

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_surc
harge')).alias('Other_amount'),col('total_amount'))
```

```
>>> my_df.show(5)
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+

|VendorID|tpep_pickup_datetime|tpep_dropoff_datetime|passenger_count|trip_distance|RatecodeID|payment_type|fare_amount|tip_amount|Other_amount|total_amount|
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+

|    2| 2024-01-01 00:57:55| 2024-01-01 01:17:43|         1|      1.72|         1|         2|      17.7|
0.0|         5.0|         22.7|

|    1| 2024-01-01 00:03:00| 2024-01-01 00:09:36|         1|      1.8|         1|         1|      10.0|
3.75|         7.5|         18.75|

|    1| 2024-01-01 00:17:06| 2024-01-01 00:35:01|         1|      4.7|         1|         1|      23.3|
3.0|         7.5|         31.3|

|    1| 2024-01-01 00:36:38| 2024-01-01 00:44:56|         1|      1.4|         1|         1|      10.0|
2.0|         7.5|         17.0|

|    1| 2024-01-01 00:46:51| 2024-01-01 00:52:57|         1|      0.8|         1|         1|       7.9|
3.2|         7.5|         16.1|
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+
```

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_datetim
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 amount by different payment types

```
>>> result_df = new_df.groupby('payment_type').agg(count('payment_type').alias('count'),
sum('total_amount').alias('total_sum'))
```

```
>>> formatted_df = result_df.withColumn('total_sum', format_number(col('total_sum'), 2))
```

```
>>> formatted_df.show()
```

```
+-----+-----+-----+
|payment_type| count|  total_sum|
+-----+-----+-----+
|      1|2298388|64,528,749.54|
|      3| 10563| 235,422.62|
|      2| 422752|10,089,171.71|
|      4| 22759| 571,936.35|
+-----+-----+-----+
```

Q2. Find all trips where the tip_amount was greater than 15% of the fare_amount

```
>>> high_tips_trip = new_df.filter(col('tip_amount') > 0.15 * col('total_amount'))
```

```
>>> high_tips_trip.show(5)
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
-----+-----+
|VendorID|passenger_count|trip_distance|RatecodeID|payment_type|fare_amount|tip_amount|
Other_amount|total_amount| pickup_datetime| dropoff_datetime|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
-----+-----+
| 2| 3| 17.77| 2| 1| 70.0| 16.04| 8.44| 96.23|2024-01-01
00:51:53|2024-01-01 01:20:07|
| 2| 1| 3.22| 1| 1| 20.5| 5.1| 5.0| 30.6|2024-01-01
00:54:53|2024-01-01 01:15:08|
| 1| 1| 0.6| 1| 1| 10.7| 20.0| 7.5| 35.7|2024-01-01
00:45:19|2024-01-01 00:56:09|
| 2| 2| 18.85| 2| 1| 70.0| 15.69| 8.44| 95.88|2024-01-01
00:00:05|2024-01-01 00:26:44|
```

	2	1	1.48	1	1	12.1	4.28	5.0	21.38	2024-01-01
00:31:47	2024-01-01	00:43:45								

```

+-----+-----+-----+-----+-----+-----+-----+-----+-----+
-----+-----+

```

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
----	--------

```

+----+-----+

```

only showing top 20 rows

Q4. Find avg duration by all week days

```

>>> 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|
+-----+-----+-----+-----+-----+-----+-----+-----+
----+-----+
```

1	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
1	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
1	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
1	2	1	3.46	1	1	20.5	2.55	5.0	28.05	2024-01-01 00:56:43	2024-01-01 01:15:28
1	2	2	3.17	1	2	18.4	0.0	5.0	23.4	2024-01-01 01:15:07	2024-01-01 01:31:25
1	2	2	6.64	1	1	32.4	11.22	5.0	48.62	2024-01-01 01:49:42	2024-01-01 02:11:46
1	2	1	0.94	1	1	8.6	2.72	5.0	16.32	2024-01-01 01:04:55	2024-01-01 01:12:10
1	2	1	1.24	1	1	7.9	2.58	5.0	15.48	2024-01-01 01:40:21	2024-01-01 01:45:32
1	2	1	1.43	1	1	8.6	2.72	5.0	16.32	2024-01-01 02:27:05	2024-01-01 02:32:30
1	2	2	0.89	1	1	10.0	5.0	5.0	20.0	2024-01-01 02:54:17	2024-01-01 03:04:24

```

+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
-----+

```

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 SQL with SparkSql

```
>>> small_df.createOrReplaceTempView('taxi_data')
```

```
>>> spark.sql(""" select
```

```
VendorId,passenger_count,trip_distance,payment_type,fare_amount,tip_amount,Other_amount,to
tal_amount,pickup_datetime,dropoff_datetime from taxi_data""").show(5)
```

```

+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
-----+

```



```
|VendorId|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					
	2	1	3.46	1	1	20.5	2.55	5.0	28.05	2024-01-01
00:56:43	2024-01-01	01:15:28		18.75	mid					
	2	2	3.17	1	2	18.4	0.0	5.0	23.4	2024-01-01
01:15:07	2024-01-01	01:31:25		16.3	mid					

```

+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
-----+-----+-----+-----+

```

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	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					
	2	1	3.46	1	1	20.5	2.55	5.0	28.05	2024-01-01
00:56:43	2024-01-01	01:15:28		18.75	mid					

	2	2	3.17	1	2	18.4	0.0	5.0	23.4	2024-01-01
01:15:07	2024-01-01	01:31:25		16.3	mid					

+-----+	-----+	-----+	-----+	-----+	-----+	-----+	-----+	-----+	-----+	-----+
-----+	-----+	-----+	-----+	-----+						

only showing top 5 rows