type	fare_amount	extra	mta_tax	tip_amount	tolls_amount	improvement_surcharge	total_amount	congestion_surcharge
1	1	31107.91	0	0	0	0	0	31107.91	null
2	1	30130.71	0	0	0	0	0	30130.71	null
3	1	25356.38	0	0	0	0	0	25356.38	null
4	1	18477.08	0	0	0	0	0	18477.08	null

Task 5

To further refine the data, assume that all fares under $1 represent bad data and filter them out.
Additionally, categorize each trip's `total_fare` into $100 groups.

1. Start with the table `FaresWithoutNegative`. Create a temporary view called `FaresWithoutNegativeFiltered` where: 0. The data set excludes all records where total_amount is below $1. 0. The data set includes a new column called `fare100`, that should be the `total_amount` grouped by 100's. For example:
2. A fare of 230 should be represdented by a value of "2".
3. A fare of 574 should be represented report a value of "6".

Performed By:- **Mahamadwahid Patel**

Solution:-

```
#Task 5

from pyspark.sql.functions import col, round
filtered_team2 = team2.filter(col("total_amount") >= 1)
result = filtered_team2.withColumn("fare100",round(col("total_amount") / 100))



result.display()
```

(1) Spark Jobs

filtered_team2: pyspark.sql.dataframe.DataFrame = [VendorID: string, tpep_pickup_datetime: string ... 16 more fields]

result: pyspark.sql.dataframe.DataFrame = [VendorID: string, tpep_pickup_datetime: string ... 17 more fields]

Table +

	fare_amount	extra	mta_tax	tip_amount	tolls_amount	improvement_surcharge	total_amount	congestion_surcharge	fare100
21	401	0.5	0.5	0	5.76	0.3	408.06	null	4
22	302	0	0	35	24.02	0.3	361.32	null	4
23	300	0	0.5	60.16	0	0.3	360.96	null	4
24	355.55	0	0	0	0	0.3	355.85	null	4
25	256.25	0	0.5	69.77	22.02	0.3	348.84	null	3
26	345.55	0	0	0	0	0.3	345.85	null	3

Task 6

Find the total fare earned by taxis at each pickup location

1. Use the `tripsdata_table` to create a view with the distinct pickup Location ID and tripID.
2. Create another view from the `faredata_table` with the tripID and the fare amount.
3. Perform a join operation on both these views to get the resulting table.

Aggregations can be run on the resulting table to find the desired amount.Perform a join operation with the two temporary views to generate a table correlating the fares with the pickup location. Use the column `tripID` to perform the join. Perform a groupBy on the Location IDs and sum the fares.

Performed By:- **Vivek Uddhav Auchar**

Solution:-

```
# Task 6
```

Cmd 30

```
%sql

CREATE TEMPORARY VIEW Tripsdata_Table AS
SELECT DISTINCT(PULocationID), VendorID FROM Team2_Table
```

_sqldf: pyspark.sql.dataframe.DataFrame

OK

Command took 0.23 seconds -- by sameerbhalerao@dolliswara2000outlook.onmicrosoft.com at 6/17/2023, 11:41:39 PM on Team2 Cluster

Cmd 31

```
%sql

CREATE TEMPORARY VIEW Faredata_Table AS
SELECT  VendorID, total_amount FROM Team2_Table
```

_sqldf: pyspark.sql.dataframe.DataFrame

OK

Command took 0.12 seconds -- by sameerbhalerao@dolliswara2000outlook.onmicrosoft.com at 6/17/2023, 11:42:14 PM on Team2 Cluster

```
%sql

SELECT t.VendorID, t.Pickup_Location,f.total_amount FROM Tripsdata_Tables AS t
INNER JOIN Faredata_Table as f
ON t.VendorID = t.VendorID
```

(3) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [VendorID: string, Pickup_Location: integer ... 1 more field]

Table

	VendorID	Pickup_Location	total_amount
1	1	43	10.8
2	1	43	10.8
3	1	43	10.8
4	1	43	10.8
5	1	43	10.8
6	1	43	10.8
7	1	43	10.8

10,000 rows | Truncated data | 2.51 seconds runtime

Refreshed now

```
%sql

SELECT t.Pickup_Location,sum(f.total_amount) FROM Tripsdata_Tables AS t
INNER JOIN Faredata_Table as f
ON t.VendorID = t.VendorID
GROUP BY t.Pickup_Location

```

(4) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [Pickup_Location: integer, sum(total_amount): double]

Table +

	Pickup_Location	sum(total_amount)
1	148	49605957.136455685
2	243	49605957.136455685
3	31	33070638.090970457
4	137	49605957.136455685
5	85	33070638.090970457
6	251	33070638.090970457
7	65	49605957.136455685

253 rows | 2.91 seconds runtime Refreshed now

SQL cell result stored as PySpark data frame `_sqldf`. Learn more

Command took 2.91 seconds -- by sameerbhalerao@dolliswara2000outlook.onmicrosoft.com at 6/17/2023, 11:48:44 PM on Team2 Cluster