YELLOW TAXI DATA ANALYSIS

Raavi Shiva Seshi Reddy (19BIT0173),

Thumma Joseph Sathwik Reddy (19BIT0178),

Shyamkumar M (19BIT0148),

Beeravelly Manish Rao (19BIT0333).

SCHOOL OF INFORMATION TECHNOLOGY AND ENGINEERING

Project Report for J component of

ITE2013 Big Data Analytics

Under guidance of Prof. Ranichandra C

Slot: D1+TD1+D2+TD2

Summer Semester V

On

Yellow Taxi Data Analysis

By

Raavi Shiva Seshi Reddy (19BIT0173),

Thumma Joseph Sathwik Reddy (19BIT0178),

Shyamkumar M (19BIT0148),

Beeravelly Manish Rao (19BIT0333).

1	**Abstract**	**4**
2	**Discovery Phase** **2.a Problem Statement** **2.b Literature Survey** **2.c S/w and H/w Requirements**	**4-5**
3	**Data Preparation Phase** **3.a Data set Description** **3.b Data Pre-processing**	**6-7**
4	**Model Planning** **4.a Module 1** **4.b Module 2**	**8-9**
5	**Model Building Phase** **5.a. Implementation of Module1** **5.b Implementation of Module 2**	**10-12**
6	**Communicating Results Phase** **6.a Output of each Module** **6.b Visualization (if used)** **6.c Comparison charts**	**13-17**
7	**Conclusion**	**18**
8	**References**	**18**
9	**Appendix**	**19-20**

1.Abstract

The yellow taxi dataset is available in NYC Open Data application. The details of the data set is available in NYC Open Data application. The data is the collection of 2018 NYC taxi data. We are going to analysis this data by following certain criteria of questions. We are going to use Hadoop components, mainly by using Hive. At initial stage we are going to explore the data and also preprocess the data for cleaning and exploration. Then doing the analysis by using Hive we are going to visualize by using the data visualization tool such as tableau to show case the reports of the data which help us to understand the data very well by visualization and also helps us to get some conclusions for analyzing the dataset. Our project main aim to provide the clear view of trends in taxi analysis and visualizing the data.

2.Discovery Phase

2.a Problem Statement

In this case study, we are giving a real world example of how to use HIVE on top of the HADOOP for different exploratory data analysis. In here, we have a predefined dataset (2018_Yellow_Taxi_Trip_Data.csv) having more than 15 columns and more than 100000 records in it. These help us to understand the trends of the yellow taxi trips which help to improve the business. These analytics reports helpful for business development in such way providing better services for customers and making some essential decision toward the activities of business.

2.b Literature Survey

- Marie Stephen Leo [1] analyze the New York Taxi data set using python and machine learning to predicting the taxi fare using Regression models such as Linear Regression, Lasso Regression Hyper tuning. They done some predictive to analytics to predict the taxi fare.

- Zhizhen Liu, Hong Chen, Yan Li, and Qi Zhang [2] done Taxi demand prediction based on combination Model in hotspots. They compare the three machine learning model approaches such as random forest, regression model, and combination forecasting model. This experimental results predicts the taxi demand by considering the different environmental conditions.

- Jun Xu, Rouhollah Rahmatizadeh, Ladislau Boloni and Damla Turgut [3] they done the real time prediction of taxi demand using recurrent neural networks. They use the one of the most best sequence learning models, Long short term memory (LSTM) for prediction. The experiment results predicts the taxi demand by considering the different parameters and features.

- Vivek Sachapara, Hrishikesh Shinde, Abhishek Puri, Shraddha Aggrawal and Prof. Sachin Wandre [4] done the Big Data Analytics on Cab Dataset using hadoop. They used Hadoop MapReduce and spark to analyze the data. They provide the analyze

report by comparing the two tools MapReduce and Spark and considering the time of execution.

- Bayan Alghuraybi, Krishna Marvaniya, Guojun xia and Jongwook Woo [5] done the Analyze NYC taxi data using Hadoop, MapReduce, Hive, Power BI and Machine Learning. They done this project by creating Hive tables in Microsoft Azure blob and then they use the machine learning model such as Logistic Regression to predict if the driver will get tip or not.

- Abhishek Singh, Ashmit Narayan Rai, Ayushi Saxena, Diti Gupta, Prabal Bhatnagar [6] done the Data Analysis using the Hadoop on youtube datato explore the trends on youtbe by using the hive.

- Rotsnarani Sethy, and Mrutyunjaya Panda [7] done Big data analytics using Hadoop which is a survey report. In this paper they introduce the Hadoop in general and focus on the MapReduce algorithms.

- Alvin Jun Yong koh, Xuan Khoa nguyen, and C. Jason woodard [8] done taxi data analysis using Hadoop and Cassandra. Basically, it is a feasibility study on usage of Hadoop and Cassandra for analyzing the large data sets.

- Umang Patel , and Anil Chandan [9] done NYC taxi trip and fare Data Analytics using BigData. They use the different components of Hadoop eco system such as HDFS to store, Pig Latin to analyze the fare to get the driver revenue. This experimental results provide the complete scenario of data analyze which helpful for business operations.

- Bijesh Dhyani and Anurag Barthwal [10] done a Survey report on Big Data Analytics and they define the various components available in Hadoop ecosystem and there usge and scope of the usibility of the components in real life.

2.c S/w and H/w Requirements

H/w requirements:

1. Intel Core 2 Duo/Quad/hex/Octa or higher end 64 bit processor PC or Laptop
2. Hard Disk capacity of 1-2TB
3. 64-512 GB RAM

S/w requirements:

1. Linus operating system environment.
2. Hadoop – HDFS
3. Hive
4. Python
5. Tableau

3. Data Preparation Phase

3.a Data set Description

In this project, we have a predefined dataset (2018_Yellow_Taxi_Trip_Data.csv) having more than 15 columns and more than 100000 records in it. The dataset has different attributes. The dataset is collected from:

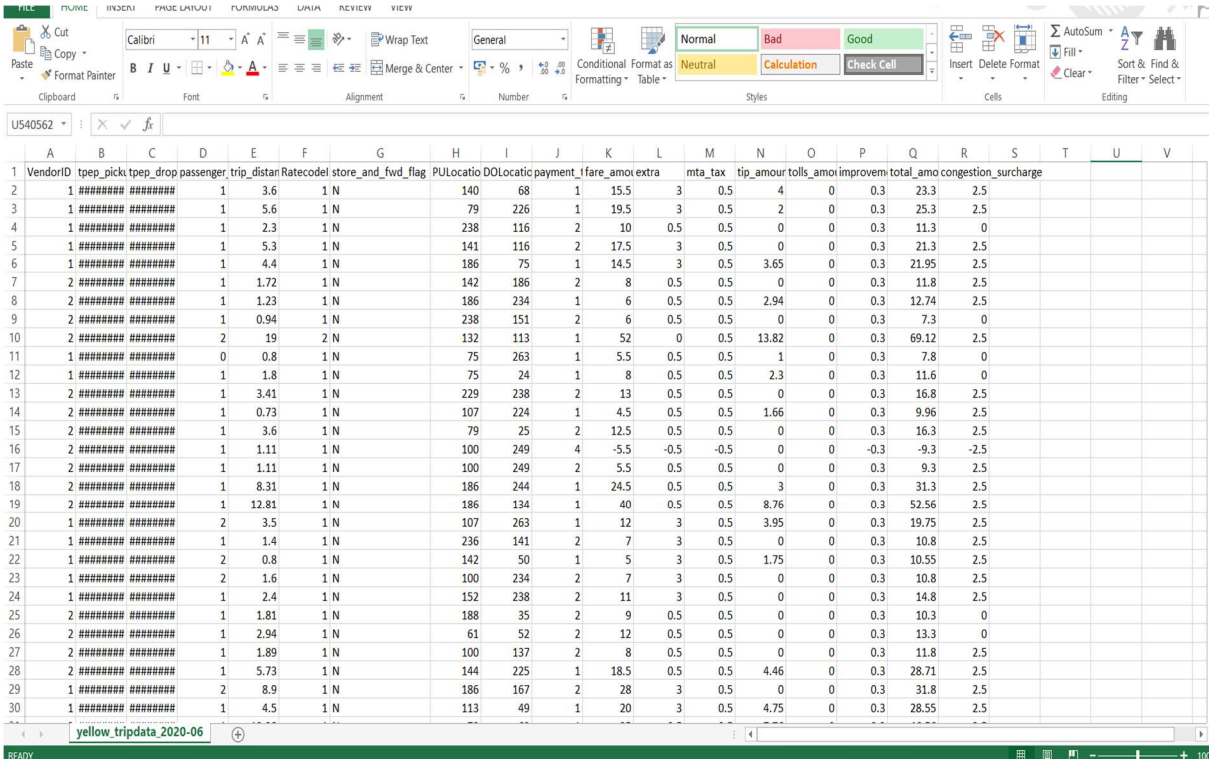
https://www.kaggle.com/datasets/microize/newyork-yellow-taxi-trip-data-2020-2019?select=yellow_tripdata_2020-06.csv

The dataset has different attributes such as:

1. **VendorID :** A code indicating the TPEP provider that provided the record.
   1= Creative Mobile Technologies, LLC; 2= VeriFone Inc.
2. **tpeppickupdatetime** : The date and time when the meter was engaged.
3. **tpepdropoffdatetime** : The date and time when the meter was disengaged.
4. **Passenger_count** : The number of passengers in the vehicle.(This is a driver-entered value)
5. **Trip_distance** : The elapsed trip distance in miles reported by the taximeter.
6. **PULocationID** : TLC Taxi Zone in which the taximeter was engaged
7. **DOLocationID** :TLC Taxi Zone in which the taximeter was disengaged
8. **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
9. **Storeandfwd_flag** : This flag indicates whether the trip record was held in vehicle memory before sending to the vendor, aka “store and forward,” because the vehicle did not have a connection to the server.
   Y= store and forward trip
   N= not a store and forward trip
10. **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
11. **Fare_amount** : The time-and-distance fare calculated by the meter.
12. **Extra :** Miscellaneous extras and surcharges. Currently, this only includes the $0.50 and $1 rush hour and overnight charges.
13. **MTA_tax :** $0.50 MTA tax that is automatically triggered based on the metered rate in use.

14. **Improvement_surcharge** : $0.30 improvement surcharge assessed trips at the flag drop. The improvement surcharge began being levied in 2015.
15. **Tip_amount** : Tip amount – This field is automatically populated for credit card tips. Cash tips are not included.
16. **Tolls_amount** : Total amount of all tolls paid in trip.
17. **Total_amount** : The total amount charged to passengers. Does not include cash tips.

Sample Data:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	VendorID	tpep_pick	tpep_drop	passenger	trip_distan	Ratecodel	store_and_fwd_flag	PULocatio	DOLocatic	payment_t	fare_amo	extra	mta_tax	tip_amour	tolls_amo	improvem	total_amo	congestion_surcharge	
2	1	########	########	1	3.6	1	N	140	68	1	15.5	3	0.5	4	0	0.3	23.3	2.5	
3	1	########	########	1	5.6	1	N	79	226	1	19.5	3	0.5	2	0	0.3	25.3	2.5	
4	1	########	########	1	2.3	1	N	238	116	2	10	0.5	0.5	0	0	0.3	11.3	0	
5	1	########	########	1	5.3	1	N	141	116	2	17.5	3	0.5	0	0	0.3	21.3	2.5	
6	1	########	########	1	4.4	1	N	186	75	1	14.5	3	0.5	3.65	0	0.3	21.95	2.5	
7	2	########	########	1	1.72	1	N	142	186	2	8	0.5	0.5	0	0	0.3	11.8	2.5	
8	2	########	########	1	1.23	1	N	186	234	1	6	0.5	0.5	2.94	0	0.3	12.74	2.5	
9	2	########	########	1	0.94	1	N	238	151	2	6	0.5	0.5	0	0	0.3	7.3	0	
10	2	########	########	2	19	2	N	132	113	1	52	0	0.5	13.82	0	0.3	69.12	2.5	
11	1	########	########	0	0.8	1	N	75	263	1	5.5	0.5	0.5	1	0	0.3	7.8	0	
12	1	########	########	1	1.8	1	N	75	24	1	8	0.5	0.5	2.3	0	0.3	11.6	0	
13	2	########	########	1	3.41	1	N	229	238	2	13	0.5	0.5	0	0	0.3	16.8	2.5	
14	2	########	########	1	0.73	1	N	107	224	1	4.5	0.5	0.5	1.66	0	0.3	9.96	2.5	
15	2	########	########	1	3.6	1	N	79	25	2	12.5	0.5	0.5	0	0	0.3	16.3	2.5	
16	2	########	########	1	1.11	1	N	100	249	4	-5.5	-0.5	-0.5	0	0	-0.3	-9.3	-2.5	
17	2	########	########	1	1.11	1	N	100	249	2	5.5	0.5	0.5	0	0	0.3	9.3	2.5	
18	2	########	########	1	8.31	1	N	186	244	1	24.5	0.5	0.5	3	0	0.3	31.3	2.5	
19	2	########	########	1	12.81	1	N	186	134	1	40	0.5	0.5	8.76	0	0.3	52.56	2.5	
20	1	########	########	2	3.5	1	N	107	263	1	12	3	0.5	3.95	0	0.3	19.75	2.5	
21	1	########	########	1	1.4	1	N	236	141	2	7	3	0.5	0	0	0.3	10.8	2.5	
22	1	########	########	2	0.8	1	N	142	50	1	5	3	0.5	1.75	0	0.3	10.55	2.5	
23	1	########	########	2	1.6	1	N	100	234	2	7	3	0.5	0	0	0.3	10.8	2.5	
24	1	########	########	1	2.4	1	N	152	238	2	11	3	0.5	0	0	0.3	14.8	2.5	
25	2	########	########	1	1.81	1	N	188	35	2	9	0.5	0.5	0	0	0.3	10.3	0	
26	2	########	########	1	2.94	1	N	61	52	2	12	0.5	0.5	0	0	0.3	13.3	0	
27	2	########	########	1	1.89	1	N	100	137	2	8	0.5	0.5	0	0	0.3	11.8	2.5	
28	2	########	########	1	5.73	1	N	144	225	1	18.5	0.5	0.5	4.46	0	0.3	28.71	2.5	
29	1	########	########	2	8.9	1	N	186	167	2	28	3	0.5	0	0	0.3	31.8	2.5	
30	1	########	########	1	4.5	1	N	113	49	1	20	3	0.5	4.75	0	0.3	28.55	2.5	

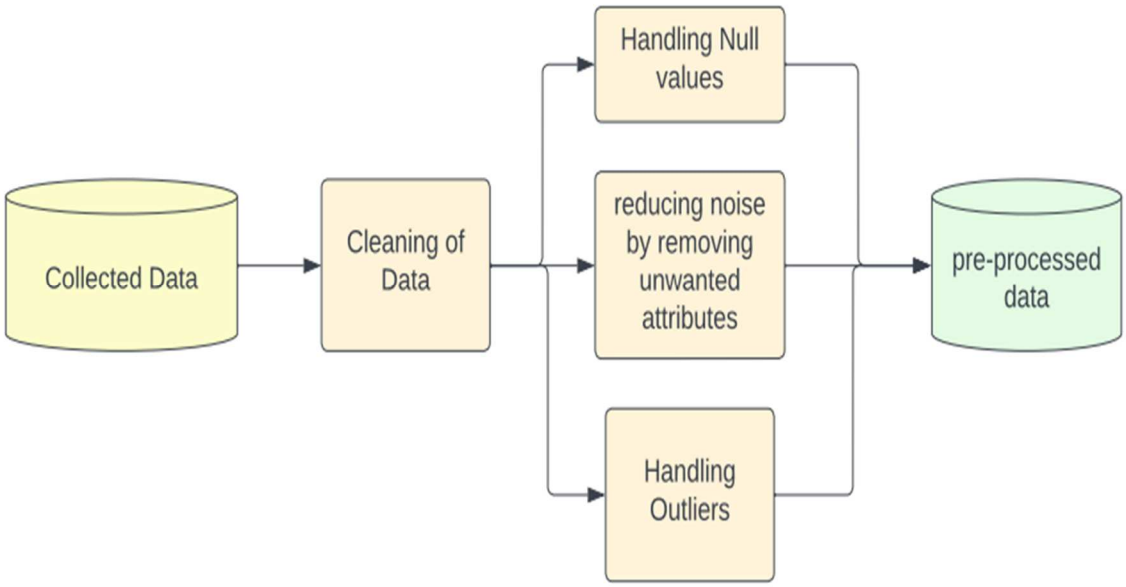
3.b Data Pre-processing

A data set collected is not directly suitable for induction (knowledge acquisition), it comprises in most cases noise, missing values, and inconsistent data set is too large, and so on. Therefore, we need to minimize the noise in data, choose a strategy for handling missing (unknown) attribute values. The process of data cleaning and preparation is highly dependent on the specific algorithm and software chosen for the classification task. The missing values, noisy, outliers are removed .The empty cells and rows are deleted and the income data is presented for use

4.Model Planning

4.a Module 1:

Organise the dataset properly and fill the missing values with 0 or either delete the missing value row.Finding out outliers and removing the outliers from the dataset manually. A python code is also developed for the same. Plot graphs for all the attributes like occupation, marital status,gender, working hours and age is income. Based on the above graphs,we arrive at a few conclusions. Python is used for data cleaning and pandas on of the powerful module of python which helps in data cleaning.

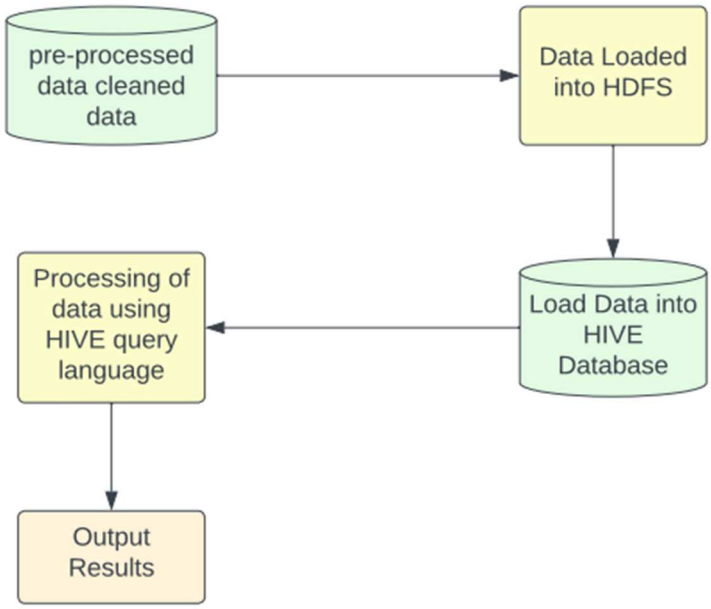


Steps involved in Data Pre-processing:

1. Read the csv format data file from system.
2. The attributes are assigned to each variable
3. The values for each are given for which the range of values specifies the category of the attribute.
4. The missing values are removed.
5. The values which exceed the given range are considered as outliers and are removed from the dataset.
6. we need to handle with null values by eliminating or assigning some value.

4.b Module 2:

After doing the data preprocessing we export the data into Hadoop and store it in HDFS. After that we transform the data stored in HDFS into Hive environment by creating the database table for the Yellow taxi dataset. Then we start the analyzing the data set based on our requirements these is also known as a type of exploration of the dataset in Hive database.



Steps involved in this module:

1. The dataset is exported to HDFS in Hadoop ecosystem.
2. A table is created in hive to store the income census data
3. The dataset is loaded into the table in hive giving the data types of the attributes as well.
4. The required queries are given to analyze and explore the data.
5. Results are displayed.

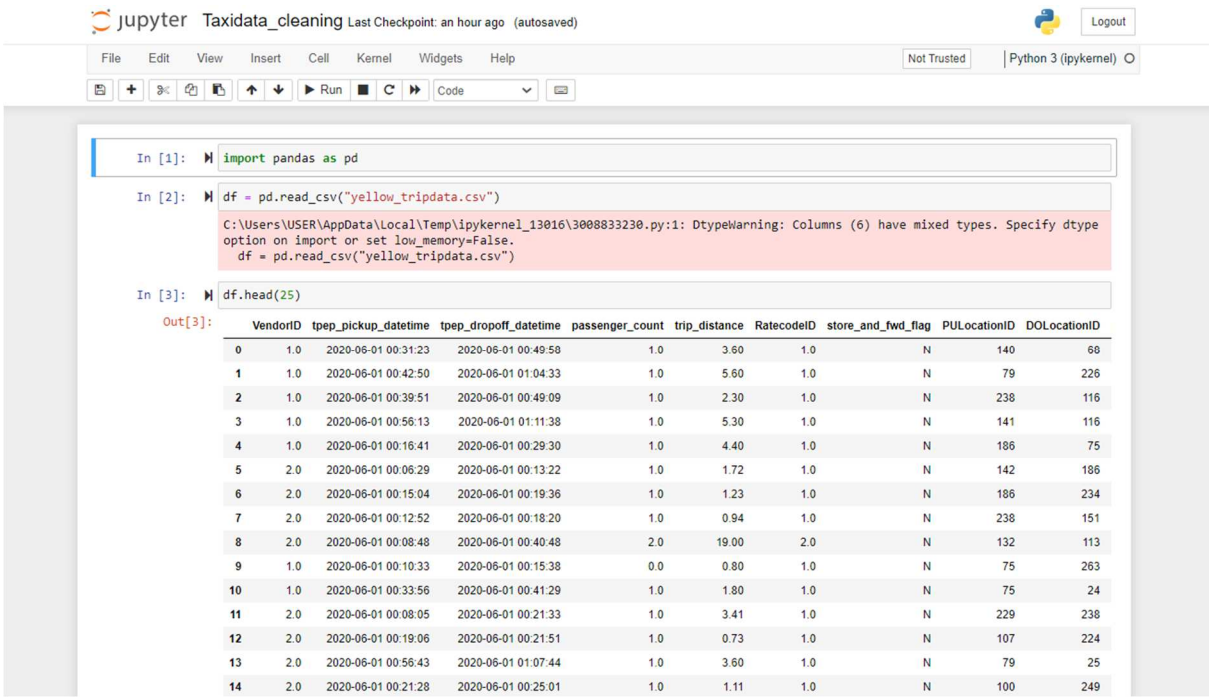
5.Model Building Phase

5.a. Implementation of Module1

Data set contains some insignificant data format here, we observed and clean the data by identifying the dataset by considering the dataset description.

The insignificant data observed in the dataset is modified as given bellow:

1. **VendorID :** A code indicating the TPEP provider that provided the record.
    a. 1= Creative Mobile Technologies, LLC;
    b. 2= VeriFone Inc.
2. **RateCodeID :** The final rate code in effect at the end of the trip.
    a. 1= Standard rate
    b. 2=JFK
    c. 3=Newark
    d. 4=Nassau or Westchester
    e. 5=Negotiated fare
    f. 6=Group ride
3. **Payment_type:** A numeric code signifying how the passenger paid for the trip.
    a. 1= Credit card
    b. 2= Cash
    c. 3= No charge
    d. 4= Dispute
    e. 5= Unknown
    f. 6= Voided trip



	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	RatecodeID	store_and_fwd_flag	PULocationID	DOLocationID
0	1.0	2020-06-01 00:31:23	2020-06-01 00:49:58	1.0	3.60	1.0	N	140	68
1	1.0	2020-06-01 00:42:50	2020-06-01 01:04:33	1.0	5.60	1.0	N	79	226
2	1.0	2020-06-01 00:39:51	2020-06-01 00:49:09	1.0	2.30	1.0	N	238	116
3	1.0	2020-06-01 00:56:13	2020-06-01 01:11:38	1.0	5.30	1.0	N	141	116
4	1.0	2020-06-01 00:16:41	2020-06-01 00:29:30	1.0	4.40	1.0	N	186	75
5	2.0	2020-06-01 00:06:29	2020-06-01 00:13:22	1.0	1.72	1.0	N	142	186
6	2.0	2020-06-01 00:15:04	2020-06-01 00:19:36	1.0	1.23	1.0	N	186	234
7	2.0	2020-06-01 00:12:52	2020-06-01 00:18:20	1.0	0.94	1.0	N	238	151
8	2.0	2020-06-01 00:08:48	2020-06-01 00:40:48	2.0	19.00	2.0	N	132	113
9	1.0	2020-06-01 00:10:33	2020-06-01 00:15:38	0.0	0.80	1.0	N	75	263
10	1.0	2020-06-01 00:33:56	2020-06-01 00:41:29	1.0	1.80	1.0	N	75	24
11	2.0	2020-06-01 00:08:05	2020-06-01 00:21:33	1.0	3.41	1.0	N	229	238
12	2.0	2020-06-01 00:19:06	2020-06-01 00:21:51	1.0	0.73	1.0	N	107	224
13	2.0	2020-06-01 00:56:43	2020-06-01 01:07:44	1.0	3.60	1.0	N	79	25
14	2.0	2020-06-01 00:21:28	2020-06-01 00:25:01	1.0	1.11	1.0	N	100	249

jupyter Taxidata_cleaning Last Checkpoint: an hour ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

In [5]:
```
df['VendorID'] = df['VendorID'].replace([1.0,2.0],['Creative Mobile technologies','VeriFone Inc'])
```

In [8]:
```
df['VendorID'].value_counts()
```

Out[8]:
```
VeriFone Inc                298603
Creative Mobile technologies    200440
Name: VendorID, dtype: int64
```

In [9]:
```
df['payment_type'] = df['payment_type'].replace([1.0,2.0,3.0,4.0,5.0,6.0],['creadit card','cash','no charge','dispute','unknc
```

In [12]:
```
df['RatecodeID'] = df['RatecodeID'].replace([1.0,2.0,3.0,4.0,5.0,6.0],['Standard rate','JFK','Newark','Nassau or Westchester
```

In [13]:
```
df.head(25)
```

Out[13]:

_dropoff_datetime	passenger_count	trip_distance	RatecodeID	store_and_fwd_flag	PULocationID	DOLocationID	payment_type	fare_amount	extra	mta_tax
020-06-01 00:49:58	1.0	3.60	Standard rate	N	140	68	creadit card	15.5	3.0	0.5
020-06-01 01:04:33	1.0	5.60	Standard rate	N	79	226	creadit card	19.5	3.0	0.5
020-06-01 00:49:09	1.0	2.30	Standard rate	N	238	116	cash	10.0	0.5	0.5
020-06-01 01:11:38	1.0	5.30	Standard rate	N	141	116	cash	17.5	3.0	0.5
020-06-01 00:29:30	1.0	4.40	Standard rate	N	186	75	creadit card	14.5	3.0	0.5
020-06-01 00:13:22	1.0	1.72	Standard rate	N	142	186	cash	8.0	0.5	0.5
020-06-01 00:19:36	1.0	1.23	Standard rate	N	186	234	creadit card	6.0	0.5	0.5
020-06-01 00:18:20	1.0	0.94	Standard rate	N	238	151	cash	6.0	0.5	0.5

jupyter Taxidata_cleaning Last Checkpoint: an hour ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

In [15]:
```
df.dtypes
```

Out[15]:
```
VendorID                 object
tpep_pickup_datetime     object
tpep_dropoff_datetime    object
passenger_count         float64
trip_distance           float64
RatecodeID               object
store_and_fwd_flag       object
PULocationID              int64
DOLocationID              int64
payment_type             object
fare_amount             float64
extra                   float64
mta_tax                 float64
tip_amount              float64
tolls_amount            float64
improvement_surcharge   float64
total_amount            float64
congestion_surcharge    float64
dtype: object
```

In [16]:
```
df['PULocationID'].value_counts()
```

Out[16]:
```
236    23098
237    22462
186    19862
140    19603
141    19006
       ...
109        3
2          3
27         2
30         2
199        1
Name: PULocationID, Length: 257, dtype: int64
```

In [17]:
```
df.to_csv(r'C:\Users\USER\Documents\Bigdata\taxidata.csv')
```

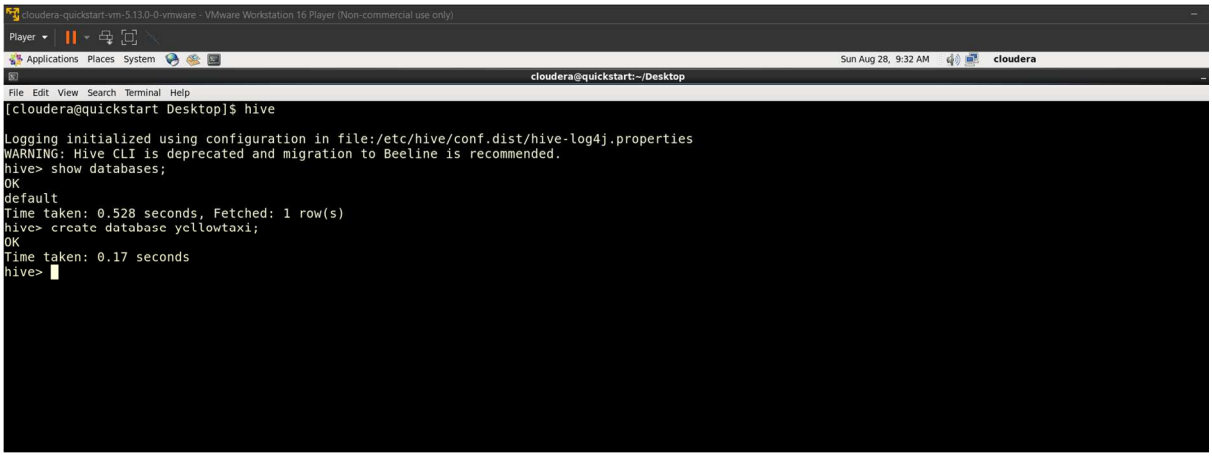
In [ ]:
```

```

5.b Implementation of Module 2

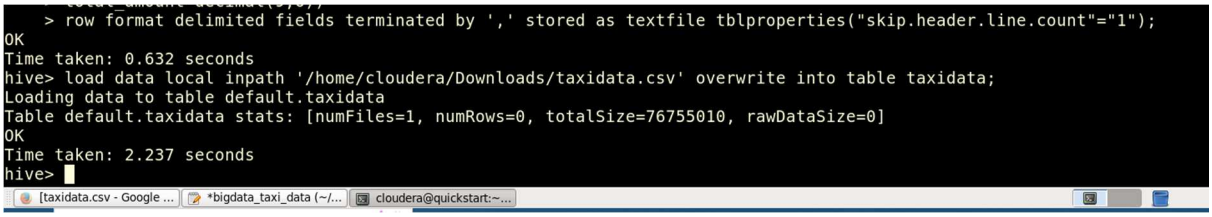
Here, we implemented the task in hive we take the preprocessed data and import the data in hive by creating the database and required table.

1. **Database creation with name yellowtaxi:**



2. **Create table taxidata with required attributes and datatypes:**

3. **Load the data to the table:**



6.Communicating Results Phase

6.a Output of each Module

4. Some set of questions are defined to analyze the data:

1. What is the total Number of trips (equal to number of rows)? Query: Select **count(*) from taxidata;**

```
hive> select count(*) from taxidata;
Query ID = cloudera_20220828094040_c14b12be-03e3-4074-8eb3-53678719cec7
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1661703302021_0001, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1661703302021_0001/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1661703302021_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2022-08-28 09:40:46,798 Stage-1 map = 0%,  reduce = 0%
2022-08-28 09:40:55,735 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.12 sec
2022-08-28 09:41:07,667 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 3.52 sec
MapReduce Total cumulative CPU time: 3 seconds 520 msec
Ended Job = job_1661703302021_0001
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 3.52 sec   HDFS Read: 76765302 HDFS Write: 7 SUCCESS
Total MapReduce CPU Time Spent: 3 seconds 520 msec
OK
549760
Time taken: 39.94 seconds, Fetched: 1 row(s)
hive>
```

2. What is the total revenue generated by all the trips ? Fare is stored in the column total_amount.
   Select sum(total_amount) as total_revenue from taxidata;

```
hive> select sum(total_amount) as total_revenue from taxidata;
Query ID = cloudera_20220828094141_d44a72ed-1b54-42ea-8114-701c65b06422
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1661703302021_0002, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1661703302021_0002/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job 1661703302021 0002
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2022-08-28 09:42:02,128 Stage-1 map = 0%,  reduce = 0%
2022-08-28 09:42:13,163 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 3.5 sec
2022-08-28 09:42:23,921 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 4.88 sec
MapReduce Total cumulative CPU time: 4 seconds 880 msec
Ended Job = job_1661703302021_0002
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 4.88 sec   HDFS Read: 76765541 HDFS Write: 12 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 880 msec
OK
10317255.83
Time taken: 31.331 seconds, Fetched: 1 row(s)
hive>
```

3. What fraction of the total is paid for tolls? Toll is stored in tolls_amount.
   Select sum(tolls_amount)/sum(total_amount) as toll_frc from taxidata;

```
hive> select sum(tolls_amount)/sum(total_amount) as total_frc from taxidata;
Query ID = cloudera_20220828094343_2bea0dbb-3624-40aa-9289-ec816ffbde7f
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1661703302021_0003, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1661703302021_0003/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1661703302021_0003
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2022-08-28 09:43:53,089 Stage-1 map = 0%,  reduce = 0%
2022-08-28 09:44:04,234 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 3.87 sec
2022-08-28 09:44:14,960 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 5.74 sec
MapReduce Total cumulative CPU time: 5 seconds 740 msec
Ended Job = job_1661703302021_0003
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 5.74 sec   HDFS Read: 76766771 HDFS Write: 25 SUCCESS
Total MapReduce CPU Time Spent: 5 seconds 740 msec
OK
0.0158247311775732132834
Time taken: 31.402 seconds, Fetched: 1 row(s)
hive>
```

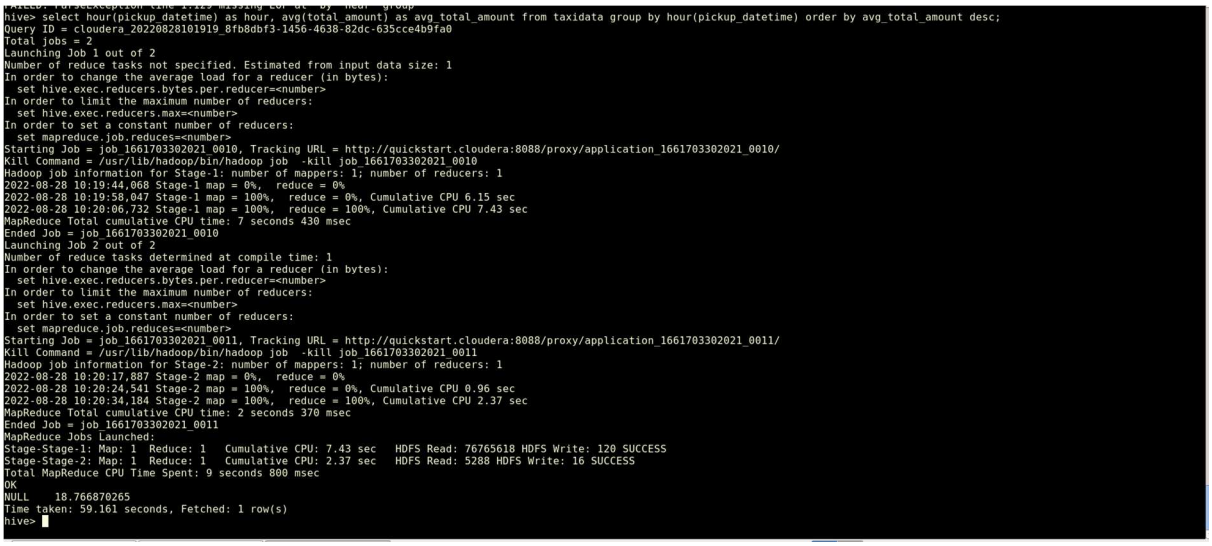
4. What fraction of it is driver tips? Tip is stored in tip_amount.
   Select sum(tip_amount)/sum(total_amount) as tip_frc from taxidata;

```
hive> select sum(tip_amount)/sum(total_amount) as tip_fraction from taxidata;
Query ID = cloudera_20220828221313_d5a8c5ba-9cb3-4811-8ac9-08c2497fdc0b
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1661748738043_0001, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1661748738043_0001/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1661748738043_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2022-08-28 22:13:38,803 Stage-1 map = 0%,  reduce = 0%
2022-08-28 22:13:49,931 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 3.74 sec
2022-08-28 22:13:59,691 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 5.42 sec
MapReduce Total cumulative CPU time: 5 seconds 420 msec
Ended Job = job_1661748738043_0001
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 5.42 sec   HDFS Read: 76766664 HDFS Write: 25 SUCCESS
Total MapReduce CPU Time Spent: 5 seconds 420 msec
OK
0.019559291087192125970 6
Time taken: 35.996 seconds, Fetched: 1 row(s)
hive>
```

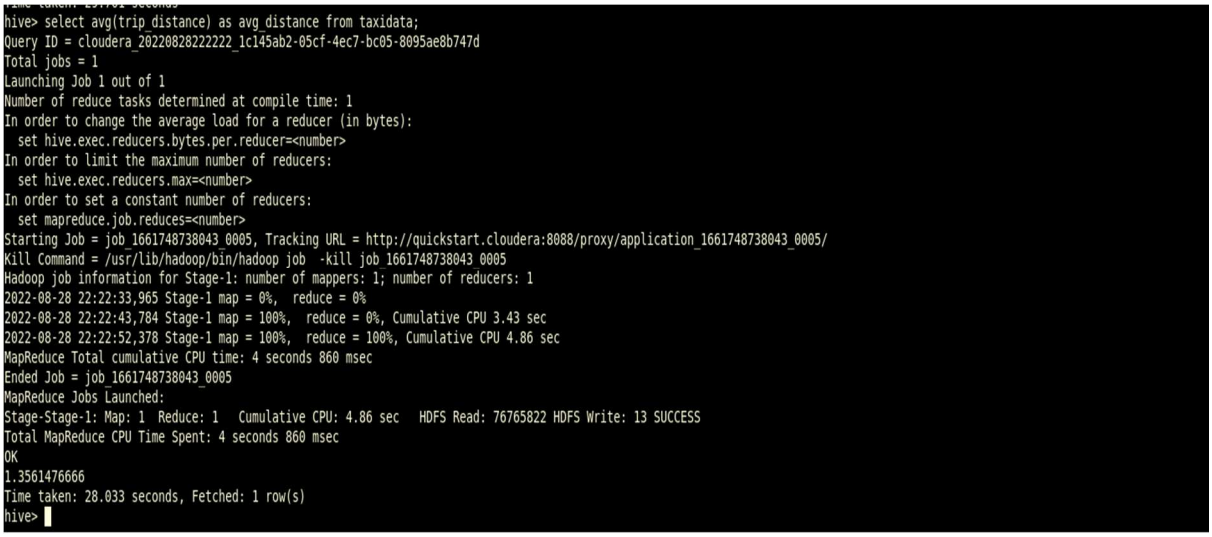
5. What is the average trip amount?
   Select avg(total_amount) as avg_tripamount from taxidata;

```
hive> select avg(total_amount) as avg_trip_amount from taxidata;
Query ID = cloudera_20220828221616_b4af9a77-9836-4edc-8569-f849f9af3952
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1661748738043_0002, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1661748738043_0002/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1661748738043_0002
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2022-08-28 22:16:09,362 Stage-1 map = 0%,  reduce = 0%
2022-08-28 22:16:19,210 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 3.36 sec
2022-08-28 22:16:28,970 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 4.72 sec
MapReduce Total cumulative CPU time: 4 seconds 720 msec
Ended Job = job_1661748738043_0002
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 4.72 sec   HDFS Read: 76765821 HDFS Write: 13 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 720 msec
OK
18.766870265
Time taken: 29.364 seconds, Fetched: 1 row(s)
hive>
```

6. On an average which hour of the day generates the highest revenue?

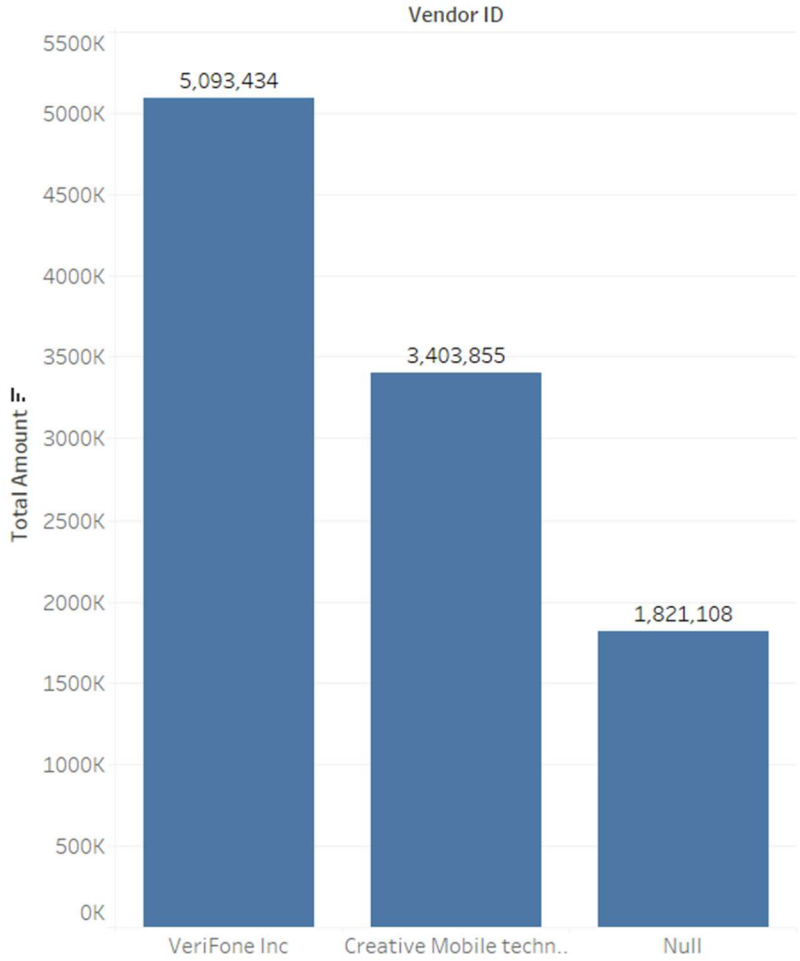


7. What is the average distance of the trips? Distance is stored in the column trip_distance.

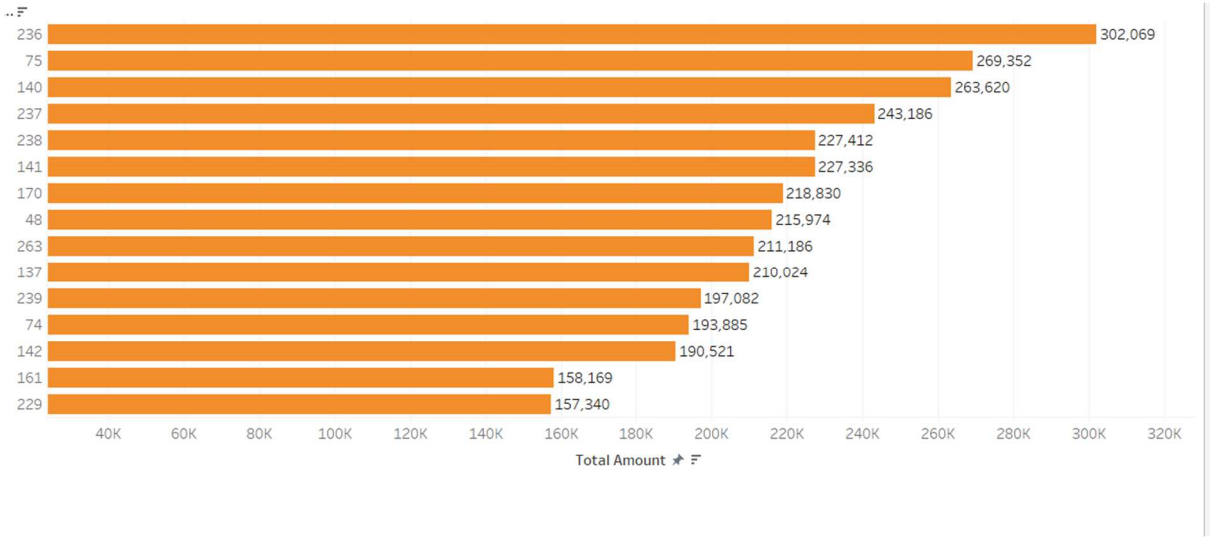


6.b Visualization / Comparison charts

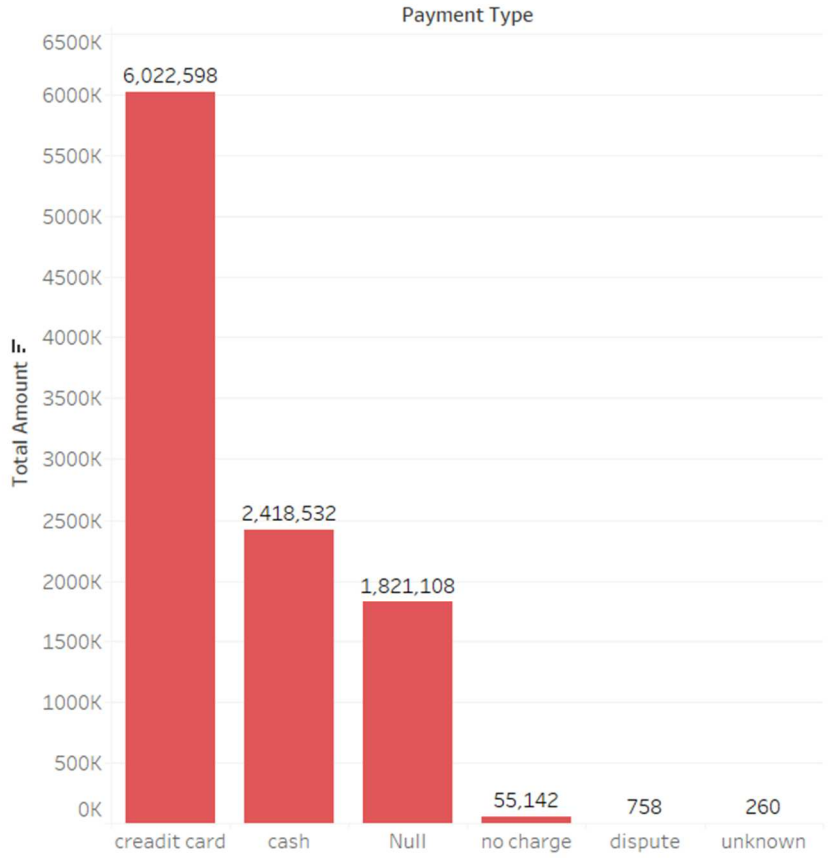
1. total revenue based on vendor



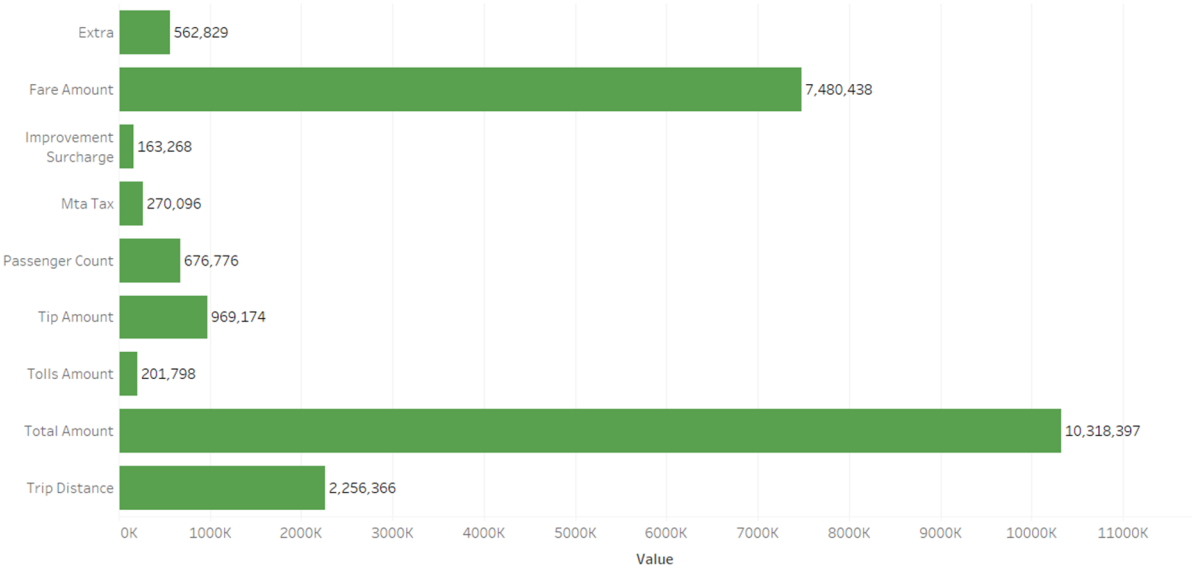
2. Top 15 locations total revenue earned based on location by considering the location id.



4. Total revenue based on the payment mode used by the customer



5. Attributes with their total sum.



7.Conclusion:

We done a descriptive data analytics for that we used the Hive which is a relational database available on top of hadoop which help as to store and handle the data to analyze yellow taxi data. Before that we done data preprocessing by using python to make data consistent. By using hive we got the results which helpful to improve the business.

Later stage we done data visualization using tableau to understand the data pattern visually. Data visualization is one of the strongest technology used in present trend to analyze and understand the data visually. By using that we derive the top 15 locations where we get more revenue so hence this help to increase the business by enhancing the taxi services in those locations as like that we also visualize the relation between revenue and vendor, revenue generated from different payment modes.

8.References:

1. Marie Stephen Leo [1] predicting the taxi fare using Regression models such as Linear Regression, Lasso Regression Hyper tuning. IEEE.
2. Zhizhen Liu, Hong Chen, Yan Li, and Qi Zhang [2]. Taxi demand prediction based on combination Model in hotspots. IEEE.
3. Jun Xu, Rouhollah Rahmatizadeh, Ladislau Boloni and Damla Turgut [3] prediction of taxi demand using recurrent neural networks. IEEE.
4. Vivek Sachapara, Hrishikesh Shinde, Abhishek Puri, Shraddha Aggrawal and Prof. Sachin Wandre [4]. Big Data Analytics on Cab Dataset using hadoop. IEEE.
5. Bayan Alghuraybi, Krishna Marvaniya, Guojun xia and Jongwook Woo [5] Analyze NYC taxi data using Big data tools. IEEE.
6. Abhishek Singh, Ashmit Narayan Rai, Ayushi Saxena, Diti Gupta, Prabal Bhatnagar [6] Youtube Data Analysis using the Hadoop.IEEE.
7. Rotsnarani Sethy, and Mrutyunjaya Panda [7] Big data analytics using Hadoop and MapReduce algorithms. IEEE.
8. Alvin Jun Yong koh, Xuan Khoa nguyen, and C. Jason woodard [8] taxi data analysis using Hadoop and Cassandra. IEEE.
9. Umang Patel , and Anil Chandan [9]. NYC taxi trip and fare Data Analytics using BigData. IEEE.
10. Bijesh Dhyani and Anurag Barthwal [10] Survey report on Big Data Analytics. IEEE.

9.Apendix

Python data preprocessing:

```
import pandas as pd

df = pd.read_csv("yellow_tripdata.csv")

df.head(25)

df['VendorID'] = df['VendorID'].replace([1.0,2.0],['Creative Mobile technologies','VeriFone Inc'])

df['VendorID'].value_counts()

df['payment_type'] = df['payment_type'].replace([1.0,2.0,3.0,4.0,5.0,6.0],['creadit card','cash','no charge','dispute','unknown','voided trip'])

df['RatecodeID'] = df['RatecodeID'].replace([1.0,2.0,3.0,4.0,5.0,6.0],['Standard rate','JFK','Newark','Nassau or Westchester','Negotiated fare','Group ride'])

df.head(25)

df.describe()

df.dtypes

df['PULocationID'].value_counts()

df.to_csv(r'C:\Users\USER\Documents\Bigdata\taxidata.csv')
```

Hive database:

1.Create a database yellowtaxi.

```
Create database yellotaxi
```

2.create a table taxidata.

```
create table if not exists taxidata(

vendor_id string,

pickup_datetime string,

dropoff_datetime string,

passenger_count int,

trip_distance decimal(9,6),

pickup_longitude decimal(9,6),

pickup_latitude decimal(9,6),

rate_code int,

store_and_fwd_flag string,
```

dropoff_longitude decimal(9,6),

droff_latitude decimal(9,6),

payment_type string,

fare_amount decimal(9,6),

extra decimal(9,6),

mta_tax decimal(9,6),

tip_amount decimal(9,6),

tolls_amount decimal(9,6),

total_amount decimal(9,6))

row format delimited fields terminated by ',' stored as textfile tblproperties("skip.header.line.count"="1");

3.Load the data to hive table.

load data local inpath '/home/cloudera/Downloads/taxidata.csv' overwrite into table taxidata;

4.Questions derived to analyze the data.

1. what is the total no of trips ?

select count(*) from taxidata;

2. what is the total revenue by all trips?

select sum(total_amount) as total_revenue from taxidata;

3. what fraction of the total is paid for tolls?

select sum(tolls_amount)/sum(total_amount) as toll_frc from taxidata;

4. what fraction of it is triver tips?

select sum(tip_amount)/sum(total_amount) as tip_frc from taxidata;

5. on an average which hour of the day generate the highest revenue?

select hour(pickup_datetime) as hour, avg(total_amount) as avg_total from taxidata where pickup_datetime is not null group by hour(pickup_datetime);

6. what is the avg trip amount?

select avg(total_amount) from taxidata;

7. what is the avg distance of the trip?

slect avg(trip_distance) as avg_distance from taxidata;