Introduction to EDA with PySpark

This presentation will guide you through an Exploratory Data Analysis (EDA) on the NYC Taxi Trip dataset using the powerful PySpark library. We'll dive into the data, uncover insights, and learn how to leverage PySpark's capabilities for effective data analysis.

Problem Statement

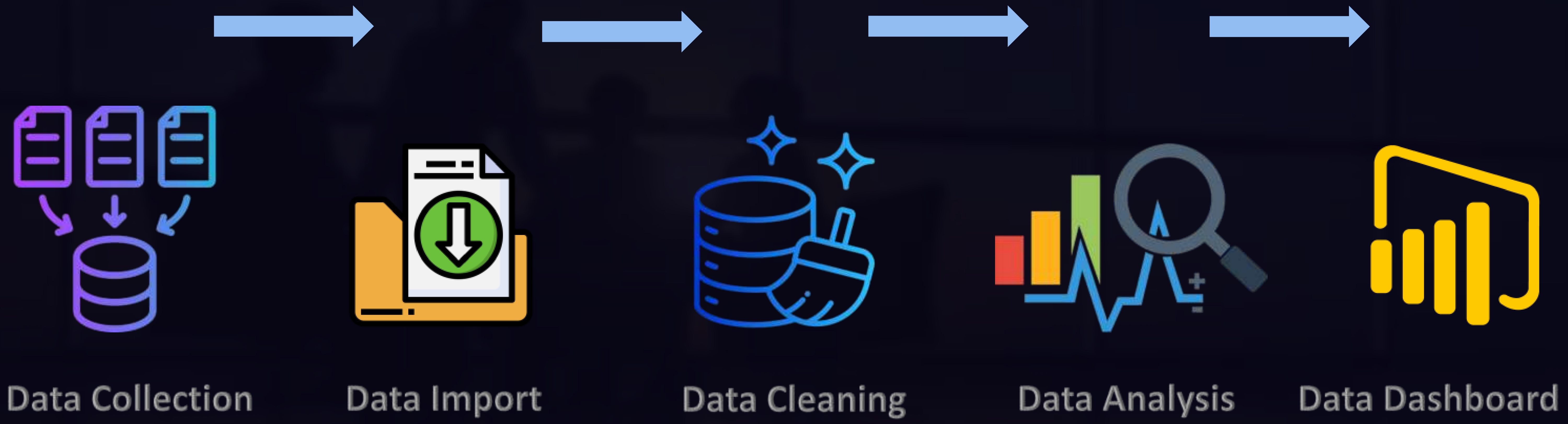
The problem statement of the NYC Taxi Data Analysis and Optimization project is to analyze the vast amount of taxi trip data from New York City (NYC) and identify opportunities to optimize various aspects of the taxi system. This includes improving operational efficiency, enhancing service quality, and maximizing revenue for taxi drivers and companies.

- What are the pickup and drop hotspots?
- Peak Hours to book a trip
- Average Passengers per trip
- Most Trip type within or intercity
- Toll Price
- Most busy days of a week
- Average Tip amount
- How distance change the price of the ride?

These are some the questions we need answer in order to increase revenue, manage the taxi availability and overall estimate the profit from each trip.

So let's get started.

Workflow of the Project



Tools for the Project

Overview of NYC Taxi Trip Data

1. ## Vendor ID

A code indicating the TPEP provider that provided the record.

1= Creative Mobile Technologies, LLC   2= VeriFone Inc.

2. ## RateCodeID

The final rate code in effect at the end of the trip.

1= Standard rate 2=JFK 3=Newark 4=Nassau or Westchester 5=Negotiated fare 6=Group ride

3. ## Payment_type

A numeric code signifying how the passenger paid for the trip.

1= Credit card 2= Cash 3= No charge 4= Dispute 5= Unknown 6= Voided trip

Overview of NYC Taxi Trip Data

Columns: [‘VendorID', 'lpep_pickup_datetime', 'lpep_dropoff_datetime', 'store_and_fwd_flag', 'RatecodeID', 'PULocationID', 'DOLocationID', 'passenger_count', 'trip_distance', 'fare_amount', 'extra', 'mta_tax', 'tip_amount', 'tolls_amount', 'ehail_fee', 'improvement_surcharge', 'total_amount', 'payment_type', 'trip_type', 'congestion_surcharge']

1 Vendor ID

A code indicating the TPEP provider that provided the record.

1= Creative Mobile Technologies, LLC   2= VeriFone Inc.

2 RateCodeID

The final rate code in effect at the end of the trip.

1= Standard rate 2=JFK 3=Newark 4=Nassau or Westchester 5=Negotiated fare 6=Group ride

3 Payment_type

A numeric code signifying how the passenger paid for the trip.

1= Credit card 2= Cash 3= No charge 4= Dispute 5= Unknown 6= Voided trip

4 Trip_type

A numeric code signifies type of trip.

1= City Trip 2= InterCity Trip

Importing the Dataset

1

Collect The Data

We collected the data from Kaggle and store it in our local disk as a csv file and description of the data as a txt file

2

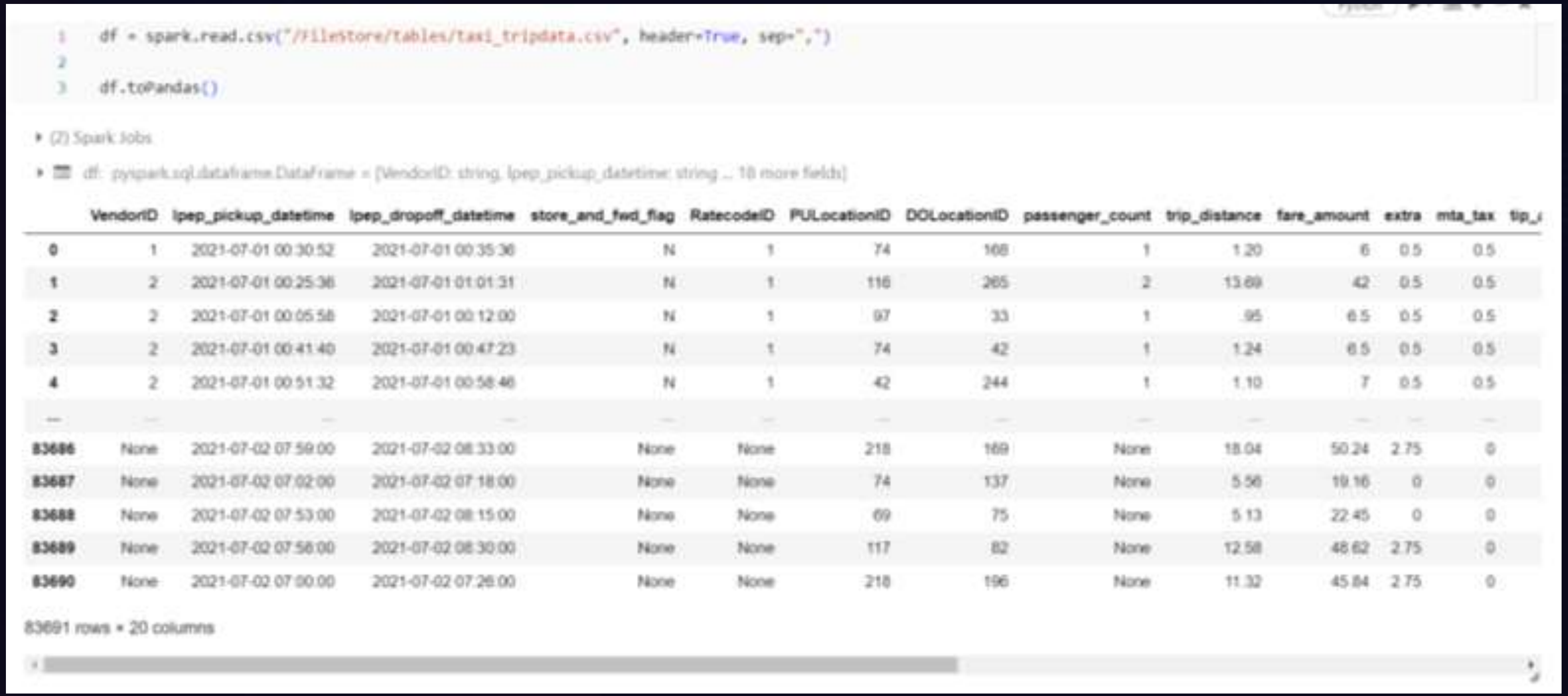
Import the data to Databricks' database

We open out Databricks profile. Go to catalog choose a database and create table. Then upload the data from the local storage and read the first row as Header.

3

Read the data in spark.read.csv

Create a spark data frame as spark.read.csv and give the file path to the table we just created



```
df = spark.read.csv("/FileStore/tables/taxi_tripdata.csv", header=True, sep=",")

df.toPandas()
```

▸ (2) Spark Jobs

▸ df: pyspark.sql.dataframe.DataFrame = [VendorID: string, lpep_pickup_datetime: string ... 18 more fields]

	VendorID	lpep_pickup_datetime	lpep_dropoff_datetime	store_and_fwd_flag	RatecodeID	PULocationID	DOLocationID	passenger_count	trip_distance	fare_amount	extra	mta_tax	tip_
0	1	2021-07-01 00:30:52	2021-07-01 00:35:36	N	1	74	168	1	1.20	6	0.5	0.5	
1	2	2021-07-01 00:25:36	2021-07-01 01:01:31	N	1	116	265	2	13.69	42	0.5	0.5	
2	2	2021-07-01 00:05:58	2021-07-01 00:12:00	N	1	97	33	1	.95	6.5	0.5	0.5	
3	2	2021-07-01 00:41:40	2021-07-01 00:47:23	N	1	74	42	1	1.24	6.5	0.5	0.5	
4	2	2021-07-01 00:51:32	2021-07-01 00:58:46	N	1	42	244	1	1.10	7	0.5	0.5	
...	...	...	...	...	...	...	...	...	...	...	...	...	
83686	None	2021-07-02 07:59:00	2021-07-02 08:33:00	None	None	218	169	None	18.04	50.24	2.75	0	
83687	None	2021-07-02 07:02:00	2021-07-02 07:18:00	None	None	74	137	None	5.56	19.16	0	0	
83688	None	2021-07-02 07:53:00	2021-07-02 08:15:00	None	None	69	75	None	5.13	22.45	0	0	
83689	None	2021-07-02 07:56:00	2021-07-02 08:30:00	None	None	117	82	None	12.58	48.62	2.75	0	
83690	None	2021-07-02 07:00:00	2021-07-02 07:26:00	None	None	218	196	None	11.32	45.84	2.75	0	

83691 rows × 20 columns

Handling Missing Values and Outliers

1. ## Identifying the Columns with Null Values

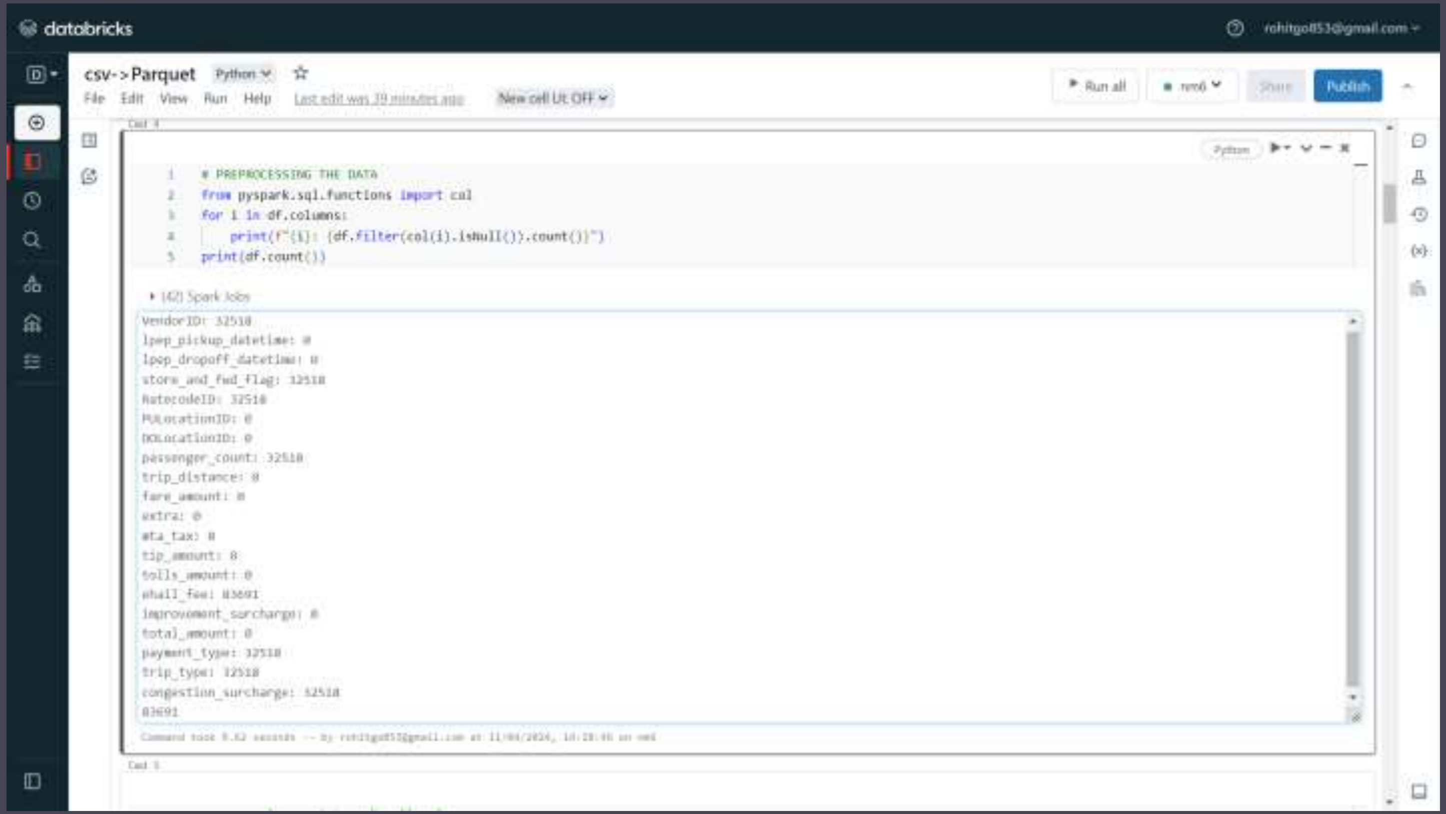
   We'll thoroughly examine the dataset for missing values and understand their potential impact on our analysis.

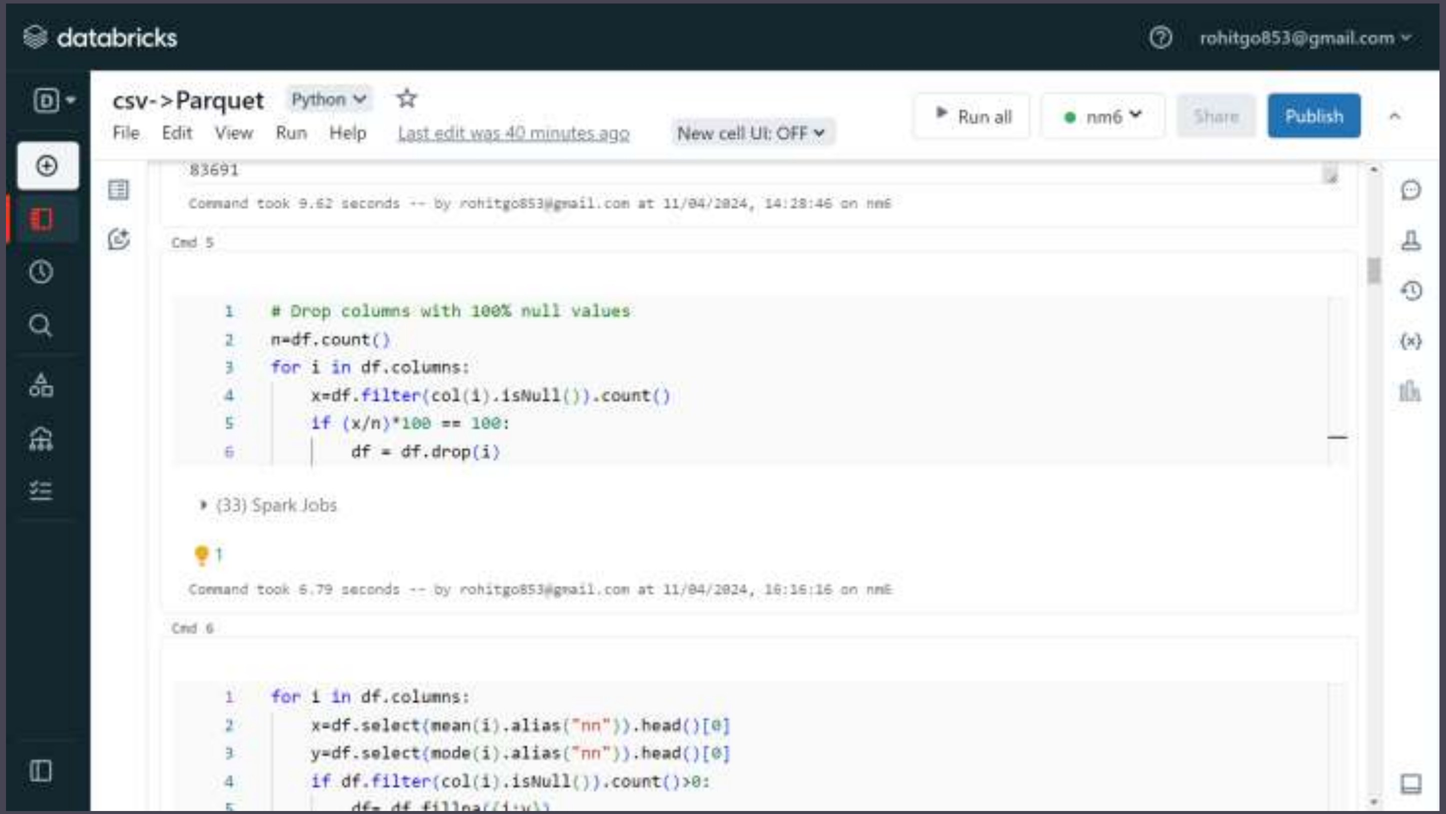
2. ## Dropping the columns with 90% Null Values

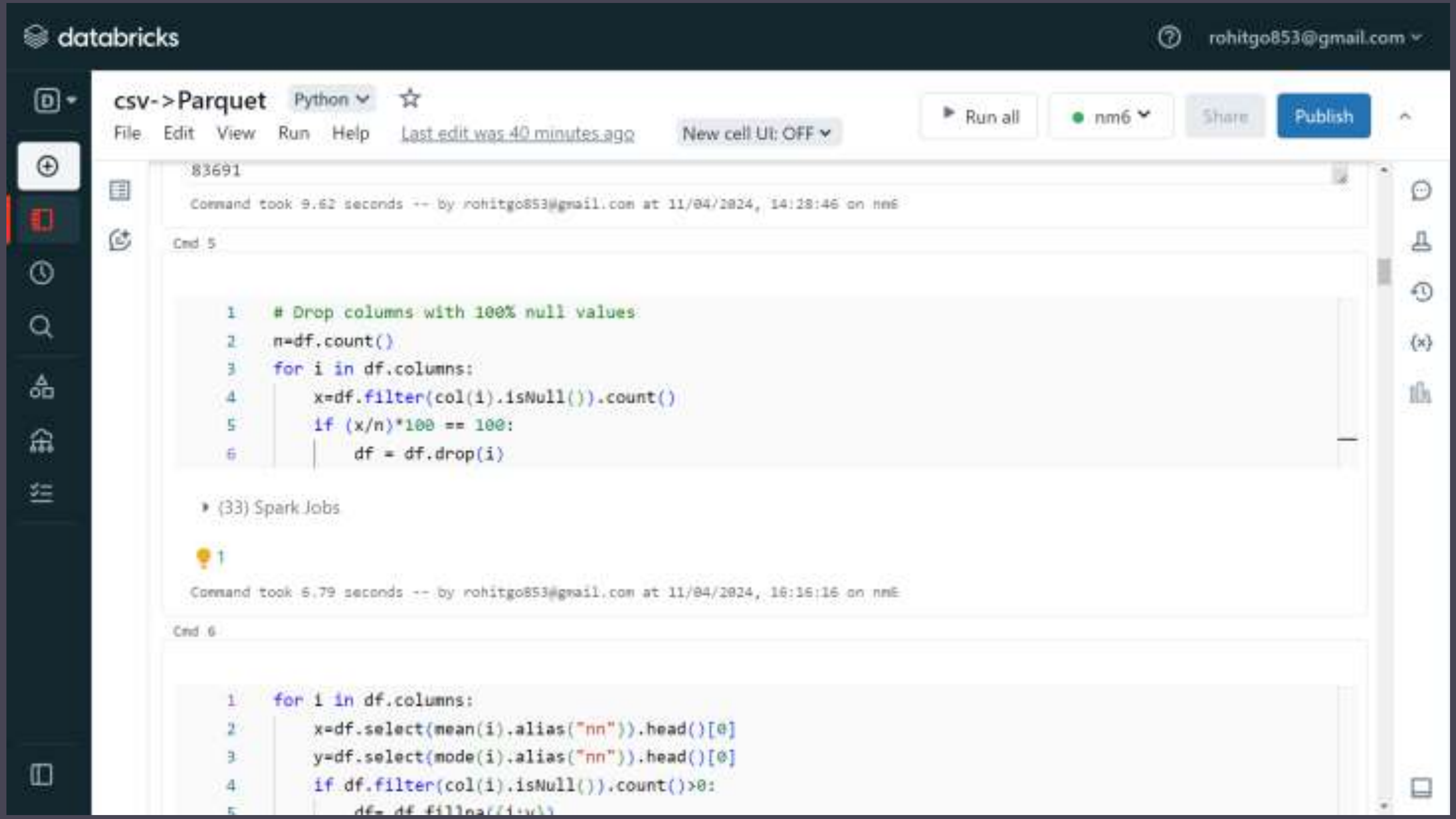
   Depending on the percentage of missing values in a column, we are filtering the dataset.

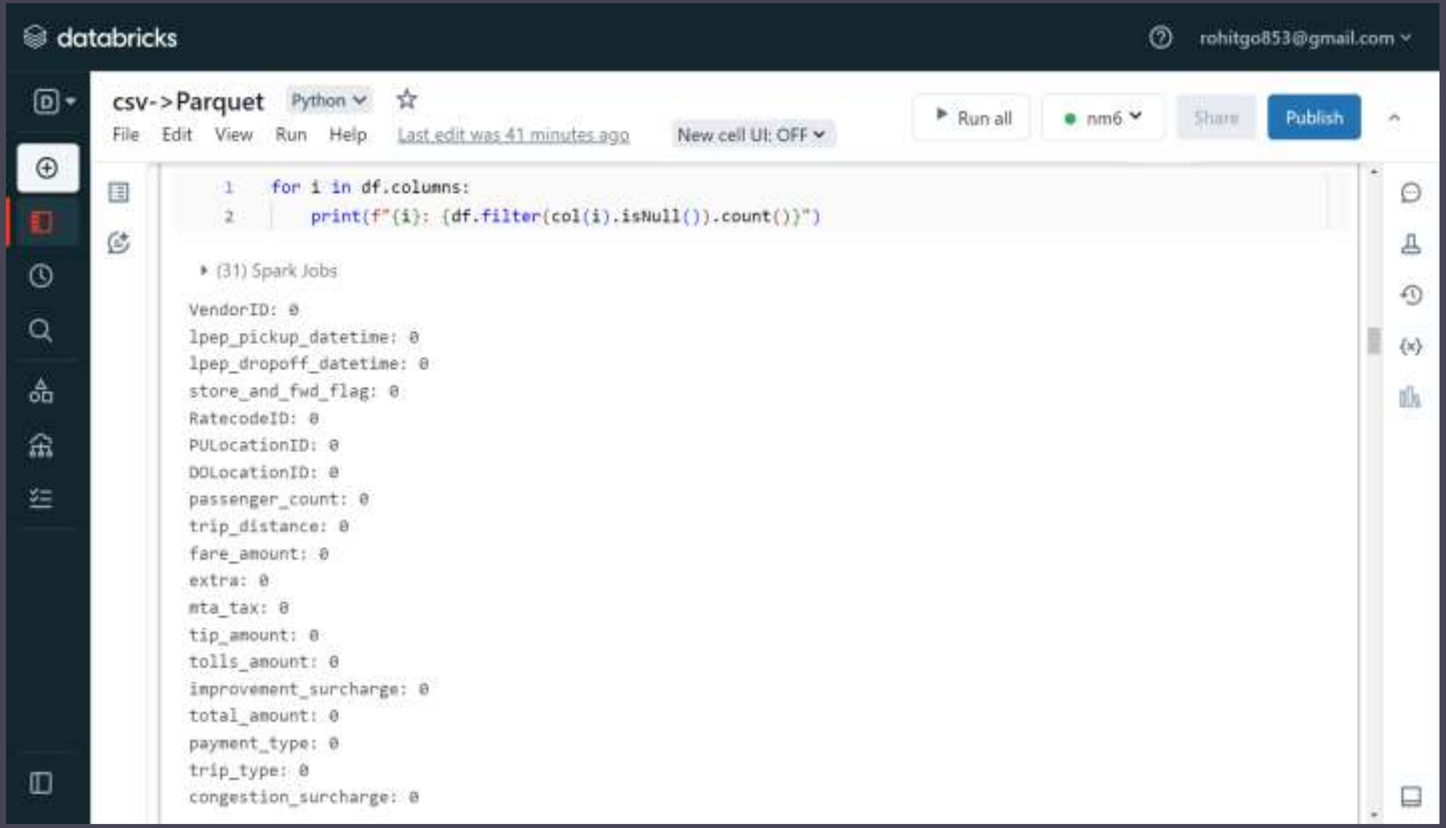
3. ## Replacing the None Values with the most common or mean value of the column

   We are filling the missing values with the mean or mode of the column based on categorical or numerical value.


databricks
csv->Parquet
# PREPROCESSING THE DATA
from pyspark.sql.functions import col
for i in df.columns:
print(f"{i}: {df.filter(col(i).isNull()).count()}")
print(df.count())
VendorID: 32518
lpep_pickup_datetime: 0
lpep_dropoff_datetime: 0
store_and_fwd_flag: 32518
RatecodeID: 32518
PULocationID: 0
DOLocationID: 0
passenger_count: 32518
trip_distance: 0
fare_amount: 0
extra: 0
mta_tax: 0
tip_amount: 0
tolls_amount: 0
ehail_fee: 83691
improvement_surcharge: 0
total_amount: 0
payment_type: 32518
trip_type: 32518
congestion_surcharge: 32518
83691


databricks
rohitgo853@gmail.com
csv->Parquet
Python
File Edit View Run Help
Last edit was 40 minutes ago
New cell UI: OFF
Run all
nm6
Share
Publish
83691
Command took 9.62 seconds -- by rohitgo853@gmail.com at 11/04/2024, 14:28:46 on nm6
Cmd 5
# Drop columns with 100% null values
n=df.count()
for i in df.columns:
x=df.filter(col(i).isNull()).count()
if (x/n)*100 == 100:
df = df.drop(i)
(33) Spark Jobs
Command took 6.79 seconds -- by rohitgo853@gmail.com at 11/04/2024, 16:16:16 on nm6
Cmd 6
for i in df.columns:
x=df.select(mean(i).alias("nn")).head()[0]
y=df.select(mode(i).alias("nn")).head()[0]
if df.filter(col(i).isNull()).count()>0:


databricks
rohitgo853@gmail.com
csv->Parquet
Python
File Edit View Run Help
Last edit was 40 minutes ago
New cell UI: OFF
Run all
nm6
Share
Publish
83691
Command took 9.62 seconds -- by rohitgo853@gmail.com at 11/04/2024, 14:28:46 on nm6
Cmd 5
# Drop columns with 100% null values
n=df.count()
for i in df.columns:
x=df.filter(col(i).isNull()).count()
if (x/n)*100 == 100:
df = df.drop(i)
(33) Spark Jobs
Command took 6.79 seconds -- by rohitgo853@gmail.com at 11/04/2024, 16:16:16 on nm6
Cmd 6
for i in df.columns:
x=df.select(mean(i).alias("nn")).head()[0]
y=df.select(mode(i).alias("nn")).head()[0]
if df.filter(col(i).isNull()).count()>0:


databricks
rohitgo853@gmail.com
csv->Parquet
Python
File Edit View Run Help
Last edit was 41 minutes ago
New cell UI: OFF
Run all
nm6
Share
Publish
for i in df.columns:
print(f"{i}: {df.filter(col(i).isNull()).count()}")
(31) Spark Jobs
VendorID: 0
lpep_pickup_datetime: 0
lpep_dropoff_datetime: 0
store_and_fwd_flag: 0
RatecodeID: 0
PULocationID: 0
DOLocationID: 0
passenger_count: 0
trip_distance: 0
fare_amount: 0
extra: 0
mta_tax: 0
tip_amount: 0
tolls_amount: 0
improvement_surcharge: 0
total_amount: 0
payment_type: 0
trip_type: 0
congestion_surcharge: 0

How man trips Recorded?

83691

```
# How man trips Recorded?
df.count()
```

(2) Spark Jobs

Out[187]: 83691

Command took 0.49 seconds -- by rohitgo853@gmail.com at 11/04/2024, 14:28:46 on nm6

#What is the average trip distance?

194.35469931055877

```
#What is the average trip distance?
df.select(mean("trip_distance").alias("Average_trip_distance")).show()

```

(2) Spark Jobs

```
+---------------------+
|Average_trip_distance|
+---------------------+
|   194.35469931055877|
+---------------------+
```

What are the top 5 PULocationIDs where trips start from?

PULocationID	count
74	8770
75	7713
41	4761
42	3229
95	2486

Cmd 13

```
# What are the top 5 PULocationIDs where trips start from?
df.groupBy("PULocationID").count().orderBy(col("count").desc()).limit(5).show()
```

▸ (2) Spark Jobs

```
+------------+-----+
|PULocationID|count|
+------------+-----+
|          74| 8770|
|          75| 7713|
|          41| 4761|
|          42| 3229|
|          95| 2486|
+------------+-----+
```

#What are the top 5 DOLocationIDs where trips end?

DOLocationID	count
74	3666
75	3122
42	2904
41	2527
236	1700

```
#What are the top 5 DOLocationIDs where trips end?
df.groupBy("DOLocationID").count().orderBy(col("count").desc()).limit(5).show()
```

▸ (2) Spark Jobs

```
+------------+-----+
|DOLocationID|count|
+------------+-----+
|          74| 3666|
|          75| 3122|
|          42| 2904|
|          41| 2527|
|         236| 1700|
```

Peak Hours for Someone to start a ride

PickUp Hours	count
10	6096
11	6092
9	5798

```
1  # Peak Hours for Someone to start a ride
2  df.groupBy(hour("lpep_pickup_datetime").alias("PickUp_Hours")).count().orderBy(col("count").desc()).limit(5).show()
```

▸ (2) Spark Jobs

```
+------------+-----+
|PickUp_Hours|count|
+------------+-----+
|          10| 6096|
|          11| 6092|
|           9| 5798|
|          12| 5766|
|          15| 5744|
+------------+-----+
```

What is the distribution of trips based on the payment type?

|payment type|count| |
1 |62508|
2	20831
3	307
4	44
5	1

```
1  # What is the distribution of trips based on the payment type?
2  df.groupBy("payment_type").count().orderBy("payment_type").show()
3
```

▸ (2) Spark Jobs

```
+------------+-----+
|payment_type|count|
+------------+-----+
|           1|62508|
|           2|20831|
|           3|  307|
|           4|   44|
|           5|    1|
+------------+-----+
```

What is the most common RatecodeID in the dataset?

|RatecodeID|count
| | 1
|81512|
| 5 |
1954|
2	158
4	41
3	26

```
Cmd 17

# What is the most common RatecodeID in the dataset?
df.groupby("RatecodeID").count().orderBy(col("count").desc()).limit(10).show()

▸ (2) Spark Jobs

+----------+-----+
|RatecodeID|count|
+----------+-----+
|         1|81512|
|         5| 1954|
|         2|  158|
|         4|   41|
|         3|   26|
+----------+-----+
```

Average fare amount for different vendors

|
1|17.013627772674567|
|
2|24.925072694291224|

```
# Average fare amount for different vendors
df = df.withColumn("total_amount", col("total_amount").cast(DoubleType()))
df.groupby("VendorID").mean("total_amount").show()

▸ (2) Spark Jobs
▸ df: pyspark.sql.dataframe.DataFrame = [VendorID: string, lpep_pickup_datetime: string ... 17 more fields]

+--------+------------------+
|VendorID| avg(total_amount)|
+--------+------------------+
|       1|17.013627772674567|
|       2|24.925072694291224|
+--------+------------------+
```

#What is the distribution of passenger counts?

1	76645
2	3922
5	1240
6	1018
3	626
4	181

```
#What is the distribution of passenger counts?
df.groupBy(col("passenger_count").alias("No. of passengers")).count().orderBy(col("count").desc()).show()
```

▸ (2) Spark Jobs

```
+-----------------+-----+
|No. of passengers|count|
+-----------------+-----+
|                1|76645|
|                2| 3922|
|                5| 1240|
|                6| 1018|
|                3|  626|
|                4|  181|
|                0|   56|
|                7|    2|
|               32|    1|
+-----------------+-----+
```

#What is the average fare amount for trips?

20.38830459667026

```
#What is the average fare amount for trips?
df.select(mean("fare_amount")).show()
```

▸ (2) Spark Jobs

```
+-----------------+
| avg(fare_amount)|
+-----------------+
|20.38830459667026|
+-----------------+
```

How many trips have a tolls amount greater than zero?

Average Tolls
6.413921953613792

Tolls _amount greater than zero
8149

```
# How many trips have a tolls amount greater than zero?
df.filter(col("tolls_amount")>0).count()
n=df.filter(col("tolls_amount")>0)
n.select(mean("tolls_amount").alias("Average Tolls_amount"),count("tolls_amount").alias("Tolls_amount greater than zero")).show()
```

▸ (4) Spark Jobs

▸ n: pyspark.sql.dataframe.DataFrame = [VendorID: string, lpep_pickup_datetime: string ... 17 more fields]

```
+--------------------+------------------------------+
|Average Tolls_amount|Tolls_amount greater than zero|
+--------------------+------------------------------+
|   6.413921953613792|                          8149|
+--------------------+------------------------------+
```

What is the distribution of trip types?

| City Trip|81931|
| InterCity Trip| 1760|

```
# What is the distribution of trip types?
df= df.withColumn("TRIPS", when(df["trip_type"]==1,"City Trip")
                    .when(df["trip_type"]==2,"InterCity Trip"))

df.groupBy(col("TRIPS").alias("No. of passengers")).count().orderBy(col("count").desc()).show()
```

▸ (2) Spark Jobs

▸ df: pyspark.sql.dataframe.DataFrame = [VendorID: string, lpep_pickup_datetime: string ... 18 more fields]

```
+-----------------+-----+
|No. of passengers|count|
+-----------------+-----+
|        City Trip|81931|
|   InterCity Trip| 1760|
+-----------------+-----+
```

What is the most common day of the week for trips

Sat	14964
Fri	14929
Sun	12862
Thu	11752
Wed|10875|
Tue|10403|
Mon	7906

```
# What is the most common day of the week for trips?
df=df.withColumn("DATE",col("lpep_pickup_datetime").cast("date")).withColumn("DayW",dayofweek(col("lpep_pickup_datetime")))

df= df.withColumn("DW", when(df["DayW"]==1,"Mon")
    .when(df["DayW"]==2,"Tue")
    .when(df["DayW"]==3,"Wed")
    .when(df["DayW"]==4,"Thu")
    .when(df["DayW"]==5,"Fri")
    .when(df["DayW"]==6,"Sat")
    .when(df["DayW"]==7,"Sun"))

df.groupBy(col("DW").alias("Day of Week")).count().orderBy(col("count").desc()).show()
df=df.drop(col("DATE"),col("DW"),col("DayW"))
```

▸ (2) Spark Jobs

▸ df: pyspark.sql.dataframe.DataFrame = [VendorID: string, lpep_pickup_datetime: string ... 18 more fields]

```
+-----------+-----+
|Day of Week|count|
+-----------+-----+
|        Sat|14964|
|        Fri|14929|
|        Sun|12862|
|        Thu|11752|
|        Wed|10875|
|        Tue|10403|
|        Mon| 7906|
+-----------+-----+
```

Is there any correlation between trip distance and total amount paid?

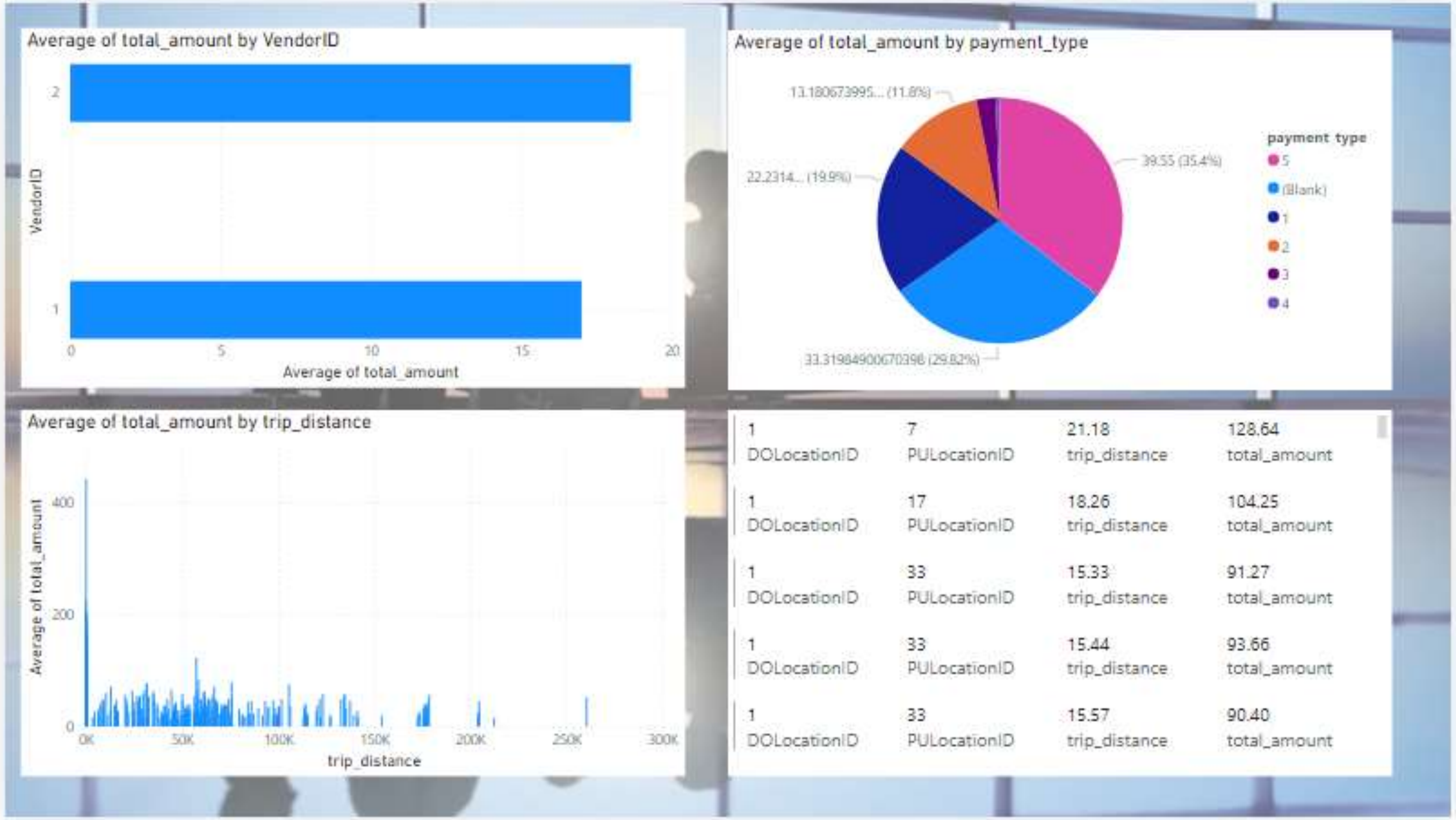
0.02507182116693
05

```
# Is there any correlation between trip distance and total amount paid?
df.select(corr("trip_distance","total_amount")).first()[0]
```

▸ (2) Spark Jobs

Out[251]: 0.0250718211669305

Data Dashboard in PowerBI



Conclusion

- Pickup Location ID hotspots are 74, 75, 41, 42, 95
- Drop Location ID hotspots are 74, 75, 42, 236
- Peak Hours for a ride is 9-12 am and 3pm
- Trips are mostly pay in credit cards and cash
- Average fare amount of Vendor 1 is 17.01 and Vendor 2 is 24.92
- Mostly number of passengers for a ride is 1, 2 or 5
- Passengers mostly use within City Trip_type
- Average Toll amount for 8149 rides is 6.41
- Saturday and Friday are the most busy days of the week
- Average tip amount is 1.05
- The relation between trip distance and total amount is weak positive 0.025.