#### **HDFS Commands:**

# Put Data File on Hadoop HDFS:

C:\Windows\system32>hadoop fs -put C:\Users\Excel\Documents\HDFS\_SPARK\_Project\yellow\_tripdata\_2024-01.parquet /project\_data

## #Import functions and types all

```
>>> from pyspark.sql.functions import *
```

>>> from pyspark.sql.types import \*

#### #Load data and PrintSchema

```
>>> df = spark.read.parquet("hdfs://localhost:9000/project_data/yellow_tripdata_2024-
01.parquet", header=True, inferSchema=True)
>>> df.printSchema()
root
|-- VendorID: integer (nullable = true)
|-- tpep_pickup_datetime: timestamp_ntz (nullable = true)
|-- tpep_dropoff_datetime: timestamp_ntz (nullable = true)
|-- passenger_count: long (nullable = true)
|-- trip_distance: double (nullable = true)
|-- RatecodeID: long (nullable = true)
|-- store_and_fwd_flag: string (nullable = true)
|-- PULocationID: integer (nullable = true)
|-- DOLocationID: integer (nullable = true)
|-- payment_type: long (nullable = true)
|-- fare_amount: double (nullable = true)
|-- extra: double (nullable = true)
|-- mta_tax: double (nullable = true)
|-- tip_amount: double (nullable = true)
|-- tolls_amount: double (nullable = true)
```

```
|-- improvement_surcharge: double (nullable = true)
|-- total_amount: double (nullable = true)
|-- congestion_surcharge: double (nullable = true)
```

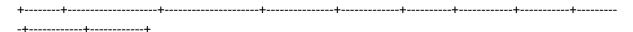
|-- Airport\_fee: double (nullable = true)

#### #Selected Columns transfer to new data frame

>>> my df =

df.select(col('VendorID'),col('tpep\_pickup\_datetime'),col('tpep\_dropoff\_datetime'),col('passenger\_c ount'),col('trip\_distance'),col('RatecodeID'),col('payment\_type'),col('fare\_amount'),col('tip\_amount'),(col('extra')+col('mta\_tax')+col('tolls\_amount')+col('improvement\_surcharge')+col('congestion\_surcharge')).alias('Other\_amount'),col('total\_amount'))

>>> my\_df.show(5)



|VendorID|tpep\_pickup\_datetime|tpep\_dropoff\_datetime|passenger\_count|trip\_distance|Rateco deID|payment\_type|fare\_amount|tip\_amount|Other\_amount|total\_amount|

++	+-	+	+-		+	+
-++						
2  2024-01-01 00:57:5   0.0    5.0    22.7	55  2024-01-01 01:17:43	1	1.72	1	2	17.7
1  2024-01-01 00:03:0 3.75  7.5  18.75	00  2024-01-01 00:09:36	1	1.8	1	1	10.0
1  2024-01-01 00:17:0 3.0  7.5  31.3	06  2024-01-01 00:35:01	1	4.7	1	1	23.3
1  2024-01-01 00:36:3 2.0  7.5  17.0	8  2024-01-01 00:44:56	1	1.4	1	1	10.0
1  2024-01-01 00:46:5 3.2  7.5  16.1	51  2024-01-01 00:52:57	1	0.8	1	1	7.9
++	+++	+	+-		+	+

only showing top 5 rows

-+----+

#### #clean Dataframe

```
>>> my_df = my_df.dropna()
>>> my_df = my_df.dropDuplicates()
>>> my_df = my_df.filter((col("trip_distance") > 0) & (col("fare_amount") > 0))
>>> my_df = my_df.distinct()
```

## #Change column name and Type

```
>>> my_df = my_df.withColumn("tpep_pickup_datetime",
to_timestamp("tpep_pickup_datetime")).withColumn("tpep_dropoff_datetime",
to timestamp("tpep dropoff datetime"))
>>> my df =
my_df.withColumn("pickup_datetime",col("tpep_pickup_datetime")).withColumn("dropoff_datetime")
e",col("tpep_dropoff_datetime"))
#Drop Extra Column
>>> column_to_drop = ['tpep_pickup_datetime','tpep_dropoff_datetime']
>>> my_df = my_df.drop(*column_to_drop)
>>> my_df.printSchema()
root
|-- VendorID: integer (nullable = true)
|-- passenger_count: long (nullable = true)
|-- trip_distance: double (nullable = true)
|-- RatecodeID: long (nullable = true)
|-- payment_type: long (nullable = true)
|-- fare_amount: double (nullable = true)
|-- tip_amount: double (nullable = true)
|-- Other_amount: double (nullable = true)
|-- total_amount: double (nullable = true)
|-- pickup_datetime: timestamp (nullable = true)
|-- dropoff_datetime: timestamp (nullable = true)
#Make Changes in datatypes
>>> double_columns = ["trip_distance", "fare_amount", "tip_amount", "Other_amount",
"total_amount"]
>>> for col_name in double_columns:
... my_df = my_df.withColumn(col_name,round(col(col_name),2))
```

# #Put clean\_df file to hdfs in parquet format with repartition 1 for 1 file

```
>>> clean_df = clean_df.repartition(1)
>>> clean_df.write.mode("overwrite").option("header",
"true").parquet('hdfs://localhost:9000/project_data/clean_data_yellow_tripdata_parquet_repartitio
n')
```

# #Put clean\_df file to hdfs in parquet format without repartition

>>> clean\_df.write.mode("overwrite").option("header",
"true").parquet('hdfs://localhost:9000/project data/clean data yellow tripdata parquet')

## #Get clean data from hdfs and read in pyspark for next task

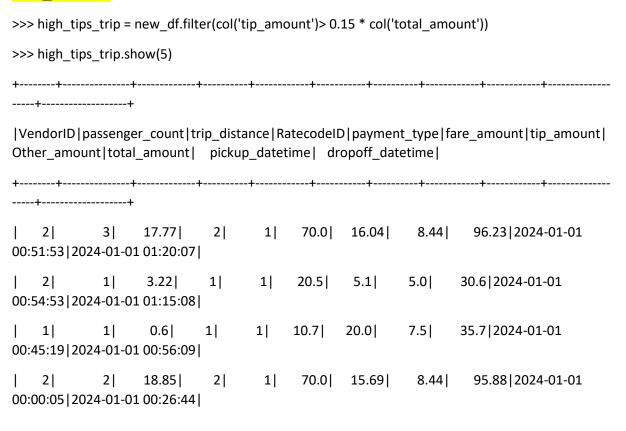
```
>>> new_df
=spark.read.parquet("hdfs://localhost:9000/project_data/clean_data_yellow_tripdata_parquet",hea
der=True,inferSchema=True)
>>> new_df.printSchema()
root
|-- VendorID: integer (nullable = true)
|-- passenger_count: long (nullable = true)
|-- trip_distance: double (nullable = true)
|-- RatecodeID: long (nullable = true)
|-- payment_type: long (nullable = true)
|-- fare_amount: double (nullable = true)
|-- tip_amount: double (nullable = true)
|-- Other_amount: double (nullable = true)
|-- total_amount: double (nullable = true)
|-- pickup_datetime: timestamp (nullable = true)
|-- dropoff_datetime: timestamp (nullable = true)
>>> new_df.count()
```

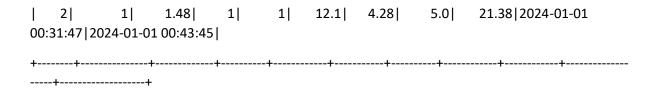
2754462

## **#Basic Analysis and Exploration**

#### Q1. Sum of total amout by different payment types

# Q2. Find all trips where the tip\_amount was greater than 15% of the fare\_amount





## Q3. Busiest hour of the day based on 'pickup\_datetime' top 5:

```
>>> busy_hour =
new_df.groupBy(hour('pickup_datetime').alias('hour')).count().sort(col('count').desc())
>>> busy_hour.show(5)
+----+
|hour| count|
+----+
| 18|198014|
| 17|193368|
| 16|180360|
| 15|179049|
| 14|173423|
+----+
```

# Q4. Find avg duration by all week days

only showing top 20 rows

```
>>> trip_duration = new_df.withColumn('Duration',(unix_timestamp(col('dropoff_datetime')) - unix_timestamp(col('pickup_datetime')))/60)
>>> avg_duration_by_day = trip_duration.groupBy(dayofweek('pickup_datetime').alias('day_of_week')).agg(avg('Duration').alias('Avg_duration'))
>>> avg_duration_by_day.show()
+-----+
| day_of_week| Avg_duration|
+-----+
| 1|14.320157142933923|
| 6|15.89541037441003|
```

```
3 | 16.180425244514616 |
     5 | 16.386867152642992 |
     4 | 16.241479365818684 |
     7 | 14.870000694803593 |
     2 | 15.844644453175697 |
+----+
Q5. Trips taken on Morning, Afternoon and Night time count
>>> trips_by_interval = new_df.withColumn('time_interval',when((hour('pickup_datetime') >= 0) &
(hour('pickup_datetime') < 6), 'Night').when((hour('pickup_datetime') >= 6) &
(hour('pickup_datetime') < 12), 'Morning').when((hour('pickup_datetime') >= 12) &
(hour('pickup datetime') < 18),
'Afternoon').otherwise('Evening')).groupBy('time_interval').count().sort(col('count').desc())
>>> trips_by_interval.show()
+----+
|time_interval| count|
+----+
| Afternoon|1042767|
   Evening | 900831 |
   Morning | 613369 |
    Night | 197495 |
+----+
# Now we are get data from hdfs (10% of data from this data ) and move to
local system
#Make data small 20%
>>> small_df = new_df.sample(fraction=0.2, seed=42)
>>> small df.show(10)
|VendorID|passenger_count|trip_distance|RatecodeID|payment_type|fare_amount|tip_amount|
Other_amount|total_amount| pickup_datetime| dropoff_datetime|
```

			0.64  00:47:04	1	1	6.5	1.72	5.0	13.22 2024-01-01
			1.61  01:05:03		1	10.0	3.0	5.0	18.0 2024-01-01
 00:			1.6  01:18:42		2	17.7	0.0	7.5	22.7 2024-01-01
	-		3.46  01:15:28	-	1	20.5	2.55	5.0	28.05 2024-01-01
 01::	-		3.17  01:31:25	1	2	18.4	0.0	5.0	23.4 2024-01-01
			6.64  02:11:46	1	1	32.4	11.22	5.0	48.62 2024-01-01
			0.94  01:12:10		1	8.6	2.72	5.0	16.32 2024-01-01
			1.24  01:45:32	1	1	7.9	2.58	5.0	15.48 2024-01-01
	-		1.43  02:32:30	1	1	8.6	2.72	5.0	16.32 2024-01-01
	-		0.89  03:04:24	1	1	10.0	5.0	5.0	20.0 2024-01-01
++									

only showing top 10 rows

# #How to get data frm hdfs to local

>>> small\_df.write.csv('hdfs://localhost:9000/project\_data/small\_data',header=True)

C:\Windows\system32>hadoop fs -get /project\_data/small\_data C:\Users\Excel\Documents\HDFS\_SPARK\_Project

# #Now working with SQI with SparkSqI

>>> small\_df.createOrReplaceTempView('taxi\_data')

>>> spark.sql(""" select

Vendorld,passenger\_count,trip\_distance,payment\_type,fare\_amount,tip\_amount,Other\_amount,to tal\_amount,pickup\_datetime,dropoff\_datetime from taxi\_data""").show(5)



Vendorld passenger_count trip_distance payment_type fare_amount tip_amount Other_amount total_amount  pickup_datetime  dropoff_datetime								
+								
2  1  0.64  1  6.5  1.72  5.0  13.22 2024-01-01 00:42:35 2024-01-01 00:47:04								
2  1  1.61  1  10.0  3.0  5.0  18.0 2024-01-01 00:57:38 2024-01-01 01:05:03								
1  1  1.6  2  17.7  0.0  7.5  22.7 2024-01-01 00:59:11 2024- 01-01 01:18:42								
2  1  0.94  1  8.6  2.72  5.0  16.32 2024-01-01 01:04:55 2024-01-01 01:12:10								
2  1  1.24  1  7.9  2.58  5.0  15.48 2024-01-01 01:40:21 2024-01-01 01:45:32								
++								
+								
only showing top 5 rows								
#Now Partition data by trip duration category for better data								
performance and efficiency								
>>> data1 = small_df.withColumn('trip_duration_min', round((unix_timestamp('dropoff_datetime') - unix_timestamp('pickup_datetime')) / 60, 2))								
>>> data1 = data1.withColumn("trip_category", when(col("trip_duration_min") < 10, "short").when((col("trip_duration_min") >= 10) & (col("trip_duration_min") < 30), "mid").otherwise("long"))								
>>> data1.show(5)								
++								
+								
VendorID passenger_count trip_distance RatecodeID payment_type fare_amount tip_amount  Other_amount total_amount  pickup_datetime  dropoff_datetime trip_duration_min trip_category								
++								
2  1  0.64  1  1  6.5  1.72  5.0  13.22 2024-01-01 00:42:35 2024-01-01 00:47:04  4.48  short								
2  1  1.61  1  1  10.0  3.0  5.0  18.0 2024-01-01 00:57:38 2024-01-01 01:05:03  7.42  short								

	1	1	1.6	1	2	17.7	0.0	7.5	22.7   2024-01-01
00:59:11 2024-01-01 01:18:42				19.52  mid					
•	•	•	3.46  01:15:28	•	•	•	•	5.0	28.05   2024-01-01
•	•	•	3.17  01:31:25	•	•	18.4  mid	0.0	5.0	23.4 2024-01-01
-	•		+		•	+	+	+	+
1.									

only showing top 5 rows

#### #Put this data on hdfs

>>>data1.write.partitionBy("trip\_category").parquet("hdfs://localhost:9000/project\_data/partition\_trip\_category")

#### #Create new table by from this data

>>> spark.sql(""" create table data\_partitioned using PARQUET PARTITIONED BY (trip\_category) AS SELECT VendorID, passenger\_count, trip\_distance, payment\_type, total\_amount, pickup\_datetime, dropoff\_datetime, trip\_duration\_min, trip\_category FROM taxi\_data\_temp """)

```
>>> partition_trip = spark.sql(""" select * from taxi_data_temp """)
>>> partition_trip.show(5)
----+-----+
|VendorID|passenger_count|trip_distance|RatecodeID|payment_type|fare_amount|tip_amount|
Other_amount|total_amount| pickup_datetime|
dropoff_datetime|trip_duration_min|trip_category|
----+----------+------+
    2|
             1|
                   0.64
                                    1|
                                          6.5
                                                1.72
                                                         5.0
                                                                13.22 | 2024-01-01
                            1|
00:42:35 | 2024-01-01 00:47:04 |
                                  4.48
                                          short
                                         10.0
                                                         5.0
                                                                 18.0 | 2024-01-01
    2|
             1|
                   1.61
                            1|
                                    1
                                                 3.0
00:57:38 | 2024-01-01 01:05:03 |
                                  7.42
                                          short|
                    1.6
                            1|
                                   2|
                                         17.7|
                                                         7.5|
    1|
             1|
                                                 0.0
                                                                22.7 | 2024-01-01
00:59:11 | 2024-01-01 01:18:42 |
                                 19.52
                                            mid|
                                                                 28.05 | 2024-01-01
    2|
             1|
                   3.46
                            1|
                                    1|
                                                 2.55
                                                          5.0
                                         20.5
00:56:43 | 2024-01-01 01:15:28 |
                                 18.75
                                            mid|
```

