In [1]:

```
# Intialization
import os
import sys

os.environ["SPARK_HOME"] = "/home/talentum/spark"
os.environ["PYLIB"] = os.environ["SPARK_HOME"] + "/python/lib"
# In below two lines, use /usr/bin/python2.7 if you want to use P
os.environ["PYSPARK_PYTHON"] = "/usr/bin/python3.6"
os.environ["PYSPARK_DRIVER_PYTHON"] = "/usr/bin/python3"
sys.path.insert(0, os.environ["PYLIB"] +"/py4j-0.10.7-src.zip")
sys.path.insert(0, os.environ["PYLIB"] +"/pyspark.zip")

# NOTE: Whichever package you want mention here.
# os.environ['PYSPARK_SUBMIT_ARGS'] = '--packages com.databricks:
# os.environ['PYSPARK_SUBMIT_ARGS'] = '--packages org.apache.spar
os.environ['PYSPARK_SUBMIT_ARGS'] = '--packages com.databricks:sp
# os.environ['PYSPARK_SUBMIT_ARGS'] = '--packages com.databricks:
```

In [2]:

```
#Entrypoint 2.x
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("Spark SQL basic example").e

# On yarn:
# spark = SparkSession.builder.appName("Spark SQL basic example")
# specify .master("yarn")

sc = spark.sparkContext
```

In [20]:

```
from pyspark.sql import SparkSession
from pyspark.sql.types import StructType, StructField, StringType

# Initialize Spark Session
spark = SparkSession.builder \
    .appName("UberRidesAnalysis") \
    .getOrCreate()

# Define the schema for the Uber rides data
schema = StructType([
    StructField("ride_id", StringType(), True),
    StructField("date", StringType(), True),
    StructField("pickup_location", StringType(), True),
    StructField("drop_location", StringType(), True),
    StructField("distance_km", DoubleType(), True),
    StructField("fare_amount", DoubleType(), True),
    StructField("payment_type", StringType(), True),
    StructField("vehicle_type", StringType(), True)
])

df = spark.read \
    .option("header", "true") \
    .schema(schema) \
    .csv("file:///home/talentum/test-jupyter/bdt/uber_rides.csv")
# Display the dataframe
print("Data Preview:")
df.show()

print("\nSchema:")
df.printSchema()


```

```
Data Preview:
+-------+----------+---------------+-------------+-----------+-----
------+------------+------------+
|ride_id|      date|pickup_location|drop_location|distance_km|fare_
amount|payment_type|vehicle_type|
+-------+----------+---------------+-------------+-----------+-----
------+------------+------------+
|   R001|2024-01-10|       Downtown|       Suburb|       12.3|
320.5|        Cash|       Sedan|
|   R002|2024-01-11|        Airport|   CityCenter|        8.5|
250.0|        Card|   Hatchback|
|   R003|2024-01-12|         Uptown|     Downtown|       15.0|
410.75|        Card|       Sedan|
|   R004|2024-01-13|        Airport|       Suburb|       22.1|
590.0|         UPI|         SUV|
|   R005|2024-01-14|       Downtown|       Uptown|        9.8|
270.25|        Cash|       Sedan|
+-------+----------+---------------+-------------+-----------+-----
------+------------+------------+


Schema:
root
 |-- ride_id: string (nullable = true)
 |-- date: string (nullable = true)
```

```
 |-- pickup_location: string (nullable = true)
 |-- drop_location: string (nullable = true)
 |-- distance_km: double (nullable = true)
 |-- fare_amount: double (nullable = true)
 |-- payment_type: string (nullable = true)
 |-- vehicle_type: string (nullable = true)
```

In [19]:

```
#find the vehicle type with highest averge fare1_df=uberrides.
Q1_df=fare1_df.groupBy("vehicle_type") \
    .avg("fare_amount") \
    .orderBy("avg(fare_amount)", ascending=False)
Q1_df.show(1)
```

```
+------------+----------------+
|vehicle_type|avg(fare_amount)|
+------------+----------------+
|         SUV|           590.0|
+------------+----------------+
only showing top 1 row
```

In [18]:

```
#find the top3 pickup locations by number of rides
Q2_df=fare1_df.groupBy("pickup_location") \
    .count() \
    .orderBy("count", ascending=False)
Q2_df.show(3)
```

```
+---------------+-----+
|pickup_location|count|
+---------------+-----+
|       Downtown|    2|
|        Airport|    2|
|         Uptown|    1|
+---------------+-----+
```

In [21]:

```
#caculate the average distance per payment_type
Q3_df=fare1_df.groupBy("payment_type") \
    .avg("distance_km")
Q3_df.show()

```

```
+------------+----------------+
|payment_type|avg(distance_km)|
+------------+----------------+
|        Card|           11.75|
|        Cash|           11.05|
|         UPI|            22.1|
+------------+----------------+
```

In [24]:

```
#Find total earnings per day

Q4_df=df.groupBy("date") \
    .sum("fare_amount") \
    .orderBy("date") \
    .show()

```

```
+----------+----------------+
|      date|sum(fare_amount)|
+----------+----------------+
|2024-01-10|           320.5|
|2024-01-11|           250.0|
|2024-01-12|          410.75|
|2024-01-13|           590.0|
|2024-01-14|          270.25|
+----------+----------------+
```

In [ ]:

```

```