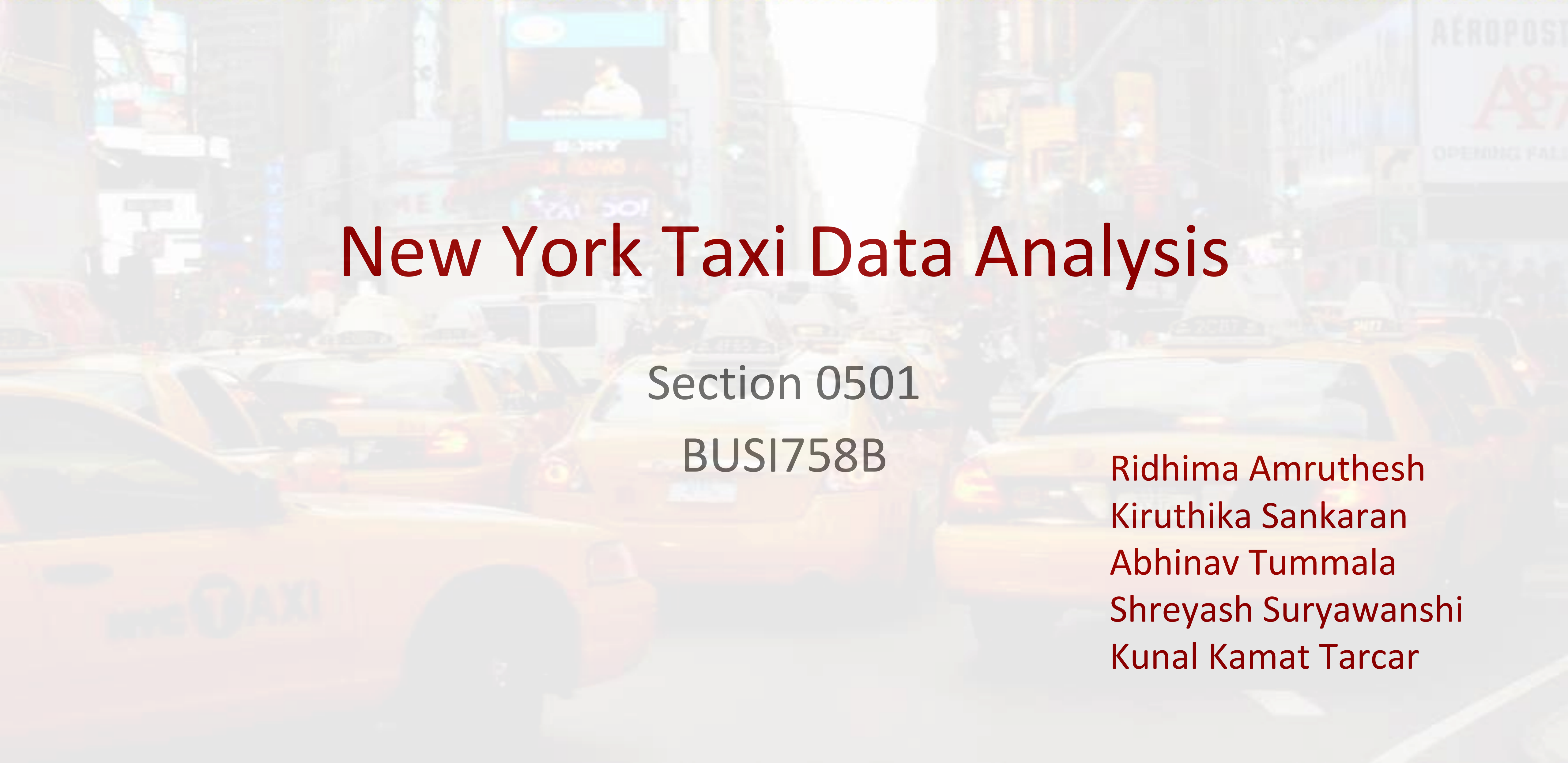
ROBERT H. SMITH
SCHOOL OF BUSINESS



New York Taxi Data Analysis

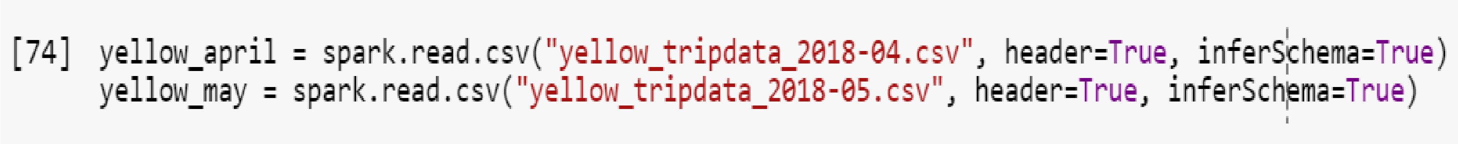
Section 0501

BUSI758B

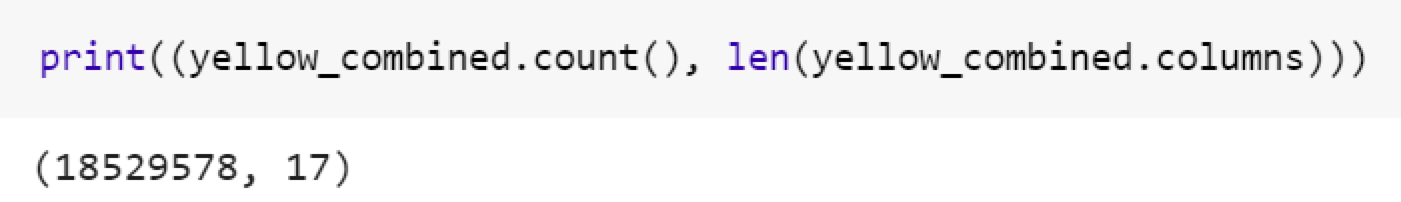
Ridhima Amruthesh
Kiruthika Sankaran
Abhinav Tummala
Shreyash Suryawanshi
Kunal Kamat Tarcar

Dataset Description

- Agency : Taxi and Limousine Commission (TLC)
- Two types of taxis:
  - Yellow
  - Green

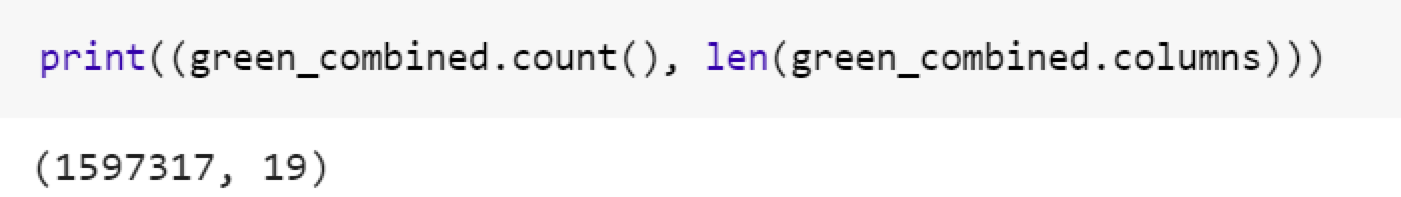


```
[74]  yellow_april = spark.read.csv("yellow_tripdata_2018-04.csv", header=True, inferSchema=True)
      yellow_may = spark.read.csv("yellow_tripdata_2018-05.csv", header=True, inferSchema=True)
```



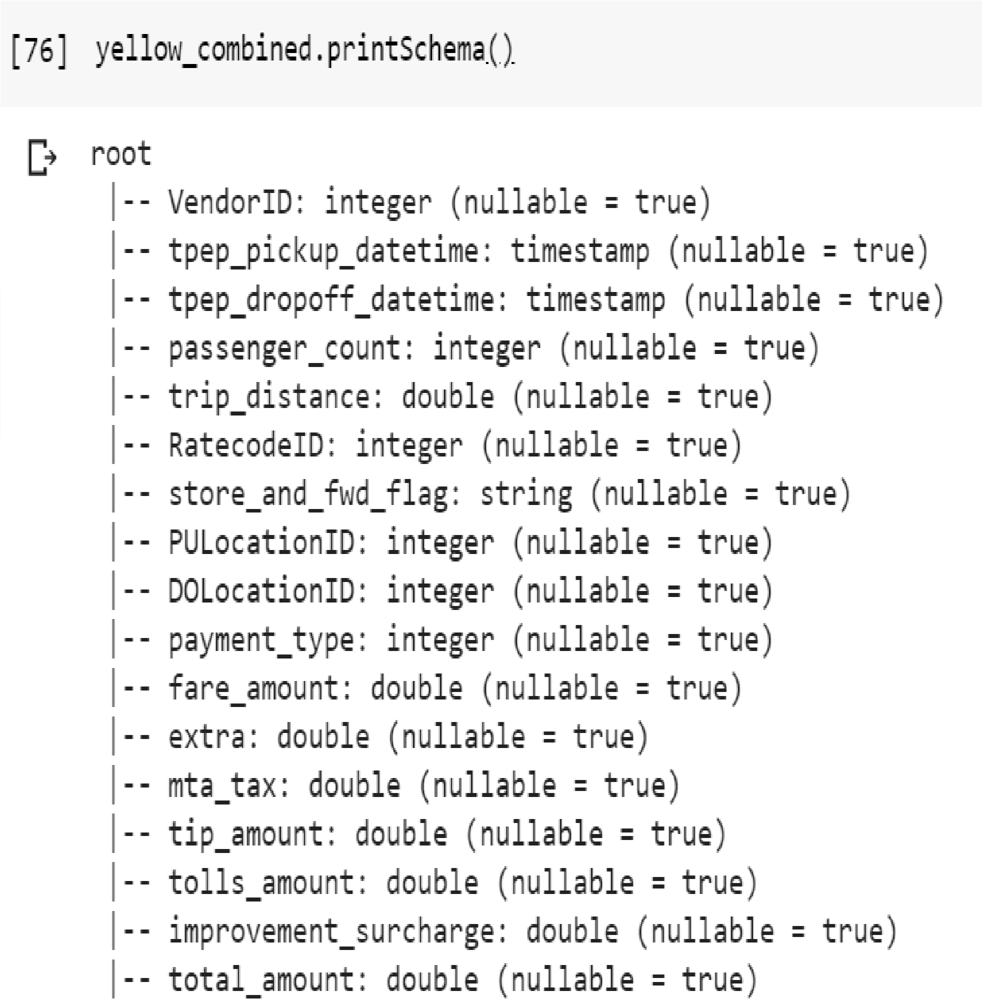
```
print((yellow_combined.count(), len(yellow_combined.columns)))
```

```
(18529578, 17)
```



```
print((green_combined.count(), len(green_combined.columns)))
```

```
(1597317, 19)
```



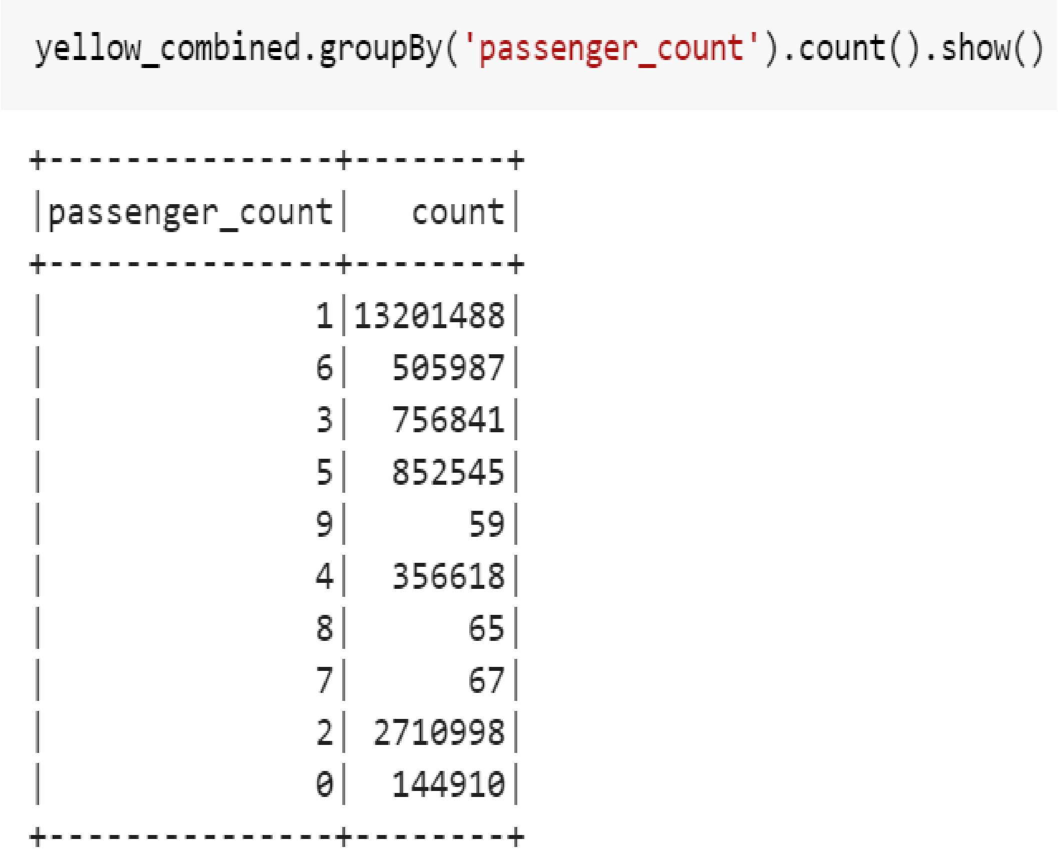
```
[76]  yellow_combined.printSchema()
```

```
root
 |-- VendorID: integer (nullable = true)
 |-- tpep_pickup_datetime: timestamp (nullable = true)
 |-- tpep_dropoff_datetime: timestamp (nullable = true)
 |-- passenger_count: integer (nullable = true)
 |-- trip_distance: double (nullable = true)
 |-- RatecodeID: integer (nullable = true)
 |-- store_and_fwd_flag: string (nullable = true)
 |-- PULocationID: integer (nullable = true)
 |-- DOLocationID: integer (nullable = true)
 |-- payment_type: integer (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)
```

Descriptive Analytics

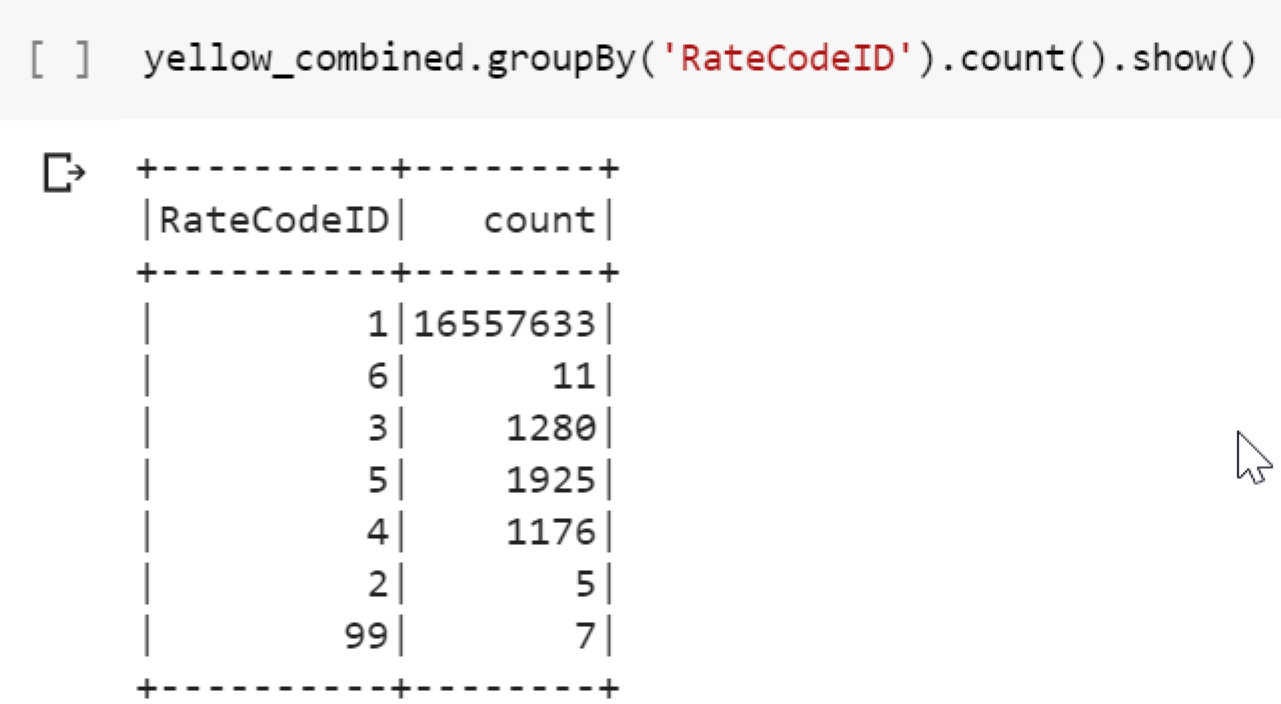
Elimination:

- 0.8% data has 0 passengers.
- No Rate Code ID named 99 in Data Dictionary.



```
yellow_combined.groupBy('passenger_count').count().show()
```

```
+---------------+--------+
|passenger_count|   count|
+---------------+--------+
|              1|13201488|
|              6|  505987|
|              3|  756841|
|              5|  852545|
|              9|      59|
|              4|  356618|
|              8|      65|
|              7|      67|
|              2| 2710998|
|              0|  144910|
+---------------+--------+
```



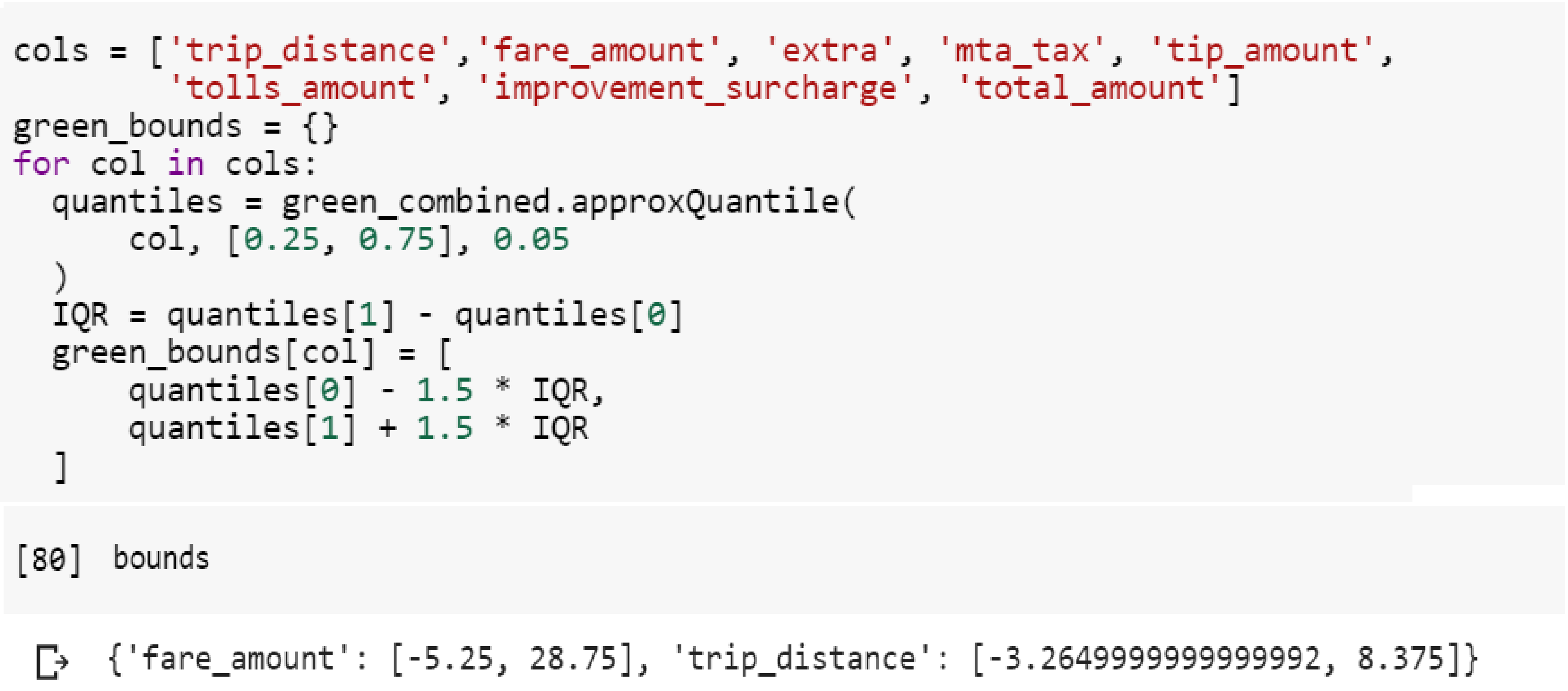
```
[ ]  yellow_combined.groupBy('RateCodeID').count().show()
```

```
+----------+--------+
|RateCodeID|   count|
+----------+--------+
|         1|16557633|
|         6|      11|
|         3|    1280|
|         5|    1925|
|         4|    1176|
|         2|       5|
|        99|       7|
+----------+--------+
```

Descriptive Analytics

Outlier Detection:

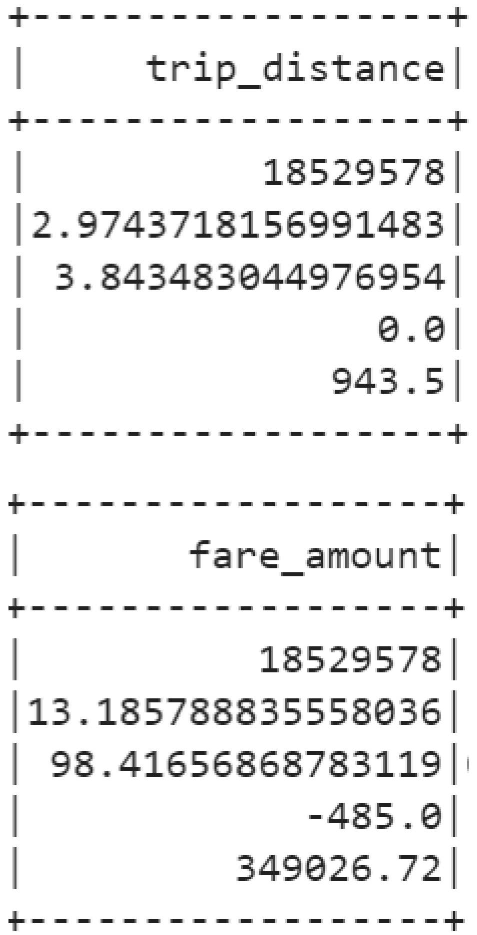
- Minimum fare amount is 2.5.
- Deleting outliers beyond 1.5 Interquartile range.



```
cols = ['trip_distance','fare_amount', 'extra', 'mta_tax', 'tip_amount',
        'tolls_amount', 'improvement_surcharge', 'total_amount']
green_bounds = {}
for col in cols:
  quantiles = green_combined.approxQuantile(
      col, [0.25, 0.75], 0.05
  )
  IQR = quantiles[1] - quantiles[0]
  green_bounds[col] = [
      quantiles[0] - 1.5 * IQR,
      quantiles[1] + 1.5 * IQR
  ]
```

```
[80]  bounds
```

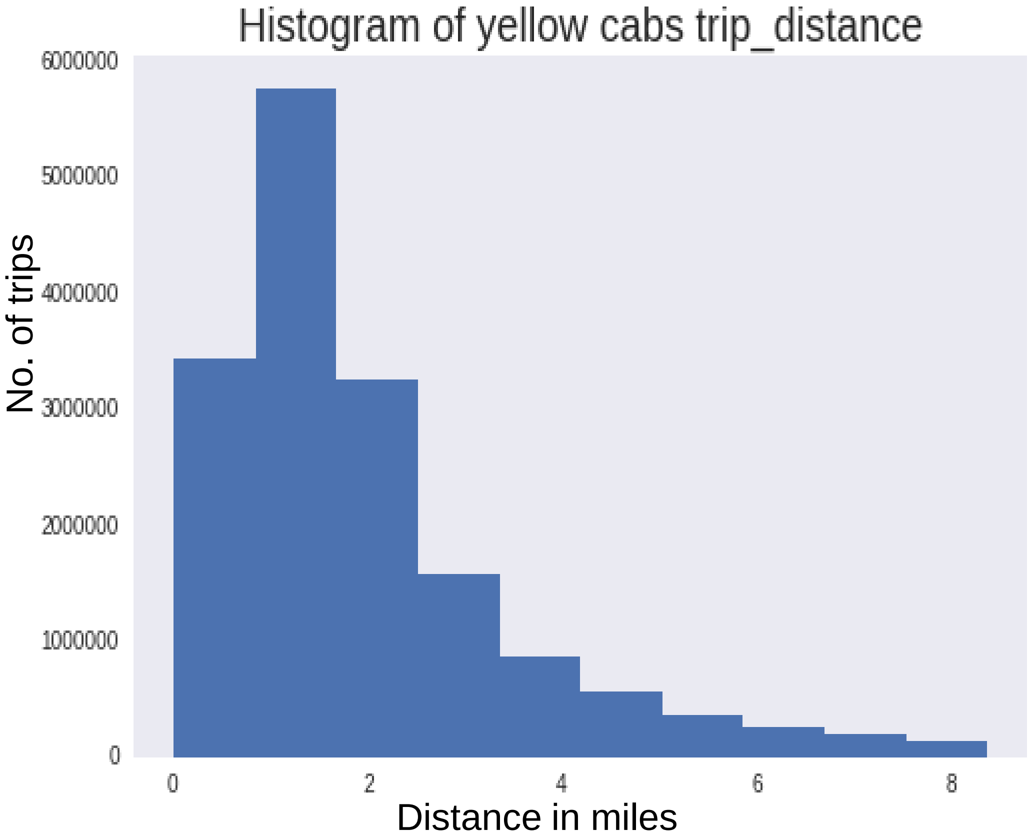
```
{'fare_amount': [-5.25, 28.75], 'trip_distance': [-3.2649999999999992, 8.375]}
```

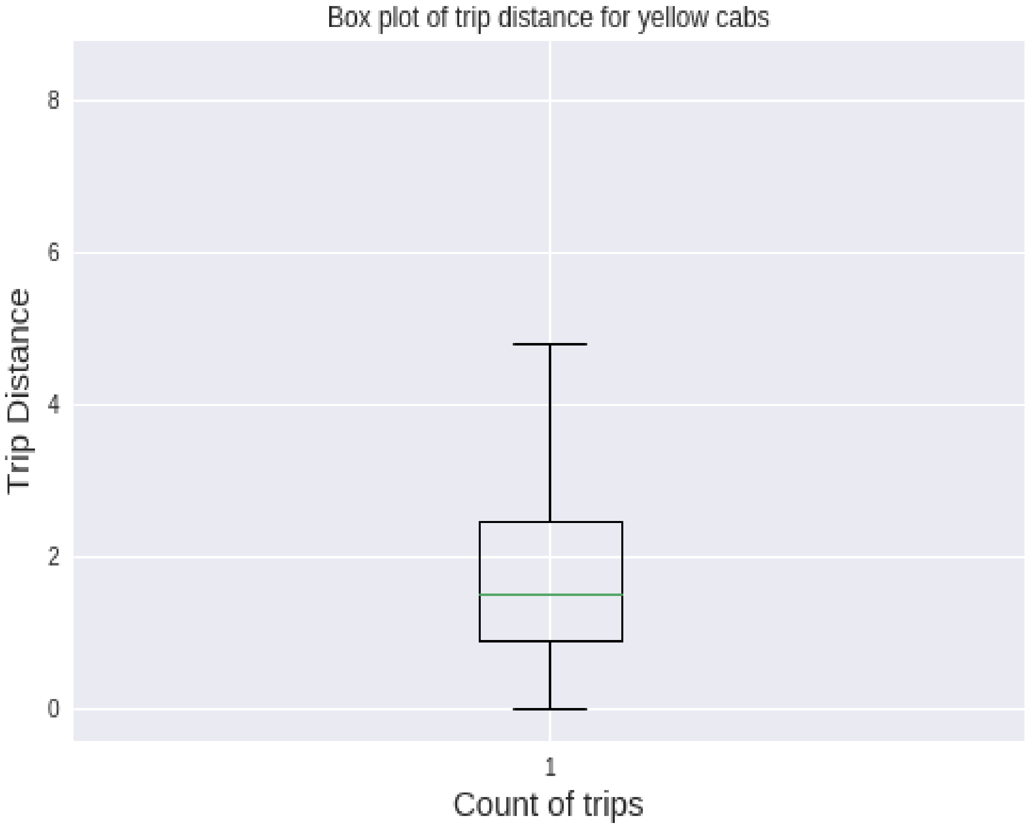


```
+------------------+
|     trip_distance|
+------------------+
|          18529578|
|2.974371815691483|
| 3.843483044976954|
|               0.0|
|             943.5|
+------------------+

+------------------+
|       fare_amount|
+------------------+
|          18529578|
|13.185788355580 36|
| 98.41656868783119|
|            -485.0|
|         349026.72|
+------------------+
```

Descriptive Analytics

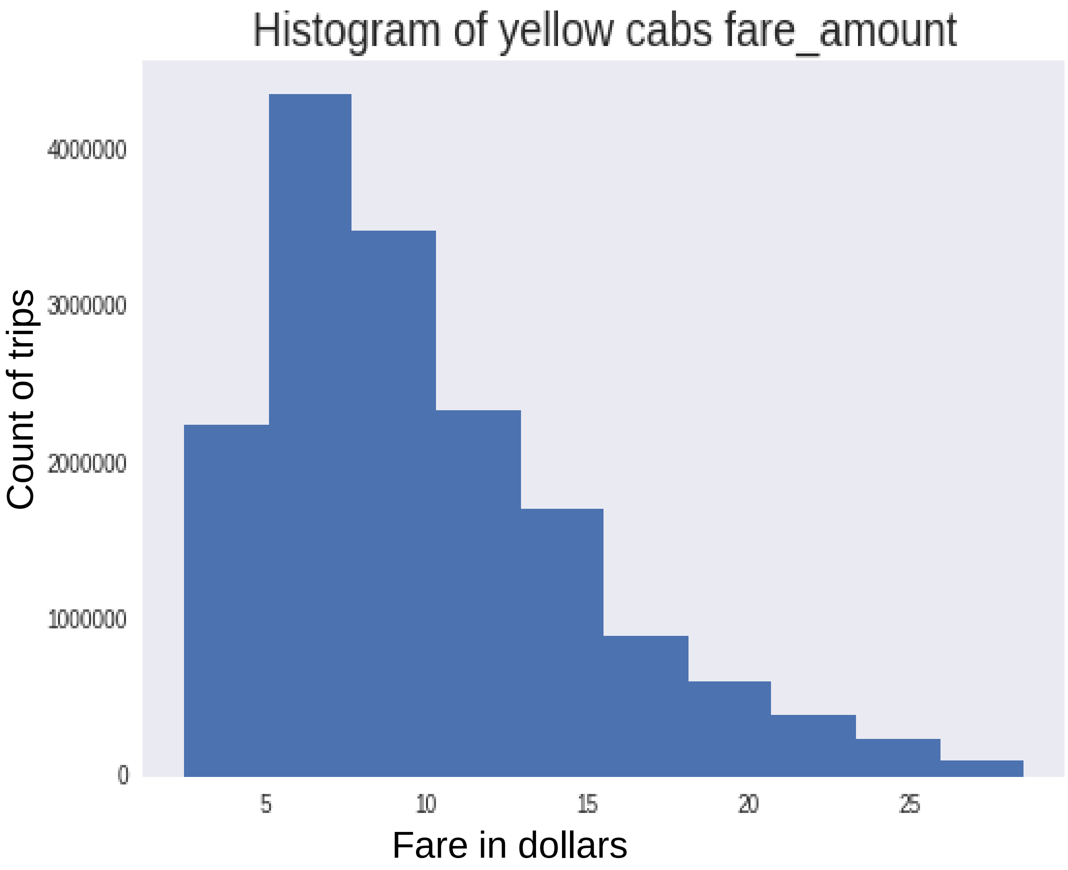
Trip Distance Analysis:

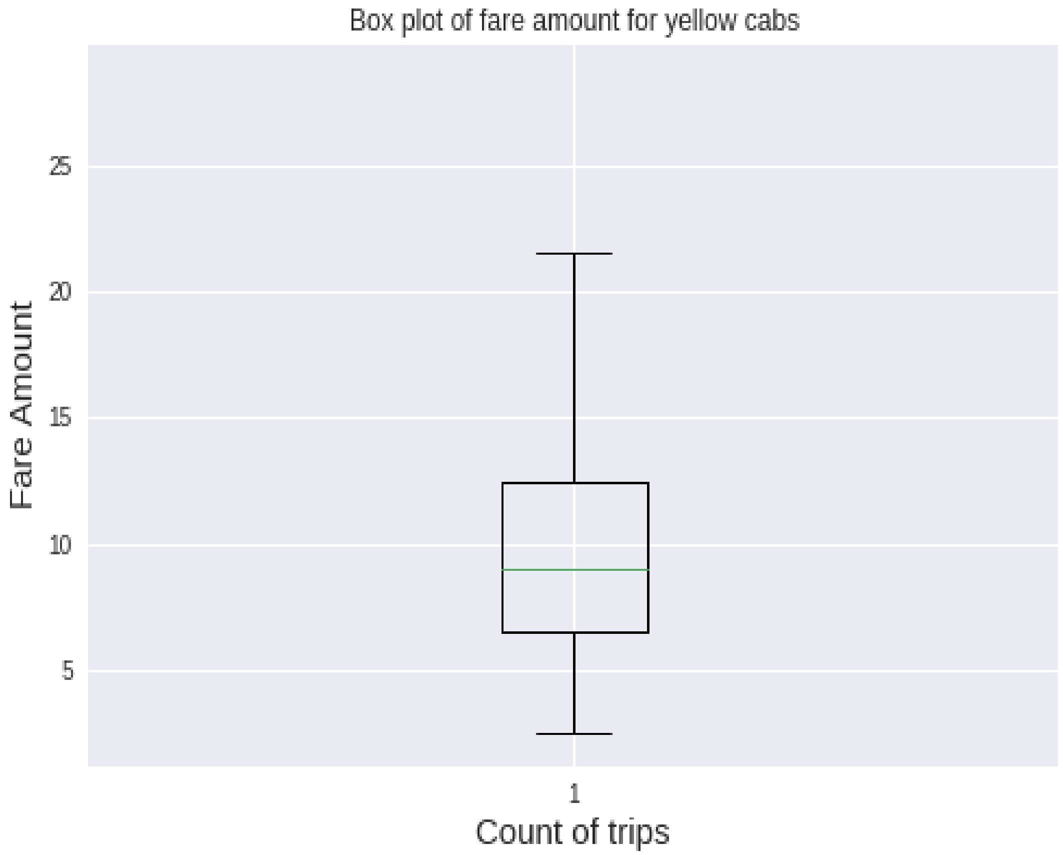




Descriptive Analytics

Fare Amount Analysis:

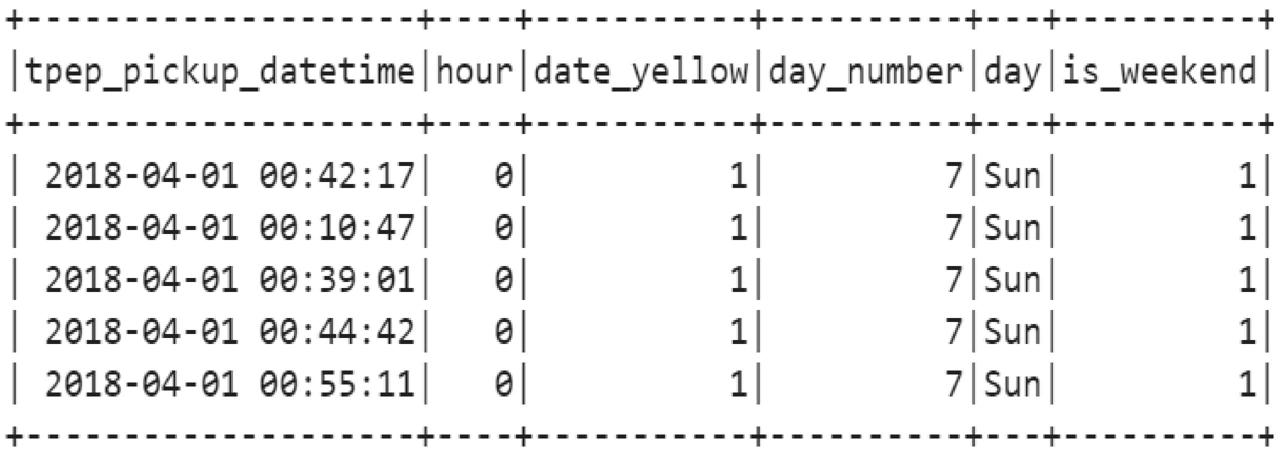




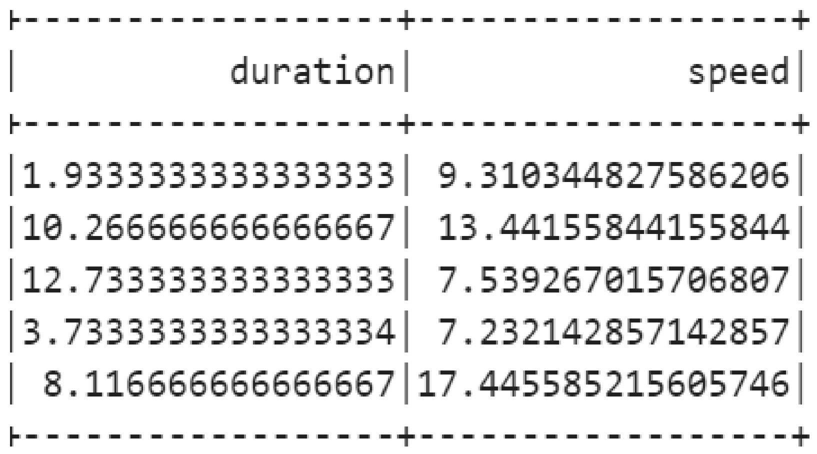
Feature Engineering

New Features:

- Time related - hour, date, day, weekend
- Duration of the ride
- Average Taxi speed



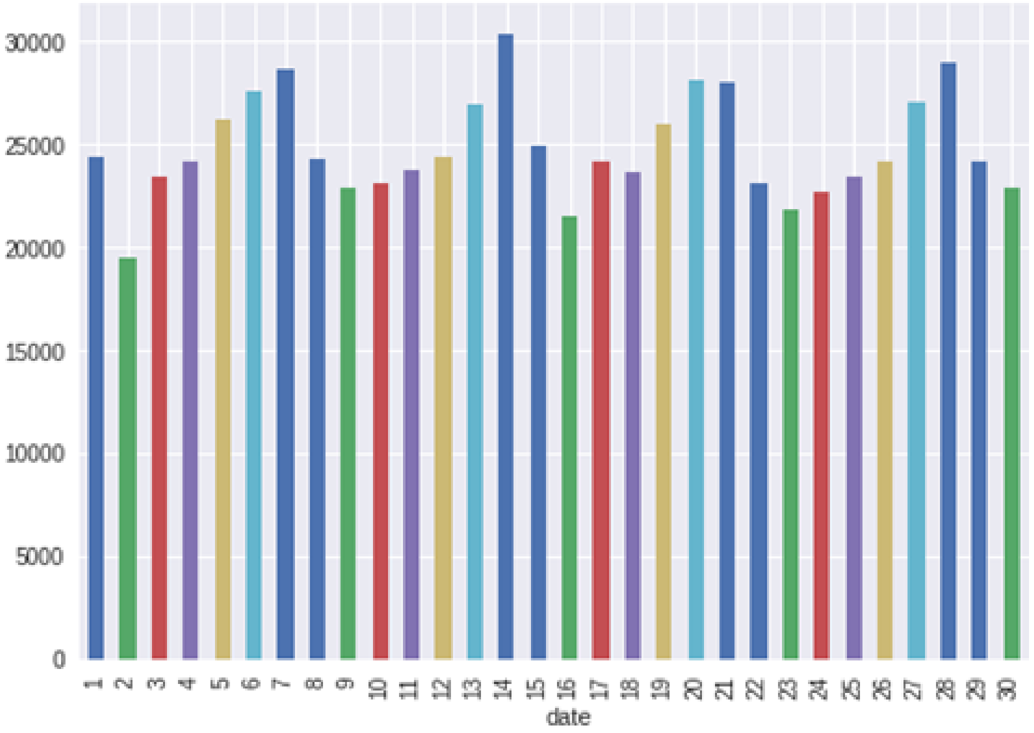
```
+--------------------+----+-----------+----------+---+----------+
|tpep_pickup_datetime|hour|date_yellow|day_number|day|is_weekend|
+--------------------+----+-----------+----------+---+----------+
| 2018-04-01 00:42:17|   0|          1|         7|Sun|         1|
| 2018-04-01 00:10:47|   0|          1|         7|Sun|         1|
| 2018-04-01 00:39:01|   0|          1|         7|Sun|         1|
| 2018-04-01 00:44:42|   0|          1|         7|Sun|         1|
| 2018-04-01 00:55:11|   0|          1|         7|Sun|         1|
+--------------------+----+-----------+----------+---+----------+
```

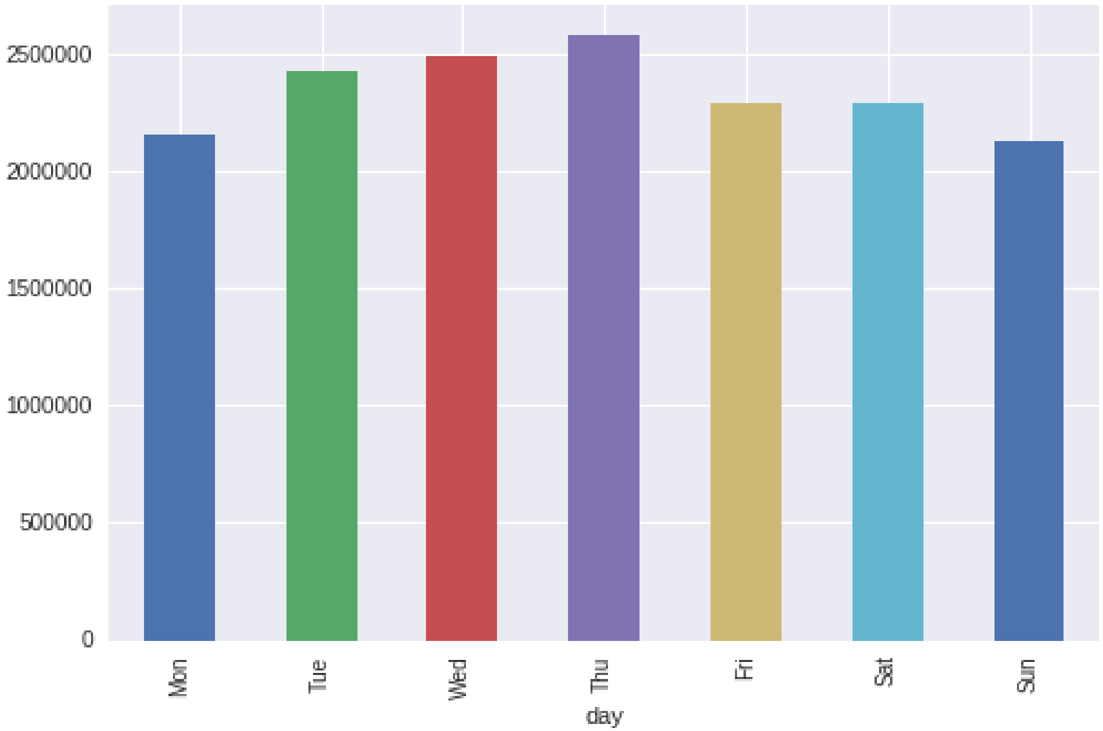


```
+------------------+------------------+
|          duration|             speed|
+------------------+------------------+
|1.9333333333333333| 9.310344827586206|
|10.266666666666667| 13.44155844155844|
|12.733333333333333| 7.539267015706807|
|3.7333333333333334| 7.232142857142857|
| 8.116666666666667|17.445585215605746|
+------------------+------------------+
```

Inferential Analytics

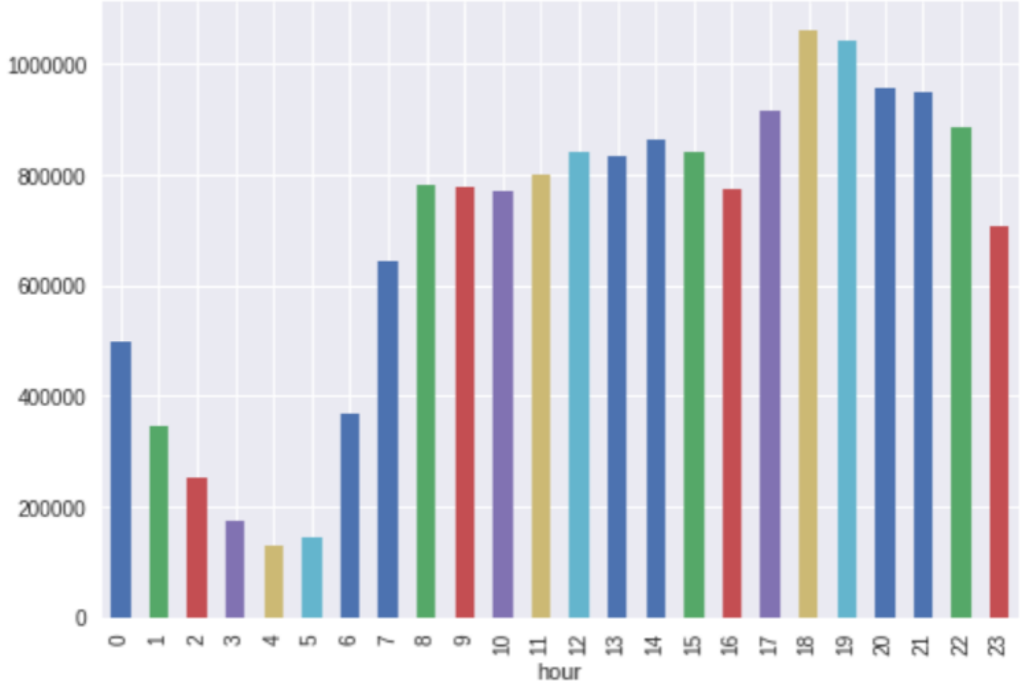
- Weekly trend observed in the daily number of bookings in April.
- Maximum bookings on Thursdays.

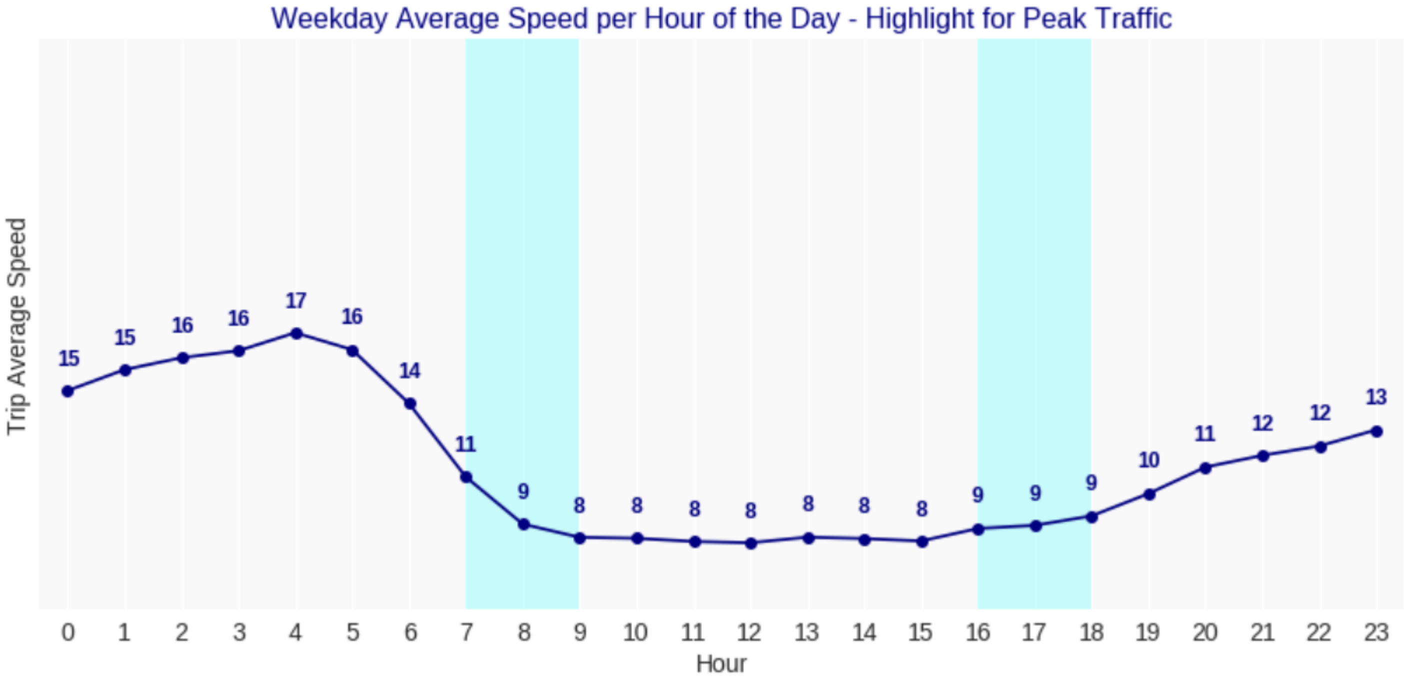




Inferential Analytics

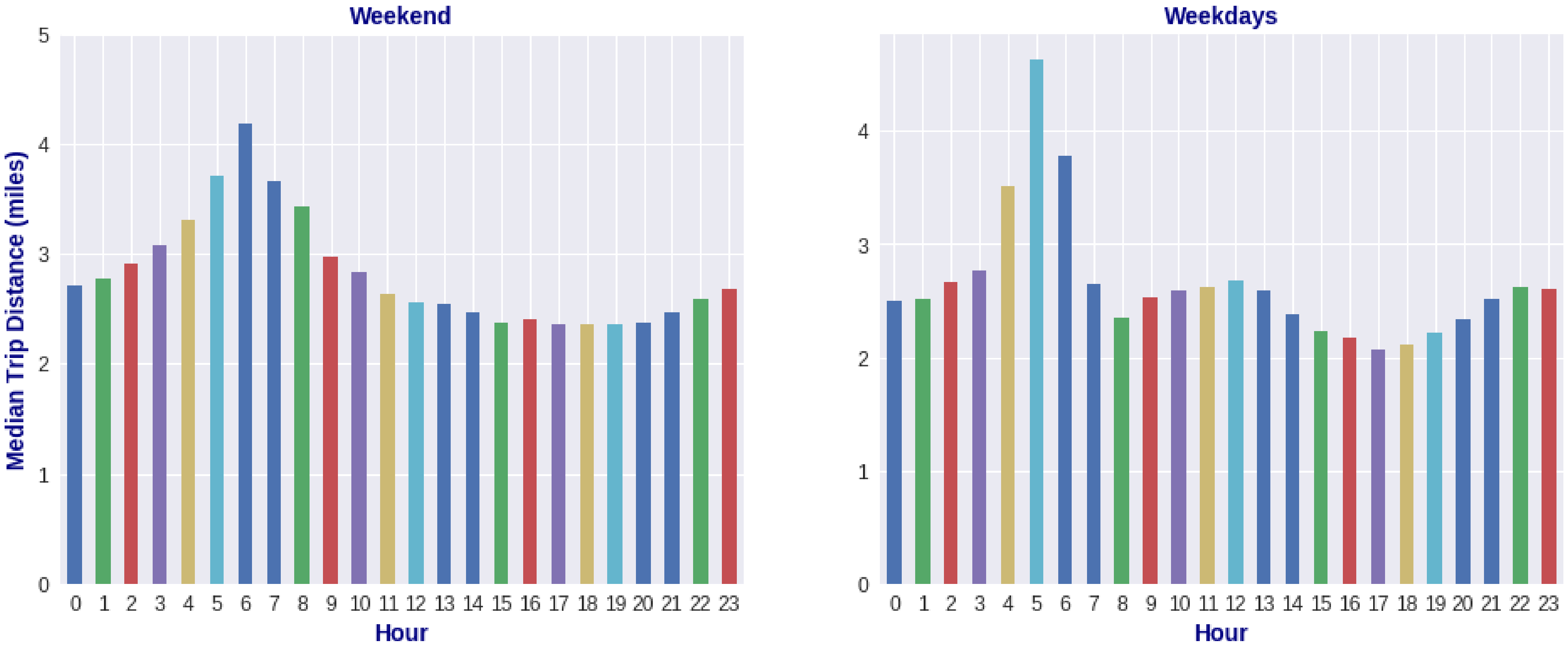
Demand for yellow cabs per hour of day
- Weekdays Peak Hours from 5pm-9pm





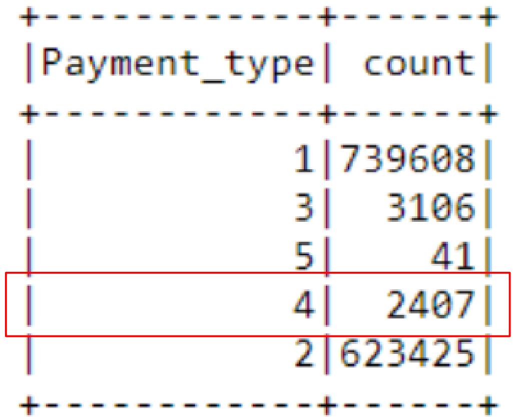
Inferential Analytics

Median Trip Distance is more during off-peak hours.



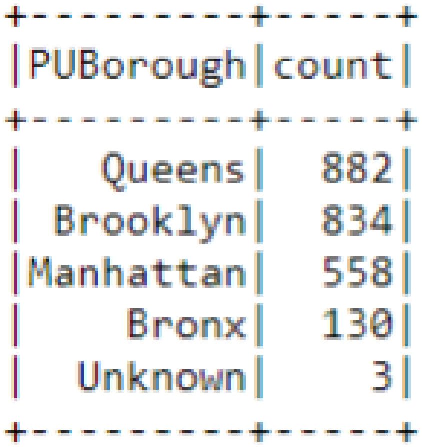
Inferential Analytics

- Total number of trips in Manhattan is significantly higher than Queens or Brooklyn.
- However, number of disputes is more in Queens and Brooklyn.
- 36.6 % of disputes happen in Queens.

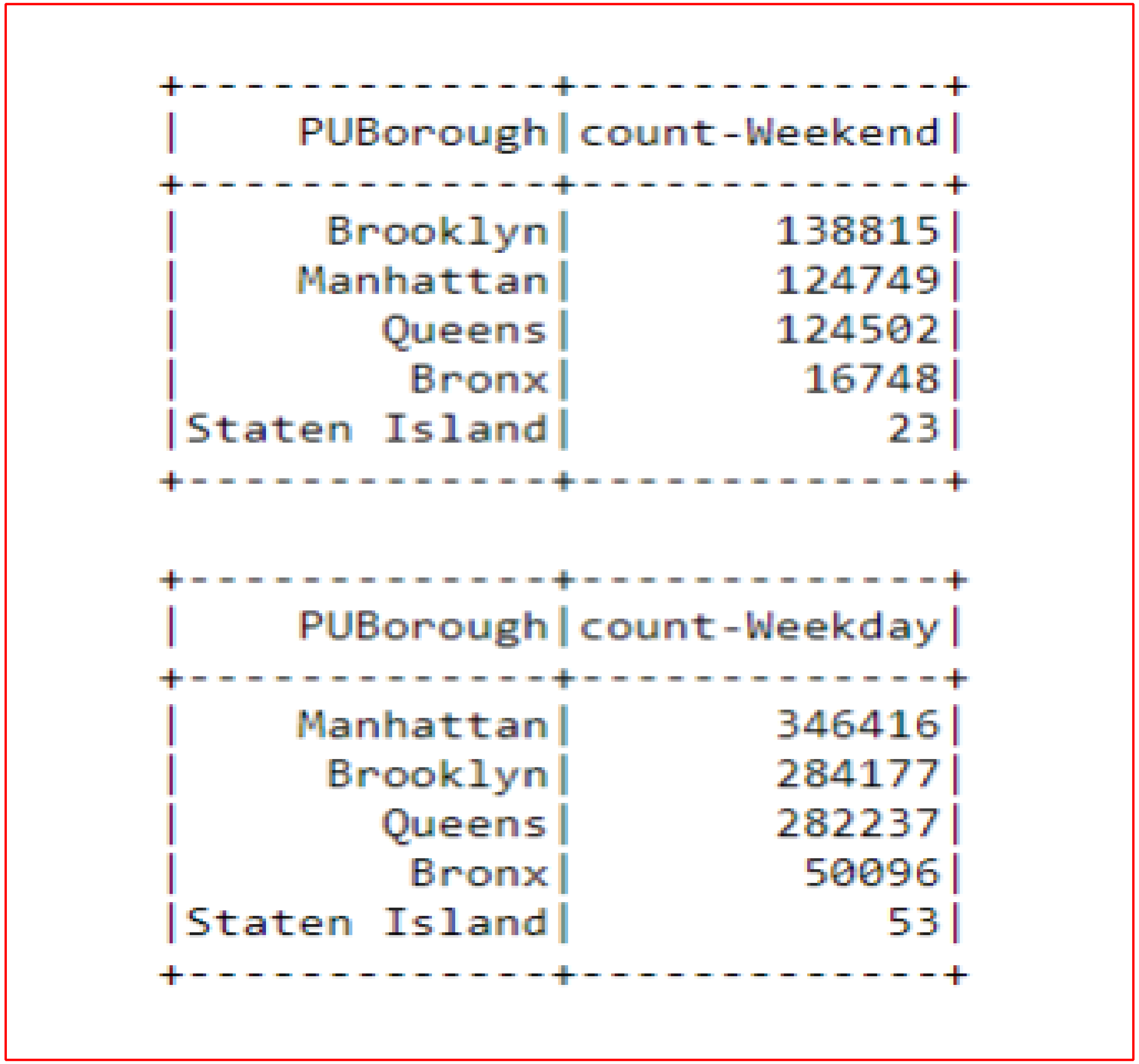


```
+------------+------+
|Payment_type| count|
+------------+------+
|           1|739608|
|           3|  3106|
|           5|    41|
|           4|  2407|
|           2|623425|
+------------+------+
```

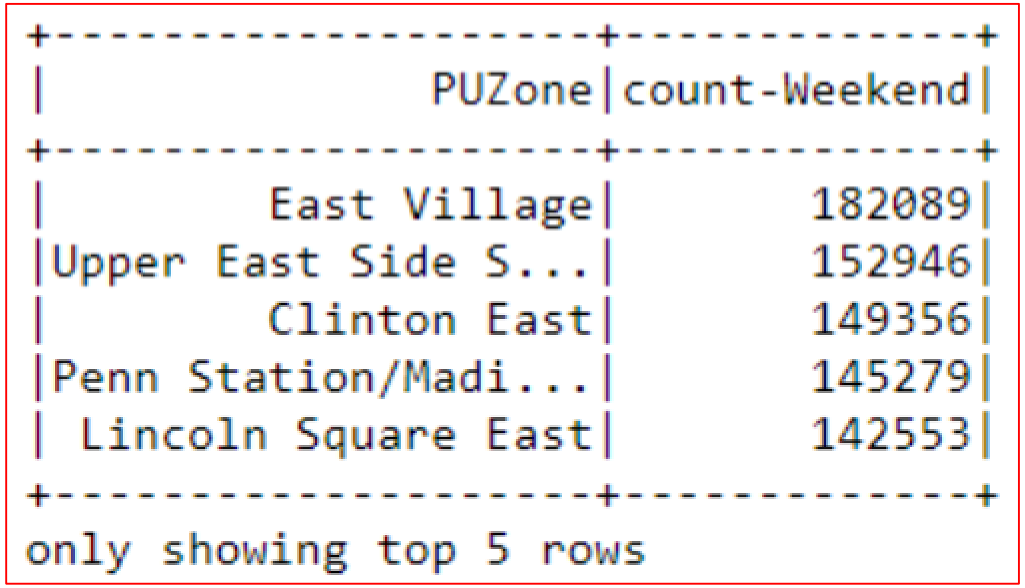
DISPUTE →



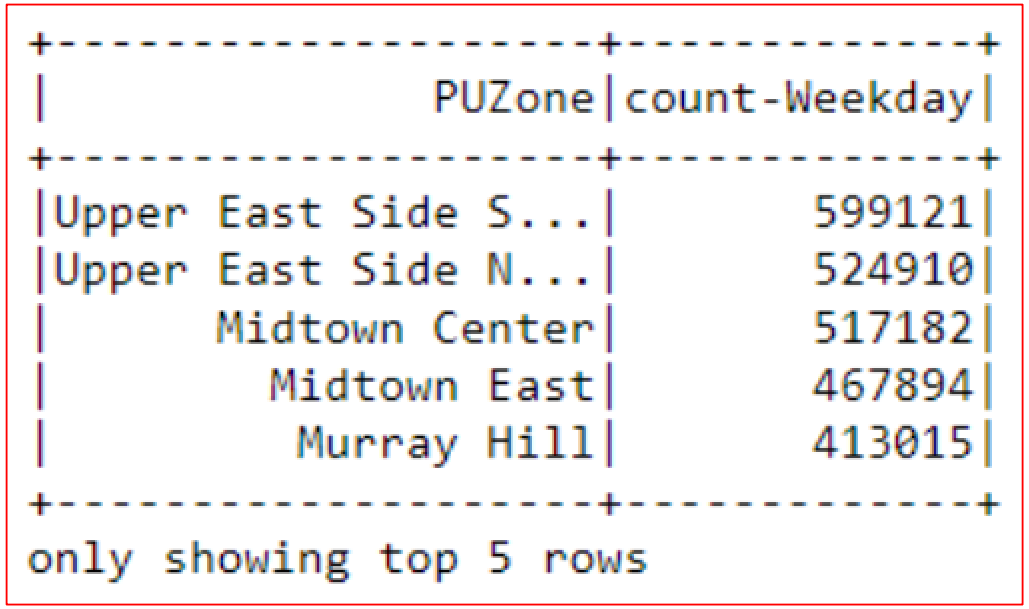
```
+---------+-----+
|PUBorough|count|
+---------+-----+
|   Queens|  882|
| Brooklyn|  834|
|Manhattan|  558|
|    Bronx|  130|
|  Unknown|    3|
+---------+-----+
```



```
+-------------+-------------+
|    PUBorough|count-Weekend|
+-------------+-------------+
|     Brooklyn|       138815|
|    Manhattan|       124749|
|       Queens|       124502|
|        Bronx|        16748|
|Staten Island|           23|
+-------------+-------------+

+-------------+-------------+
|    PUBorough|count-Weekday|
+-------------+-------------+
|    Manhattan|       346416|
|     Brooklyn|       284177|
|       Queens|       282237|
|        Bronx|        50096|
|Staten Island|           53|
+-------------+-------------+
```

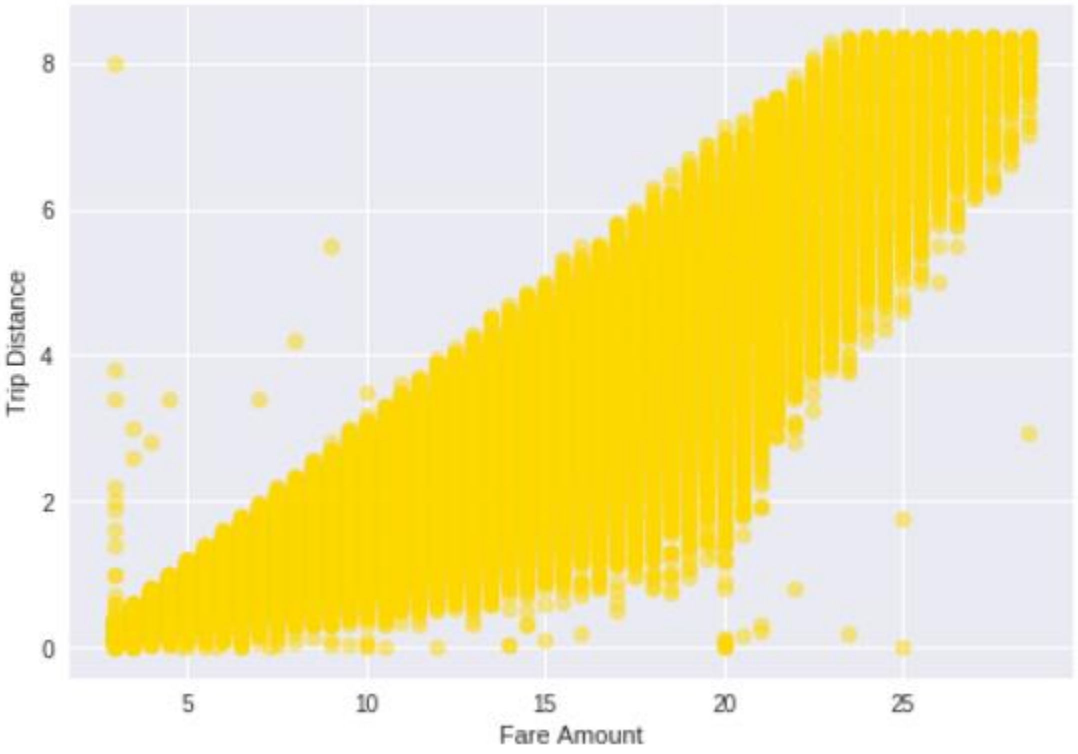


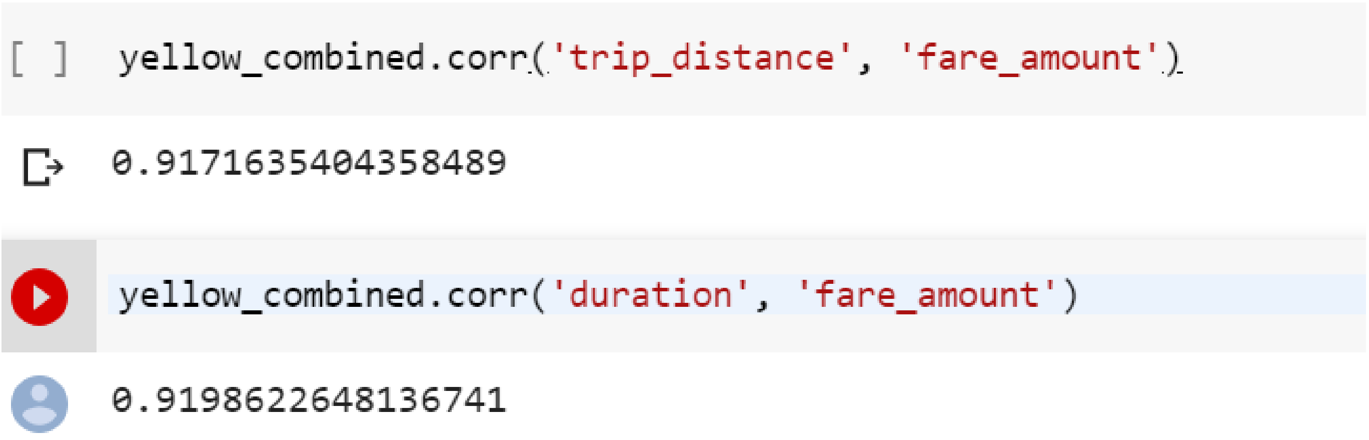
```
+--------------------+-------------+
|              PUZone|count-Weekend|
+--------------------+-------------+
|        East Village|       182089|
|Upper East Side S...|       152946|
|        Clinton East|       149356|
|Penn Station/Madi...|       145279|
| Lincoln Square East|       142553|
+--------------------+-------------+
only showing top 5 rows
```



```
+--------------------+-------------+
|              PUZone|count-Weekday|
+--------------------+-------------+
|Upper East Side S...|       599121|
|Upper East Side N...|       524910|
|      Midtown Center|       517182|
|        Midtown East|       467894|
|         Murray Hill|       413015|
+--------------------+-------------+
only showing top 5 rows
```

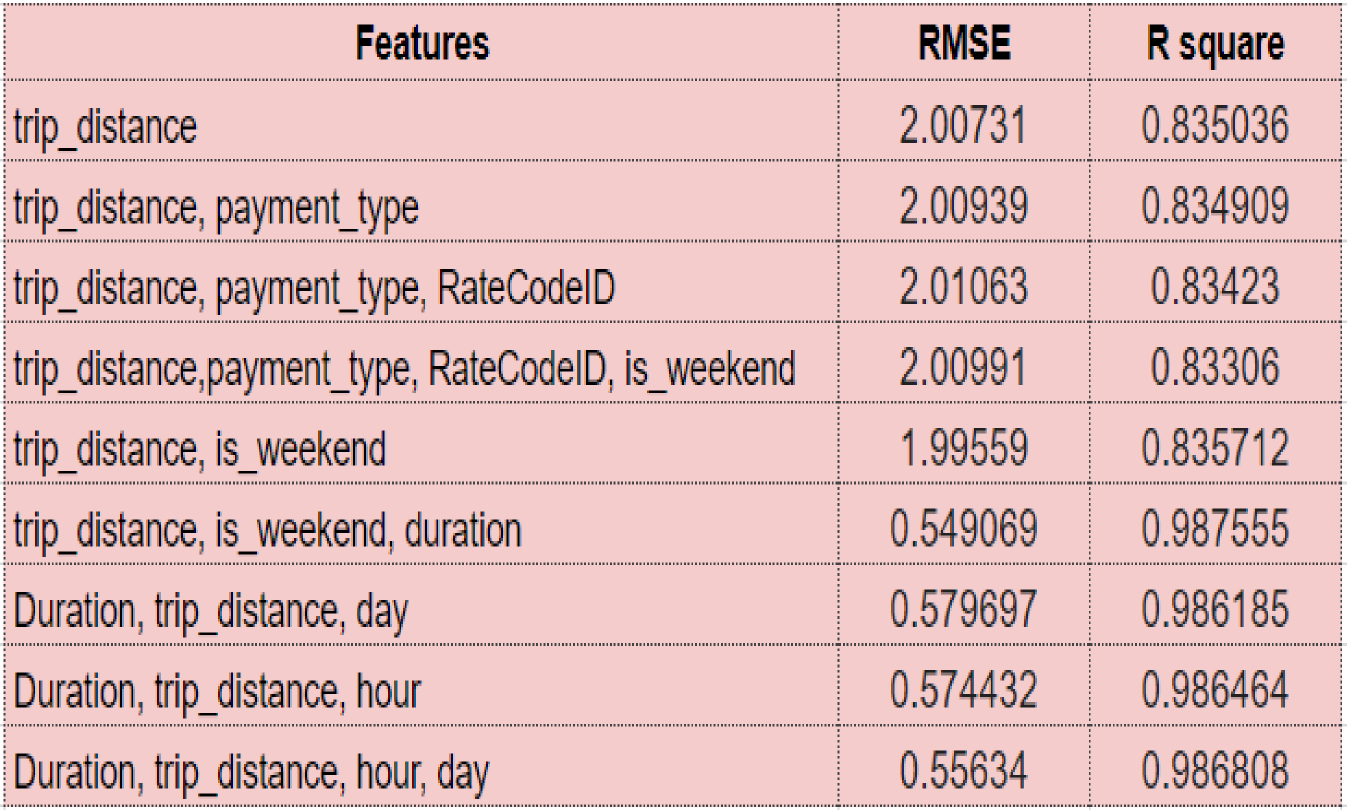
Fare Amount VS Trip Distance, Duration



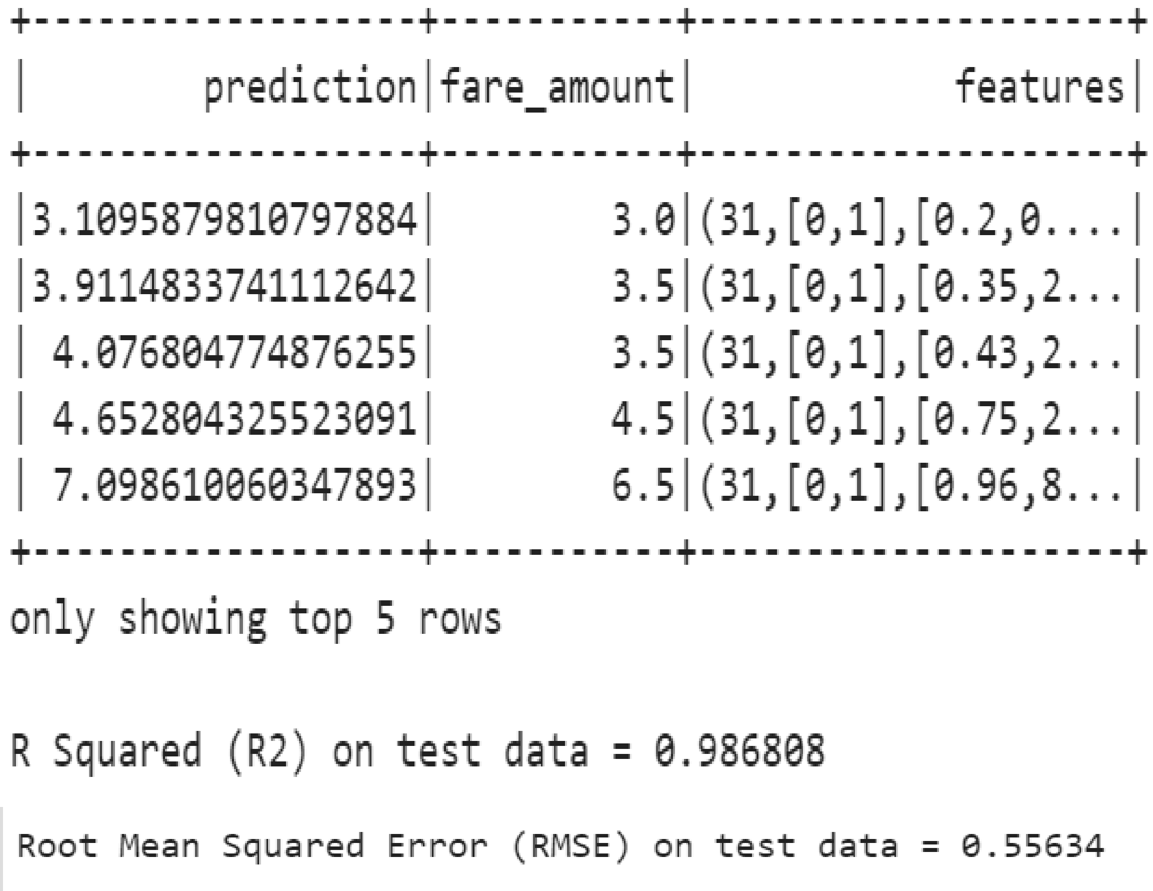


```
[ ]  yellow_combined.corr('trip_distance', 'fare_amount')

     0.9171635404358489

     yellow_combined.corr('duration', 'fare_amount')

     0.9198622648136741
```

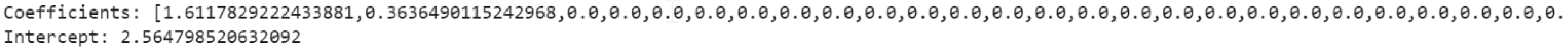
Predictive Model



Features	RMSE	R square
trip_distance	2.00731	0.835036
trip_distance, payment_type	2.00939	0.834909
trip_distance, payment_type, RateCodeID	2.01063	0.83423
trip_distance,payment_type, RateCodeID, is_weekend	2.00991	0.83306
trip_distance, is_weekend	1.99559	0.835712
trip_distance, is_weekend, duration	0.549069	0.987555
Duration, trip_distance, day	0.579697	0.986185
Duration, trip_distance, hour	0.574432	0.986464
Duration, trip_distance, hour, day	0.55634	0.986808



```
+------------------+-----------+--------------------+
|        prediction|fare_amount|            features|
+------------------+-----------+--------------------+
|3.1095879810797884|        3.0|(31,[0,1],[0.2,0....|
|3.9114833741112642|        3.5|(31,[0,1],[0.35,2...|
| 4.076804774876255|        3.5|(31,[0,1],[0.43,2...|
| 4.652804325523091|        4.5|(31,[0,1],[0.75,2...|
| 7.098610060347893|        6.5|(31,[0,1],[0.96,8...|
+------------------+-----------+--------------------+
only showing top 5 rows

R Squared (R2) on test data = 0.986808

Root Mean Squared Error (RMSE) on test data = 0.55634
```



```
Coefficients: [1.6117829222433881,0.3636490115242968,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
Intercept: 2.564798520632092
```

Recommendation

- Taxi fares are fixed throughout the year, regardless of seasonality.
- Competitors are capitalizing on peak hours when demand is greater.
- Variable Pricing - Capitalize on peak hours.
- Location Targeting based on weekday/weekend and hour of the day.

THANK YOU

Appendix - Code

```
!apt-get install openjdk-8-jdk-headless -qq > /dev/null
!wget -q https://www-eu.apache.org/dist/spark/spark-2.4.0/spark-2.4.0-bin-hadoop2.7.tgz
!tar xf spark-2.4.0-bin-hadoop2.7.tgz
!pip install -q findspark

# Set environmental variables
import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
#os.environ["SPARK_HOME"] = "/content/spark-2.2.1-bin-hadoop2.7"
os.environ["SPARK_HOME"] = "/content/spark-2.4.0-bin-hadoop2.7"
```

```
import findspark
findspark.init()
from pyspark.sql import SparkSession

spark = SparkSession.builder.master("local[*]").getOrCreate()

spark

from pyspark.sql import SparkSession
spark = SparkSession.builder.appName('NYC').getOrCreate()
```

Appendix - Code

```
!wget https://s3.amazonaws.com/nyc-tlc/trip+data/yellow_tripdata_2018-04.csv
!wget https://s3.amazonaws.com/nyc-tlc/trip+data/yellow_tripdata_2018-05.csv
print(spark.version)
yellow_april = spark.read.csv("yellow_tripdata_2018-04.csv", header=True, inferSchema=True)
yellow_may = spark.read.csv("yellow_tripdata_2018-05.csv", header=True, inferSchema=True)
green_april = spark.read.csv("green_tripdata_2018-04.csv", header=True, inferSchema=True)
green_may = spark.read.csv("green_tripdata_2018-05.csv", header=True, inferSchema=True)
yellow_combined = yellow_april.union(yellow_may)
yellow_combined.printSchema()
green_combined = green_april.union(green_may)
yellow_combined = yellow_combined.filter(yellow_combined["VendorID"]!=4)
```

```
yellow_combined = yellow_combined.filter(yellow_combined["passenger_count"]!=0)
green_combined = green_combined.filter(green_combined["passenger_count"]!=0)
#Outliers Detection
cols = ['trip_distance','fare_amount', 'extra', 'mta_tax', 'tip_amount',
        'tolls_amount', 'improvement_surcharge', 'total_amount']
bounds = {}
for col in cols:
  quantiles = yellow_combined.approxQuantile(
      col, [0.25, 0.75], 0.05
  )
  IQR = quantiles[1] - quantiles[0]
  bounds[col] = [
      quantiles[0] - 1.5 * IQR,
      quantiles[1] + 1.5 * IQR
  ]
bounds
```

Appendix - Code

```
yellow_combined =
yellow_combined.filter((yellow_combined["trip_distance"]  > 0) &
                                  (yellow_combined["trip_distance"] <
bounds['trip_distance'][1]))
yellow_combined =
yellow_combined.filter((yellow_combined["fare_amount"]  > 2.5) &
                                  (yellow_combined["fare_amount"] <
bounds['fare_amount'][1]))
print((yellow_combined.count(), len(yellow_combined.columns)))
cols = ['trip_distance','fare_amount', 'extra', 'mta_tax', 'tip_amount',
       'tolls_amount', 'improvement_surcharge', 'total_amount']
green_bounds = {}
for col in cols:
  quantiles = green_combined.approxQuantile(
     col, [0.25, 0.75], 0.05
  )
  IQR = quantiles[1] - quantiles[0]
  green_bounds[col] = [
     quantiles[0] - 1.5 * IQR,
     quantiles[1] + 1.5 * IQR
```

```
green_combined =
green_combined.filter((green_combined["trip_distance"]  > 0) &
                                  (green_combined["trip_distance"] <
green_bounds['trip_distance'][1]))
green_combined =
green_combined.filter((green_combined["fare_amount"]  > 2.5) &
                                  (green_combined["fare_amount"] <
green_bounds['fare_amount'][1]))
yellow_combined =
yellow_combined.filter(yellow_combined["RateCodeID"]!= 99)
from pyspark.sql import functions as F
timeFmt = "yyyy-MM-dd'T'HH:mm:ss.SSS"
timeDiff = (F.unix_timestamp('tpep_dropoff_datetime',
format=timeFmt)
            - F.unix_timestamp('tpep_pickup_datetime', format=timeFmt))
yellow_combined = yellow_combined.withColumn("duration",
timeDiff/60)
```

Appendix - Code

```
cols = ['duration']
time_bounds = {}
for col in cols:
  quantiles = yellow_combined.approxQuantile(
    col, [0.25, 0.75], 0.05
  )
  IQR = quantiles[1] - quantiles[0]
  time_bounds[col] = [
    quantiles[0] - 1.5 * IQR,
    quantiles[1] + 1.5 * IQR
  ]
  time_bounds
yellow_combined = yellow_combined.filter((yellow_combined["duration"]  > 0) & (yellow_combined["duration"] < time_bounds['duration'][1]))
from pyspark.sql import functions as F
timeFmt = "yyyy-MM-dd'T'HH:mm:ss.SSS"
```

```
timeDiff = (F.unix_timestamp('lpep_dropoff_datetime', format=timeFmt)
            - F.unix_timestamp('lpep_pickup_datetime', format=timeFmt))
green_combined = green_combined.withColumn("duration", timeDiff/60)
cols = ['duration']
time_bounds = {}
for col in cols:
  quantiles = green_combined.approxQuantile(
    col, [0.25, 0.75], 0.05
  )
  IQR = quantiles[1] - quantiles[0]
  time_bounds[col] = [
    quantiles[0] - 1.5 * IQR,
    quantiles[1] + 1.5 * IQR
  ]
time_bounds
green_combined = green_combined.filter((green_combined["duration"]  > 0) &
```

Appendix - Code

```
from pyspark.sql.functions import date_format

from pyspark.sql.functions import isnan, when, count, col


date_green = green_combined.select('lpep_pickup_datetime',

                       date_format('lpep_pickup_datetime', 'H').alias('hour'),

                       date_format('lpep_pickup_datetime', 'd').alias('date'),

                       date_format('lpep_pickup_datetime',
'u').alias('day_number'), date_format('lpep_pickup_datetime',
'E').alias('day'))

#green.show()

date_green = date_green.withColumn('is_weekend',
when((date_green.day == 'Sun') | (date_green.day ==
'Sat'),1).otherwise(0))

# Convert string to numeric


from pyspark.sql.types import IntegerType

date_green = date_green.withColumn("hour",
date_green["hour"].cast(IntegerType()))

date_green = date_green.withColumn("date",
date_green["date"].cast(IntegerType()))
```

```
date_green = date_green.withColumn("day_number",
date_green["day_number"].cast(IntegerType()))

date_green = date_green.withColumn("is_weekend",
date_green["is_weekend"].cast(IntegerType()))

from pyspark.sql.functions import date_format

from pyspark.sql.functions import isnan, when, count, col


date_yellow = yellow_combined.select('tpep_pickup_datetime',

                       date_format('tpep_pickup_datetime',
'H').alias('hour'),

                       date_format('tpep_pickup_datetime',
'd').alias('date_yellow'),

                       date_format('tpep_pickup_datetime',
'u').alias('day_number'), date_format('tpep_pickup_datetime',
'E').alias('day'))

#yellow_combined.show()

date_yellow =
date_yellow.withColumn('is_weekend',when((date_yellow.day ==
'Sun') | (date_yellow.day == 'Sat'),1).otherwise(0))

# Convert string to numeric
```

Appendix - Code

```
from pyspark.sql.types import IntegerType
date_yellow = date_yellow.withColumn("hour",
date_yellow["hour"].cast(IntegerType()))
date_yellow = date_yellow.withColumn("date_yellow",
date_yellow["date_yellow"].cast(IntegerType()))
date_yellow = date_yellow.withColumn("day_number",
date_yellow["day_number"].cast(IntegerType()))
date_yellow = date_yellow.withColumn("is_weekend",
date_yellow["is_weekend"].cast(IntegerType()))
from pyspark.sql.functions import monotonically_increasing_id
green_combined = green_combined.drop("lpep_pickup_datetime")
green_combined = green_combined.withColumn("id",
monotonically_increasing_id())


date_green = date_green.withColumn("id",
monotonically_increasing_id())
green_combined = date_green.join(green_combined, "id",
"outer").drop("id")


green_combined.show()
```

```
from pyspark.sql.functions import monotonically_increasing_id
yellow_combined = yellow_combined.drop("tpep_pickup_datetime")
yellow_combined = yellow_combined.withColumn("id",
monotonically_increasing_id())
date_yellow = date_yellow.withColumn("id",
monotonically_increasing_id())
yellow_combined = date_yellow.join(yellow_combined, "id",
"outer").drop("id")


yellow_combined.show(5)
yellow_combined = yellow_combined.withColumn("speed",
yellow_combined['trip_distance']*60/yellow_combined['duration'])
green_combined = green_combined.withColumn("speed",
green_combined['trip_distance']*60/green_combined['duration'])
```

Appendix - Code

```
#Visualizations
##Histogram
import matplotlib.pyplot as plt
# Show histogram of the 'C1' column
bins, counts = yellow.select('trip_distance').rdd.flatMap(lambda x: x).histogram(10)
plt.hist(bins[:-1], bins=bins, weights=counts)
bins, counts = green_new.select('trip_distance').rdd.flatMap(lambda x: x).histogram(10)
plt.hist(bins[:-1], bins=bins, weights=counts)
##Barplot
# Create dataframe with frequencies with date
hour_plot = joined1.groupBy('hour').count()
# Convert the dataframe to pandas dataframe
hour_plot = hour_plot.toPandas()
# Set the hour column as index
hour_plot =hour_plot.set_index('hour')
hour_plot.sort_index(inplace=True)
hour_plot.T.squeeze().plot.bar()
```

```
##Boxplot
# BoxPlot
d2 = yellow.select('fare_amount').rdd.flatMap(lambda x: x).collect()
fig1, ax1 = plt.subplots()
ax1.set_title('Basic Plot')
ax1.boxplot(d2)
##Scatterplot
scar = yellow.select('trip_distance','fare_amount')
scar.show(5)
s1 = scar.select('fare_amount').rdd.flatMap(lambda x: x).collect()
s2 = scar.select('trip_distance').rdd.flatMap(lambda x: x).collect()
plt.scatter(s1,s2, alpha = 0.5, color = 'red', cmap='viridis', marker=r'$\clubsuit$', label="Luck")
plt.xlabel('Fare Amount')
plt.ylabel('Trip Distance')
plt.legend(loc='upper left')
plt.show()
```

Appendix - Code

```
#Model


df.groupBy("ID").pivot("Text").agg(F.lit(1)).na.fill(0).show()

yellow =
yellow_combined.select('trip_distance','duration','fare_amount','day','
hour')

from pyspark.sql.functions import monotonically_increasing_id

yellow = yellow.withColumn("id", monotonically_increasing_id())
```

```
from pyspark.sql import functions as F

new_yellow = yellow.groupBy("id").pivot("day").agg(F.lit(1)).na.fill(0)

from pyspark.sql import functions as F

new_yellow2 = yellow.groupBy("id").pivot("hour").agg(F.lit(1)).na.fill(0)

dummy_encoded_df = yellow.join(new_yellow, "id", "outer")

dummy_encoded_df = dummy_encoded_df.join(new_yellow2, "id",
"outer").drop("id")

dummy_encoded_df.show(5)

from pyspark.ml.feature import VectorAssembler


vectorAssembler = VectorAssembler(inputCols =
['trip_distance',’duration’,
'Mon','Wed','Thu','Fri','Sat','Sun','0','1','2','3','4','6','7','8','9','10','11','12'
,'13','14','15','16','17','18','19','20','21','22','23'], outputCol = 'features')

regression_df = vectorAssembler.transform(dummy_encoded_df)

regression_df = regression_df.select(['features', 'fare_amount'])

regression_df.show(3)
```

Appendix - Code

```
splits = regression_df.randomSplit([0.7, 0.3])
train_df = splits[0]
test_df = splits[1]
from pyspark.ml.regression import LinearRegression
lr = LinearRegression(featuresCol = 'features', labelCol='fare_amount',
maxIter=10, regParam=0.3, elasticNetParam=0.8)
lr_model = lr.fit(train_df)
print("Coefficients: " + str(lr_model.coefficients))
print("Intercept: " + str(lr_model.intercept))
lr_predictions = lr_model.transform(test_df)
lr_predictions.select("prediction","fare_amount","features").show(5)
from pyspark.ml.evaluation import RegressionEvaluator
lr_evaluator = RegressionEvaluator(predictionCol="prediction", \
            labelCol="fare_amount",metricName="r2")
print("R Squared (R2) on test data = %g" %
lr_evaluator.evaluate(lr_predictions))
```