In [0]:

```
%pip install shapely
```

```
Python interpreter will be restarted.
Collecting shapely
  Downloading Shapely-1.7.1-cp37-cp37m-manylinux1_x86_64.whl (1.0 MB)
Installing collected packages: shapely
Successfully installed shapely-1.7.1
Python interpreter will be restarted.
```

In [0]:

```
%pip install geopandas
```

```
Python interpreter will be restarted.
Collecting geopandas
  Downloading geopandas-0.9.0-py2.py3-none-any.whl (994 kB)
Requirement already satisfied: shapely>=1.6 in /local_disk0/.ephemeral_nfs/envs/
pythonEnv-5fb2cc4b-c1b7-45b0-8ae8-fa185c0dbc1d/lib/python3.7/site-packages (from
geopandas) (1.7.1)
Collecting fiona>=1.8
  Downloading Fiona-1.8.20-cp37-cp37m-manylinux1_x86_64.whl (15.4 MB)
Collecting pyproj>=2.2.0
  Downloading pyproj-3.1.0-cp37-cp37m-manylinux2010_x86_64.whl (6.6 MB)
Requirement already satisfied: pandas>=0.24.0 in /databricks/python3/lib/python
3.7/site-packages (from geopandas) (1.0.1)
Collecting click>=4.0
  Downloading click-8.0.1-py3-none-any.whl (97 kB)
Collecting munch
  Downloading munch-2.5.0-py2.py3-none-any.whl (10 kB)
Collecting click-plugins>=1.0
  Downloading click_plugins-1.1.1-py2.py3-none-any.whl (7.5 kB)
Collecting attrs>=17
  Downloading attrs-21.2.0-py2.py3-none-any.whl (53 kB)
Requirement already satisfied: certifi in /databricks/python3/lib/python3.7/site
-packages (from fiona>=1.8->geopandas) (2020.6.20)
Collecting cligj>=0.5
  Downloading cligj-0.7.2-py3-none-any.whl (7.1 kB)
Requirement already satisfied: six>=1.7 in /databricks/python3/lib/python3.7/sit
e-packages (from fiona>=1.8->geopandas) (1.14.0)
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-packa
ges (from fiona>=1.8->geopandas) (45.2.0)
Collecting importlib-metadata
  Downloading importlib_metadata-4.4.0-py3-none-any.whl (17 kB)
Requirement already satisfied: python-dateutil>=2.6.1 in /databricks/python3/li
b/python3.7/site-packages (from pandas>=0.24.0->geopandas) (2.8.1)
Requirement already satisfied: numpy>=1.13.3 in /databricks/python3/lib/python3.
7/site-packages (from pandas>=0.24.0->geopandas) (1.18.1)
Requirement already satisfied: pytz>=2017.2 in /databricks/python3/lib/python3.
7/site-packages (from pandas>=0.24.0->geopandas) (2019.3)
Collecting typing-extensions>=3.6.4
  Downloading typing_extensions-3.10.0.0-py3-none-any.whl (26 kB)
Collecting zipp>=0.5
  Downloading zipp-3.4.1-py3-none-any.whl (5.2 kB)
Installing collected packages: zipp, typing-extensions, importlib-metadata, clic
k, munch, cligj, click-plugins, attrs, pyproj, fiona, geopandas
Successfully installed attrs-21.2.0 click-8.0.1 click-plugins-1.1.1 cligj-0.7.2
fiona-1.8.20 geopandas-0.9.0 importlib-metadata-4.4.0 munch-2.5.0 pyproj-3.1.0 t
yping-extensions-3.10.0.0 zipp-3.4.1
Python interpreter will be restarted.
```

In [0]:

```
from pyspark.sql.functions import *
from pyspark.sql.functions import date_format
from pyspark.sql.functions import to_date
from shapely.geometry import Point, LineString, Polygon,point
from pyspark.sql.functions import udf
import shapely
from pyspark.sql.types import *
from pyspark.sql import SparkSession
import pandas as pd
import geopandas
import datetime
import time
import shapely.speedups
shapely.speedups.enable()
from pyspark.sql.functions import *
from pyspark.sql import Window
```

In [0]:

```
geo_schema = StructType([
  StructField("properties", StructType([
    StructField("@id", StringType(), True),
    StructField("borough", StringType(), True),
    StructField("boroughCode", IntegerType(), True)
  ]), True),
  StructField("geometry", StructType([
    StructField("coordinates", ArrayType(ArrayType(ArrayType(FloatType()))), True),
    StructField("type", StringType(), True)
  ]), True),
])

nyc_schema = StructType([
  StructField("medallion", StringType(), True),
  StructField("hack_license", StringType(), True),
  StructField("vendor_id", StringType(), True),
  StructField("rate_code", IntegerType(), True),
  StructField("store_and_fwd_flag", StringType(), True),
  StructField("pickup_datetime", StringType(), True),
  StructField("dropoff_datetime", StringType(), True),
  StructField("passenger_count", IntegerType(), True),
  StructField("pickup_longitude", FloatType(), True),
  StructField("pickup_latitude", FloatType(), True),
  StructField("dropoff_longitude", FloatType(), True),
  StructField("dropoff_latitude", FloatType(), True),
])
```

In [0]:

```
df_nyc = (spark.read
         .schema(nyc_schema)
        .format("csv")
        .option("header",True)
        .load("dbfs:/FileStore/shared_uploads/kingbillyintartu@gmail.com/Sample_NYC_Data.csv"))
display(df_nyc)
```

medallion	hack_license	vendor
89D227B655E5C82AECF13C3F540D4CF4	BA96DE419E711691B9445D6A6307C170	C
0BD7C8F5BA12B88E0B67BED28BEA73D8	9FD8F69F0804BDB5549F40E9DA1BE472	C
0BD7C8F5BA12B88E0B67BED28BEA73D8	9FD8F69F0804BDB5549F40E9DA1BE472	C
DFD2202EE08F7A8DC9A57B02ACB81FE2	51EE87E3205C985EF8431D850C786310	C
DFD2202EE08F7A8DC9A57B02ACB81FE2	51EE87E3205C985EF8431D850C786310	C
20D9ECB2CA0767CF7A01564DF2844A3E	598CCE5B9C1918568DEE71F43CF26CD2	C
496644932DF3932605C22C7926FF0FE0	513189AD756FF14FE670D10B92FAF04C	C
0B57B9633A2FECD3D3B1944AFC7471CF	CCD4367B417ED6634D986F573A552A62	C
2C0E91FF20A856C891483ED63589F982	1DA2F6543A62B8ED934771661A9D2FA0	C
2D4B95E2FA7B2E85118EC5CA4570FA58	CD2F522EEE1FF5F5A8D8B679E23576B3	C

In [0]:

```
# Calculate the difference between dropoff_time and pickup_time
def time_difference(pick_up, drop_off):
  pickupTime = time.mktime(datetime.datetime.strptime(pick_up, '%d-%m-%y %H:%M').timetu
ple())
  dropoffTime = time.mktime(datetime.datetime.strptime(drop_off, '%d-%m-%y %H:%M').time
tuple())
  return dropoffTime-pickupTime

timeDiff = udf(time_difference, FloatType())

df_nyc = (df_nyc.
          withColumn('duration', timeDiff(col('pickup_datetime'), col('dropoff_datetim
e')))
          .filter((col('duration') > 0) & (col('duration') < 4*3600)) # if difference i
s negative or greater than 4 hours, drop it
       )

display(df_nyc)
```

medallion	hack_license	vendor
89D227B655E5C82AECF13C3F540D4CF4	BA96DE419E711691B9445D6A6307C170	C
0BD7C8F5BA12B88E0B67BED28BEA73D8	9FD8F69F0804BDB5549F40E9DA1BE472	C
0BD7C8F5BA12B88E0B67BED28BEA73D8	9FD8F69F0804BDB5549F40E9DA1BE472	C
DFD2202EE08F7A8DC9A57B02ACB81FE2	51EE87E3205C985EF8431D850C786310	C
DFD2202EE08F7A8DC9A57B02ACB81FE2	51EE87E3205C985EF8431D850C786310	C
20D9ECB2CA0767CF7A01564DF2844A3E	598CCE5B9C1918568DEE71F43CF26CD2	C
496644932DF3932605C22C7926FF0FE0	513189AD756FF14FE670D10B92FAF04C	C
0B57B9633A2FECD3D3B1944AFC7471CF	CCD4367B417ED6634D986F573A552A62	C
2C0E91FF20A856C891483ED63589F982	1DA2F6543A62B8ED934771661A9D2FA0	C
2D4B95E2FA7B2E85118EC5CA4570FA58	CD2F522EEE1FF5F5A8D8B679E23576B3	C

In [0]:

```
df_geospatial = (spark.read
      .schema(geo_schema)
      .json("dbfs:/FileStore/shared_uploads/kingbillyintartu@gmail.com/nyc_boroughs.geo
json")
      .filter(col('geometry').isNotNull())
      .withColumn("coordinates", explode(col("geometry.coordinates")))
      .withColumn("borough", col("properties.borough"))
      .withColumn("boroughcode", col("properties.boroughCode"))
      .select(col("coordinates"), col("boroughcode"), col("borough"))
      .sort(col("boroughCode").asc())
)
```

In [0]:

```
display(df_geospatial.take(2))
```

coordinates	boroughcode	borough
List(List(-74.01092529296875, 40.68449020385742), List(-74.01193237304688, 40.68388748168945), List(-74.01217651367188, 40.6840934753418), List(-74.00835418701172, 40.6864013671875), List(-74.00816345214844, 40.68617630004883), List(-74.01092529296875, 40.68449020385742))	1	Manhattan
List(List(-74.0050048828125, 40.68760681152344), List(-74.00563049316406, 40.68678283691406), List(-74.0078353881836, 40.68738555908203), List(-74.0074234008789, 40.68820571899414), List(-74.0050048828125, 40.68760681152344))	1	Manhattan

In [0]:

```
# Prepare geospatial datageopandas.GeoDataFrame(
df_geospatial_pandas = df_geospatial.select(col('coordinates'), col('borough')).toPanda
s()
polygons=[]
for index, row in df_geospatial_pandas.iterrows():
  poly_points = Polygon([(pts[0], pts[1]) for pts in row[0]])
  polygons.append(poly_points)
df_geospatial_pandas['Polygons'] = polygons
df_geospatial_pandas.drop('coordinates', inplace=True, axis=1)
df_geospatial_data = geopandas.GeoDataFrame(df_geospatial_pandas)
```

In [0]:

```
df_geospatial_data
```

Out[9]:

	borough	Polygons
0	Manhattan	POLYGON ((-74.01092529296875 40.68449020385742...
1	Manhattan	POLYGON ((-74.0050048828125 40.68760681152344,...
2	Manhattan	POLYGON ((-74.00382232666016 40.68893051147461...
3	Manhattan	POLYGON ((-74.00297546386719 40.69042587280273...
4	Manhattan	POLYGON ((-74.04387664794922 40.69018936157227...
...	...	...
102	Queens	POLYGON ((-73.89144897460938 40.77622222900391...
103	Staten Island	POLYGON ((-74.05050659179688 40.56642150878906...
104	Staten Island	POLYGON ((-74.05313873291016 40.57770156860352...
105	Staten Island	POLYGON ((-74.15945434570312 40.64144897460938...
106	Staten Island	POLYGON ((-74.08221435546875 40.64828109741211...

107 rows × 2 columns

In [0]:

```
def findBorough(longitude, latitude):
  mgdf = df_geospatial_data.apply(lambda x: x['borough'] if x['Polygons'].intersects(Po
int(longitude, latitude)) else None, axis=1)
  idx = mgdf.first_valid_index()
  first_valid_value = mgdf.loc[idx] if idx is not None else None
  return first_valid_value

findBorough = udf(findBorough, StringType())
```

In [0]:

```
sc.broadcast(df_geospatial_data)

df_nyc = (df_nyc
          .withColumn("Pickup_location", findBorough(col('pickup_longitude'), col('pick
up_latitude')))
          .withColumn('Dropoff_location', findBorough(col('dropoff_longitude'), col('dr
opoff_latitude')))
       )
```

In [0]:

```
display(df_nyc)
```

medallion	hack_license	vendor
89D227B655E5C82AECF13C3F540D4CF4	BA96DE419E711691B9445D6A6307C170	C
0BD7C8F5BA12B88E0B67BED28BEA73D8	9FD8F69F0804BDB5549F40E9DA1BE472	C
0BD7C8F5BA12B88E0B67BED28BEA73D8	9FD8F69F0804BDB5549F40E9DA1BE472	C
DFD2202EE08F7A8DC9A57B02ACB81FE2	51EE87E3205C985EF8431D850C786310	C
DFD2202EE08F7A8DC9A57B02ACB81FE2	51EE87E3205C985EF8431D850C786310	C
20D9ECB2CA0767CF7A01564DF2844A3E	598CCE5B9C1918568DEE71F43CF26CD2	C
496644932DF3932605C22C7926FF0FE0	513189AD756FF14FE670D10B92FAF04C	C
0B57B9633A2FECD3D3B1944AFC7471CF	CCD4367B417ED6634D986F573A552A62	C
2C0E91FF20A856C891483ED63589F982	1DA2F6543A62B8ED934771661A9D2FA0	C
2D4B95E2FA7B2E85118EC5CA4570FA58	CD2F522EEE1FF5F5A8D8B679E23576B3	C

In [0]:

```
df_last_nyc = (df_nyc
               .select(
                 col('medallion'),
                 col('hack_license'),
                 col('pickup_datetime'),
                 col('dropoff_datetime'),
                 col('passenger_count'),
                 col('duration'),
                 col('Pickup_location'),
                 col('Dropoff_location'))
              )
```

Queries

Query 1

Utilization per taxi/driver: The sum of idle time aggregated by destination borough

In [0]:

```
window_spec = Window.partitionBy("medallion").orderBy(col("pickup_datetime").asc())

query_1 = (df_last_nyc
       .withColumn("dropoff_lagging", lag('dropoff_datetime').over(window_spec))
       .filter(col("dropoff_lagging").isNotNull())
       )

query_1 = (query_1.withColumn("idle", timeDiff(col('dropoff_lagging'), col('pickup_date
time'))) # timeDiff function returned seconds
         .filter((col('idle') < 4*3600) & (col('idle') > 0))
         .filter(col('Dropoff_location').isNotNull())
        ) # if idle less than 4 hours keep it
```

In [0]:

```
# End of Query 1
display(query_1.groupBy(col('Dropoff_location')).agg(sum(col('idle'))))
```

Dropoff_location	sum(idle)
Queens	8310600.0
Brooklyn	4736700.0
Staten Island	2940.0
Manhattan	8.28003E7
Bronx	617400.0

The total idle time per taxi/driver aggregated by destination borough:
Queens: 8,310,600 seconds
Brooklyn: 4,736,700 seconds
Staten Island: 2940 seconds
Manhattan: 82,800,300 seconds
Bronx: 617,400 seconds

Query 2

The average time it takes for a taxi to find its next fare per destination borough.

In [0]:

```
query_2 = (query_1.groupBy(col("Dropoff_location")).agg(avg(col("idle")))) # returns av
erage time as seconds
display(query_2)
```

Dropoff_location	avg(idle)
Queens	1934.9476135040745
Brooklyn	1822.5086571758368
Staten Island	980.0
Manhattan	1066.3820415732941
Bronx	2166.315789473684

The average time it takes for a taxi to find its next fare per destination borough:
Queens: 1934.95 seconds
Brooklyn: 1822.51 seconds
Staten Island: 980 seconds
Manhattan: 1066.38 seconds
Bronx: 2166.32 seconds

Query 3

The number of trips that started and ended in the same borough

In [0]:

```
query_3 = (df_last_nyc.filter(col("Pickup_location") == col("Dropoff_location")).count
())
query_3
```

```
Out[18]: 85944
```

85,944 trips started and ended in the same borough.

Query 4

The number of trips that started in one borough and ended in another

In [0]:

```
query_4 = (df_last_nyc.filter(col("Pickup_location") != col("Dropoff_location")).count
())
query_4
```

Out[19]: 11431

11,431 trips started in one borough and ended in another.