```
%md ##                                   Aarhus City Road Traffic data  Analysis
```

FINISHED

Aarhus City Road Traffic data Analysis

FINISHED

```
%pyspark
### Import Libraries
import pandas as pd
from pyspark.sql.types import StringType
from pyspark import SQLContext
#Spark RDD
#road_rdd = sc.textFile("/Users/jyothi/Desktop/capstone/trafficData.csv")
#Pandas Dataframe
#road_pd = pd.read_csv("/Users/jyothi/Desktop/capstone/trafficData158324.csv")
```

```
['status', 'avgMeasuredTime', 'avgSpeed', 'extID', 'medianMeasuredTime', 'TIMESTAMP', 'vehi
cleCount', '_id', 'REPORT_ID']
```

FINISHED

```
%pyspark
#Loading Data Files as spark Dataframe.
road_df = sqlContext.read.format("com.databricks.spark.csv").option("header", "true").load
## List columns in a dataset
road_columns = list(road_df.columns)
road_columns
```

```
['status', 'avgMeasuredTime', 'avgSpeed', 'extID', 'medianMeasuredTime', 'TIMESTAMP', 'vehi
cleCount', '_id', 'REPORT_ID']
```

FINISHED

```
%pyspark
road_df.show()
```

```
+------+---------------+-------+-----+-----------------+-------------------+-----------
-+------+---------+
|status|avgMeasuredTime|avgSpeed|extID|medianMeasuredTime|          TIMESTAMP|vehicleCoun
t|   _id|REPORT_ID|
+------+---------------+-------+-----+-----------------+-------------------+-----------
-+------+---------+
|    OK|             92|      91|  646|               92|2014-02-13T11:30:00|          3
5|189978|   158446|
|    OK|             84|     100|  646|               84|2014-02-13T11:35:00|          2
6|190427|   158446|
|    OK|             78|     108|  646|               78|2014-02-13T11:40:00|          3
1|190876|   158446|
|    OK|             79|     106|  646|               79|2014-02-13T11:45:00|          2
8|191325|   158446|
|    OK|             77|     109|  646|               77|2014-02-13T11:50:00|          3
5|191774|   158446|
|    OK|             83|     101|  646|               83|2014-02-13T11:55:00|          3
```

FINISHED

```
%pyspark
#Loading Meta Data as PySpark Dataframe.
meta_df = sqlContext.read.format("com.databricks.spark.csv").option("header", "true").load
meta_df = meta_df.drop('extID')
meta_df = meta_df.drop('_id')
metadata_columns = list(meta_df.columns)
metadata_columns
```

```
['POINT_1_STREET', 'DURATION_IN_SEC', 'POINT_1_NAME', 'POINT_1_CITY', 'POINT_2_NAME', 'POIN
T_2_LNG', 'POINT_2_STREET', 'NDT_IN_KMH', 'POINT_2_POSTAL_CODE', 'POINT_2_COUNTRY', 'POINT_
1_STREET_NUMBER', 'ORGANISATION', 'POINT_1_LAT', 'POINT_2_LAT', 'POINT_1_POSTAL_CODE', 'POI
NT_2_STREET_NUMBER', 'POINT_2_CITY', 'ROAD_TYPE', 'POINT_1_LNG', 'REPORT_ID', 'POINT_1_COUN
TRY', 'DISTANCE_IN_METERS', 'REPORT_NAME', 'RBA_ID']
```

FINISHED

```
%pyspark
meta_df.printSchema()
```

```
root
 |-- POINT_1_STREET: string (nullable = true)
 |-- DURATION_IN_SEC: string (nullable = true)
 |-- POINT_1_NAME: string (nullable = true)
 |-- POINT_1_CITY: string (nullable = true)
 |-- POINT_2_NAME: string (nullable = true)
 |-- POINT_2_LNG: string (nullable = true)
 |-- POINT_2_STREET: string (nullable = true)
 |-- NDT_IN_KMH: string (nullable = true)
 |-- POINT_2_POSTAL_CODE: string (nullable = true)
 |-- POINT_2_COUNTRY: string (nullable = true)
 |-- POINT_1_STREET_NUMBER: string (nullable = true)
 |-- ORGANISATION: string (nullable = true)
 |-- POINT_1_LAT: string (nullable = true)
 |-- POINT_2_LAT: string (nullable = true)
 |-- POINT_1_POSTAL_CODE: string (nullable = true)
 |-- POINT_2_STREET_NUMBER: string (nullable = true)
 |-- POINT_2_CITY: string (nullable = true)
```

FINISHED

```
%pyspark
## Joining two datasets. Removed repeated columns
df = road_df.join(meta_df, (road_df.REPORT_ID == meta_df.REPORT_ID) ).drop(meta_df.REPORT_
df.printSchema()
```

```
root
 |-- status: string (nullable = true)
 |-- avgMeasuredTime: string (nullable = true)
 |-- avgSpeed: string (nullable = true)
 |-- extID: string (nullable = true)
 |-- medianMeasuredTime: string (nullable = true)
 |-- TIMESTAMP: string (nullable = true)
 |-- vehicleCount: string (nullable = true)
 |-- _id: string (nullable = true)
 |-- REPORT_ID: string (nullable = true)
 |-- POINT_1_STREET: string (nullable = true)
 |-- DURATION_IN_SEC: string (nullable = true)
 |-- POINT_1_NAME: string (nullable = true)
 |-- POINT_1_CITY: string (nullable = true)
 |-- POINT_2_NAME: string (nullable = true)
 |-- POINT_2_LNG: string (nullable = true)
 |-- POINT_2_STREET: string (nullable = true)
 |-- NDT_IN_KMH: string (nullable = true)
```

FINISHED

```
%pyspark
df.show()
```

```
+------+--------------+-------+-----+----------------+----------------+----------
-+------+---------+----------------+--------------+-----------+-----------+---------
--+----------------+----------------+---------+-----------------+--------------+--
------------------+-----------+----------------+----------------+-----------------+
--------------------+----------+---------+---------------+-------------+--------
---------+------------------+------+
|status|avgMeasuredTime|avgSpeed|extID|medianMeasuredTime|          TIMESTAMP|vehicleCoun
t|    _id|REPORT_ID|    POINT_1_STREET|DURATION_IN_SEC|POINT_1_NAME|POINT_1_CITY|POINT_2_NA
ME|       POINT_2_LNG|     POINT_2_STREET|NDT_IN_KMH|POINT_2_POSTAL_CODE|POINT_2_COUNTRY|PO
INT_1_STREET_NUMBER|ORGANISATION|       POINT_1_LAT|       POINT_2_LAT|POINT_1_POSTAL_CODE|
POINT_2_STREET_NUMBER|POINT_2_CITY| ROAD_TYPE|       POINT_1_LNG|POINT_1_COUNTRY|DISTANCE_
IN_METERS|         REPORT_NAME|RBA_ID|
+------+--------------+-------+-----+----------------+----------------+----------
-+------+---------+----------------+--------------+-----------+-----------+---------
--+----------------+----------------+---------+-----------------+--------------+--
------------------+-----------+----------------+----------------+-----------------+
--------------------+----------+---------+---------------+-------------+--------
```

FINISHED

```
%pyspark
# Writing in to a single file.
df.repartition(1).write.csv("/Users/jyothi/Downloads/mycsv1.csv")
```

FINISHED

```
%pyspark



# prepare for Python version 3x features and functions
from __future__ import division, print_function

# import packages for multivariate analysis
import pandas as pd  # DataFrame structure and operations
import numpy as np  # arrays and numerical processing
from sklearn.cluster import KMeans  # cluster analysis by partitioning
from sklearn.metrics import silhouette_score as silhouette_score

# read data from comma-delimited text file... create DataFrame object
traffic = pd.read_csv('/Users/jyothi/Desktop/capstone/trafficData158324.csv', sep = ',')
#print(traffic.head)  # check the structure of the data frame
```

FINISHED

```
%pyspark
# examine the demographic variable age
print(traffic['_id'].unique())

#print(traffic['_id'].value_counts(sort = True))#print(traffic['_id'].describe())
type(traffic)
```

```
[  190000   190449   190898 ..., 14353052 14353465 14353801]
<class 'pandas.core.frame.DataFrame'>
```

FINISHED

```
%pyspark
print(traffic['avgSpeed'].describe())
```

```
count    32075.000000
mean        61.103757
std         11.443108
min          0.000000
25%         55.000000
50%         60.000000
75%         67.000000
max        132.000000
Name: avgSpeed, dtype: float64
```

FINISHED

```
%pyspark
import plotly.plotly as py
import plotly.graph_objs as go
import pandas as pd
import numpy as np # for generating random data
from datetime import datetime
from pyspark.sql import functions as F
from pyspark.sql.functions import col,udf, unix_timestamp
from pyspark.sql.types import DateType
from pyspark.sql.functions import from_unixtime
```

FINISHED

```
%md ### Converting String into timestamp for timeseries analysis
```

Converting String into timestamp for timeseries analysis

FINISHED

```
%pyspark
#Hours and Minutes extracted from Timestamp
format = "yyyy-MM-dd'T'HH:mm:ss"
df2 = tr.select('avgSpeed','TIMESTAMP','vehicleCount', from_unixtime(unix_timestamp('TIMES
#ts = unix_timestamp( tr$TIMESTAMP,"MM/dd/yyyy HH:mm:ss").cast("timestamp")
f = df2.select('avgSpeed','TIMESTAMP','vehicleCount', F.hour('date').alias('hour'),F.minut
f.show()
```

```
+--------+-------------------+------------+----+---+
|avgSpeed|          TIMESTAMP|vehicleCount|hour|min|
+--------+-------------------+------------+----+---+
|      56|2014-02-13T11:30:00|           7|  11| 30|
|      53|2014-02-13T11:35:00|           5|  11| 35|
|      53|2014-02-13T11:40:00|           6|  11| 40|
|      52|2014-02-13T11:45:00|           3|  11| 45|
|      57|2014-02-13T11:50:00|           6|  11| 50|
|      49|2014-02-13T11:55:00|           9|  11| 55|
|      50|2014-02-13T12:00:00|          11|  12|  0|
|      62|2014-02-13T12:05:00|           8|  12|  5|
|      60|2014-02-13T12:10:00|          10|  12| 10|
|      58|2014-02-13T12:15:00|          12|  12| 15|
|      59|2014-02-13T12:20:00|          16|  12| 20|
|      59|2014-02-13T12:25:00|          16|  12| 25|
|      62|2014-02-13T12:30:00|           8|  12| 30|
|      55|2014-02-13T12:35:00|           9|  12| 35|
|      57|2014-02-13T12:40:00|           8|  12| 40|
```

FINISHED

```
%pyspark
 f.registerTempTable("minutedata")
```

FINISHED

```
%sql
select  TIMESTAMP , avgSpeed   from minutedata
```

settings

No Da

```
Results are limited by 1000.
```

FINISHED

```
%md ###Hourly TimeSeries Graph :     X -Axis Hour of the day.  Y-Axis Average speed by eac
```

Hourly TimeSeries Graph : X -Axis Hour of the day. Y-Axis Average speed by each hour

```
%md ####Day time average traffic speed is lesser than night time traffic.
```

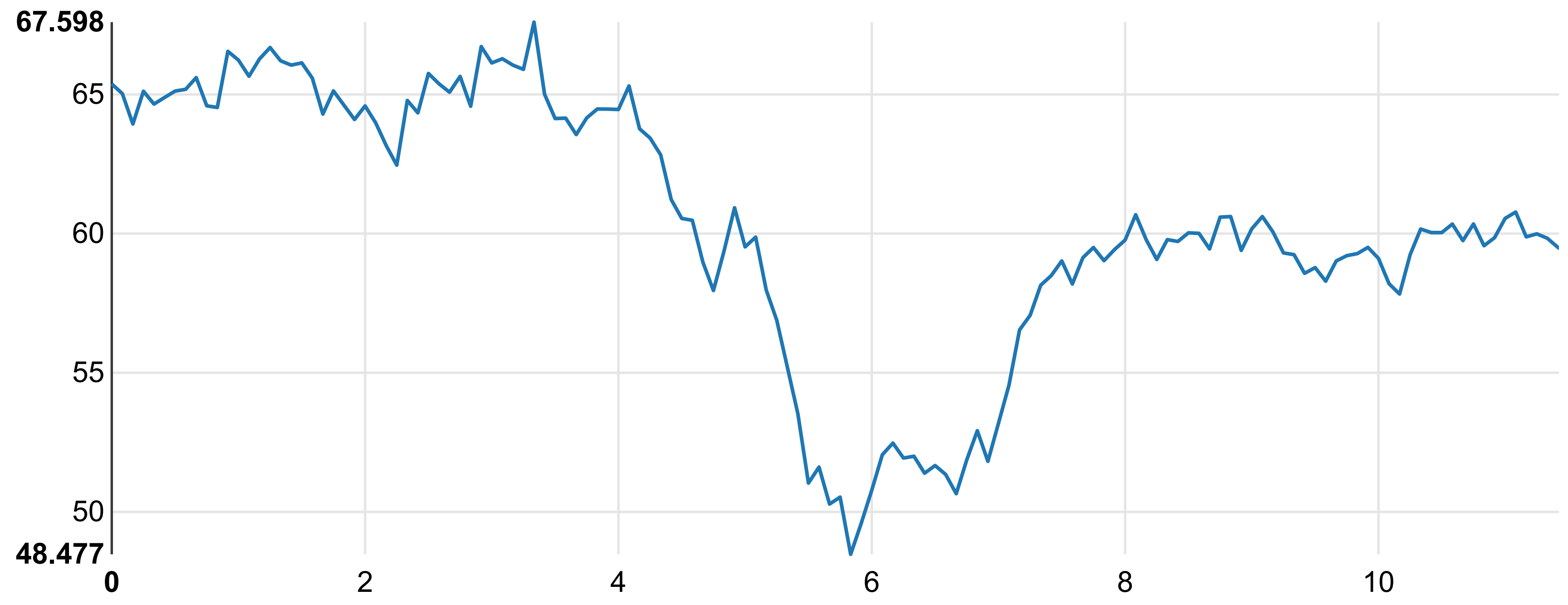
FINISHED

Day time average traffic speed is lesser than night time traffic.

```
%sql
select (hour*60 + min)/60 as time  ,avg(avgSpeed) as avg_speed   from minutedata group by l
```

FINISHED

settings



```
%md ### Graph represents Average Speed Vs  Vehicle count
```

FINISHED

Graph represents Average Speed Vs Vehicle count

```
%md ### When Vehicle increases average speed decreases
```

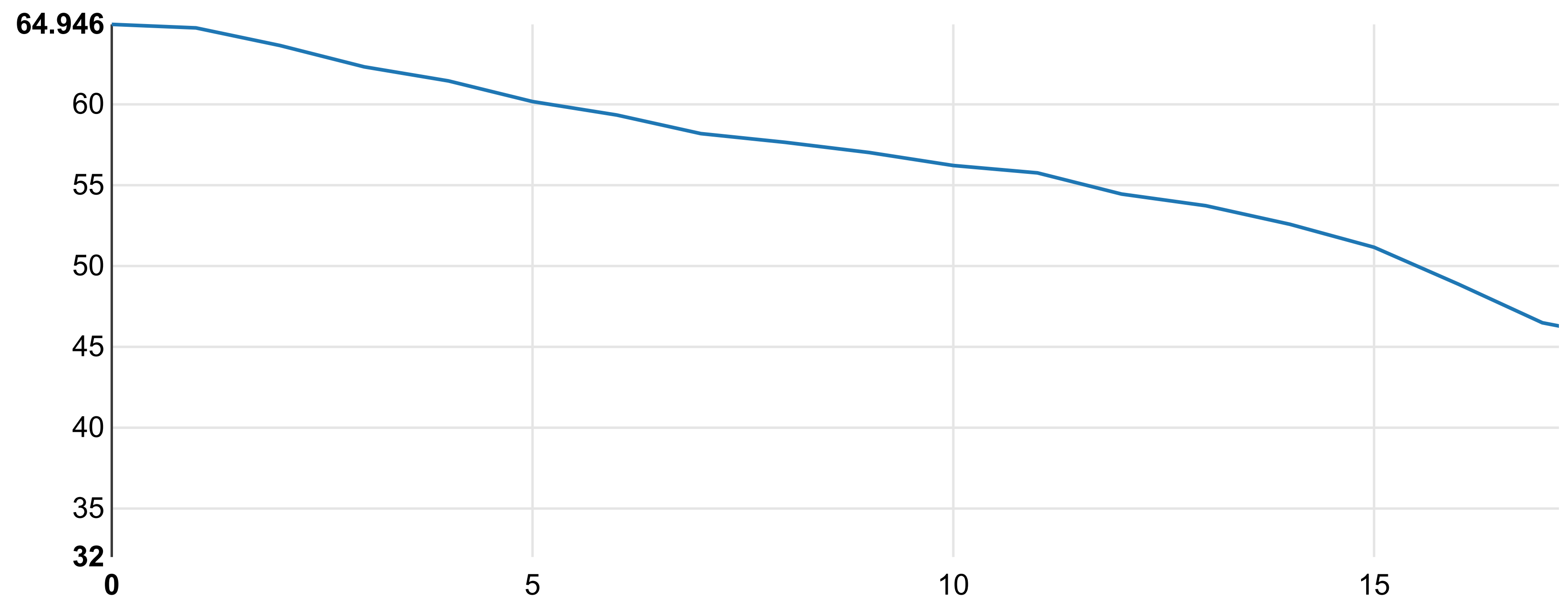
FINISHED

When Vehicle increases average speed decreases

FINISHED

```
%sql
select  vehicleCount, avg(avgSpeed) as speed  from minutedata  group by vehicleCount order
```

settings

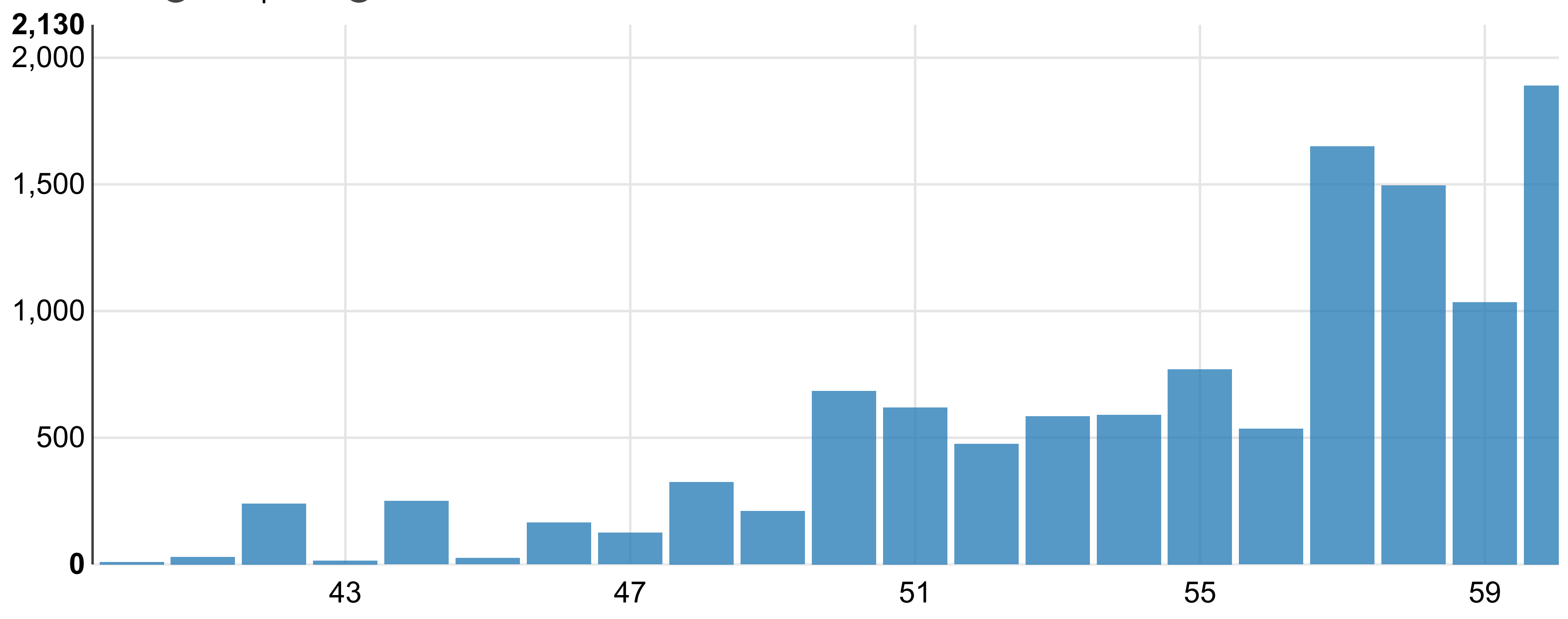


FINISHED

```
%sql
select * from minute_data
```

settings

● Grouped ○ Stacked



```
Results are limited by 1000.
```

```
%pyspark
traffic = f.toPandas()
dat = [go.Histogram(x=traffic['avgSpeed'])]
```

FINISHED

```
%pyspark
py.plot(dat, filename='Traffic-histogram')
```

```
'https://plot.ly/~jkunaparaju/101'
```

FINISHED

```
%pyspark



py.iplot(dat, filename='Traffic-histogram')
```

```
<plotly.tools.PlotlyDisplay object>
```

FINISHED

```

from datetime import datetime
from pyspark.sql.functions import col,udf, unix_timestamp
from pyspark.sql.types import DateType
from pyspark.sql.types import  TimestampType
func =  udf (lambda x:  TimestampType.strptime(x, '%M/%d/%Y'), DateType())
df1 = tr.withColumn('newdate', func(col('TIMESTAMP')))
df1.show()
```

ERROR

```
%pyspark
import seaborn as sns
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
```

FINISHED

```
%pyspark
def show(p):
  img = StringIO.StringIO()
  p.savefig(img, format='svg')
  img.seek(0)
  print( "%html " + img.buf)
df = sqlContext.sql("select vehicleCount,avg(avgSpeed) as avgSpeed , hour     from minuted
data = df.toPandas()
```

FINISHED

FINISHED

```
%pyspark

value = "avgSpeed"
x = "vehicleCount"
grouping = ["hour"]

heatmap_data = data.pivot_table(values=value, index=x, columns=grouping)
heatmap_data = heatmap_data[0:100]
```

FINISHED

```
%pyspark
a4_dims = (len(heatmap_data.columns),50)
fig, ax = plt.subplots(figsize=a4_dims)
ax.set_title("Avg Speed")
sns.heatmap(heatmap_data, ax=ax, annot=True, fmt=".02f")
```

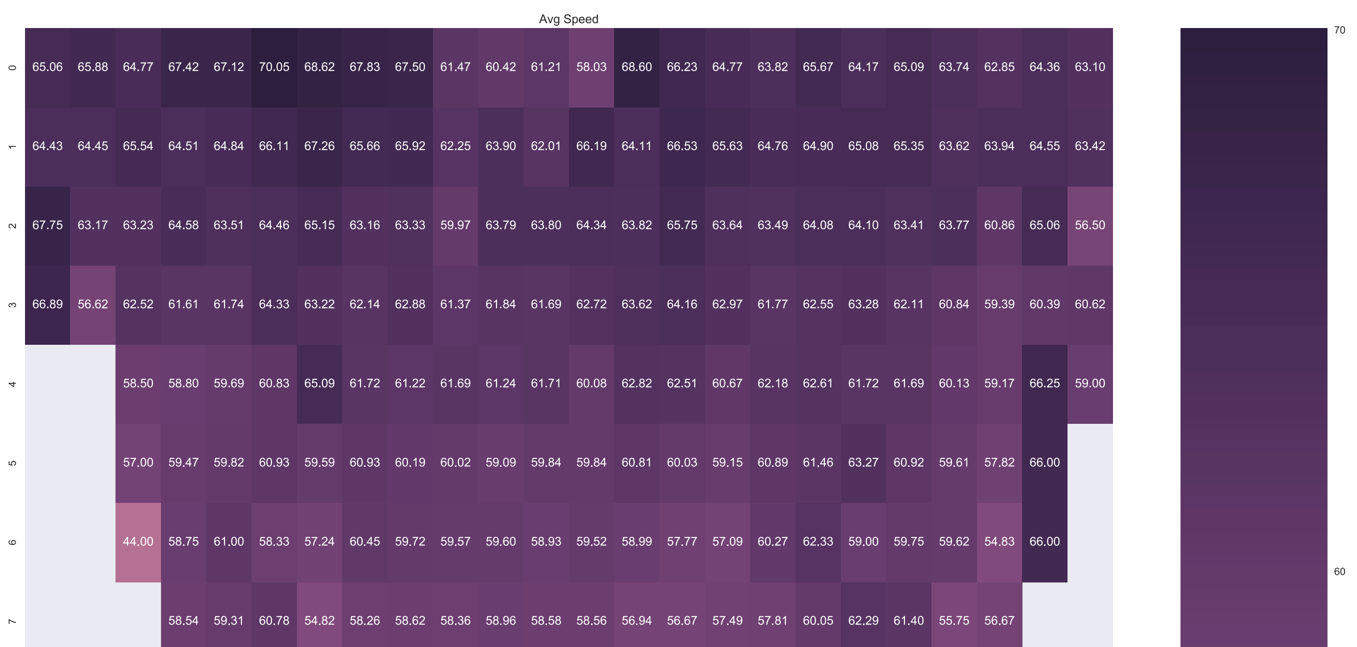
```
<matplotlib.axes._subplots.AxesSubplot object at 0x119aec0b8>
```

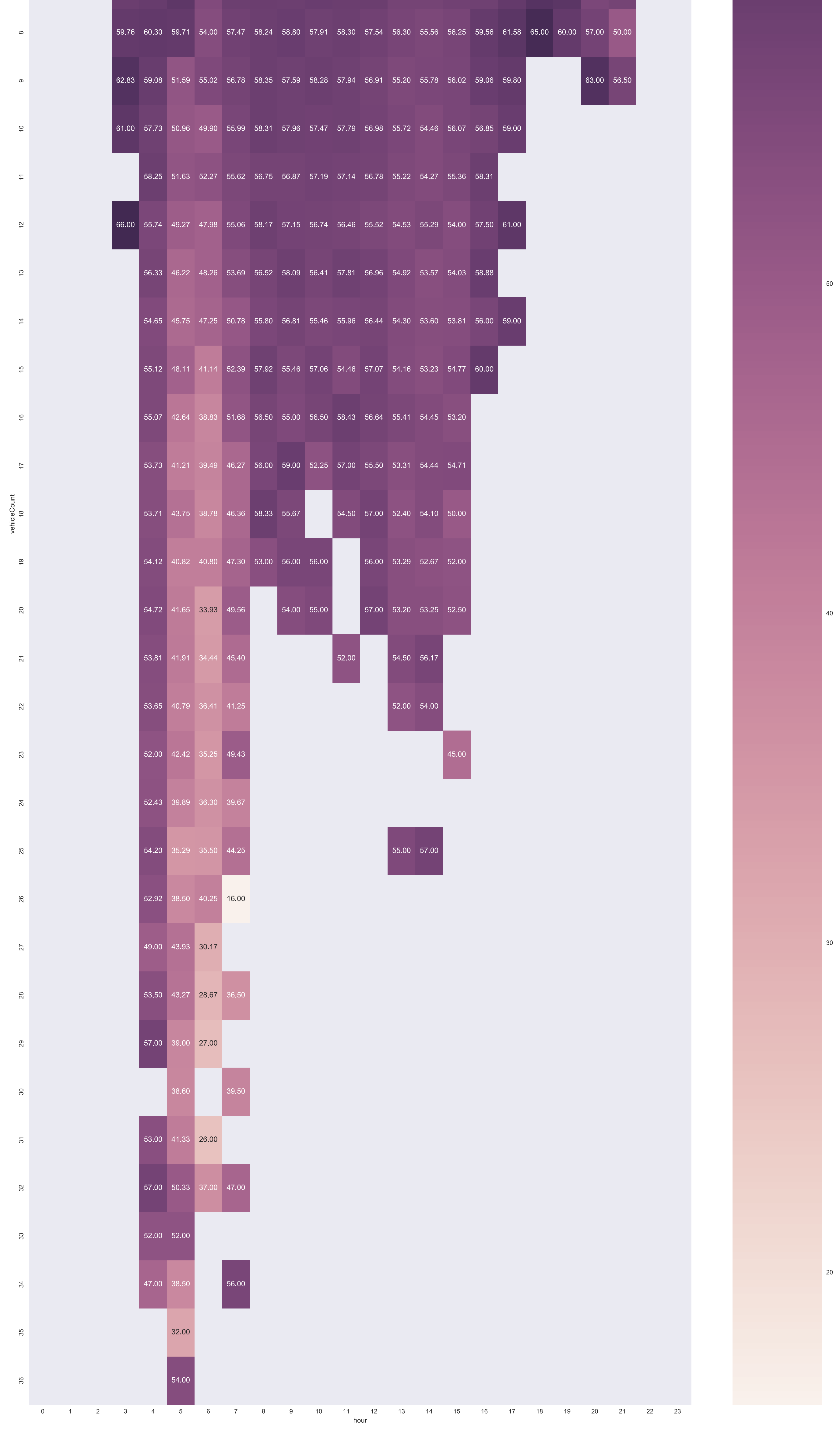
FINISHED

```
%pyspark
def show(p):
  img = StringIO()
  p.savefig(img, format='svg')
  img.seek(0)
  print( "%html " + img.read())
```

FINISHED

```
%pyspark
show(plt)
```




vehicleCount
hour
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
50
40
30
20
59.76 60.30 59.71 54.00 57.47 58.24 58.80 57.91 58.30 57.54 56.30 55.56 56.25 59.56 61.58 65.00 60.00 57.00 50.00
62.83 59.08 51.59 55.02 56.78 58.35 57.59 58.28 57.94 56.91 55.20 55.78 56.02 59.06 59.80 63.00 56.50
61.00 57.73 50.96 49.90 55.99 58.31 57.96 57.47 57.79 56.98 55.72 54.46 56.07 56.85 59.00
58.25 51.63 52.27 55.62 56.75 56.87 57.19 57.14 56.78 55.22 54.27 55.36 58.31
66.00 55.74 49.27 47.98 55.06 58.17 57.15 56.74 56.46 55.52 54.53 55.29 54.00 57.50 61.00
56.33 46.22 48.26 53.69 56.52 58.09 56.41 57.81 56.96 54.92 53.57 54.03 58.88
54.65 45.75 47.25 50.78 55.80 56.81 55.46 55.96 56.44 54.30 53.60 53.81 56.00 59.00
55.12 48.11 41.14 52.39 57.92 55.46 57.06 54.46 57.07 54.16 53.23 54.77 60.00
55.07 42.64 38.83 51.68 56.50 55.00 56.50 58.43 56.64 55.41 54.45 53.20
53.73 41.21 39.49 46.27 56.00 59.00 52.25 57.00 55.50 53.31 54.44 54.71
53.71 43.75 38.78 46.36 58.33 55.67 54.50 57.00 52.40 54.10 50.00
54.12 40.82 40.80 47.30 53.00 56.00 56.00 56.00 53.29 52.67 52.00
54.72 41.65 33.93 49.56 54.00 55.00 57.00 53.20 53.25 52.50
53.81 41.91 34.44 45.40 52.00 54.50 56.17
53.65 40.79 36.41 41.25 52.00 54.00
52.00 42.42 35.25 49.43 45.00
52.43 39.89 36.30 39.67
54.20 35.29 35.50 44.25 55.00 57.00
52.92 38.50 40.25 16.00
49.00 43.93 30.17
53.50 43.27 28.67 36.50
57.00 39.00 27.00
38.60 39.50
53.00 41.33 26.00
57.00 50.33 37.00 47.00
52.00 52.00
47.00 38.50 56.00
32.00
54.00

READY