Project#4: Notebook Exploratory Data Analysis using Databricks

(https://databricks.com)

# Project#4: Exploratory Data Analysis using Databricks

## (Total Points: 10)

## Instruction:

**Perform exploratory data analysis (EDA) to gain insights using Community Edition of Databricks.**

## Use the Notebook "Project#3: Exploratory Data Analysis using Databricks" DBC file for the assigment

**“Databricks Overview Labs CS 5165 | 6065 v1_1.dbc” file has been provided for review. It is in Canvas.**

## Homework Tasks:

**1. Calculate the Total count of Crimes for each of the 6 United States cities listed below using 2016 crime data in dbfs:/mnt/training/crime-data-2016**

**2. Provide Total count of different Types of Crimes for each of 6 United States cities listed below using crime-data-2016**

**3. Calculate the total Robbery count for each of the 3 United States cities listed below using crime-data-2016**

**4. Find the months with the Highest and Lowest Robbery counts for each of the 3 United States cities listed below using crime-data-2016**

**5. Combine all three cities robberies-per-month views into one and Find the month with the Highest and Lowest combined Robbery counts using crime-data-2016**

**6. Plot the robberies per month for each of the three cities, producing one Graph using the contents of combinedRobberiesByMonthDF**

**7. Find the "per capita robbery rates" using ther Hint below, and plot graph as above for the per capita robbery rates per month for each of the three cities, producing one Graph using the contents of robberyRatesByCityDF**

**8. Find the monthly HOMICIDE counts for each of the 2 United States cities listed below using crime-data-2016, and Combine both cities HOMICIDE-per-month views into one and Find the month with the Highest and Lowest combined HOMICIDE-per-month counts. See Question 6.**

**9. Find the "per capita HOMICIDE rates" using the Hint in Qn 7, and plot graph as above for the per capita HOMICIDE rates per month for each of the two cities, producing one Graph using the contents of HOMICIDERatesByCityDF**

**10. A stretch goal to address a Data Science question on predicting crimes for a city as a time series model. How would you predict future values for monthly Robbery count rate for Log Angeles using the historical values from crime-data-2016. Data Science is a multi-year discipline. I am not expecting anything fancy. There is no wrong answer. I do expect students to explore and give their best shot.**

Submission: 1. Export the completed "Project#4: Exploratory Data Analysis using Databricks.dbc" as DBC Archive file with all the information in Blackboard 2. Your spark code/command and results included

## Exploratory Data Analysis using Databricks

Perform exploratory data analysis (EDA) to gain insights from a data lake.

# Instructions

In `dbfs:/mnt/training/crime-data-2016`, there are a number of Parquet files containing 2016 crime data from seven United States cities:

- New York
- Los Angeles
- Chicago
- Philadelphia
- Dallas
- Boston

The data is cleaned up a little, but has not been normalized. Each city reports crime data slightly differently, so examine the data for each city to determine how to query it properly.

## Getting Started

Run the following cell to configure our "classroom."

```
%run ./Includes/Classroom-Setup
```

Initialized classroom variables & functions...

Datasets are already mounted to **/mnt/training** from **s3a://databricks-corp-training/common**

Imported Test Library...

Created user-specific database

Using the database **ghantasrisaiganesh_gmail_com_db**.

All done!

```
print("username: " + username)
print("userhome: " + userhome)
```

```
username: ghantasrisaiganesh@gmail.com
userhome: dbfs:/user/ghantasrisaiganesh@gmail.com
```

# Question 1: Calculate the Total count of Crimes for each of the 6 United States cities listed below using 2016 crime data in dbfs:/mnt/training/crime-data-2016:

- New York
- Los Angeles
- Chicago
- Philadelphia
- Dallas
- Boston

**Hint:**

Start by creating DataFrames for Los Angeles, Philadelphia, and Dallas data.

Use `spark.read.parquet` to create named DataFrames for the files you choose.

To read in the parquet file, use
`crimeDataNewYorkDF = spark.read.parquet("/mnt/training/crime-data-2016/C`

Use the following view names:

| City | DataFrame Name | Path to DBFS file |
|---|---|---|
| Los Angeles | `crimeDataLosAngelesDF` | `dbfs:/mnt/training/crime-dat` |
| Philadelphia | `crimeDataPhiladelphiaDF` | `dbfs:/mnt/training/crime-dat` |
| Dallas | `crimeDataDallasDF` | `dbfs:/mnt/training/crime-dat` |
| etc... | | |

**Hint:** Los Angeles

```
# TODO

crimeDataNewYorkDF = spark.read.parquet("/mnt/training/crime-data-
2016/Crime-Data-New-York-2016.parquet")
print("Counts in Newyork city:",crimeDataNewYorkDF.count())
crimeDataLosAngelesDF = spark.read.parquet("dbfs:/mnt/training/crime-data-
2016/Crime-Data-Los-Angeles-2016.parquet")
print("Counts in LosAngles city:",crimeDataLosAngelesDF.count())
crimeDataChicagoDF = spark.read.parquet("/mnt/training/crime-data-
2016/Crime-Data-Chicago-2016.parquet")
print("Counts in Chicago city:",crimeDataChicagoDF.count())
crimeDataPhiladelphiaDF = spark.read.parquet("/mnt/training/crime-data-
2016/Crime-Data-Philadelphia-2016.parquet")
print("Counts in Philadelphia city:",crimeDataPhiladelphiaDF.count())
crimeDataDallasDF = spark.read.parquet("/mnt/training/crime-data-2016/Crime-
Data-Dallas-2016.parquet")
print("Counts in Dallas city:",crimeDataDallasDF.count())
crimeDataBostonDF = spark.read.parquet("/mnt/training/crime-data-2016/Crime-
Data-Boston-2016.parquet")
print("Counts in Boston city:",crimeDataBostonDF.count())
```

```
Counts in Newyork city: 468241
Counts in LosAngles city: 217945
Counts in Chicago city: 267872
Counts in Philadelphia city: 168664
Counts in Dallas city: 99642
Counts in Boston city: 218610
```

# Question 2: Provide Total count of different Types of Crimes for each of 6 United States cities listed below using crime-data-2016:

- New York
- Los Angeles
- Chicago
- Philadelphia
- Dallas
- Boston

## Notice in the Dataframes:

- The `crimeDataNewYorkDF` and `crimeDataBostonDF` DataFrames use different names for the columns.
- The data itself is formatted differently and different names are used for similar concepts.

This is common in a Data Lake. Often files are added to a Data Lake by different groups at different times. The advantage of this strategy is that anyone can contribute information to the Data Lake and that Data Lakes scale to store arbitrarily large and diverse data. The tradeoff for this ease in storing data is that it doesn't have the rigid structure of a traditional relational data model, so the person querying the Data Lake will need to normalize data before extracting useful insights.

# Same Type of Data, Different Structure

**Hint:** Los Angeles

```
# TODO

print("Total Different types of crime in New York
are:",crimeDataNewYorkDF.select("keyCode").distinct().count())
print("Total Different types of crime in Los Angeles
are:",crimeDataLosAngelesDF.select("crimeCode").distinct().count())
print("Total Different types of crime in Chicago
are:",crimeDataChicagoDF.select("primaryType").distinct().count())
print("Total Different types of crime in Philadelphia
are:",crimeDataPhiladelphiaDF.select("ucr_general").distinct().count())
print("Total Different types of crime in Dallas
are:",crimeDataDallasDF.select("typeOfIncident").distinct().count())
print("Total Different types of crime in Boston
are:",crimeDataBostonDF.select("OFFENSE_CODE").distinct().count())
```

```
Total Different types of crime in New York are: 67
Total Different types of crime in Los Angeles are: 128
Total Different types of crime in Chicago are: 33
Total Different types of crime in Philadelphia are: 26
Total Different types of crime in Dallas are: 370
Total Different types of crime in Boston are: 221
```

# Question 3: Calculate the total Robbery count for each of the 3 United States cities listed below using crime-data-2016:

- Los Angeles
- Philadelphia
- Dallas

**Hint:** For each table, examine the data to figure out how to extract *robbery* statistics.

Each city uses different values to indicate robbery. Commonly used terminology is "larceny", "burglary" or "robbery." These challenges are common in data lakes. To simplify things, restrict yourself to only the word "robbery" (and not attempted-roberty, larceny, or burglary).

Explore the data for the three cities until you understand how each city records robbery information. If you don't want to worry about upper- or lower-case, remember to use the DataFrame `lower()` method to converts column values to lowercase.

Create a DataFrame containing only the robbery-related rows, as shown in the table below.

**Hint:** For each table, focus your efforts on the column listed below.

**Hint:** Los Angeles

```
from pyspark.sql.functions import col, lower
robberyLosAngelesDF =
crimeDataLosAngelesDF.select(lower(col("crimeCodeDescription")))
robberyPhiladelphiaDF =
crimeDataPhiladelphiaDF.select(lower(col("ucr_general_description")))
robberyDallasDF  = crimeDataDallasDF.select(lower(col("typeOfIncident")))
print("Total Robbery Count for Los Angeles
is:",robberyLosAngelesDF.filter("lower(crimeCodeDescription) like
'%robbery%'").count())
print("Total Robbery Count for Philadelphia
is:",robberyPhiladelphiaDF.filter("lower(ucr_general_description) like
'%robbery%'").count())
print("Total Robbery Count for Dallas
is:",robberyDallasDF.filter("lower(typeOfIncident) like
'%robbery%'").count())
```

```
Total Robbery Count for Los Angeles is: 10275
Total Robbery Count for Philadelphia is: 6149
Total Robbery Count for Dallas is: 6852
```

# Question 4: Find the months with the Highest and Lowest Robbery counts for each of the 3 United States cities listed below using crime-data-2016:

- Los Angeles
- Philadelphia
- Dallas

**Hint:** Now that you have DataFrames of only the robberies in each city, create DataFrames for each city summarizing the number of robberies in each month.

Your DataFrames must contain two columns:

- `month`: The month number (e.g., 1 for January, 2 for February, etc.).
- `robberies`: The total number of robberies in the month.

Use the following DataFrame names and date columns:

| City | DataFrame Name | Date Column |
|---|---|---|
| Los Angeles | `robberiesByMonthLosAngelesDF` | `timeOccurred` |
| Philadelphia | `robberiesByMonthPhiladelphiaDF` | `dispatch_date_time` |

**Hint:** Los Angeles

```
# TODO

from pyspark.sql.functions import lower, col, split
import pyspark.sql
LosAngelsData =
crimeDataLosAngelesDF.withColumn("monthOccurred",split(col("timeOccurred"),"
-").getItem(1)).withColumn("crime",lower(col("crimeCodeDescription")))
robberiesByMonthLosAngelesDF = LosAngelsData.select("monthOccurred","crime")
robberiesByMonthLosAngelesDF = robberiesByMonthLosAngelesDF.filter("crime
like '%robbery%'")
highest_robbery_count_LA =
robberiesByMonthLosAngelesDF.groupBy("monthOccurred").count().sort('count',a
scending=False).withColumnRenamed("count","Highest Robbery Count in
LosAngeles").take(1)
display(highest_robbery_count_LA)
lowest_robbery_count_LA =
robberiesByMonthLosAngelesDF.groupBy("monthOccurred").count().sort('count',a
scending=True).withColumnRenamed("count"," Lowest robbery count in
LosAngels").take(1)
display(lowest_robbery_count_LA)


Philadephiadata =
crimeDataPhiladelphiaDF.withColumn("monthOccurred",split(col("dispatch_date"
),"-").getItem(1)).withColumn("crime",lower(col("ucr_general_description")))
robberiesByMonthPhiladelphiaDF =
Philadephiadata.select("monthOccurred","crime")
robberiesByMonthPhiladelphiaDF =
robberiesByMonthPhiladelphiaDF.filter("crime like '%robbery%'")
highest_robbery_count_PH =
robberiesByMonthPhiladelphiaDF.groupBy("monthOccurred").count().sort('count'
,ascending=False).withColumnRenamed("count"," Highest robbery Count in
Philadelphia ").take(1)
display(highest_robbery_count_PH)
lowest_robbery_count_PH =
robberiesByMonthPhiladelphiaDF.groupBy("monthOccurred").count().sort('count'
,ascending=True).withColumnRenamed("count"," Lowest robbery Count in
Philadelphia").take(1)
display(lowest_robbery_count_PH)


Dallasdata =
crimeDataDallasDF.withColumn("monthOccurred",split(col("callReceivedDateTime
"),"/").getItem(0)).withColumn("crime",lower(col("typeOfIncident")))
robberiesByMonthDallasDF = Dallasdata.select("monthOccurred","crime")
robberiesByMonthDallasDF = robberiesByMonthDallasDF.filter("crime like
'%robbery%'")
highest_robbery_count_Dallas =
robberiesByMonthDallasDF.groupBy("monthOccurred").count().sort('count',ascen
ding=False).withColumnRenamed("count","Highest Robbery Count in Dallas
```

```
").take(1)
display(highest_robbery_count_Dallas)
lowest_robbery_count_Dallas =
robberiesByMonthDallasDF.groupBy("monthOccurred").count().sort('count',ascen
ding=True).withColumnRenamed("count","Lowest Crime Count in Dallas
is").take(1)
display(lowest_robbery_count_Dallas)
```

Table

| | monthOccurred | Highest Robbery Count in LosAngeles |
|---|---|---|
| 1 | 12 | 958 |

1 row

Table

| | monthOccurred | Lowest robbery count in LosAngels |
|---|---|---|
| 1 | 02 | 757 |

1 row

Table

| | monthOccurred | Highest robbery Count in Philadelphia |
|---|---|---|
| 1 | 10 | 570 |

1 row

Table

| | monthOccurred | Lowest robbery Count in Philadelphia |
|---|---|---|
| 1 | 02 | 412 |

1 row

Table

| | monthOccurred | Highest Robbery Count in Dallas |
|---|---|---|
| 1 | 01 | 743 |

1 row

Table

| | monthOccurred | Lowest Crime Count in Dallas is |
|---|---|---|
| 1 | 03 | 410 |

1 row

# Question 5: Combine all three cities robberies-per-month views into one and Find the month with the Highest and Lowest combined Robbery counts using crime-data-2016:

- Los Angeles
- Philadelphia
- Dallas

Create another DataFrame called `combinedRobberiesByMonthDF`, that combines all three robberies-per-month views into one. In creating this view, add a new column called `city`, that identifies the city associated with each row. The final

**Hint:** Combined 3 Cities

```
# TODO

from pyspark.sql.functions import *
robbery_per_month_dallas_city =
robberiesByMonthDallasDF.groupBy("monthOccurred").count().withColumn("city",
lit("Dallas"))
robbery_per_month_losangels_city =
robberiesByMonthLosAngelesDF.groupBy("monthOccurred").count().withColumn("ci
ty",lit("LosAngeles"))
robbery_per_month_philadelphia_city =
robberiesByMonthPhiladelphiaDF.groupBy("monthOccurred").count().withColumn("
city",lit("Philadelphia"))
combinedRobberiesByMonthDF =
robbery_per_month_dallas_city.union(robbery_per_month_losangels_city).union(
robbery_per_month_philadelphia_city).withColumnRenamed("count","robbery").so
rt("monthOccurred",ascending=True)
highest_robbery =
combinedRobberiesByMonthDF.select("monthOccurred","robbery","city").groupBy(
"monthOccurred").sum("robbery").withColumnRenamed("sum(robbery)","robbery").
sort("robbery",ascending=False).take(1)
display(highest_robbery)
```

Table

| | monthOccurred | robbery | |
|---|---|---|---|
| 1 | 12 | 2175 | |

1 row

```
#Lowest Combined Count for cities
lowest_robbery_for_a_city =
combinedRobberiesByMonthDF.select("monthOccurred","robbery","city").groupBy(
"monthOccurred").sum("robbery").withColumnRenamed("sum(robbery)","robbery").
sort("robbery",ascending=True).take(1)
display(lowest_robbery_for_a_city)
```

Table

| | monthOccurred | robbery |
|---|---|---|
| 1 | 02 | 1608 |

1 row

```
#The highest robbery count present in the city out of given three cities.
highest_robbery_for_a_city =
combinedRobberiesByMonthDF.sort("robbery",ascending=False).take(1)
display(highest_robbery_for_a_city)
```

Table

| | monthOccurred | robbery | city |
|---|---|---|---|
| 1 | 12 | 958 | LosAngeles |

1 row

```
#The lowest robbery count city out of three cities.
lowest_robbery_city_out_of_3 =
combinedRobberiesByMonthDF.sort("robbery",ascending=True).take(1)
display(lowest_robbery_city_out_of_3)
```

Table

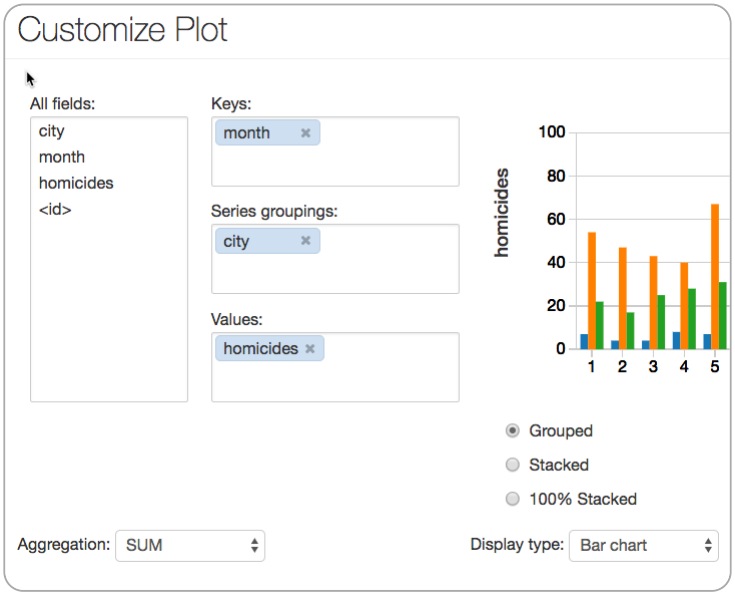
| | monthOccurred | robbery | city |
|---|---|---|---|
| 1 | 03 | 410 | Dallas |

1 row

# Question 6: Plot the robberies per month for each of the three cities, producing one Graph using the contents of `combinedRobberiesByMonthDF`

Adjust the plot options to configure the plot properly, as shown below:

When you first run the cell, you'll get an HTML table as the result. To configure the plot:




1. Click the graph button.
2. If the plot doesn't look correct, click the **Plot Options** button.
3. Configure the plot similar to the following example.

**Hint:** Order your results by `month`, then `city`.

2

```
# TODO

#Plot the graph for the robberies per month for each of the three cities
display(combinedRobberiesByMonthDF)
```

Table

| | monthOccurred | robbery | city |
|---|---|---|---|
| 1 | 01 | 524 | Philadelphia |
| 2 | 01 | 835 | LosAngeles |

| | | | |
|---|---|---|---|
| 3 | 01 | 743 | Dallas |
| 4 | 02 | 757 | LosAngeles |
| 5 | 02 | 412 | Philadelphia |
| 6 | 02 | 439 | Dallas |

36 rows

# Question 7: Find the "per capita robbery rates" using ther Hint below, and plot graph as above for the per capita robbery rates per month for each of the three cities, producing one Graph using the contents of `robberyRatesByCityDF`:

- Los Angeles
- Philadelphia
- Dallas

While the above graph is interesting, it's flawed: it's comparing the raw numbers of robberies, not the per capita robbery rates.

The DataFrame (already created) called `cityDataDF` (dbfs:/mnt/training/City-Data.parquet) contains, among other data, estimated 2016 population values for all United States cities with populations of at least 100,000. (The data is from Wikipedia (https://en.wikipedia.org/wiki/List_of_United_States_cities_by_population).)

- Use the population values in that table to normalize the robberies so they represent per-capita values (total robberies divided by population).
- Save your results in a DataFrame called `robberyRatesByCityDF`.
- The robbery rate value must be stored in a new column, `robberyRate`.

Next, graph the results, as above.

```
# TODO

#Plot the Graph for the "per capita robbery rates" per month for each of the three cities
cityDataDF = spark.read.parquet("/mnt/training/City-Data.parquet")
population_LosAngeles = cityDataDF.filter("city = 'Los Angeles'").select("estPopulation2016").collect()[0]["estPopulation2016"]
population_Philadelphia = cityDataDF.filter("city = 'Philadelphia'").select("estPopulation2016").collect()[0]["estPopulation2016"]
population_Dallas = cityDataDF.filter("city = 'Dallas'").select("estPopulation2016").collect()[0]["estPopulation2016"]
robbery_rates_for_LA = combinedRobberiesByMonthDF.withColumn("robberyRate", (combinedRobberiesByMonthDF.robbery)/population_LosAngeles).where("city = 'LosAngeles'")
robbery_rates_for_philadelphia = combinedRobberiesByMonthDF.withColumn("robberyRate", (combinedRobberiesByMonthDF.robbery)/population_Philadelphia).where("city = 'Philadelphia'")
robbery_rates_for_Dallas = combinedRobberiesByMonthDF.withColumn("robberyRate", (combinedRobberiesByMonthDF.robbery)/population_Dallas).where("city = 'Dallas'")
robberyRatesByCityDF = robbery_rates_for_LA.union(robbery_rates_for_philadelphia).union(robbery_rates_for_Dallas).sort("monthOccurred",ascending=True)
display(robberyRatesByCityDF)
```

Table

| | monthOccurred | robbery | city | robberyRate |
|---|---|---|---|---|
| 1 | 01 | 524 | Philadelphia | 0.0003342109559964079 |
| 2 | 01 | 835 | LosAngeles | 0.0002099930538824572 |
| 3 | 01 | 743 | Dallas | 0.0005637632983263893 |
| 4 | 02 | 757 | LosAngeles | 0.00019037693627427558 |
| 5 | 02 | 412 | Philadelphia | 0.00026277655318801535 |
| 6 | 02 | 439 | Dallas | 0.00033309836872851266 |

36 rows

```
display(robberyRatesByCityDF)
```

Table

| | monthOccurred | robbery | city | robberyRate |
|---|---|---|---|---|
| 1 | 01 | 524 | Philadelphia | 0.0003342109559964079 |
| 2 | 01 | 835 | LosAngeles | 0.0002099930538824572 |

| | | | | |
|---|---|---|---|---|
| 3 | 01 | 743 | Dallas | 0.0005637632983263893 |
| 4 | 02 | 757 | LosAngeles | 0.00019037693627427558 |
| 5 | 02 | 412 | Philadelphia | 0.00026277655318801535 |
| 6 | 02 | 439 | Dallas | 0.00033309836872851266 |

36 rows

# Question 8: Find the monthly HOMICIDE counts for each of the 2 United States cities listed below using crime-data-2016, and Combine both cities HOMICIDE-per-month views into one and Find the month with the Highest and Lowest combined HOMICIDE-per-month counts. See Question 6.

- New York
- Boston

**Hint:** Same Type of Data, Different Structure In this section, we examine crime data to determine how to extract homicide statistics. Each city may use different field names and values to indicate homicides, dates, etc. For example:

- Some cities use the value "HOMICIDE", "CRIMINAL HOMICIDE" or "MURDER".
- In the New York data, the column is named `offenseDescription` while in the Boston data, the column is named `OFFENSE_CODE_GROUP`.
- In the New York data, the date of the event is in the `reportDate`, while in the Boston data, there is a single column named `MONTH`.

To get started, create a temporary view containing only the homicide-related rows. At the same time, normalize the data structure of each table so all the columns (and their values) line up with each other. In the case of New York and Boston, here are the unique characteristics of each data set:

```
crimeDataNYDF =
crimeDataNewYorkDF.select(lower(col("offenseDescription")),month("reportDate
")).withColumnRenamed("lower(offenseDescription)","offenseDescription").with
ColumnRenamed("month(reportDate)","month").filter('offenseDescription like
"%murder%" or offenseDescription like
"%homicide%"').groupBy("month").count()
crimeDataBosDF =
crimeDataBostonDF.select(lower(col("offense_code_group")),"month").withColum
nRenamed("lower(offense_code_group)","offenseDescription").filter('offenseDe
scription like "%homicide%"').groupBy("month").count()
combinedHomicide =
crimeDataNYDF.union(crimeDataBosDF).groupBy("month").sum("count").sort("mont
h",ascending=True).withColumnRenamed("sum(count)","Homicide Count")
display(combinedHomicide)
```

Table

| | month | Homicide Count |
|---|---|---|
| 1 | 1 | 29 |
| 2 | 2 | 21 |
| 3 | 3 | 29 |
| 4 | 4 | 36 |
| 5 | 5 | 38 |
| 6 | 6 | 42 |

12 rows

```
#The highest homicide count
highest_homicide_count = combinedHomicide.sort("homicide
count",ascending=False).take(1)
display(highest_homicide_count)
```

Table

| | month | Homicide Count |
|---|---|---|
| 1 | 8 | 50 |

1 row

```
#The lowest combined homicide count
lowest_homicide_count = combinedHomicide.sort("Homicide
Count",ascending=True).take(1)
display(lowest_homicide_count)
```

Table

| | month | Homicide Count |
|---|---|---|

| | month | Homicide Count | |
|---|---|---|---|
| 1 | 2 | 21 | |

1 row

```
#The highest homicide count city out of all given three cities,
HomicidesByMonthNY =
crimeDataNewYorkDF.select(lower(col("offenseDescription")),month("reportDate
")).withColumnRenamed("lower(offenseDescription)","offenseDescription").with
ColumnRenamed("month(reportDate)","month")
HomicidesByMonthNY = HomicidesByMonthNY.filter('offenseDescription like
"%murder%" or offenseDescription like
"%homicide%"').groupBy("month").count().withColumn("city",lit("New York"))
crimeDataBosDF =
crimeDataBostonDF.select(lower(col("offense_code_group")),"month").withColum
nRenamed("lower(offense_code_group)","offenseDescription")
crimeDataBosDF = crimeDataBosDF.filter('offenseDescription like
"%homicide%"').groupBy("month").count().withColumn("city",lit("Boston"))
combinedhomicidesByMonthDF = HomicidesByMonthNY.union(crimeDataBosDF)
highest_homicide_city_per_count =
combinedhomicidesByMonthDF.sort("count",ascending=False).take(1)
display(highest_homicide_city_per_count)
```

Table

| | month | count | city | |
|---|---|---|---|---|
| 1 | 8 | 36 | New York | |

1 row

```
#The lowest homicide count out of all cities
lowest_homicide_count_per_city =
combinedhomicidesByMonthDF.sort("count",ascending=True).take(1)
display(lowest_homicide_count_per_city)
```

Table

| | month | count | city | |
|---|---|---|---|---|
| 1 | 2 | 4 | Boston | |

1 row

# Question 9: Find the "per capita HOMICIDE rates" using the Hint in Qn 7, and plot graph as above for the per capita HOMICIDE rates

```
Tot_population_NY = cityDataDF.filter("city = 'New
York'").select("estPopulation2016").collect()[0]["estPopulation2016"]
Tot_population_Boston = cityDataDF.filter("city =
'Boston'").select("estPopulation2016").collect()[0]["estPopulation2016"]
crimeDataNY = crimeDataNYDF.withColumn("city",lit("New
York")).withColumnRenamed("count","homicides")
crimeDataBos =
crimeDataBosDF.withColumn("city",lit("Boston")).withColumnRenamed("count","h
omicides")
combinedhomicideDF =
crimeDataNY.union(crimeDataBos).sort("month",ascending=True)
Tot_homiciderates_Boston =
crimeDataBos.withColumn("homicideRate",crimeDataBos.homicides/Tot_population
_Boston)
Tot_homiciderates_NY =
crimeDataNY.withColumn("homicideRate",crimeDataNY.homicides/Tot_population_N
Y)
homicideratesCityDF =
Tot_homiciderates_NY.union(Tot_homiciderates_Boston).sort("month",ascending=
True)
display(homicideratesCityDF)
```

Table

| | month | homicides | city | homicideRate |
|---|---|---|---|---|
| 1 | 1 | 7 | Boston | 0.000010398345771735514 |
| 2 | 1 | 22 | New York | 0.000002576814548882347 7 |
| 3 | 2 | 4 | Boston | 0.00000594191186956315 |
| 4 | 2 | 17 | New York | 0.000019911748786818143 |
| 5 | 3 | 4 | Boston | 0.00000594191186956315 |
| 6 | 3 | 25 | New York | 0.000029281983510026 68 |

24 rows

```
display(homicideratesCityDF)
```

Table

| | month | homicides | city | homicideRate |
|---|---|---|---|---|
| 1 | 1 | 22 | New York | 0.000002576814548882347 7 |
| 2 | 1 | 7 | Boston | 0.000010398345771735514 |
| 3 | 2 | 17 | New York | 0.000019911748786818143 |
| 4 | 2 | 4 | Boston | 0.00000594191186956315 |
| 5 | 3 | 25 | New York | 0.000029281983510026 68 |
| 6 | 3 | 4 | Boston | 0.00000594191186956315 |

24 rows

# Question 10: A stretch goal to address a Data Science question on predicting crimes for a city as a time series model. How would you predict future values for monthly Robbery

## References

The crime data used in this notebook comes from the following locations:

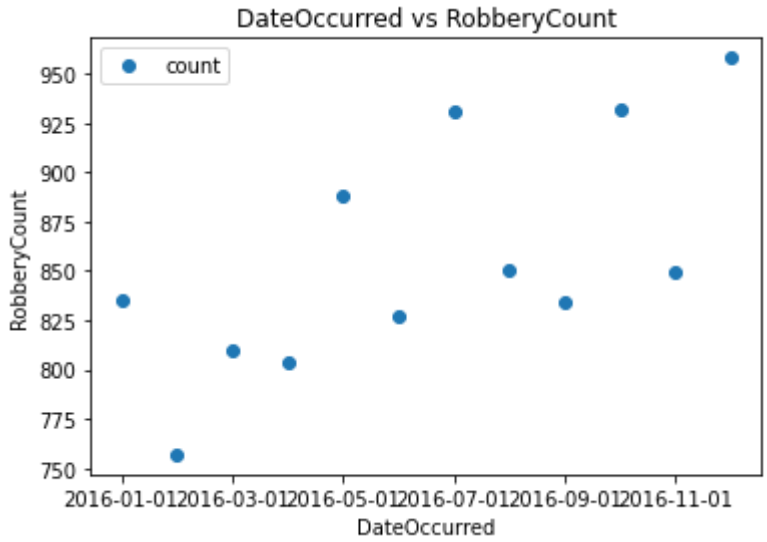
| City | Original Data |
|---|---|
| Boston | https://data.boston.gov/group/public-safety (https://data.boston.gov/group/public-safety) |
| Chicago | https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-present/ijzp-q8t2 (https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-present/ijzp-q8t2) |
| Dallas | https://www.dallasopendata.com/Public-Safety/Police-Incidents/tbnj-w5hb/data (https://www.dallasopendata.com/Public-Safety/Police-Incidents/tbnj-w5hb/data) |
| Los Angeles | https://data.lacity.org/A-Safe-City/Crime-Data-From-2010-to-Present/y8tr-7khq (https://data.lacity.org/A-Safe-City/Crime-Data-From-2010-to-Present/y8tr-7khq) |
| New Orleans | https://data.nola.gov/Public-Safety-and-Preparedness/Electronic-Police-Report-2016/4gc2-25he/data (https://data.nola.gov/Public-Safety-and-Preparedness/Electronic-Police-Report-2016/4gc2-25he/data) |
| New York | https://data.cityofnewyork.us/Public-Safety/NYPD-Complaint-Data-Historic/qgea-i56i (https://data.cityofnewyork.us/Public-Safety/NYPD-Complaint-Data-Historic/qgea-i56i) |

| City | Original Data |
| --- | --- |

```
from statsmodels.tsa.arima_model import ARIMA
import pandas as pd
from matplotlib import pyplot
from sklearn.metrics import mean_squared_error
import datetime
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as seabornInstance
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
LAcolumndata =
crimeDataLosAngelesDF.select("timeOccurred",lower(col("crimeCodeDescription"
))).withColumnRenamed("lower(crimeCodeDescription)","crime")
robberyLADF =
LAcolumndata.withColumn("DateOccurred",split(col("timeOccurred")," 
").getItem(0)).withColumn("MonthOccurred",split(col("timeOccurred"),"-
").getItem(1))
robberiesByMonthLosAngelesDF =
robberyLADF.select("DateOccurred","crime","MonthOccurred").filter("crime
like '%robbery%'")
LADF =
robberiesByMonthLosAngelesDF.groupBy("MonthOccurred").count().sort("MonthOcc
urred",ascending=True).withColumn("year",lit("2016-
")).withColumn("day",lit("-01")).withColumn("date",concat("year",
"MonthOccurred")).withColumn("DateOccurred",concat("date","day"))
TimeSeriesDF = LADF.select("DateOccurred","count").toPandas()
TimeSeriesDF
```

| | DateOccurred | count |
|---|---|---|
| **0** | 2016-01-01 | 835 |
| **1** | 2016-02-01 | 757 |
| **2** | 2016-03-01 | 810 |
| **3** | 2016-04-01 | 804 |
| **4** | 2016-05-01 | 888 |
| **5** | 2016-06-01 | 827 |
| **6** | 2016-07-01 | 931 |
| **7** | 2016-08-01 | 850 |
| **8** | 2016-09-01 | 834 |
| **9** | 2016-10-01 | 932 |
| **10** | 2016-11-01 | 849 |
| **11** | 2016-12-01 | 958 |

```
TimeSeriesDF.plot(x='DateOccurred', y='count', style='o')
plt.title('DateOccurred vs RobberyCount')
plt.xlabel('DateOccurred')
plt.ylabel('RobberyCount')
plt.show()
```




```
TimeSeriesDF.index = pd.to_datetime(TimeSeriesDF['DateOccurred'])
TimeSeriesDF = TimeSeriesDF.drop(['DateOccurred'],axis=1)
TimeSeriesDF
```

| DateOccurred | count |
|---|---|
| **2016-01-01** | 835 |
| **2016-02-01** | 757 |
| **2016-03-01** | 810 |
| **2016-04-01** | 804 |
| **2016-05-01** | 888 |
| **2016-06-01** | 827 |
| **2016-07-01** | 931 |
| **2016-08-01** | 850 |
| **2016-09-01** | 834 |
| **2016-10-01** | 932 |
| **2016-11-01** | 849 |
| **2016-12-01** | 958 |

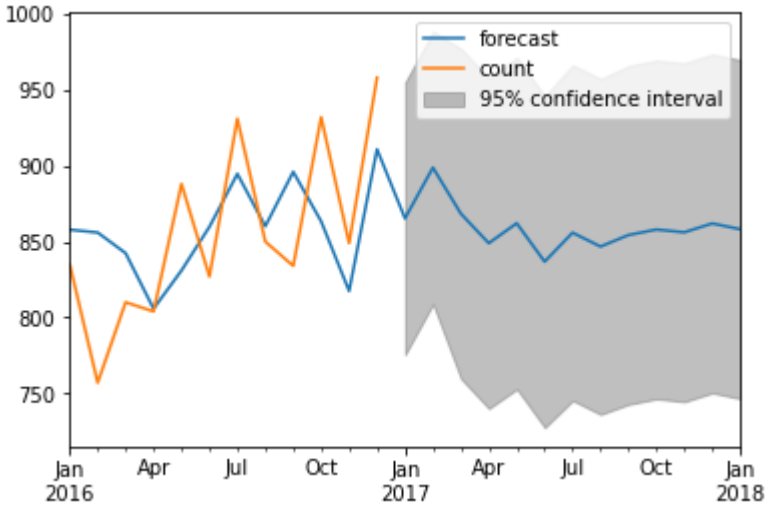
```
from statsmodels.tsa.arima_model import ARMA
mod = ARMA(TimeSeriesDF,order=(4,0))
res = mod.fit()
```

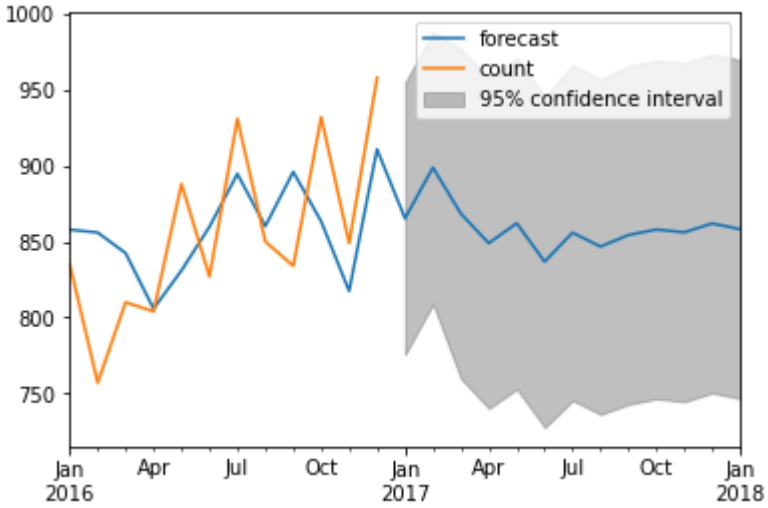
```
/databricks/python/lib/python3.9/site-packages/statsmodels/tsa/arima_model.p
y:472: FutureWarning:
statsmodels.tsa.arima_model.ARMA and statsmodels.tsa.arima_model.ARIMA have
been deprecated in favor of statsmodels.tsa.arima.model.ARIMA (note the .
between arima and model) and
statsmodels.tsa.SARIMAX. These will be removed after the 0.12 release.

statsmodels.tsa.arima.model.ARIMA makes use of the statespace framework and
is both well tested and maintained.

To silence this warning and continue using ARMA and ARIMA until they are
removed, use:

import warnings
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARMA',
                        FutureWarning)
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARIMA',
                        FutureWarning)

  warnings.warn(ARIMA_DEPRECATION_WARN, FutureWarning)
/databricks/python/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_mode
```

```
res.plot_predict(start='2016-01-01',end='2018-01-01')
```

```
/databricks/python/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_mode
l.py:132: FutureWarning: The 'freq' argument in Timestamp is deprecated and
```

```
will be removed in a future version.
  date_key = Timestamp(key, freq=base_index.freq)
```



ARMA Model Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | count | **No. Observations:** | 12 |
| **Model:** | ARMA(4, 0) | **Log Likelihood** | -63.494 |
| **Method:** | css-mle | **S.D. of innovations** | 45.766 |
| **Date:** | Mon, 17 Apr 2023 | **AIC** | 138.989 |
| **Time:** | 04:00:47 | **BIC** | 141.898 |
| **Sample:** | 01-01-2016 - 12-01-2016 | **HQIC** | 137.912 |

| | coef | std err | z | P>\|z\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **const** | 857.8547 | 21.161 | 40.539 | 0.000 | 816.379 | 899.330 |
| **ar.L1.count** | 0.0427 | 0.257 | 0.166 | 0.868 | -0.461 | 0.546 |
| **ar.L2.count** | 0.6817 | 0.311 | 2.194 | 0.028 | 0.073 | 1.291 |
| **ar.L3.count** | 0.0047 | 0.343 | 0.014 | 0.989 | -0.667 | 0.677 |
| **ar.L4.count** | -0.3724 | 0.321 | -1.160 | 0.246 | -1.001 | 0.257 |

Roots

| | Real | Imaginary | Modulus | Frequency |
|---|---|---|---|---|
| **AR.1** | -1.1269 | -0.6242j | 1.2883 | -0.4195 |
| **AR.2** | -1.1269 | +0.6242j | 1.2883 | 0.4195 |
| **AR.3** | 1.1333 | -0.5778j | 1.2721 | -0.0750 |
| **AR.4** | 1.1333 | +0.5778j | 1.2721 | 0.0750 |