Final BD Project

December 20, 2021

1 Big Data Final Project Notebook

1.1 Loading SparkSession

[1]:
```
from pyspark.sql import SparkSession

spark = SparkSession\
        .builder\
        .appName("BD Project")\
        .getOrCreate()
```

```
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use
setLogLevel(newLevel).
21/12/20 18:19:18 WARN NativeCodeLoader: Unable to load native-hadoop library
for your platform… using builtin-java classes where applicable
```

1.2 All Imports

[2]:
```
#pip install nbformat
#pip install plotly
#pip install folium
#pip install seaborn

from pyspark.sql.functions import *

#Importing Plotly for Plotting Data
import plotly.graph_objects as go

# Imports for plotting Map
import numpy as np
import seaborn as sns
from folium import plugins
%matplotlib inline
import folium

import matplotlib.pyplot as plt
```

```
plt.style.use('seaborn')
```

[3]:
```
#Loading Rat Sightings CSV Data
df_main = spark.read \
    .format("csv") \
    .option("header", "true") \
    .option("inferSchema","true") \
    .load("project_datasets/Rat_Sightings.csv")

df_main.printSchema()
```

```
[Stage 1:==============================>                          (1 + 1) / 2]
root
 |-- Unique Key: integer (nullable = true)
 |-- Created Date: string (nullable = true)
 |-- Closed Date: string (nullable = true)
 |-- Agency: string (nullable = true)
 |-- Agency Name: string (nullable = true)
 |-- Complaint Type: string (nullable = true)
 |-- Descriptor: string (nullable = true)
 |-- Location Type: string (nullable = true)
 |-- Incident Zip: string (nullable = true)
 |-- Incident Address: string (nullable = true)
 |-- Street Name: string (nullable = true)
 |-- Cross Street 1: string (nullable = true)
 |-- Cross Street 2: string (nullable = true)
 |-- Intersection Street 1: string (nullable = true)
 |-- Intersection Street 2: string (nullable = true)
 |-- Address Type: string (nullable = true)
 |-- City: string (nullable = true)
 |-- Landmark: string (nullable = true)
 |-- Facility Type: string (nullable = true)
 |-- Status: string (nullable = true)
 |-- Due Date: string (nullable = true)
 |-- Resolution Action Updated Date: string (nullable = true)
 |-- Community Board: string (nullable = true)
 |-- Borough: string (nullable = true)
 |-- X Coordinate (State Plane): integer (nullable = true)
 |-- Y Coordinate (State Plane): integer (nullable = true)
 |-- Park Facility Name: string (nullable = true)
 |-- Park Borough: string (nullable = true)
 |-- Vehicle Type: string (nullable = true)
 |-- Taxi Company Borough: string (nullable = true)
 |-- Taxi Pick Up Location: string (nullable = true)
 |-- Bridge Highway Name: string (nullable = true)
 |-- Bridge Highway Direction: string (nullable = true)
```

```
|-- Road Ramp: string (nullable = true)
|-- Bridge Highway Segment: string (nullable = true)
|-- Latitude: double (nullable = true)
|-- Longitude: double (nullable = true)
|-- Location: string (nullable = true)
```

[ ]:

2 Rat Sightings by Borough

[4]:
```
df_location = df_main.select('Unique Key','Incident Zip','City','Address␣
↪Type','Location Type','Borough')
df_borough = df_location.groupBy("Borough").count()
```

[5]:
```
#Filtering Null Rows
df_borough = df_borough.na.drop(subset=["Borough"])
df_borough = df_borough.filter(df_borough.Borough!="Unspecified")
```

[6]:
```
#Collecting data in lists
Borough = df_borough.select(collect_list('Borough')).first()[0]
BoroughCount = df_borough.select(collect_list('count')).first()[0]
```

[7]:
```
fig = go.Figure(
    data=[go.Pie(labels = Borough , values = BoroughCount)],
    layout_title_text="Borough vs Count of  Rat Sightings"
)
fig.show()
```

3 Rat Sightings by Apartment Type

[8]:
```
df_apartment_type = df_location.groupBy("Location Type").count()
```

[9]:
```
ApartmentType = df_apartment_type.select(collect_list('Location Type')).
↪first()[0]
ApartmentTypeCount = df_apartment_type.select(collect_list('count')).first()[0]
```

[10]:
```
fig1 = go.Figure(
    data=[go.Bar(x = ApartmentType , y = ApartmentTypeCount)],
    layout_title_text="Apartment Type vs Count of Rat Sightings"
```

```
)
fig1.show()
```

4 Rat Sightings by City

[11]:
```
city_data = df_main.groupBy("City").count()
city_data = city_data.sort(col('count').desc())
```

[12]:
```
CityData = city_data.select(collect_list('City')).first()[0]
CityCount = city_data.select(collect_list('count')).first()[0]
```

[13]:
```
fig3 = go.Figure(
    data=[go.Bar(x = CityData , y = CityCount)],
    layout_title_text="City vs Count of Rat Sightings"
)
fig3.show()
```

[ ]:
```

```

5 Rat Sightings by Month

[14]:
```
df_month = df_main.select('Unique Key','Created Date')
df_month=df_month.withColumn("Month",df_month["Created Date"].substr(0,2))
```

[15]:
```
monthsDict = { "01" : "January", "02": "February", "03": "March", "04":␣
↪"April", "05": "May", "06": "June", "07": "July",
         "08" : "August", "09": "September", "10": "October", "11":␣
↪"November", "12": "December"}
```

[16]:
```
df_month = df_month.groupBy("Month").count()
df_month = df_month.sort(df_month.Month.asc())
```

[17]:
```
months = df_month.select(collect_list('Month')).first()[0]
sightingsCount = df_month.select(collect_list('count')).first()[0]
months = [monthsDict[x] for x in months]
```

[18]:
```
fig4 = go.Figure([go.Bar(x=months, y=sightingsCount, marker_color='crimson')],
                 layout_title_text="Month vs Count of Rat Sightings")
fig4.update_xaxes(title_text="Months")
fig4.update_yaxes(title_text="No. of Rats Sightings")
fig4.show()
```

6 Rat Sightings by Season

[19]:
```
SeasonDict = {
    "Winter": ["December", "January", "February"],
    "Spring": ["March", "April", "May"],
    "Summer": ["June", "July", "August"],
    "Autumn": ["September", "October", "November"]
}
```

[20]:
```
Seasoncount = []
seasons = []

for season in SeasonDict:
    seasons.append(season)
    count = 0
    for month in SeasonDict[season]:
        idx = months.index(month)
        count += sightingsCount[idx]

    Seasoncount.append(count)
```

[ ]:
```
```

[21]:
```
fig5 = go.Figure([go.Bar(x=seasons, y=Seasoncount, marker_color='blue')],
                 layout_title_text="Seasons vs Count of Rat Sightings")
fig5.update_xaxes(title_text="Seasons")
fig5.update_yaxes(title_text="No. of Rats Sightings")
fig5.show()
```

[22]:
```
fig6 = go.Figure(
    data=[go.Pie(labels = seasons , values = Seasoncount)],
    layout_title_text="Season vs Count of  Rat Sightings"
)
fig6.show()
```

7 Rat Sightings by Year

[23]:
```
df_year = df_main.select('Unique Key','Created Date')
df_year = df_year.withColumn("Year",df_year["Created Date"].substr(0,10).
 ↪substr(7,8))
```

[24]:
```
df_year = df_year.groupBy("Year").count()
df_year = df_year.sort(df_year.Year.asc())
years = df_year.select(collect_list('Year')).first()[0]
sightingsCountPerYear = df_year.select(collect_list('count')).first()[0]
```

[25]:
```
fig7 = go.Figure([go.Scatter(x=years, y=sightingsCountPerYear,␣
 ↪marker_color='green')],
                 layout_title_text="Year vs Count of Rat Sightings")
fig7.update_xaxes(title_text="Year")
fig7.update_yaxes(title_text="No. of Rats Sightings")
fig7.show()
```

8 Rat Sightings Heat Map around NYC Subway Stations

[26]:
```
df_subway = spark.read\
    .option("header","true")\
    .option("inferSchema","true")\
    .option("delimiter",',')\
    .csv('project_datasets/NYC_Transit_Subway_Entrance_And_Exit_Data.csv')

df_subway.printSchema()
```

```
root
 |-- Division: string (nullable = true)
 |-- Line: string (nullable = true)
 |-- Station Name: string (nullable = true)
 |-- Station Latitude: double (nullable = true)
 |-- Station Longitude: double (nullable = true)
 |-- Route1: string (nullable = true)
 |-- Route2: string (nullable = true)
 |-- Route3: string (nullable = true)
 |-- Route4: string (nullable = true)
 |-- Route5: string (nullable = true)
 |-- Route6: string (nullable = true)
 |-- Route7: string (nullable = true)
 |-- Route8: integer (nullable = true)
 |-- Route9: integer (nullable = true)
 |-- Route10: integer (nullable = true)
 |-- Route11: integer (nullable = true)
 |-- Entrance Type: string (nullable = true)
 |-- Entry: string (nullable = true)
 |-- Exit Only: string (nullable = true)
 |-- Vending: string (nullable = true)
 |-- Staffing: string (nullable = true)
 |-- Staff Hours: string (nullable = true)
 |-- ADA: boolean (nullable = true)
 |-- ADA Notes: string (nullable = true)
 |-- Free Crossover: boolean (nullable = true)
 |-- North South Street: string (nullable = true)
 |-- East West Street: string (nullable = true)
```

```
 |-- Corner: string (nullable = true)
 |-- Entrance Latitude: double (nullable = true)
 |-- Entrance Longitude: double (nullable = true)
 |-- Station Location: string (nullable = true)
 |-- Entrance Location: string (nullable = true)
```

[ ]:

[27]:
```
df1_station1=df_subway.withColumn("Station Name",concat(col('Station␣
↪Name'),lit(' '),col('Line')))
df1_station=df1_station1.select('Station Name','Station Location').distinct()
```

[28]:
```
df1_lat_long = df1_station1.select('Entrance Latitude','Entrance Longitude').
↪distinct()
df1_station.orderBy('Station Name').show()
```

```
+--------------------+--------------------+
|        Station Name|    Station Location|
+--------------------+--------------------+
|   103rd St 8 Avenue|(40.796092, -73.9…|
|103rd St Broadway…|(40.799446, -73.9…|
|   103rd St Flushing|(40.749865, -73.8…|
|  103rd St Lexington|(40.7906, -73.947…|
|104th St-102nd St…|(40.695178, -73.8…|
|104th St-Oxford A…|(40.681711, -73.8…|
|  110th St Lexington|(40.79502, -73.94…|
|110th St-Central …|(40.799075, -73.9…|
|111th St Broadway…|(40.697418, -73.8…|
|   111th St Flushing|(40.75173, -73.85…|
|111th St-Greenwoo…|(40.684331, -73.8…|
|   116th St 8 Avenue|(40.805085, -73.9…|
|      116th St Lenox|(40.802098, -73.9…|
|  116th St Lexington|(40.798629, -73.9…|
|116th St-Columbia…|(40.807722, -73.9…|
|121st St Broadway…|(40.700492, -73.8…|
|   125th St 8 Avenue|(40.811109, -73.9…|
|125th St Broadway…|(40.815581, -73.9…|
|      125th St Lenox|(40.807754, -73.9…|
|  125th St Lexington|(40.804138, -73.9…|
+--------------------+--------------------+
only showing top 20 rows
```

[29]:
```
from pyspark.sql import functions as F

df3=df_main.where(df_main.Location!="").select('Unique Key','Location')
```

```
df3.printSchema()
```

```
root
 |-- Unique Key: integer (nullable = true)
 |-- Location: string (nullable = true)
```

[30]:
```
df4=df3.withColumn('Location',regexp_replace(col("Location"), "\(*.\)", ""))
df5=df4.withColumn('Location',regexp_replace(col("Location"), "\(", ""))

rat_sightings = df5.select('Unique Key','Location').withColumn('Location',␣
 ↪split(col('Location'),',').cast('array<double>'))

rat_sightings.cache()
```

[30]:
```
DataFrame[Unique Key: int, Location: array<double>]
```

[31]:
```
df1_pre=df1_station.withColumn('Station Location',regexp_replace(col("Station␣
 ↪Location"), "\(*.\)", ""))
df1_post=df1_pre.withColumn('Station Location',regexp_replace(col("Station␣
 ↪Location"), "\(", ""))

subway_station = df1_post.select('Station Name','Station Location').
 ↪withColumn('Station Location', split(col('Station Location'),',').
 ↪cast('array<double>'))

subway_station.show(5,False)
subway_station.printSchema()
subway_station.cache()
```

```
+--------------------------------+----------------------+
|Station Name                    |Station Location      |
+--------------------------------+----------------------+
|Classon Av Crosstown            |[40.688873, -73.96]   |
|East 143rd St-St Mary's St Pelham|[40.808719, -73.90765]|
|York St 6 Avenue                |[40.699743, -73.98688]|
|34th St Broadway                |[40.749567, -73.9879] |
|Mets - Willets Point Flushing   |[40.754622, -73.84562]|
+--------------------------------+----------------------+
only showing top 5 rows

root
 |-- Station Name: string (nullable = true)
 |-- Station Location: array (nullable = true)
 |    |-- element: double (containsNull = true)
```

[31]:
```
DataFrame[Station Name: string, Station Location: array<double>]
```

[32]:
```
#generating folium map
map_subway = folium.Map([40.7, -73.9], zoom_start=11)
map_rats = folium.Map([40.7, -73.9], zoom_start=11)
```

[ ]:
```
```

[33]:
```
df_2=df_main.where(df_main.Location!="").select('Latitude','Longitude')
df2_lat=df_2.withColumn('Latitude',df_2['Latitude'].cast("float").
 ↪alias('Latitude'))
df2_long=df2_lat.withColumn('Longitude',df2_lat['Longitude'].cast("float").
 ↪alias('Longitude'))
df_loc=df_main.where(df_main.Location!="").select('Location')
```

8.1 Rat Sightings Heat Map

[34]:
```
df1_latlong_temp1=df1_lat_long.withColumn('Entrance␣
 ↪Latitude',df1_lat_long['Entrance Latitude'].cast("float").alias('Entrance␣
 ↪Latitude'))
df1_latlong_temp2=df1_latlong_temp1.withColumn('Entrance␣
 ↪Longitude',df1_latlong_temp1['Entrance Longitude'].cast("float").
 ↪alias('Entrance Longitude'))
locs=np.array(df1_latlong_temp2.select('Entrance Latitude','Entrance␣
 ↪Longitude').collect())
rats=np.array(df2_long.select('Latitude','Longitude').collect())


plugins.MarkerCluster(locs).add_to(map_subway)
map_subway.add_children(plugins.HeatMap(rats, radius=15))
map_rats.add_children(plugins.HeatMap(rats, radius=15))
display(map_rats)
```

```
/opt/conda/envs/bigdata/lib/python3.7/site-packages/ipykernel_launcher.py:8:
FutureWarning:

Method `add_children` is deprecated. Please use `add_child` instead.

/opt/conda/envs/bigdata/lib/python3.7/site-packages/ipykernel_launcher.py:9:
FutureWarning:

Method `add_children` is deprecated. Please use `add_child` instead.


<folium.folium.Map at 0x7f651668c710>
```

[35]:
```
display(map_subway)
```

```
<folium.folium.Map at 0x7f651668c790>
```

[ ]:
```

```

9 Rat Sightings Heat Map around NYC Litter Baskets

[36]:
```
df_litter=spark.read\
    .option("header","true")\
    .option("inferSchema","true")\
    .option("delimiter",',')\
    .csv('project_datasets/DSNY_Litter_Basket_Inventory.csv')
df_litter.show(5,False)
df_litter.printSchema()
```

```
+--------+----------+---------+-----+-------------------------+---------+------
-+-----------------+----------------+------------------+-------------------+--
---------+-----------+---------------------------------------------+
|BASKETID|BASKETTYPE|DIRECTION|FID  |LOCATION_DESCRIPTION     |OWNERTYPE|SECTIO
N|STATEPLANE_LABELX|STATEPLANE_LABELY|STATEPLANE_SNAPPEDX|STATEPLANE_SNAPPEDY|ST
REETNAME1|STREETNAME2|point                                        |
+--------+----------+---------+-----+-------------------------+---------+------
-+-----------------+----------------+------------------+-------------------+--
---------+-----------+---------------------------------------------+
|30720110|S         |W        |12267|W corner of 4 AV and 36 ST|D        |BKS072
|982933.32545614  |177651.46200422  |982975.08455898    |177673.7063823     |4
AV       |36 ST      |POINT (-74.00474671499995 40.65429701700003)|
|40780011|H         |null     |12054|null                     |P        |QE078
|1041044.98088615 |228077.28044371  |null               |null
|null       |null       |POINT (-73.79488819299996 40.79252126700004)|
|40830037|R         |null     |11967|null                     |D        |QE083
|1034275.14895339 |206947.41522321  |null               |null
|null       |null       |POINT (-73.81949439499994 40.73456626300003)|
|20710099|H         |null     |9752 |null                     |P        |BX071
|1010945.30420898 |255598.10040197  |null               |null
|null       |null       |POINT (-73.90348256799996 40.86820017500003)|
|21210022|S         |null     |5830 |Unknown corner on BURKE AV|D        |BX121
|1020926.56639865 |256725.31632339  |1020957.31568098   |256764.63481429
|null       |BURKE AV   |POINT (-73.86738843799998 40.87125820800003)|
+--------+----------+---------+-----+-------------------------+---------+------
-+-----------------+----------------+------------------+-------------------+--
---------+-----------+---------------------------------------------+
only showing top 5 rows

root
```

```
 |-- BASKETID: integer (nullable = true)
 |-- BASKETTYPE: string (nullable = true)
 |-- DIRECTION: string (nullable = true)
 |-- FID: integer (nullable = true)
 |-- LOCATION_DESCRIPTION: string (nullable = true)
 |-- OWNERTYPE: string (nullable = true)
 |-- SECTION: string (nullable = true)
 |-- STATEPLANE_LABELX: double (nullable = true)
 |-- STATEPLANE_LABELY: double (nullable = true)
 |-- STATEPLANE_SNAPPEDX: double (nullable = true)
 |-- STATEPLANE_SNAPPEDY: double (nullable = true)
 |-- STREETNAME1: string (nullable = true)
 |-- STREETNAME2: string (nullable = true)
 |-- point: string (nullable = true)
```

[37]:
```
#Preprocessing of Data
df_bin = df_litter.withColumn('point',regexp_replace(col("point"), "\(*.\)",␣
 ↪""))
df_refined = df_bin.withColumn('point',regexp_replace(col("point"), "\(", ""))
df_refined1 = df_refined.withColumn('point', regexp_replace('point', 'POINT ',␣
 ↪''))
df_final = df_refined1.select("point")

df_final1 = df_final.select('point').withColumn('point', split(col('point'),'␣
 ↪'))
df_final2 = df_final1.select('point')


df_final3 = df_final.withColumn('Latitude', split(df_final['point'], ' ').
 ↪getItem(0)) \
       .withColumn('Longitude', split(df_final['point'], ' ').getItem(1))
df_final3.printSchema()
df_final3.show(5,False)
```

```
root
 |-- point: string (nullable = true)
 |-- Latitude: string (nullable = true)
 |-- Longitude: string (nullable = true)

+-----------------------------------+------------------+----------------+
|point                              |Latitude          |Longitude       |
+-----------------------------------+------------------+----------------+
|-74.00474671499995 40.6542970170000|-74.00474671499995|40.6542970170000|
|-73.79488819299996 40.7925212670000|-73.79488819299996|40.7925212670000|
|-73.81949439499994 40.7345662630000|-73.81949439499994|40.7345662630000|
|-73.90348256799996 40.8682001750000|-73.90348256799996|40.8682001750000|
```

```
|-73.86738843799998 40.8712582080000|-73.86738843799998|40.8712582080000|
+-----------------------------------+------------------+---------------+
only showing top 5 rows
```

[38]:
```
df_final4 = df_final3.select('Latitude','Longitude')

df_final5 = df_final4.withColumn('Longitude',df_final4['Longitude'].
 ↪cast("float").alias('Longitude'))
df_final6 = df_final5.withColumn('Latitude',df_final5['Latitude'].cast("float").
 ↪alias('Latitude'))
bin_points = np.array(df_final6.select('Longitude','Latitude').collect())
```

[ ]:
```

```

[39]:
```
litter_map = folium.Map([40.7, -73.9], zoom_start=11)
plugins.MarkerCluster(bin_points).add_to(litter_map)

litter_map.add_children(plugins.HeatMap(rats, radius=15))
map_rats.add_children(plugins.HeatMap(rats, radius=15))

display(litter_map)
```

```
/opt/conda/envs/bigdata/lib/python3.7/site-packages/ipykernel_launcher.py:4:
FutureWarning:

Method `add_children` is deprecated. Please use `add_child` instead.

/opt/conda/envs/bigdata/lib/python3.7/site-packages/ipykernel_launcher.py:5:
FutureWarning:

Method `add_children` is deprecated. Please use `add_child` instead.


<folium.folium.Map at 0x7f650dacd210>
```

[ ]:
```

```

10 Average Response Time of Department of Health and Mental Hygiene

[40]:
```
df_response = df_main.na.drop(subset=["Closed Date"])
df_response = df_response.select('Unique Key','Created Date','Closed Date')
```

[41]:
```
df_temp1 = df_response.withColumn('Created_date',to_timestamp('Created Date',␣
↪'MM/dd/yyyy hh:mm:ss a'))
df_temp2 = df_temp1.withColumn('Closed_date',to_timestamp('Closed Date', 'MM/dd/
↪yyyy hh:mm:ss a'))
```

[42]:
```
df_temp3 = df_temp2.where((unix_timestamp(col("Closed_date")) -␣
↪unix_timestamp(col("Created_date")))>0)\
                    .withColumn("difference",␣
↪unix_timestamp(col("Closed_date"))- unix_timestamp(col("Created_date")))

df_temp3 = df_temp3.groupBy().avg("difference").
↪withColumnRenamed("avg(difference)", "Avg_RT_in_sec")

df_response_final = df_temp3.withColumn("Avg_RT_in_min",␣
↪round(col("Avg_RT_in_sec")/60))\
                     .withColumn("Avg_RT_in_hour", round(col("Avg_RT_in_min")/
↪60))

df_response_final.show()
```

```
[Stage 115:============================>                            (1 + 1) / 2]

+------------------+-------------+--------------+
|     Avg_RT_in_sec|Avg_RT_in_min|Avg_RT_in_hour|
+------------------+-------------+--------------+
|1720536.9605394606|      28676.0|         478.0|
+------------------+-------------+--------------+
```

[43]:
```
df_temp2 = df_temp2.withColumn("Year", year("Created_date"))

df_temp3 = df_temp2.where((unix_timestamp(col("Closed_date")) -␣
↪unix_timestamp(col("Created_date")))>0)\
                    .withColumn("difference",␣
↪unix_timestamp(col("Closed_date"))- unix_timestamp(col("Created_date")))

df_temp3 = df_temp3.groupBy('Year').avg("difference").
↪withColumnRenamed("avg(difference)", "Avg_RT_in_sec")

df_temp4 = df_temp3.withColumn("Avg_RT_in_min", round(col("Avg_RT_in_sec")/60)).
↪withColumn("Avg_RT_in_hour", round(col("Avg_RT_in_min")/60))
df_response_final_year = df_temp4.sort(df_temp4.Year.asc())
df_response_final_year.show()
```

```
[Stage 118:>                                                            (0 + 2) / 2]
```

```
+----+------------------+------------+--------------+
|Year|     Avg_RT_in_sec|Avg_RT_in_min|Avg_RT_in_hour|
+----+------------------+------------+--------------+
|2010|2911244.5276630884|     48521.0|         809.0|
|2011|1553890.6670320034|     25898.0|         432.0|
|2012|1712044.8312883435|     28534.0|         476.0|
|2013| 1091473.111942959|     18191.0|         303.0|
|2014|1147740.6896406808|     19129.0|         319.0|
|2015|1271387.8375653827|     21190.0|         353.0|
|2016|1069469.4299427564|     17824.0|         297.0|
|2017|1125708.6382476583|     18762.0|         313.0|
|2018| 1008897.302240919|     16815.0|         280.0|
|2019|3644794.1353983497|     60747.0|        1012.0|
|2020| 3729748.577942736|     62162.0|        1036.0|
|2021|1905050.0201822917|     31751.0|         529.0|
+----+------------------+------------+--------------+
```

[44]:
```
Years = df_response_final_year.select(collect_list('Year')).first()[0]
RT_HR = df_response_final_year.select(collect_list('Avg_RT_in_hour')).first()[0]
```

[45]:
```
fig8 = go.Figure([go.Scatter(x=Years, y=RT_HR, marker_color='red')],
                 layout_title_text="Year vs Avg. Response Time in Hour")
fig8.update_xaxes(title_text="Year")
fig8.update_yaxes(title_text="Time in Hour")
fig8.show()
```

[ ]:
```

```