

Analysis of the Factors Contributing to Changes in Crime Rates and Property Values in New York City

BIG DATA GROUP PROJECT



NextGen Analytical Solutions

C-SUITE EMPLOYEES:

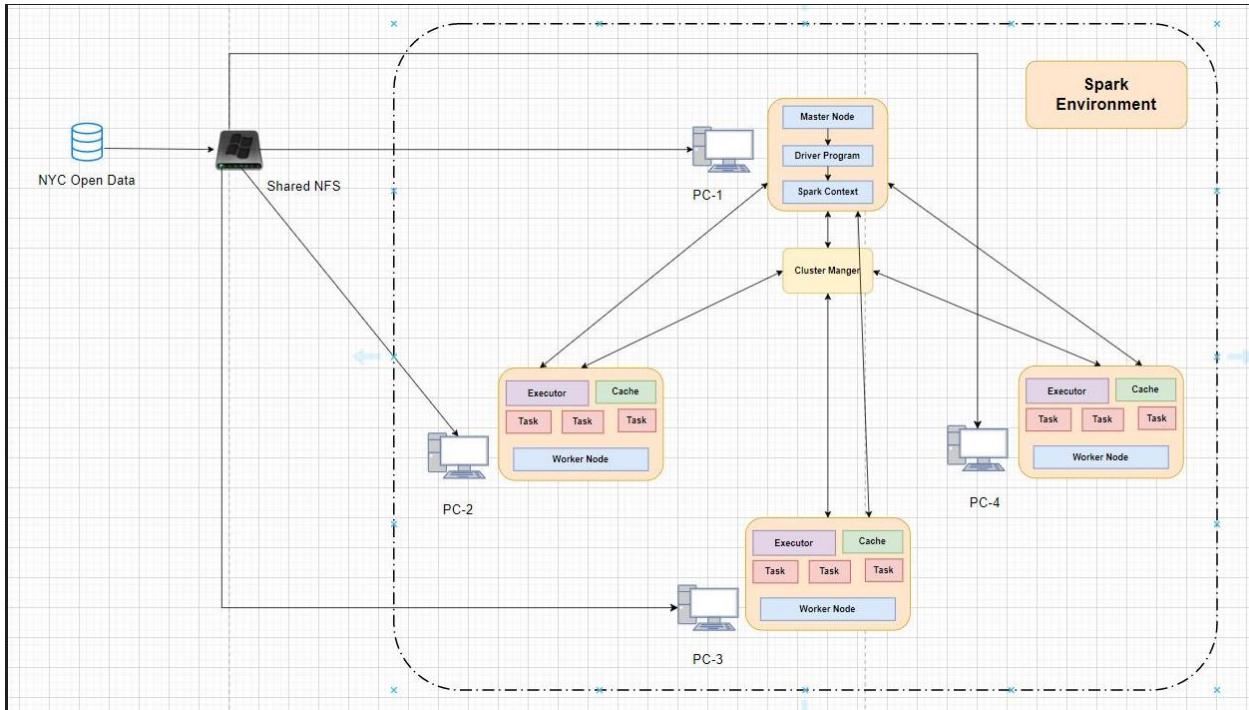
Satya Vinod Kantipudi - SXK210365 (CEO- Chief Executive Office)

Anurag S S S M - AXS210475 (CTO – Chief Technical Officer)

Sriya Meghana Nandam – SMN210004 (CIO – Chief Information Officer)

Lekhashree Jeezula - LXJ220002 (COO – Chief Operating Officer)

ARCHITECTURE



Distributed Computing System: The architecture you described is a distributed computing system. This means that the processing of data is distributed across multiple nodes in the system to improve performance and efficiency.

Network File System (NFS): The system uses a network file system (NFS) to load data onto the system. NFS is a protocol that allows a user to access files over a network as if they were stored on the local machine.

Master-Slave Architecture: The distributed computing system consists of a master node and one or more slave nodes. The master node is responsible for coordinating the processing of data and distributing tasks to the slave nodes. The slave nodes perform the computations on the data.

Data Loading: To start the data processing, the master node loads the data from the NFS server into memory. The data is partitioned into smaller chunks that can be processed by the slave nodes in parallel.

Parallel Processing: The slave nodes are equipped with processing power and memory to perform computations on the data. The master node assigns the data chunks to the slave nodes, which perform computations on the data in parallel.

Results Combination: Once the slave nodes have processed their assigned data chunks, they send the results back to the master node. The master node combines the results from each of the slave nodes and presents a final output to the user.

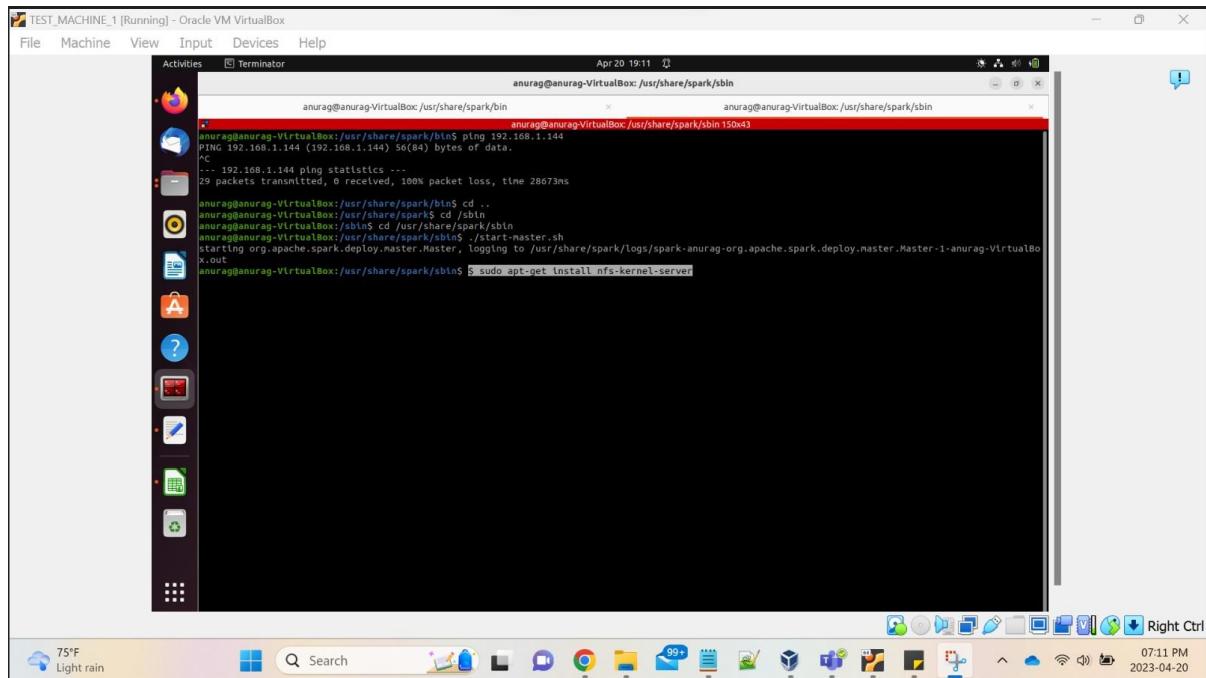
Performance Improvement: The architecture you described is often used in big data processing systems such as Hadoop and Apache Spark. By distributing the workload across multiple nodes, the processing time can be reduced significantly, improving performance and efficiency.

Network File System (NFS):

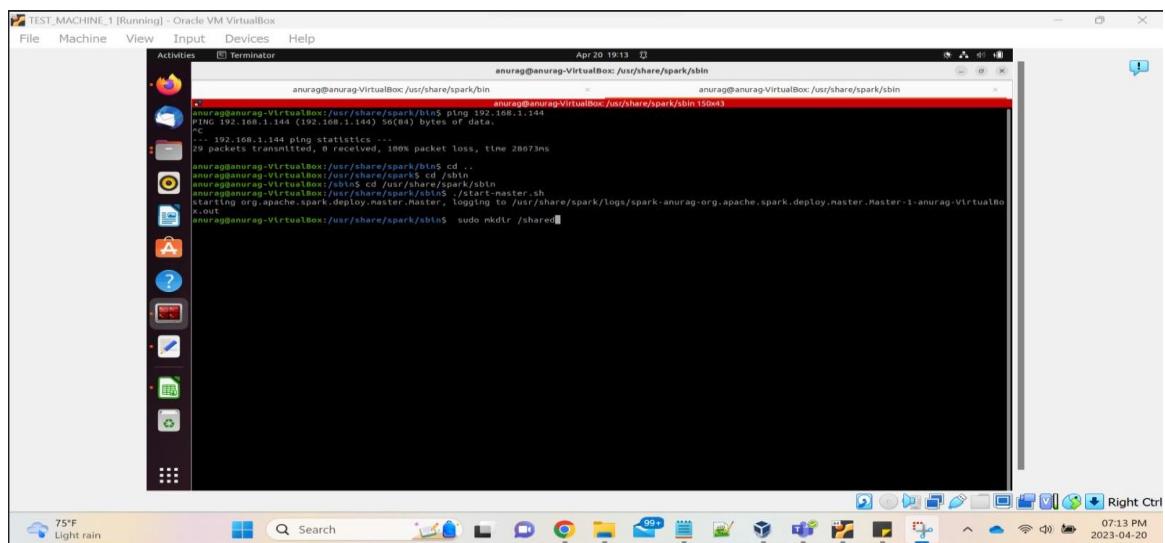
Steps to install NFS on both master nodes and worker nodes:

Below are the NFS screenshots :-

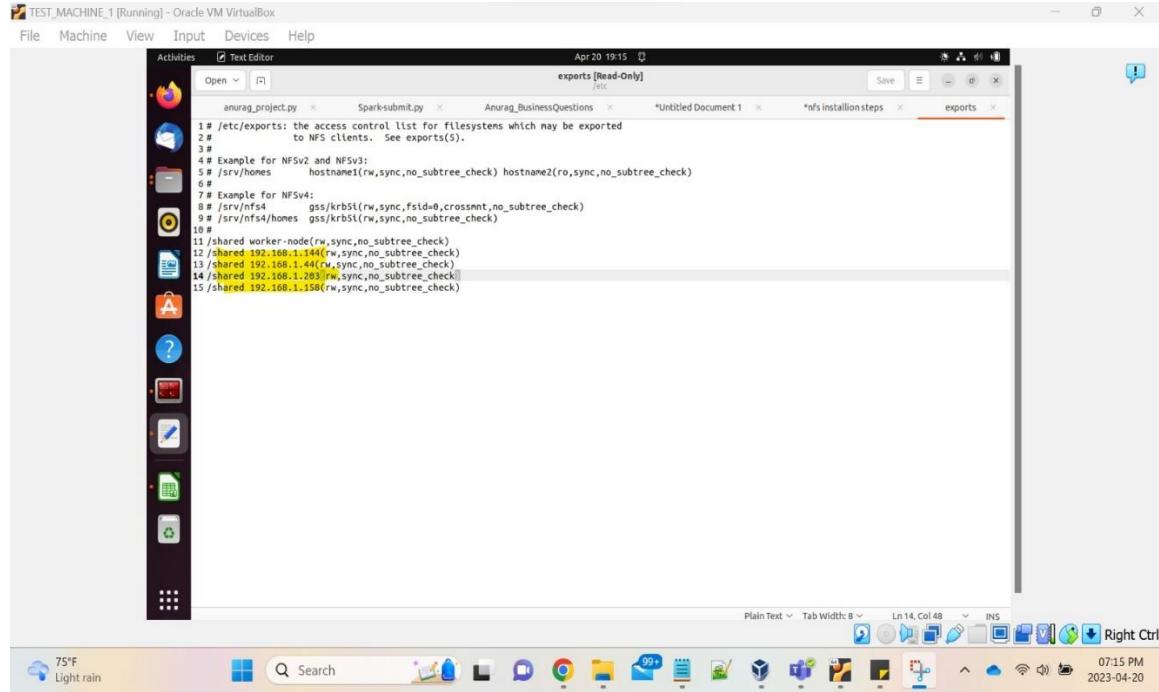
1. Master node NFS installation (Anurag)



2. Creating a shared folder in the master node



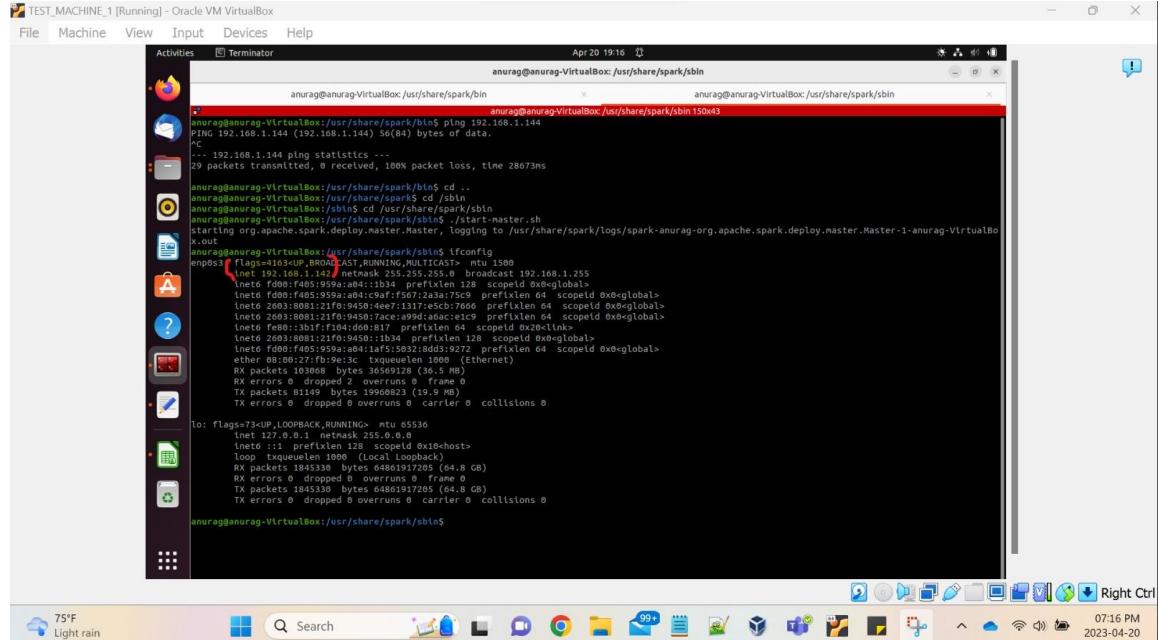
3. Adding worker node ip's in the /etc/export file in order to give the worker nodes access to the shared folder.



The screenshot shows a Windows desktop environment with an Oracle VM VirtualBox window titled "TEST_MACHINE_1 [Running]". Inside the virtual machine, a terminal window is open with the command "cat /etc/export" running. The output of the command is displayed, listing several IP addresses followed by "/(rw,sync,no_subtree_check)". The desktop taskbar at the bottom shows various icons and the date/time as 2023-04-20 07:15 PM.

```
anurag@anurag-VirtualBox:~$ cat /etc/export
1# /etc(exports: the access control list for filesystems which may be exported
2# to NFS clients. See exports(5).
3#
4# Example for NFSv2 and NFSv3:
5# /srv/homes hostname!(rw,sync,no_subtree_check) hostname2(ro,sync,no_subtree_check)
6#
7# Example for NFSv4:
8# /srv/nfs4 gss/krb5!(rw,sync,fsid=0,crossmnt,no_subtree_check)
9# /srv/nfs4/homes gss/krb5!(rw,sync,no_subtree_check)
10#
11/shared worker-node(rw,sync,no_subtree_check)
12/shared 192.168.1.144 (rw,sync,no_subtree_check)
13/shared 192.168.1.144 (ro,sync,no_subtree_check)
14/shared 192.168.1.143(rw,sync,no_subtree_check)
15/shared 192.168.1.158(rw,sync,no_subtree_check)
```

4. IP address of the master node 192.168.1.142



The screenshot shows a Windows desktop environment with an Oracle VM VirtualBox window titled "TEST_MACHINE_1 [Running]". Inside the virtual machine, a terminal window is open with the command "ifconfig" running. The output shows detailed network statistics for the eth0 interface, including RX/TX bytes, errors, and collisions. The desktop taskbar at the bottom shows various icons and the date/time as 2023-04-20 07:16 PM.

```
anurag@anurag-VirtualBox:~$ ifconfig
eth0      flags=700<NOARP,BROADCAST,MULTICAST  mtu 1500
          inet6 brd ff:ff:ff:ff:ff:ff  scopeid 0x0<global>
          inet6 fd00:7405:559a:a04::1b34  prefixlen 128  scoprid 0x0<global>
          inet6 fd00:7405:559a:a04::1b34  prefixlen 64  scoprid 0x0<global>
          inet6 fd00:7405:559a:a04::1b34  prefixlen 64  scoprid 0x0<global>
          inet6 fd00:7405:559a:a04::1b34  prefixlen 64  scoprid 0x0<global>
          ether 00:0c:29:1f:04:d6  brd ff:ff:ff:ff:ff:ff
          RX packets 1845330  bytes 0x4861917205 (64.8 GB)
          RX errors 0  dropped 0  overruns 0  frame 0
          TX packets 1845330  bytes 0x4861917205 (64.8 GB)
          TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

lo        flags=73<NOARP,BROADCAST,RUNNING  mtu 65536
          inet 127.0.0.1  netmask 255.0.0.0
          inet6 ::1  prefixlen 128  scopeid 0x0<host>
          loop      flags=73<NOARP,BROADCAST,RUNNING  mtu 65536
          RX packets 1845330  bytes 0x4861917205 (64.8 GB)
          RX errors 0  dropped 0  overruns 0  frame 0
          TX packets 1845330  bytes 0x4861917205 (64.8 GB)
          TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

anurag@anurag-VirtualBox:~$
```

5. Worker node nfs-common installation (192.168.1.158)

```

sriya@sriya-VirtualBox: ~ $ cd /usr/share/spark/btl
sriya@sriya-VirtualBox: ~ $ ./start-worker.sh
starting org.apache.spark.deploy.worker.Worker, logging to /usr/share/spark/logs/spark-sriya-org.apache.spark.deploy.worker.Worker-1-sriya-VirtualBox.out
org.apache.spark.deploy.worker.Worker running as process 2044. Stop it first.
sriya@sriya-VirtualBox: ~ $ ./stop-worker.sh
stop-worker.sh stop-workers.sh
sriya@sriya-VirtualBox: ~ $ ./stop-worker.sh
stopping org.apache.spark.deploy.worker.Worker
sriya@sriya-VirtualBox: ~ $ ./start-worker.sh
starting org.apache.spark.deploy.worker.Worker, logging to /usr/share/spark/logs/spark-sriya-org.apache.spark.deploy.worker.Worker-1-sriya-VirtualBox.out
sriya@sriya-VirtualBox: ~ $ ./stop-worker.sh
stopping org.apache.spark.deploy.worker.Worker
sriya@sriya-VirtualBox: ~ $ ./start-worker.sh
starting org.apache.spark.deploy.worker.Worker, logging to /usr/share/spark/logs/spark-sriya-org.apache.spark.deploy.worker.Worker-1-sriya-VirtualBox.out
sriya@sriya-VirtualBox: ~ $ sudo apt-get install nfs-common
[sudo] password for sriya:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
nfs-common is already the newest version (1:2.6.1-1ubuntu1.2).
0 upgraded, 0 newly installed, 0 to remove and 112 not upgraded.
sriya@sriya-VirtualBox: ~ $ ifconfig
enp0s3: flags=4163 mtu 1500
    inet 192.168.1.158 brd 192.168.1.255 netmask 255.255.255.0 broadcast 192.168.1.255
        inet6 fe80::6d90:1ff:fe:1735%enp0s3 brd fe80::ff:fe:1735%enp0s3 scopeid 0x20<link>
            inet6 2603:8081:21f0:9450:a4c9:09fe:b40c:1c60 prefixlen 64 scopelen 0x0<global>
            inet6 fd00:f405:959a:a04:8e41:db51:2f38:eff8 prefixlen 64 scopelen 0x0<global>
            inet6 fd00:f405:959a:a04:1958 prefixlen 128 scopelen 0x0<global>
            inet6 fd00:f405:959a:a04:1c4f:2fa4:2fa4 prefixlen 64 scopelen 0x0<global>
            inet6 2603:8081:21f0:9450:ae0b:838c:71f9:6795 prefixlen 64 scopelen 0x0<global>
            link layer [ether] brd ff:ff:ff:ff:ff:ff
            ether 08:00:27:f2:c8:a2 txqueuelen 1000 (Ethernet)
            RX packets 210718 bytes 282457416 (282.4 MB)
            RX errors 0 dropped 3 overruns 0 frame 0
            TX packets 21745 bytes 4666397 (4.6 MB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73 mtu 65536
    inet 127.0.0.1 brd 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopelen 0x10<host>
            loop txqueuelen 1000 (Local Loopback)
            RX packets 17840 bytes 510348827 (510.3 MB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 17840 bytes 510348827 (510.3 MB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
sriya@sriya-VirtualBox: ~ $ lsof

```

74°F Heavy t-storms Search ENG IN 19:27 20-04-2023 Right Ctrl

6. Accessing shared folder by mounting the master /shared location to the worker shared location.

```

sriya@sriya-VirtualBox: ~ $ ifconfig
enp0s3: flags=4163 mtu 1500
    inet 192.168.1.158 brd 192.168.1.255 netmask 255.255.255.0 broadcast 192.168.1.255
        inet6 fe80::6d90:1ff:fe:1735%enp0s3 brd fe80::ff:fe:1735%enp0s3 scopeid 0x20<link>
            inet6 2603:8081:21f0:9450:a4c9:09fe:b40c:1c60 prefixlen 64 scopelen 0x0<global>
            inet6 fd00:f405:959a:a04:8e41:db51:2f38:eff8 prefixlen 64 scopelen 0x0<global>
            inet6 fd00:f405:959a:a04:1958 prefixlen 128 scopelen 0x0<global>
            inet6 fd00:f405:959a:a04:1c4f:2fa4:2fa4 prefixlen 64 scopelen 0x0<global>
            inet6 2603:8081:21f0:9450:ae0b:838c:71f9:6795 prefixlen 64 scopelen 0x0<global>
            inet6 2603:8081:21f0:9450::1958 prefixlen 128 scopelen 0x0<global>
            ether 08:00:27:f2:c8:a2 txqueuelen 1000 (Ethernet)
            RX packets 210718 bytes 282457416 (282.4 MB)
            RX errors 0 dropped 3 overruns 0 frame 0
            TX packets 21745 bytes 4666397 (4.6 MB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73 mtu 65536
    inet 127.0.0.1 brd 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopelen 0x10<host>
            loop txqueuelen 1000 (Local Loopback)
            RX packets 17840 bytes 510348827 (510.3 MB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 17840 bytes 510348827 (510.3 MB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
sriya@sriya-VirtualBox: ~ $ sudo mount 192.168.1.142:/shared /shared
sriya@sriya-VirtualBox: ~ $
sriya@sriya-VirtualBox: ~ $ cd /shared
sriya@sriya-VirtualBox: /shared $ ls -l
total 672680
-rwxrwxr-x 1 root root 111626932 Apr 19 01:30 chunk_1final.csv
-rwxrwxr-x 1 root root 113101085 Apr 19 01:35 chunk_2final.csv
-rwxrwxr-x 1 root root 61344889 Apr 19 01:35 chunk_3final.csv
-rwxrwxr-x 1 root root 113101085 Apr 19 01:35 chunk_4final.csv
-rwxrwxr-x 1 root root 76824547 Apr 19 01:35 chunk_5final.csv
-rwxrwxr-x 1 root root 33006518 Apr 19 01:35 chunk_6final.csv
-rwxrwxr-x 1 root root 32920844 Apr 19 01:35 chunk_7final.csv
-rwxrwxr-x 1 root root 55364679 Apr 19 01:35 chunk_8final.csv
-rwxrwxr-x 1 root root 81607759 Apr 19 01:35 chunk_9final.csv
-rwxrwxr-x 1 sriya sriya 58446 Apr 20 04:02 House_Sales.csv
-rwxrwxr-x 1 root root 1842 Apr 20 08:56 population.csv
sriya@sriya-VirtualBox: /shared $ 
```

74°F Heavy t-storms Search ENG IN 19:30 20-04-2023 Right Ctrl

We have repeated this step for all the worker nodes like 192.168.1.144(Vinod) 192.168.1.44(Lekha), 192.168.1.158(Meghana). Now all the worker nodes have access to the shared folder from where the data is ingested into our program.

CLUSTER ARCHITECTURE:

Steps performed to enable cluster processing:

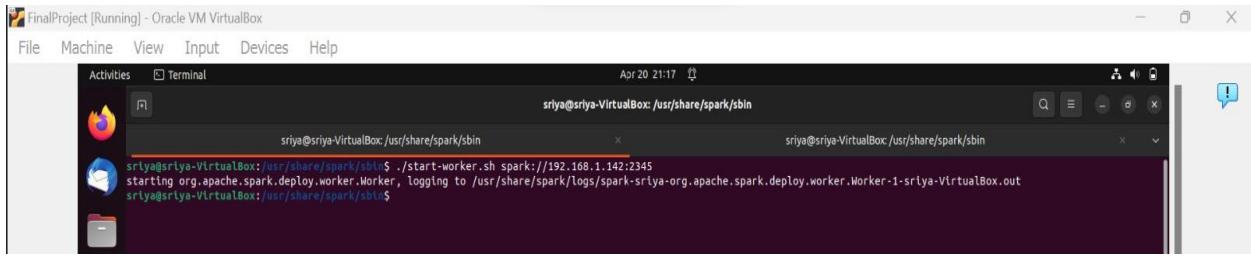
Step 1:- We have set up the SPARK MASTER NODE and SPARK MASTER PORT configuration in /conf/spark-env.sh.

This configuration is set for all the Worker nodes to specify who the master node is and which port we need to use to connect to the master node.

The screenshot shows a Windows desktop environment with several open windows. At the top is the taskbar with icons for File Explorer, Edge browser, FileZilla, and others. Below the taskbar is a system tray showing battery level (73%), signal strength, and the date/time (08:03 PM, 2023-04-20). The main focus is a terminal window titled 'Text Editor' with the file path 'spark-env.sh'. The terminal contains a large block of shell script code for setting up a Spark cluster. The code includes environment variable assignments like SPARK_MASTER_IP=192.168.1.142, SPARK_PUBLIC_DNS=192.168.1.142, and SPARK_LOCAL_IP=192.168.1.142. It also includes comments explaining the purpose of each variable and the source of configuration files. The terminal window has a dark theme and is running on a virtual machine named 'TEST_MACHINE_1 [Running] - Oracle VM VirtualBox'. The background of the desktop shows a blue and white abstract pattern.

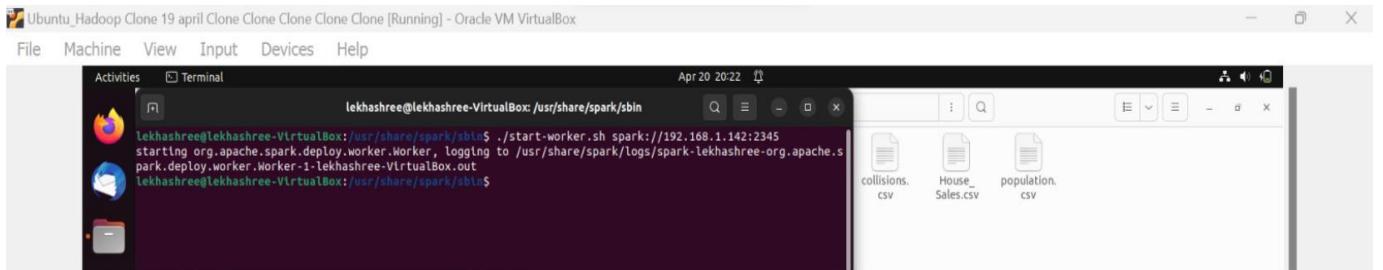
Step 2:- After adding the conf properties, we will have to run the Master node. using above command.

Step 3 – Start Worker nodes



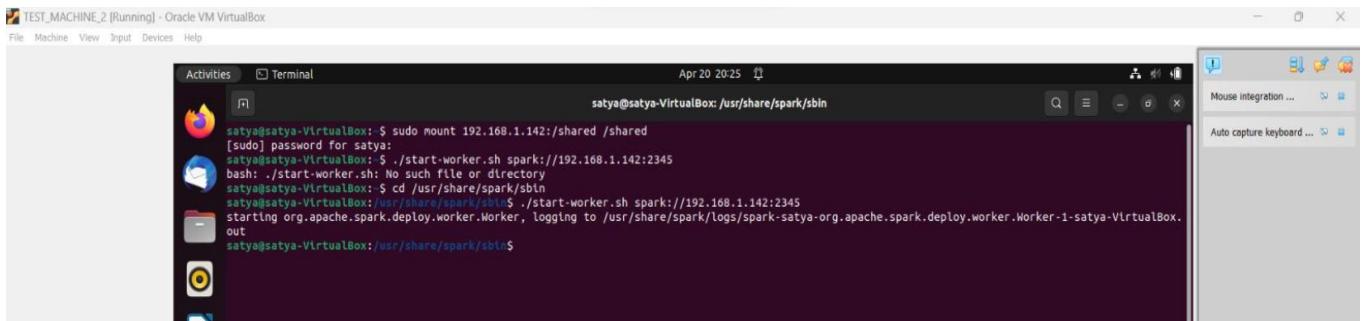
```
sriya@sriya-VirtualBox: /usr/share/spark/sbin$ ./start-worker.sh spark://192.168.1.142:2345
starting org.apache.spark.deploy.worker.Worker, logging to /usr/share/spark/logs/spark-sriya-org.apache.spark.deploy.worker.Worker-1-sriya-VirtualBox.out
sriya@sriya-VirtualBox: /usr/share/spark/sbin$
```

Worker node 192.162.1.158(Meghana) started.



```
lekhashree@lekhshree-VirtualBox: /usr/share/spark/sbin$ ./start-worker.sh spark://192.168.1.142:2345
starting org.apache.spark.deploy.worker.Worker, logging to /usr/share/spark/logs/spark-lekhashree-org.apache.spark.deploy.worker.Worker-1-lekhashree-VirtualBox.out
lekhashree@lekhshree-VirtualBox: /usr/share/spark/sbin$
```

Worker node 192.168.1.44 (Lekha) started.



```
satya@satya-VirtualBox: $ sudo mount 192.168.1.142:/shared /shared
[sudo] password for satya:
satya@satya-VirtualBox: $ ./start-worker.sh spark://192.168.1.142:2345
bash: ./start-worker.sh: No such file or directory
satya@satya-VirtualBox: $ cd /usr/share/spark/sbin
satya@satya-VirtualBox: /usr/share/spark/sbin$ ./start-worker.sh spark://192.168.1.142:2345
starting org.apache.spark.deploy.worker.Worker, logging to /usr/share/spark/logs/spark-satya-org.apache.spark.deploy.worker.Worker-1-satya-VirtualBox.out
satya@satya-VirtualBox: /usr/share/spark/sbin$
```

Worker node 192.168.1.144(Vinod) started.

CLUSTER PROCESSING:

The screenshot shows the Spark Master Web UI running in a Firefox browser window titled "Spark Master at spark://192.168.1.142:2345". The UI displays the following information:

- URL:** spark://192.168.1.142:2345
- Alive Workers:** 3
- Cores in use:** 6 Total, 6 Used
- Memory in use:** 2.8 GB Total, 2.8 GB Used
- Resources in use:**
- Applications:** 1 Running, 8 Completed
- Drivers:** 0 Running, 0 Completed
- Status:** ALIVE

Workers (3)

Worker ID	Address	State	Cores	Memory	Resources
worker-20230420202205-192.168.1.144-43399	192.168.1.144:43399	ALIVE	1 (1 Used)	6.8 GB (476.0 MB Used)	
worker-20230420202551-192.168.1.144-41203	192.168.1.144:41203	ALIVE	4 (4 Used)	3.8 GB (1904.0 MB Used)	
worker-20230420211536-192.168.1.158-36987	192.168.1.158:36987	ALIVE	1 (1 Used)	2.8 GB (476.0 MB Used)	

Running Applications (1)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20230420211456-0008	Spark-submit.py	6	476.0 MB		2023/04/20 21:14:56	anurag	RUNNING	33 s

Completed Applications (8)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20230420210503-0007	Spark-submit.py	4	476.0 MB		2023/04/20 21:05:03	anurag	FINISHED	4.5 min
app-20230420205342-0006	Spark-submit.py	4	476.0 MB		2023/04/20 20:53:42	anurag	FINISHED	53 s
app-20230420205209-0005	Spark-submit.py	0	476.0 MB		2023/04/20 20:52:09	anurag	FINISHED	52 s
app-20230420204809-0004	Spark-submit.py	0	476.0 MB		2023/04/20 20:48:09	anurag	FINISHED	3.8 min
app-20230420204421-0003	Spark-submit.py	4	476.0 MB		2023/04/20 20:44:21	anurag	FINISHED	60 s

This is the Master Web UI where it shows the Number of worker nodes involved in the process and amount of memory used by each node for processing the data.

The screenshot shows the Spark Worker Web UI running in a Firefox browser window titled "Spark Worker at 192.168.1.144:43399". The UI displays the following information:

- ID:** worker-20230420202205-192.168.1.144-43399
- Master URL:** spark://192.168.1.142:2345
- Cores:** 1 (0 Used)
- Memory:** 6.8 GB (0.0 B Used)
- Resources:**
- Back to Master**

Running Executors (0)

ExecutorID	State	Cores	Memory	Resources	Job Details	Logs
------------	-------	-------	--------	-----------	-------------	------

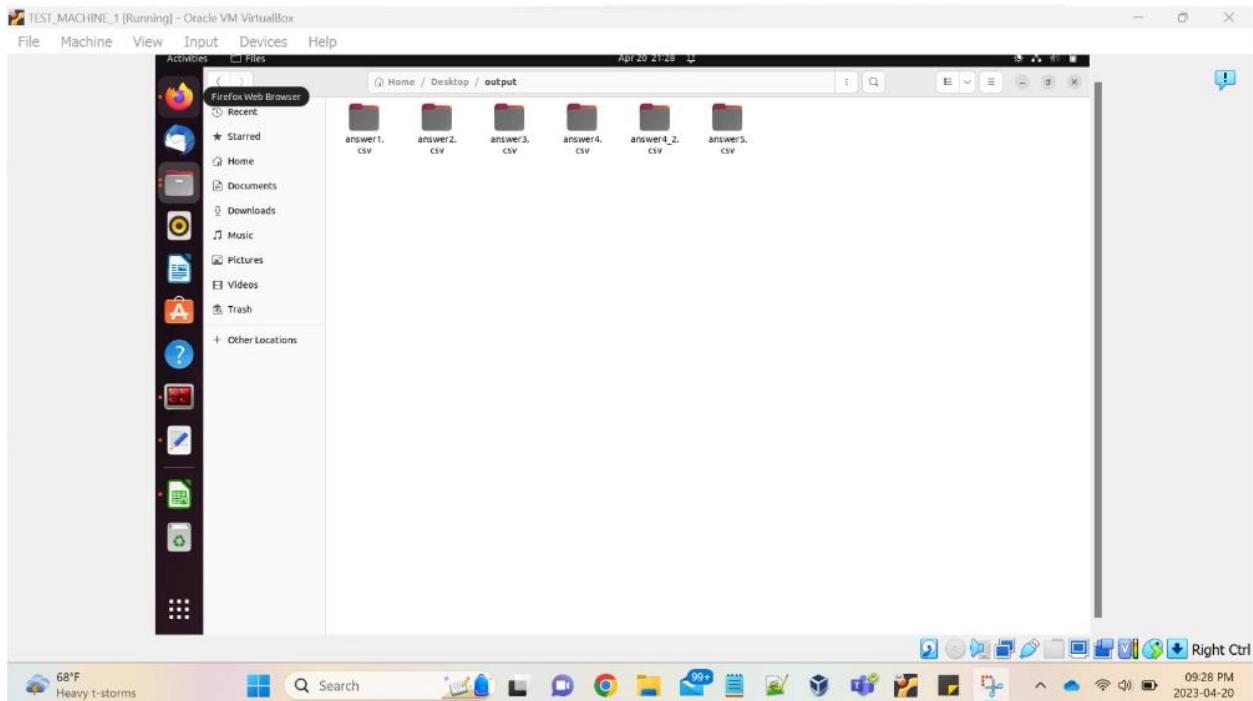
Finished Executors (1)

ExecutorID	State	Cores	Memory	Resources	Job Details	Logs
4	KILLED	1	476.0 MB		ID: app-20230420211456-0008 Name: Spark-submit.py User: anurag	stdout stderr

Log - one among the user node that shows the execution and the memory utilized for processing of the data.

A screenshot of a Linux desktop environment, likely Ubuntu, showing a terminal window with a large amount of text output from a Spark application. The terminal title is "anurag@anurag-VirtualBox: /usr/share/spark/bin". The log output includes several INFO messages from the DAGScheduler, MemoryStore, and TaskScheduler, as well as a warning about missing parents. It also shows the execution of a job with 111 tasks, the completion of a stage, and the final output taking 0.136189 seconds. Below the terminal, the desktop taskbar displays icons for various applications like a file manager, browser, and system monitor. The bottom right corner shows the date and time as "09:27 PM 2023-04-20".

Pyspark shell - background execution of the Spark-submit.py code.



The output file is written in this location. The result from the data analysis is written into a csv file and stored here.

BUSINESS QUESTIONS

- 1. How can we correlate the population of a neighborhood, crime rate and the average sale price of homes in that neighborhood?*

Business Conclusion:

There may be a correlation between the population of a neighborhood and the average sale price of homes in that neighborhood, although other factors may also play a role in determining home prices.

In general, areas with larger populations may have higher demand for housing, which could drive up prices. Additionally, neighborhoods with higher populations may be more attractive to businesses, leading to more jobs and economic activity in the area, which could also contribute to higher home prices.

However, it is important to note that other factors, such as the quality of schools, crime rates, and access to amenities such as parks and shopping centers, can also play a significant role in determining home prices. As such, it is important to consider a variety of factors when analyzing the relationship between population and home prices.

To determine whether there is a correlation between population and home prices in a particular neighborhood, you could analyze data on population and home sales in the area. One approach might be to plot population against home sale prices on a scatter plot, and then calculate the correlation coefficient between the two variables. A positive correlation coefficient would suggest that as population increases, home sale prices also tend to increase.

Overall, understanding the relationship between population and home prices can be useful for businesses and individuals who are looking to invest in real estate or develop new properties. By analyzing data on population and home prices, businesses can gain insights into the factors that influence demand for housing in a particular area and make more informed decisions about where to invest resources.

We can prevent this from happening by installing cameras, increased patrolling, increasing penalties for violent crime etc.

Output:

TEST_MACHINE_1 [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Activities Terminator Apr 20 22:08

anurag@anurag-VirtualBox: /usr/share/spark/bin

```
anurag@anurag-VirtualBox: /usr/share/spark/bin$ ./spark-submit --class BusinessAnswers --master local[*] /home/anurag/Desktop/output/business_answers.jar
```

anurag@anurag-VirtualBox: /usr/share/spark/bin\$

```
... StructField("% of share of NYC", FloatType(), true))+-----+-----+-----+-----+  
| BORO_NY|HOUSE_SALE_COUNT|CRIME_CASES_COUNT|% OF SHARE OF NYC|  
+-----+-----+-----+-----+  
| STATEN ISLAND| 330| 239115| 5.18|  
| BROOKLYN| 3201| 107074| 27.85|  
| BRONX| 281| 1139704| 16.93|  
| MANHATTAN| 2360| 1277512| 19.36|  
| BROOKLYN| 14454| 1581609| 30.97|  
+-----+-----+-----+-----+  
>>> selected_business_answer1.write.format("csv").option("header","true").mode("overwrite").save("/home/anurag/Desktop/output/answer1.csv")  
(0 + 1) / 1
```

67°F Cloudy

Search

Right Ctrl

SQL Code:

```
average=house_sales_df.groupBy().avg("AVERAGE_SALE_PRICE").collect()[0][0]
result_set=house_sales_df.filter(house_sales_df['AVERAGE_SALE_PRICE']>average).select('borough','number_of_sales')
most_expensive_neighborhood=result_set.groupBy("BOROUGH").sum("NUMBER_OF_SALES").alias("NUMBER_OF_HOUSES")
most_expensive_neighborhood=most_expensive_neighborhood.orderBy(desc("sum(NUMBER_OF_SALES)"))

crime_rate=nyc_crime_data.groupBy("BORO_NM").count().orderBy("count")
nyc_expensive_crime=most_expensive_neighborhood.join(crime_rate,most_expensive_neighborhood["BOROUGH"]==crime_rate["BORO_NM"])

nyc_expensive_crime=nyc_expensive_crime.orderBy("count")
business_answer1=nyc_expensive_crime.withColumnRenamed("sum(NUMBER_OF_SALES)","HOUSE_SALE_COUNT").withColumnRenamed("count","CRIME_CASES_COUNT")

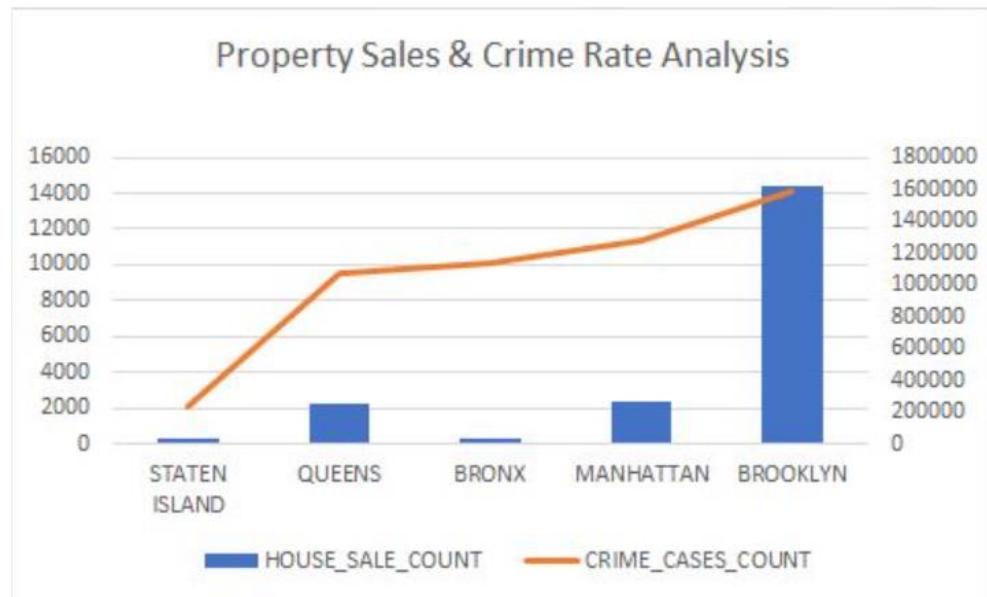
population_yearwise=population_df.filter(population_df["YEAR"]=="2020").select("borough","% of share of NYC")
business_answer1=business_answer1.join(population_yearwise,business_answer1["BOROUGH"]==population_yearwise["borough"])
business_answer1=business_answer1.orderBy("CRIME_CASES_COUNT")
column_to_remove=business_answer1.columns[-2]
business_answer1=business_answer1.drop(column_to_remove)

#mode("overwrite")
selected_columns=["BORO_NM","HOUSE_SALE_COUNT","CRIME_CASES_COUNT","% OF SHARE OF NYC"]
selected_business_answer1=business_answer1.select(selected_columns)
selected_business_answer1.show()
```

Code Output Explanation:

Based on the analysis of New York house sales data and crime data, it can be concluded that the neighborhoods with the highest number of house sales in New York City tend to have higher crime rates than other neighborhoods, and are typically the most densely populated areas. Specifically, the analysis found that Brooklyn had the highest number of both house sales and crime cases followed by Manhattan.

Visualization:

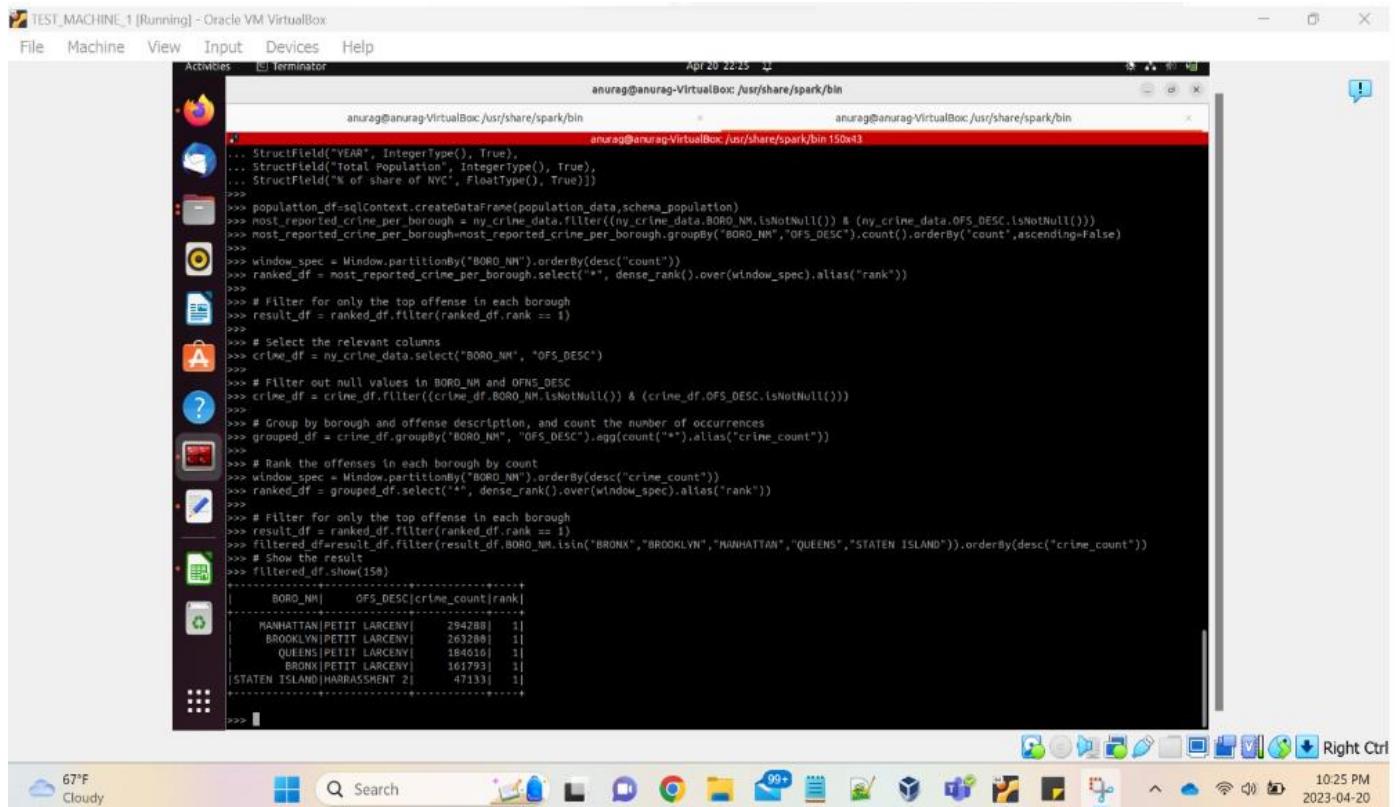


2. What type of crimes are most commonly reported in each borough, and are there any patterns or trends that can be identified?

Business Conclusion:

Petit larceny and harassment are two of the most commonly committed crimes in several boroughs. To tackle these issues, several strategies can be employed. Firstly, increasing law enforcement presence in areas where these crimes are prevalent can act as a deterrent and lead to more arrests and prosecutions. Secondly, the installation of surveillance cameras in public spaces such as parking lots and parks can deter criminals and provide valuable evidence for law enforcement. Additionally, the use of social media can help raise awareness of these issues among the public. Ensuring proper lighting and visibility in public spaces is also essential in reducing crime, as poor lighting can make it easier for criminals to commit petit larceny and harassment. Addressing underlying social and economic issues, such as poverty and substance abuse, can also contribute to reducing crime in the long run. However, it is important to note that reducing crime requires a comprehensive and sustained effort from law enforcement, community members, and other stakeholders.

Output:



The screenshot shows a Linux desktop environment with a terminal window open in the Terminator application. The terminal window has two tabs: 'anurag@anurag-VirtualBox:/usr/share/spark/bin' and 'anurag@anurag-VirtualBox:/usr/share/spark/bin 150w4'. The user is running a Python script to analyze crime data. The script uses PySpark to filter and group crime data by borough and offense type, then ranks the offenses within each borough. Finally, it filters the results to show only the top offense in each borough. The output of the script is displayed in the terminal, showing a table with columns 'BORO_NM', 'OFS_DESC', and 'crime_count'. The data shows that Petit Larceny is the most common offense in all boroughs, with Manhattan having the highest count of 294280.

```
anurag@anurag-VirtualBox:/usr/share/spark/bin
anurag@anurag-VirtualBox:/usr/share/spark/bin 150w4

... StructField("YEAR", IntegerType(), True),
... StructField("Total Population", IntegerType(), True),
... StructField("% of share of NYC", FloatType(), True))
...
>>> population_df = sqlContext.createDataFrame(population_data, schema_population)
>>> most_reported_crime_per_borough = ny_crime_data.filter((ny_crime_data.BORO_NM.isNotNull()) & (ny_crime_data.OFS_DESC.isNotNull()))
>>> most_reported_crime_per_borough=most_reported_crime_per_borough.groupBy("BORO_NM","OFS_DESC").count().orderBy("count",ascending=False)
>>>
>>> window_spec = Window.partitionBy("BORO_NM").orderBy(desc("count"))
>>> ranked_df = most_reported_crime_per_borough.select("", dense_rank().over(window_spec).alias("rank"))
>>>
>>> # Filter for only the top offense in each borough
>>> result_df = ranked_df.filter(ranked_df.rank == 1)
>>>
>>> # Select the relevant columns
>>> crime_df = ny_crime_data.select("BORO_NM", "OFS_DESC")
>>>
>>> # Filter out null values in BORO_NM and OFS_DESC
>>> crime_df = crime_df.filter((crime_df.BORO_NM.isNotNull()) & (crime_df.OFS_DESC.isNotNull()))
>>>
>>> # Group by borough and offense description, and count the number of occurrences
>>> grouped_df = crime_df.groupby("BORO_NM", "OFS_DESC").agg(count("*").alias("crime_count"))
>>>
>>> # Rank the offenses in each borough by count
>>> window_spec = Window.partitionBy("BORO_NM").orderBy(desc("crime_count"))
>>> ranked_df = grouped_df.select("", dense_rank().over(window_spec).alias("rank"))
>>>
>>> # Filter for only the top offense in each borough
>>> result_df = ranked_df.filter(ranked_df.rank == 1)
>>> filtered_df=result_df.filter(result_df.BORO_NM.isin("BRONX","BROOKLYN","MANHATTAN","QUEENS","STATEN ISLAND")).orderBy(desc("crime_count"))
>>> filtered_df.show(150)

+-----+-----+
| BORO_NM | OFS_DESC|crime_count|
+-----+-----+
| MANHATTAN|PETIT LARCENY| 294280| 1|
| BROOKLYN|PETIT LARCENY| 263280| 1|
| QUEENS|PETIT LARCENY| 184610| 1|
| BRONX|PETIT LARCENY| 161793| 1|
| STATEN ISLAND|HARRASSMENT 2| 47133| 1|
+-----+-----+
```

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
	BORO_NM	OFS_DESC	crime_count	rank											
1	BORO_NM	PETIT LARCENY	294260	1											
2	MANHATTAN	PETIT LARCENY	263280	1											
3	BROOKLYN	PETIT LARCENY	263280	1											
4	QUEENS	PETIT LARCENY	164616	1											
5	BRONX	PETIT LARCENY	161793	1											
6	STATEN ISLAND	HARRASSMENT	2	47133	1										
7															
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															

SQL Code:

```

most_reported_crime_per_borough =
ny_crime_data.filter((ny_crime_data.BORO_NM.isNotNull()) &
(ny_crime_data.OFS_DESC.isNotNull()))
most_reported_crime_per_borough=most_reported_crime_per_borough.groupBy("BORO_NM", "OFS_DESC").count().orderBy("count", ascending=False)

window_spec = Window.partitionBy("BORO_NM").orderBy(desc("count"))
ranked_df = most_reported_crime_per_borough.select("*",
dense_rank().over(window_spec).alias("rank"))

# Filter for only the top offense in each borough
result_df = ranked_df.filter(ranked_df.rank == 1)

# Select the relevant columns
crime_df = ny_crime_data.select("BORO_NM", "OFS_DESC")

# Filter out null values in BORO_NM and OFNS_DESC
crime_df = crime_df.filter((crime_df.BORO_NM.isNotNull()) &
(crime_df.OFS_DESC.isNotNull()))

# Group by borough and offense description, and count the number of
occurrences
grouped_df = crime_df.groupBy("BORO_NM",
"OFS_DESC").agg(count("*").alias("crime_count"))

# Rank the offenses in each borough by count
window_spec = Window.partitionBy("BORO_NM").orderBy(desc("crime_count"))

```

```

ranked_df = grouped_df.select("*",
dense_rank().over(window_spec).alias("rank"))

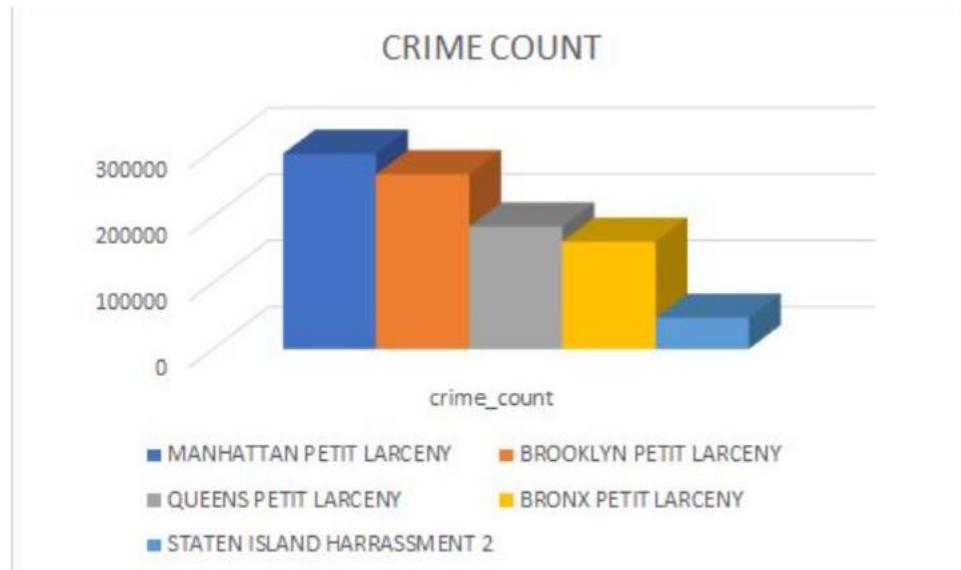
# Filter for only the top offense in each borough
result_df = ranked_df.filter(ranked_df.rank == 1)
filtered_df=result_df.filter(result_df.BORO_NM.isin("BRONX", "BROOKLYN", "MANHATTAN", "QUEENS", "STATEN ISLAND")).orderBy(desc("crime_count"))

```

Code Output Explanation:

According to the analysis of sales and crime data in New York, it can be inferred that the Manhattan neighborhoods with the highest crime rates have predominantly reported incidents of petit larceny, with a total count of over 250,000 reported crimes. This trend is followed by Brooklyn, Queens, and the Bronx. In contrast, the highest recorded incidents of harassment were reported in Staten Island.

Visualization:

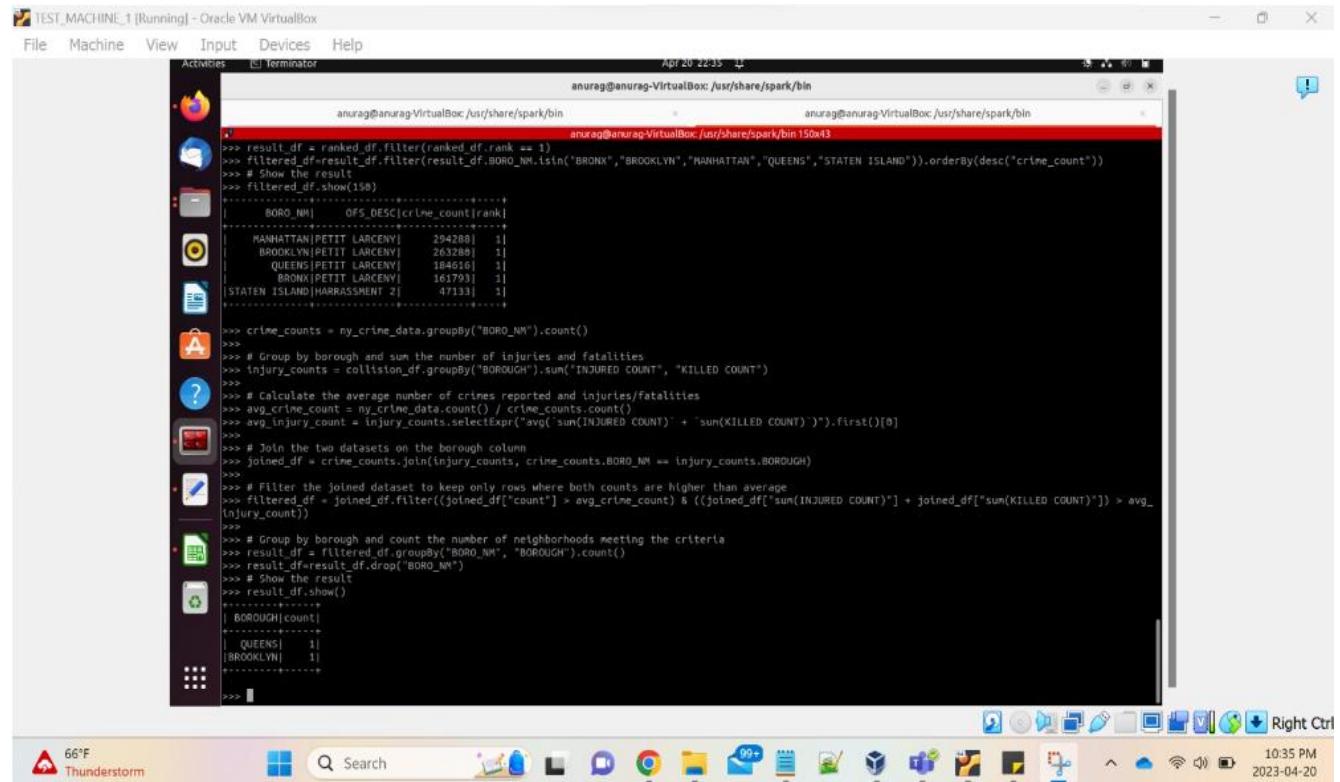


3. Are there any specific neighborhoods in New York City where both the number of crimes reported and the number of motor vehicle accidents where there is either injury or person killed are higher than average?

Business Conclusion:

In neighborhoods with high rates of motor vehicle accidents and crime, businesses may need to adjust their operations to mitigate risk and address safety concerns. Insurance companies may need to increase premiums or adjust policies, while real estate companies may have an opportunity to revitalize distressed areas and improve safety. Car rental companies may need to offer additional safety features or reassurances, and businesses relying on transportation and logistics may need to invest in additional safety training and equipment. Technology companies specializing in safety and security may see increased demand, and emergency response services may also experience increased demand in these areas.

Output:



The screenshot shows a Linux desktop environment with a terminal window open. The terminal window title is "TEST_MACHINE_1 [Running] - Oracle VM VirtualBox". The terminal content displays Python code for data analysis:

```
>>> result_df = ranked_df.filter(ranked_df.rank == 1)
>>> filtered_df=result_df.filter(result_df.BORO_NM.isin(["BRONX","BROOKLYN","MANHATTAN","QUEENS","STATEN ISLAND"])).orderBy(desc("crime_count"))
>>> # Show the result
>>> filtered_df.show(158)
+-----+
| BORO_NM| OF5_DESC|crime_count|rank|
+-----+
| MANHATTAN|PETIT LARCENY| 294200| 1|
| BROOKLYN|PETIT LARCENY| 263280| 1|
| QUEENS|PETIT LARCENY| 184616| 1|
| BRONX|PETIT LARCENY| 161793| 1|
| STATEN ISLAND|HARRASSMENT Z| 47133| 1|
+-----+
>>> crime_counts = ny_crime_data.groupBy("BORO_NM").count()
>>> # Group by borough and sum the number of injuries and fatalities
>>> injury_counts = collision_df.groupby("BOROUGH").sum("INJURED COUNT", "KILLED COUNT")
>>>
>>> # calculate the average number of crimes reported and injuries/fatalities
>>> avg_crime_count = ny_crime_data.count() / crime_counts.count()
>>> avg_injury_count = injury_counts.selectExpr("avg(sum(INJURED COUNT) + sum(KILLED COUNT))").first()[0]
>>>
>>> # Join the two datasets on the borough column
>>> joined_df = crime_counts.join(injury_counts, crime_counts.BORO_NM == injury_counts.BOROUGH)
>>>
>>> # Filter the joined dataset to keep only rows where both counts are higher than average
>>> filtered_df = joined_df.filter((joined_df["count"] > avg_crime_count) & ((joined_df["sum(INJURED COUNT)"] + joined_df["sum(KILLED COUNT)"]) > avg_injury_count))
>>>
>>> # Group by borough and count the number of neighborhoods meeting the criteria
>>> result_df = filtered_df.groupBy("BORO_NM", "BOROUGH").count()
>>> result_df=result_df.drop("BORO_NM")
>>> # Show the result
>>> result_df.show()
+-----+
| BOROUGH|count|
+-----+
| QUEENS| 1|
| BROOKLYN| 1|
+-----+
```

The screenshot shows the LibreOffice Calc application window. The title bar reads "part-00000-0bfa7a7d-2e22-46a5-9b5d-2e738798ae8b-c000.csv - LibreOffice Calc". The menu bar includes File, Edit, View, Insert, Format, Styles, Sheet, Data, Tools, Window, and Help. The toolbar contains various icons for file operations, text styles, and data manipulation. The spreadsheet has a single sheet with 17 columns labeled A through P. Row 1 contains the header "BOROUGH" in column A and "count" in column B. Row 2 contains the data "QUEENS" in column A and "1" in column B. Rows 3 through 34 are empty. The status bar at the bottom shows the file path "part-00000-0bfa7a7d-2e22-46a5-9b5d-2e738798ae8b-c000.csv" and various search and filter options.

SQL Code:

```

crime_counts = ny_crime_data.groupBy("BORO_NM").count()

# Group by borough and sum the number of injuries and fatalities
injury_counts = collision_df.groupBy("BOROUGH").sum("INJURED COUNT",
"KILLED COUNT")

# Calculate the average number of crimes reported and injuries/fatalities
avg_crime_count = ny_crime_data.count() / crime_counts.count()
avg_injury_count = injury_counts.selectExpr("avg(`sum(INJURED COUNT)` +
`sum(KILLED COUNT)`)").first()[0]

# Join the two datasets on the borough column
joined_df = crime_counts.join(injury_counts, crime_counts.BORO_NM ==
injury_counts.BOROUGH)

# Filter the joined dataset to keep only rows where both counts are higher
than average
filtered_df = joined_df.filter((joined_df["count"] > avg_crime_count) &
((joined_df["sum(INJURED COUNT)"] + joined_df["sum(KILLED COUNT)"]) >
avg_injury_count))

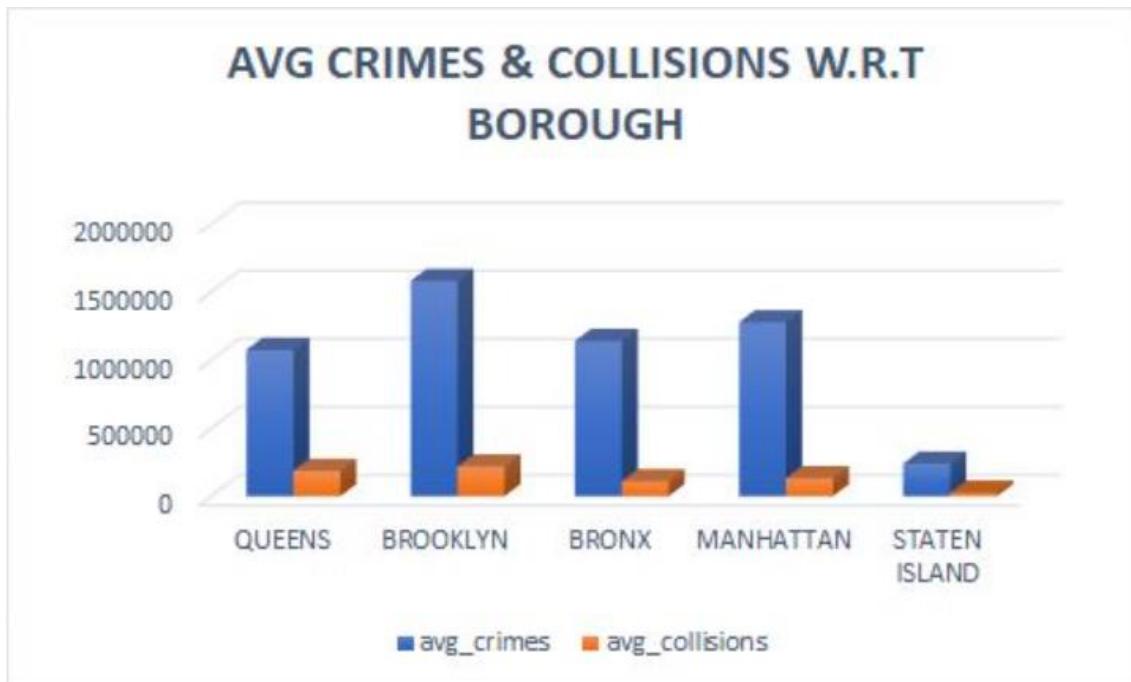
# Group by borough and count the number of neighborhoods meeting the
criteria
result_df = filtered_df.groupBy("BORO_NM", "BOROUGH").count()
result_df=result_df.drop("BORO_NM")
# Show the result
result_df.show()

```

Code Output Explanation:

The analytics indicate that the incidence of reported crimes and motor vehicle accidents resulting in injury or fatality are above the average in neighborhoods located in Queens and Brooklyn, New York City.

Visualization:

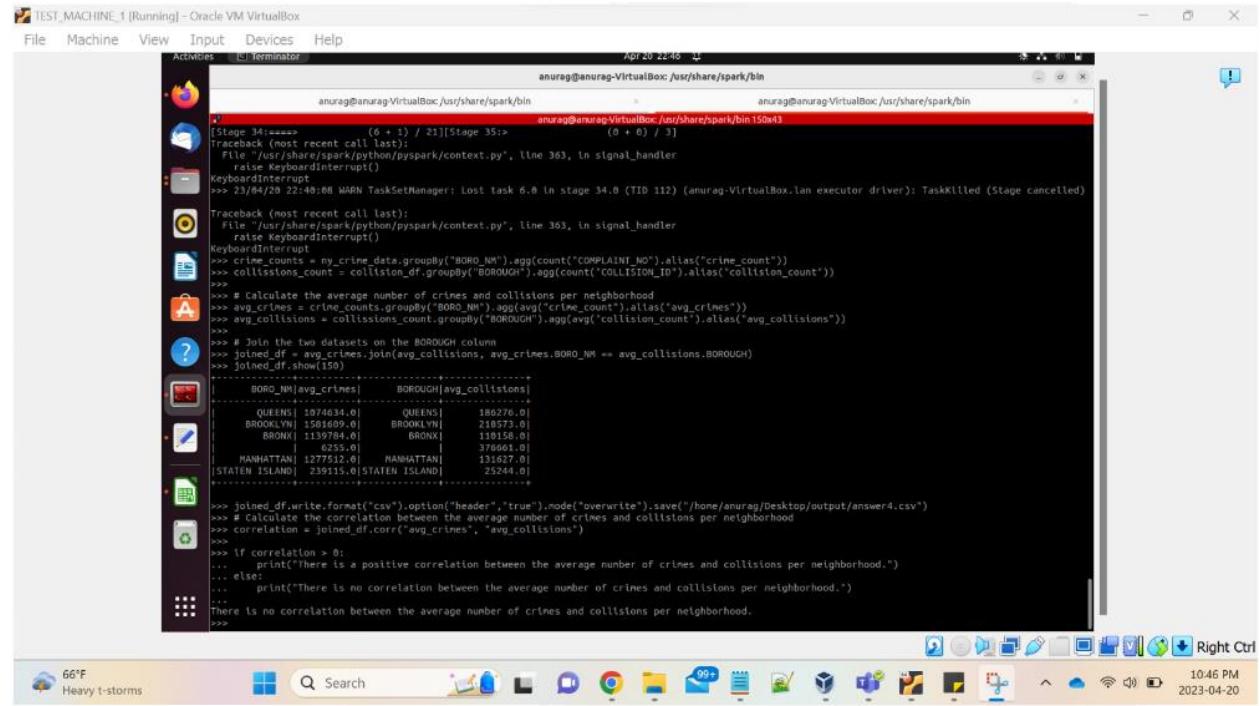


4. IS there any relation between crime and the accidents happening?

Business Conclusion:

Companies in the automotive industry, such as car dealerships, auto parts stores, and safety equipment manufacturers, may benefit from increased demand for safety equipment in areas with high numbers of motor vehicle collisions. Real estate companies may need to invest in security upgrades to make their properties more attractive to potential tenants or buyers in high-crime and high-accident areas. Transportation companies may need to adjust their routes to avoid problem areas, and companies with fleets of vehicles may need to invest in additional safety training and equipment. Retail stores may need to implement additional security measures to deter theft and ensure safety. High-crime areas may also have higher rates of accidents due to poor road conditions or a lack of infrastructure.

Output:



The screenshot shows a Linux desktop environment with a terminal window open. The terminal window title is "anurag@anurag-VirtualBox: /usr/share/spark/bin". The terminal content displays a Python script running on a Spark environment. The script performs several operations:

- It reads data from two datasets: "ny_crime_data" and "ny_collision_data".
- It groups the data by "BORO_NM" and calculates the count of complaints and collisions per neighborhood.
- It calculates the average number of crimes and collisions per neighborhood.
- It joins the two datasets on the "BOROUGH" column.
- It saves the joined dataset to a CSV file named "answer4.csv".
- It calculates the correlation between the average number of crimes and collisions per neighborhood.

The terminal output shows the following data for neighborhoods:

BORO_NM[avg_crimes]	BOROUGH[avg_collisions]
QUEENS 1074634.0	QUEENS 186276.0
BROOKLYN 1501609.0	BROOKLYN 218573.0
BRONX 1139784.0	BRONX 119156.0
MANHATTAN 1277512.0	MANHATTAN 131837.0
STATEN ISLAND 239115.0	STATEN ISLAND 25244.0

The script concludes with a message: "There is no correlation between the average number of crimes and collisions per neighborhood."

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	BORO_NM	avg_crimes	BOROUGH	avg_collisions										
2	QUEENS	1074834	QUEENS	186278										
3	BROOKLYN	1581609	BROOKLYN	210573										
4	BRONX	1139784	BRONX	110158										
5		6255		376661										
6	MANHATTAN	1277512	MANHATTAN	131627										
7	STATEN ISLAND	239115	STATEN ISLAND	25244										

SQL Code:

```

crime_counts =
ny_crime_data.groupBy("BORO_NM").agg(count("COMPLAINT_NO")).alias("crime_count"))
collissions_count =
collision_df.groupBy("BOROUGH").agg(count("COLLISION_ID")).alias("collision_count"))
# Calculate the average number of crimes and collisions per neighborhood
avg_crimes =
crime_counts.groupBy("BORO_NM").agg(avg("crime_count")).alias("avg_crimes")
)
avg_collisions =
collissions_count.groupBy("BOROUGH").agg(avg("collision_count")).alias("avg_collisions"))
# Join the two datasets on the BOROUGH column
joined_df = avg_crimes.join(avg_collisions, avg_crimes.BORO_NM ==
avg_collisions.BOROUGH)
joined_df.write.format("csv").option("header","true").mode("overwrite").save("/home/anurag/Desktop/output/answer4.csv")
# Calculate the correlation between the average number of crimes and
collisions per neighborhood
correlation = joined_df.corr("avg_crimes", "avg_collisions")
if correlation > 0:
    print("There is a positive correlation between the average number of
crimes and collisions per neighborhood.")
else:
    print("There is no correlation between the average number of crimes
and collisions per neighborhood.")

```

Code Output Explanation:

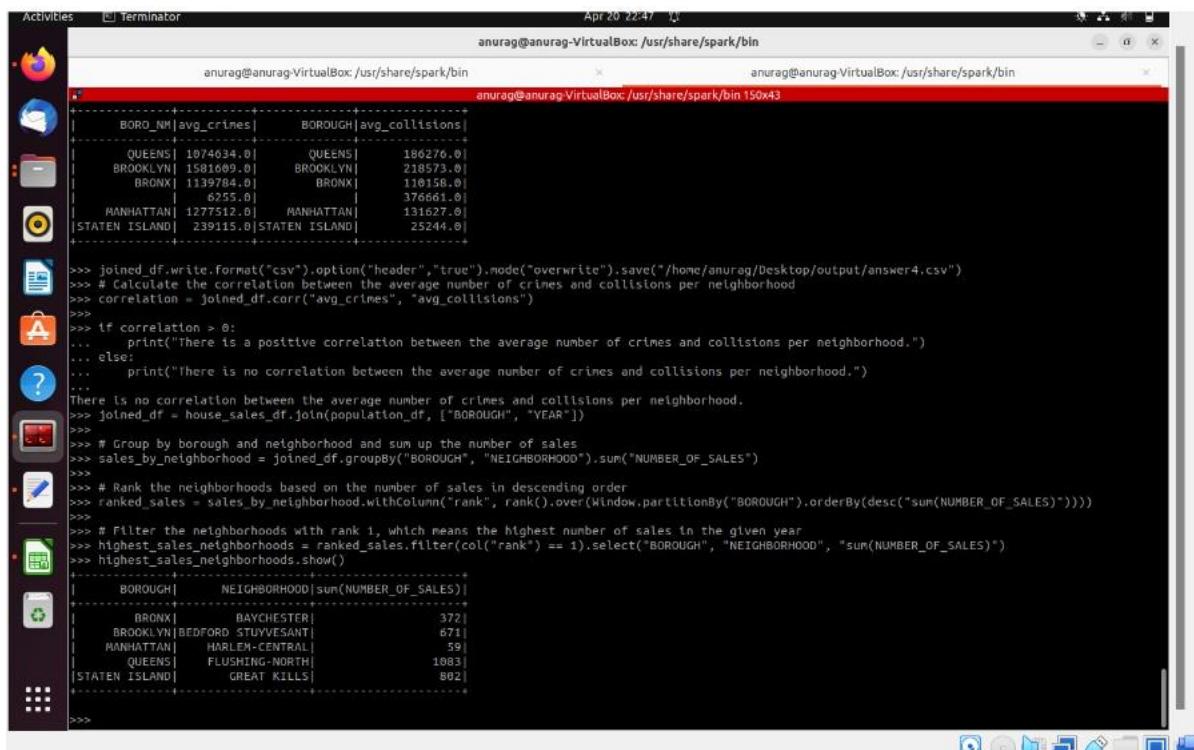
Based on our analysis, we have found that there is no correlation between the average number of motor vehicle collisions and the frequency of certain types of crimes such as theft or assault in specific areas. Manhattan had been recorded to have highest average number of crimes, followed by the Bronx, whereas Brooklyn had the second-highest average number of collisions, and the Bronx was fourth on the list. Therefore, we can conclude that there is no direct relationship between the occurrence of crimes and motor vehicle collisions in a given area.

5.Which neighborhood in each borough had the highest number of sales ?

Business Conclusion:

The high number of property sales in a given neighborhood can be influenced by various factors such as location, affordability, housing quality, market conditions, effective marketing and advertising, investment and development, and demographic changes. Neighborhoods located in desirable areas close to employment hubs or popular attractions are often in high demand, as well as affordable neighborhoods that offer good value for money. Housing quality, favorable market conditions, effective marketing, and advertising campaigns can also attract potential buyers. Investment and development efforts, as well as demographic changes, such as an influx of young professionals or retirees, can further drive up demand and lead to increased property sales in a neighborhood.

Output:



The screenshot shows a terminal window titled "Terminator" running on a Linux desktop. The terminal has two tabs open, both showing the command line prompt "anurag@anurag-VirtualBox: /usr/share/spark/bin". The code in the terminal is as follows:

```
anurag@anurag-VirtualBox: /usr/share/spark/bin
anurag@anurag-VirtualBox: /usr/share/spark/bin 150x43
+-----+
| BORO_NM|avg_crimes|      | BOROUGH|avg_collisions|
+-----+-----+-----+-----+
| QUEENS| 1074634.0|      | QUEENS| 186276.0|
| BROOKLYN| 1581609.0|      | BROOKLYN| 218573.0|
| BRONX| 1139784.0|      | BRONX| 110156.0|
|       | 6255.0|      |       | 376661.0|
| MANHATTAN| 1277512.0|      | MANHATTAN| 131627.0|
| STATEEN ISLAND| 239115.0|      | STATEEN ISLAND| 25244.0|
+-----+-----+-----+-----+
>>> joined_df.write.format("csv").option("header","true").mode("overwrite").save("/home/anurag/Desktop/output/answer4.csv")
>>> # Calculate the correlation between the average number of crimes and collisions per neighborhood
>>> correlation = joined_df.corr("avg_crimes", "avg_collisions")
>>>
>>> if correlation > 0:
...     print("There is a positive correlation between the average number of crimes and collisions per neighborhood.")
... else:
...     print("There is no correlation between the average number of crimes and collisions per neighborhood.")
...
There is no correlation between the average number of crimes and collisions per neighborhood.
>>> joined_df = house_sales_df.join(population_df, ["BOROUGH", "YEAR"])
>>>
>>> # Group by borough and neighborhood and sum up the number of sales
>>> sales_by_neighborhood = joined_df.groupBy("BOROUGH", "NEIGHBORHOOD").sum("NUMBER_OF_SALES")
>>>
>>> # Rank the neighborhoods based on the number of sales in descending order
>>> ranked_sales = sales_by_neighborhood.withColumn("rank", rank().over(Window.partitionBy("BOROUGH").orderBy(desc("sum(NUMBER_OF_SALES)))))
>>>
>>> # Filter the neighborhoods with rank 1, which means the highest number of sales in the given year
>>> highest_sales_neighborhoods = ranked_sales.filter(col("rank") == 1).select("BOROUGH", "NEIGHBORHOOD", "sum(NUMBER_OF_SALES)")
>>> highest_sales_neighborhoods.show()
+-----+
| BOROUGH| NEIGHBORHOOD|sum(NUMBER_OF_SALES)|
+-----+
| BRONX| BAYCHESTER| 372|
| BROOKLYN| BEDFORD STUYVESANT| 671|
| MANHATTAN| HARLEM-CENTRAL| 59|
| QUEENS| FLUSHING-NORTH| 1083|
| STATEEN ISLAND| GREAT KILLS| 802|
+-----+
>>>
```

The screenshot shows a LibreOffice Calc spreadsheet window titled "part-00000-318ca2ce-93b5-4874-9e13-da96b1d35c09-c000.csv - LibreOffice Calc". The table has columns labeled A, B, and C. Column A contains Borough names, column B contains Neighborhood names, and column C contains the sum of Number of Sales. The data is as follows:

A	B	C
1	BOROUGH	NEIGHBORHOOD
2	BRONX	BAYCHESTER
3	BROOKLYN	BEDFORD STUYVESANT
4	MANHATTAN	HARLEM-CENTRAL
5	QUEENS	FLUSHING-NORTH
6	STATEN ISLAND	GREAT KILLS
7		sum(NUMBER_OF_SALES)
8		372
9		671
10		59
11		1083
12		802
13		
14		
15		
16		
17		
18		
19		
20		
21		
22		
23		
24		
25		
26		
27		
28		

SQL Code:

```
# Join sales_data and population_data dataframes based on borough and year
joined_df = house_sales_df.join(population_df, ["BOROUGH", "YEAR"])

# Group by borough and neighborhood and sum up the number of sales
sales_by_neighborhood = joined_df.groupBy("BOROUGH",
"NEIGHBORHOOD").sum("NUMBER_OF_SALES")

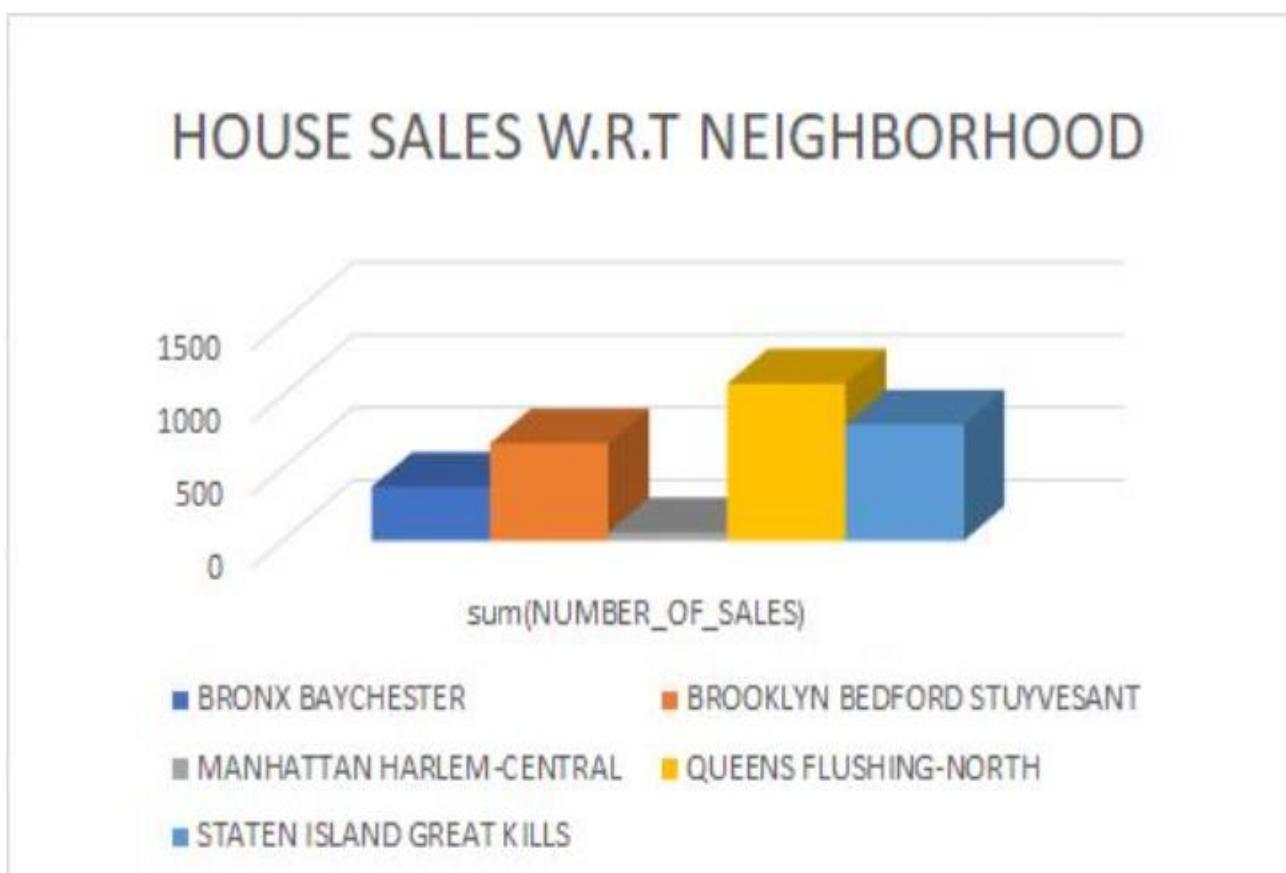
# Rank the neighborhoods based on the number of sales in descending order
ranked_sales = sales_by_neighborhood.withColumn("rank",
rank().over(Window.partitionBy("BOROUGH").orderBy(desc("sum(NUMBER_OF_SALES)"))))

# Filter the neighborhoods with rank 1, which means the highest number of
# sales in the given year
highest_sales_neighborhoods = ranked_sales.filter(col("rank") ==
1).select("BOROUGH", "NEIGHBORHOOD", "sum(NUMBER_OF_SALES)")
highest_sales_neighborhoods.show()
```

Code Output Explanation:

The analysis revealed that Flushing North neighborhood in Queens had the highest number of sales, with a total of approximately 1083. Great Kills neighborhood in Staten Island ranked second with around 802 sales. On the other hand, Harlem Central in Manhattan had the least number of sales, recording only 59 in the given year.

Visualization:



6. Is there any correlation between the population of a neighborhood and the average sale price of homes in that neighborhood?

Business Conclusion:

The correlation between the population of a neighborhood and the average sale price of homes in that neighborhood can have several business implications. Real estate developers and investors can use this information to make informed decisions about where to focus their efforts, such as identifying neighborhoods with a growing population and potential for increased property values. It can also inform pricing strategies for properties in different neighborhoods, with higher population areas potentially commanding higher prices. Real estate agents can use this information to tailor their marketing and outreach efforts to attract potential buyers in areas with a specific population size. Additionally, businesses such as retail stores and restaurants can use this information to identify areas with a growing population to establish their presence and attract customers. Overall, understanding the correlation between population and home prices can inform various business decisions within the real estate and retail industries.

Output:

```
Activities Terminator Apr 20 22:49
anurag@anurag-VirtualBox:/usr/share/spark/bin

anurag@anurag-VirtualBox:/usr/share/spark/bin>>> ranked_sales = sales_by_neighborhood.withColumn("rank", rank().over(Window.partitionBy("BOROUGH").orderBy(desc("sum(NUMBER_OF_SALES)))))
>>>
>>> # Filter the neighborhoods with rank 1, which means the highest number of sales in the given year
>>> highest_sales_neighborhoods = ranked_sales.filter(col("rank") == 1).select("BOROUGH", "NEIGHBORHOOD", "sum(NUMBER_OF_SALES)")
>>> highest_sales_neighborhoods.show()
+-----+-----+-----+
| BOROUGH| NEIGHBORHOOD|sum(NUMBER_OF_SALES)|
+-----+-----+-----+
| BRONX| BAYCHESTER| 372|
| BROOKLYN| BEDFORD STUYVESANT| 671|
| MANHATTAN| HARLEM-CENTRAL| 59|
| QUEENS| FLUSHING-NORTH| 1083|
| STATEN ISLAND| GREAT KILLS| 802|
+-----+-----+-----+
>>> joined_df = house_sales_df.join(population_df, ["BOROUGH", "YEAR"])
>>>
>>> # Group by borough and neighborhood and calculate the average sale price and total population
>>>
>>> avg_price_population = joined_df.groupBy("BOROUGH").agg(avg("AVERAGE_SALE_PRICE").alias("avg_sale_price"), sum("Total Population").alias("total_population"))
>>>
>>> avg_price_population.show(150)
+-----+-----+
| BOROUGH| avg_sale_price|total_population|
+-----+-----+
| QUEENS| 732068.9616613418| 716856627|
| BROOKLYN| 1249836.084592145| 860873347|
| BRONX| 619251.4270833334| 271862016|
| MANHATTAN| 5307903.37113402| 156240449|
| STATEN ISLAND| 509913.2541000067| 114724025|
+-----+-----+
>>> # Calculate the correlation between the average sale price and total population
>>> correlation = avg_price_population.select(corr("avg_sale_price", "total_population")).collect()[0][0]
>>>
>>> if correlation > 0:
...     print("There is a positive correlation between the avg_sale_price and total_population.")
... else:
...     print("There is no correlation between the avg_sale_price and total_population.")
...
There is no correlation between the avg_sale_price and total_population.
>>>
```

SQL Code:

```
joined_df = house_sales_df.join(population_df, ["BOROUGH", "YEAR"])

# Group by borough and neighborhood and calculate the average sale price
# and total population

avg_price_population=joined_df.groupBy("BOROUGH").agg(avg("AVERAGE_SALE_PRICE").alias("avg_sale_price"), sum("Total Population").alias("total_population"))

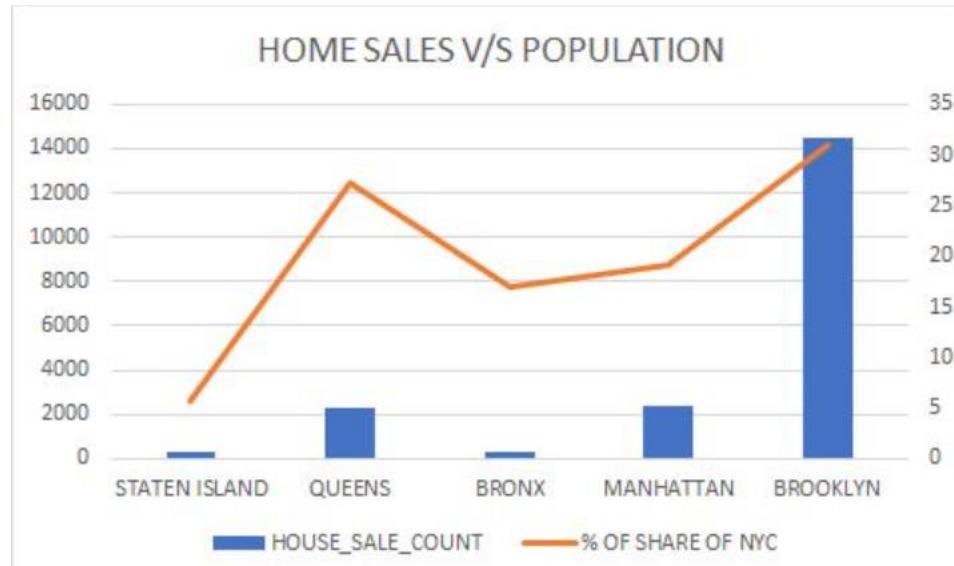
avg_price_population.show(150)
# Calculate the correlation between the average sale price and total population
correlation = avg_price_population.select(corr("avg_sale_price", "total_population")).collect()[0][0]

if correlation > 0:
    print("There is a positive correlation between the avg_sale_price and total_population.")
else:
    print("There is no correlation between the avg_sale_price and total_population.")
```

Code Output Explanation:

Based on the analysis, it can be concluded that there is no correlation or association between the population of a neighborhood and the average sale price of homes in that neighborhood.

Visualization:



REFERENCES

NEW YORK HOME SALES DATA: <https://data.cityofnewyork.us/City-Government/DOF-Summary-of-Neighborhood-Sales-by-Neighborhood-/5ebm-myj7> (0.00578GB)

NYPD COMPLAINT DATA: <https://data.cityofnewyork.us/Public-Safety/NYPD-Complaint-Data-Historic/qgea-i56i> (2.45GB)

MOTOR VEHICAL COLLISIONS DATA: <https://data.cityofnewyork.us/Public-Safety/Motor-Vehicle-Collisions-Crashes/h9gi-nx95> (0.21GB)

NEW YORK POPULATION BY BOROUGH: <https://data.cityofnewyork.us/City-Government/New-York-City-Population-by-Borough-1950-2040/xywu-7bv9> (2KB)

Total Dataset size = 2.7GB (approx..)



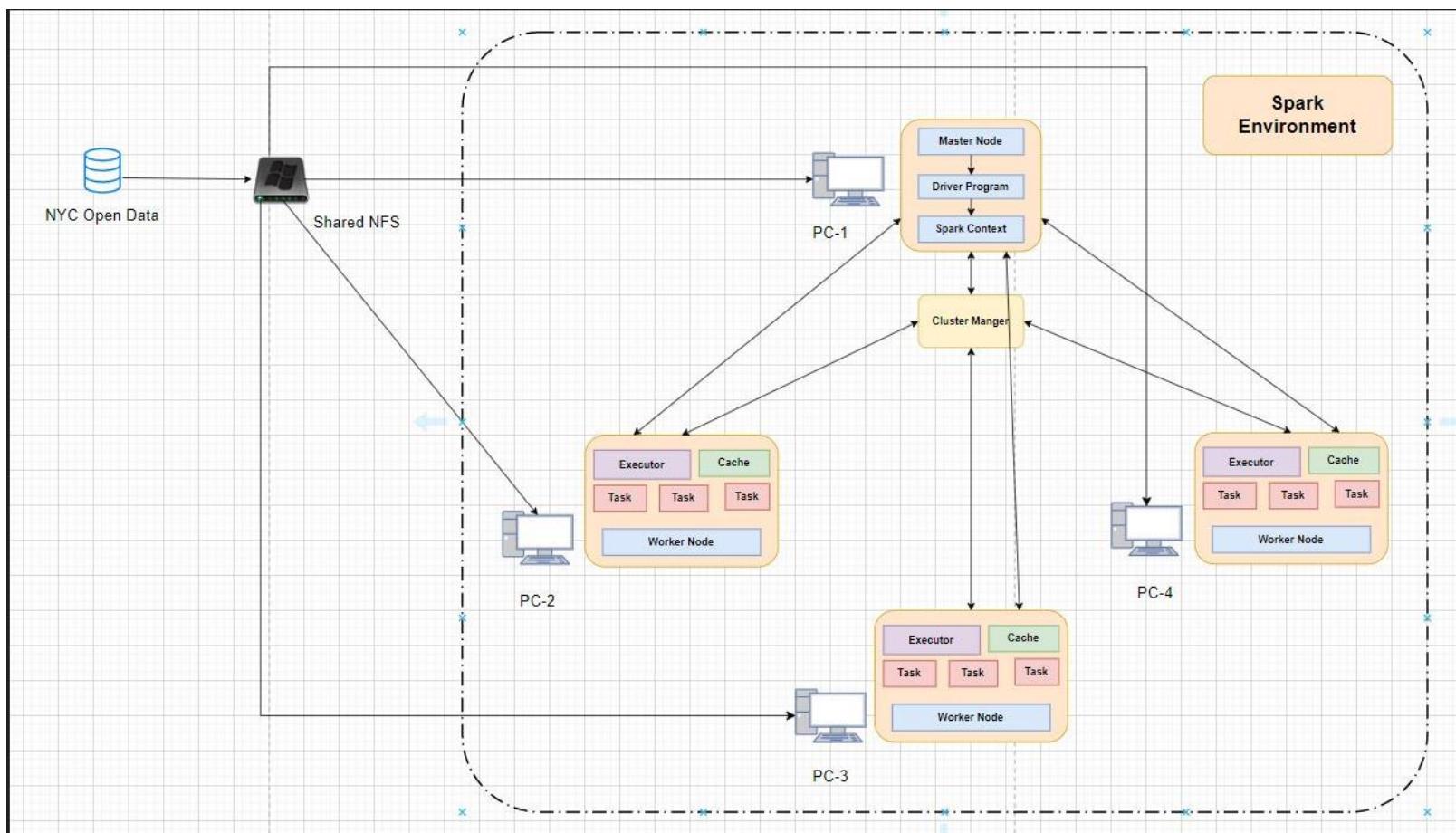
NextGen Analytical Solutions

Analysis of the Factors Contributing to Changes in Crime Rates and Property Values in New York City

Anurag S S S M
Lekhashree J
Meghana N
Satya Vinod K



ARCHITECTURE



- **Distributed Computing System**
- **Network File System (NFS)**
- **Master-Slave Architecture**
- **Data Loading**
- **Parallel Processing**

NFS

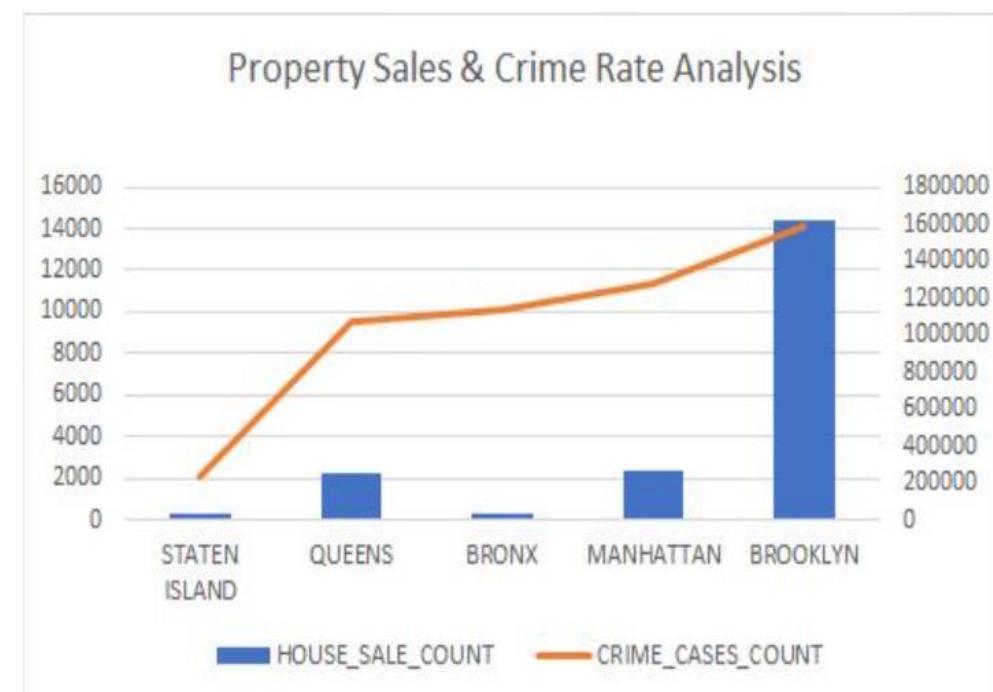
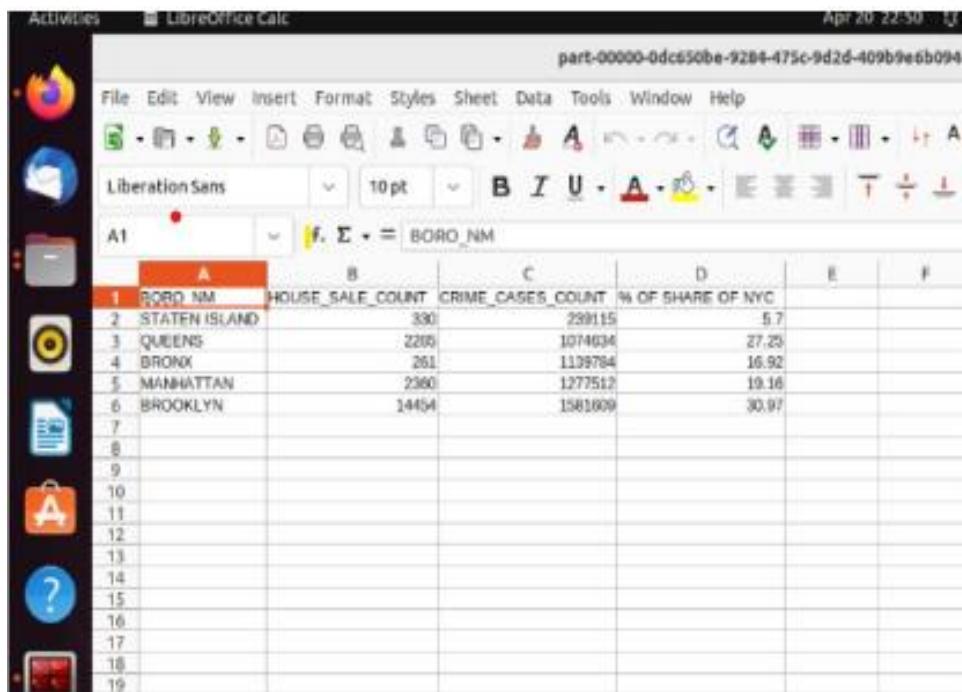
NFS (Network File System) is a distributed file system protocol that enables multiple users to access and share files over a network. It provides a scalable and centralized storage solution for big data applications, allowing data to be stored on a central server and accessed by multiple clients. NFS can improve the performance of big data applications by providing high-speed access to data, and it can handle large volumes of data transfers with minimal latency. Overall, NFS is a valuable tool for managing and storing big data.

Cluster architecture

Cluster architecture is commonly used in large-scale computing systems, such as big data processing, scientific computing, and high-performance computing. It allows for efficient processing of large volumes of data by dividing the workload among multiple nodes, enabling parallel processing and reducing the time it takes to process data.

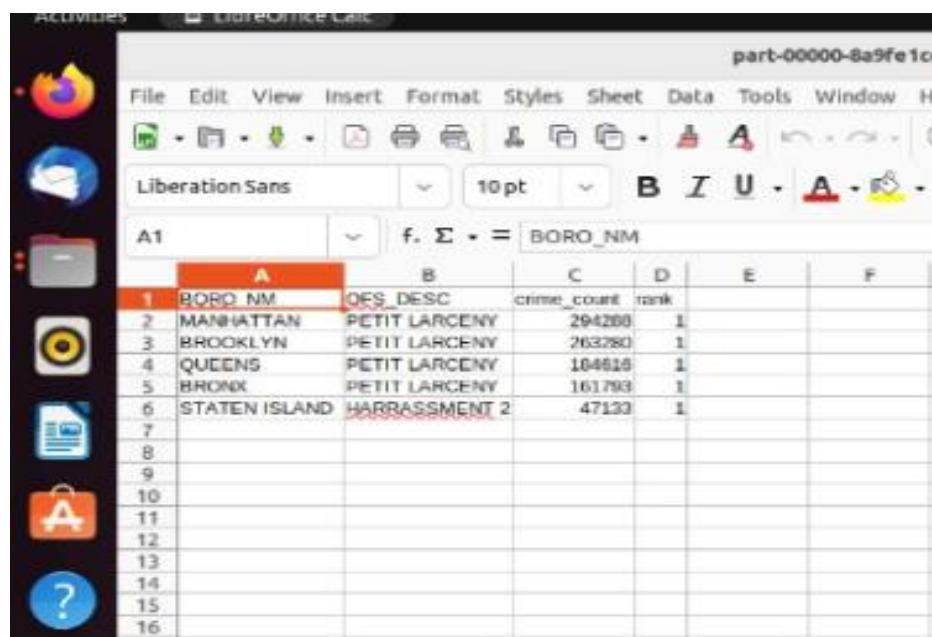
1. How can we correlate the population of a neighborhood, crime rate and the average sale price of homes in that neighborhood?

The correlation between population, crime rate, and home prices in a neighborhood can affect several businesses. Real estate and homebuilding businesses can identify profitable investment opportunities based on this data, while retail businesses can benefit from areas with higher populations and purchasing power. Security system and neighborhood watch businesses can benefit from areas with higher crime rates, while other businesses may suffer if an area becomes less desirable to customers.



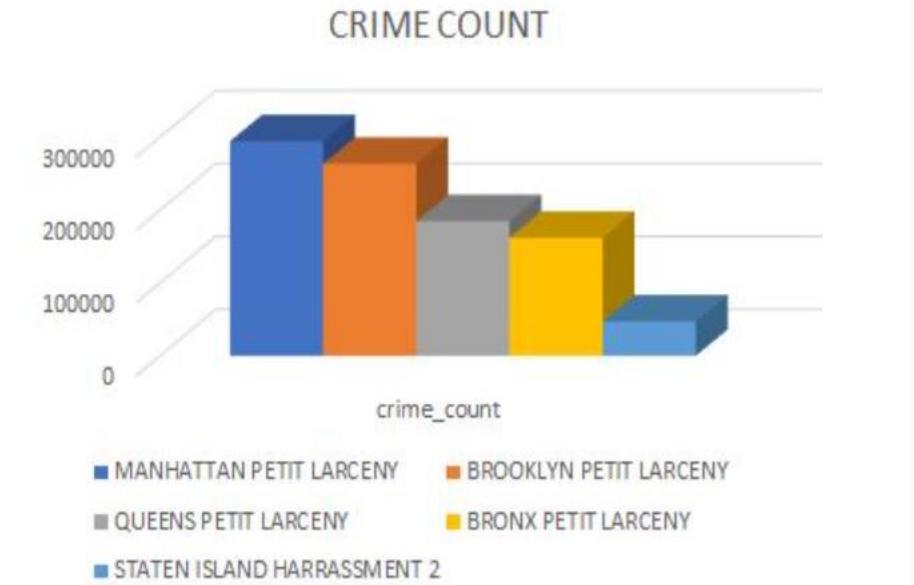
2.What type of crimes are most commonly reported in each borough, and are there any patterns or trends that can be identified?

Businesses operating in areas with high crime rates such as petit larceny and harassment may need to implement safety measures to protect their customers and employees. This could include increasing security personnel or installing surveillance cameras. They may also face financial implications such as higher insurance premiums and loss of revenue. Technology companies specializing in safety and security may see increased demand. Addressing the underlying causes of crime may require a collaborative effort from businesses, law enforcement, and community members.



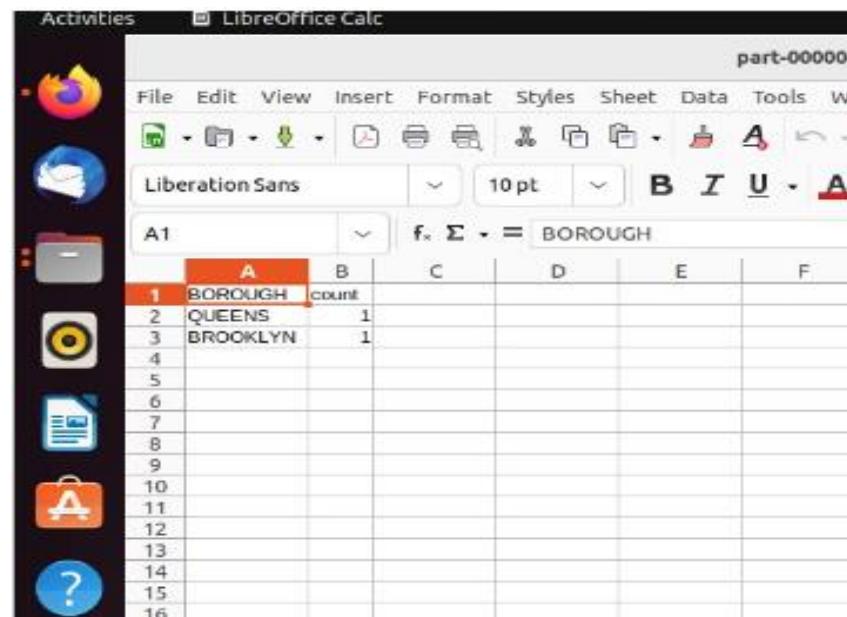
A screenshot of a Microsoft Word document titled "part-00000-8a9fe1ce". The document contains a table with the following data:

BORO_NM	OES_DESC	crime_count	rank
MANHATTAN	PETIT LARCENY	294200	1
BROOKLYN	PETIT LARCENY	263280	1
QUEENS	PETIT LARCENY	164618	1
BRONX	PETIT LARCENY	161793	1
STATEN ISLAND	HARRASSMENT 2	47133	1



3.Are there any specific neighborhoods in New York City where both the number of crimes reported and the number of motor vehicle accidents where there is either injury or person killed are higher than average?

Businesses operating in high-crime and high-accident areas should prioritize safety to mitigate risks and address safety concerns. This may involve offering additional safety features, investing in safety training and equipment, and partnering with technology companies specializing in safety and security. Revitalization of distressed areas and improvement of safety by real estate companies may provide opportunities, while emergency response services and insurance companies may see increased demand.



The screenshot shows a LibreOffice Calc spreadsheet window titled "LibreOffice Calc". The window has a dark interface with a toolbar at the top containing icons for file operations, text styling, and data manipulation. The main area displays a table with the following data:

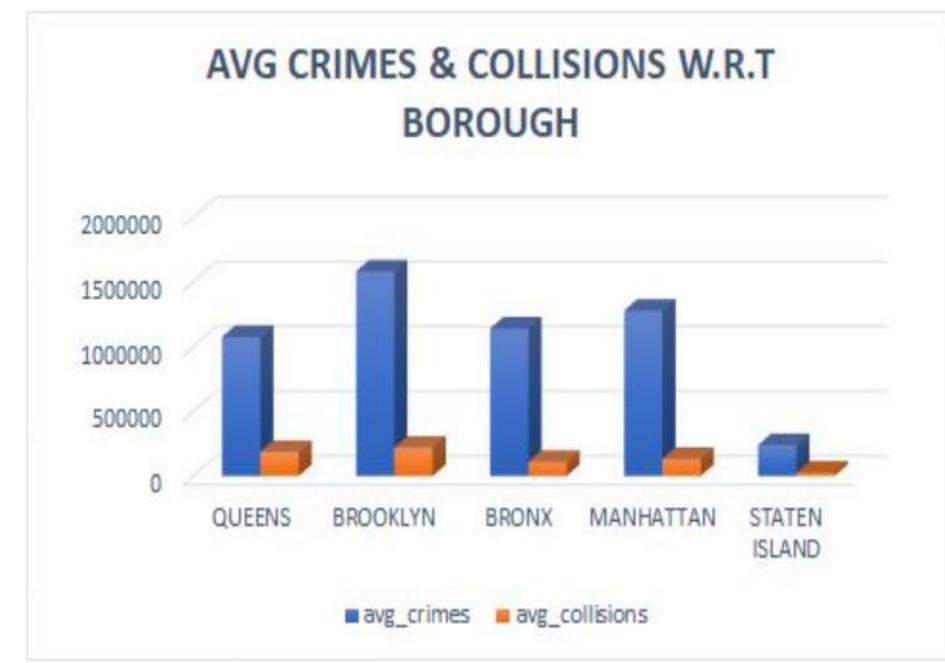
A	B	C	D	E	F
1	BOROUGH	count			
2	QUEENS	1			
3	BROOKLYN	1			
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					

4. Is there any relation between crime and the accidents happening?

Businesses in the automotive industry may experience higher demand for safety equipment in areas with high rates of motor vehicle collisions, while real estate companies may need to invest in security upgrades to attract tenants or buyers in high-crime areas. Transportation companies may need to adjust their operations for safety, and retail stores may need to implement additional security measures. Poor road conditions and inadequate infrastructure in high-crime areas may also affect businesses relying on transportation and logistics.

The screenshot shows a LibreOffice Calc spreadsheet titled "part-00000-7b326a2f-7623-4d8f-9406-". The spreadsheet contains data for five boroughs: QUEENS, BROOKLYN, BRONX, MANHATTAN, and STATEN ISLAND. Column A lists the borough names, column B lists the average number of crimes, and column C lists the average number of collisions. The data is as follows:

A	B	C
1 BORO_NM	avg_crimes	BOROUGH
2 QUEENS	1074634	QUEENS
3 BROOKLYN	1501609	BROOKLYN
4 BRONX	1139784	BRONX
5	6255	
6 MANHATTAN	1277512	MANHATTAN
7 STATEN ISLAND	239115	STATEN ISLAND
8		
9		
10		
11		
12		
13		
14		
15		
16		
17		
18		
19		



5. Which neighborhood in each borough had the highest number of sales?

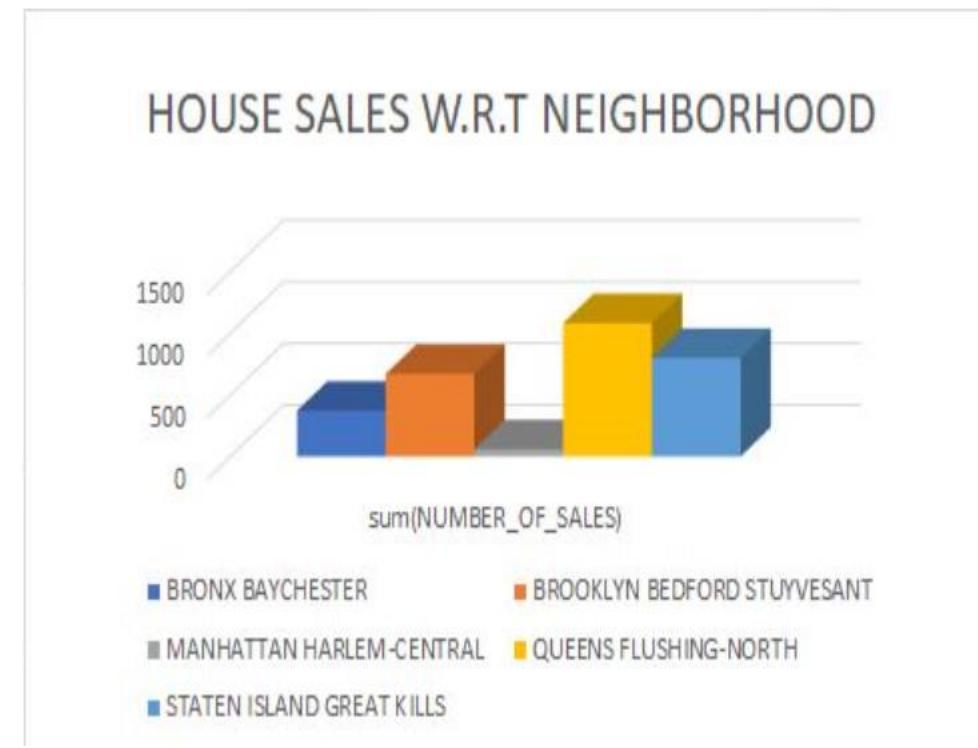
Real estate businesses can benefit from identifying the factors that influence the number of property sales in a neighborhood. These include location, affordability, housing quality, market conditions, effective marketing, investment and development, and demographic changes. Neighborhoods located in desirable areas, offering good value for money, and experiencing investment and development efforts or demographic changes are often in high demand, leading to increased property sales. Businesses can develop strategies to target potential buyers based on these factors.

Activities LibreOffice Calc Apr 20 22:54 part-00000-318ca2ce-93b5-4874-9e13-da96b1d3:

File Edit View Insert Format Styles Sheet Data Tools Window Help

Liberation Sans 10 pt B I U A f. Σ = BOROUGH

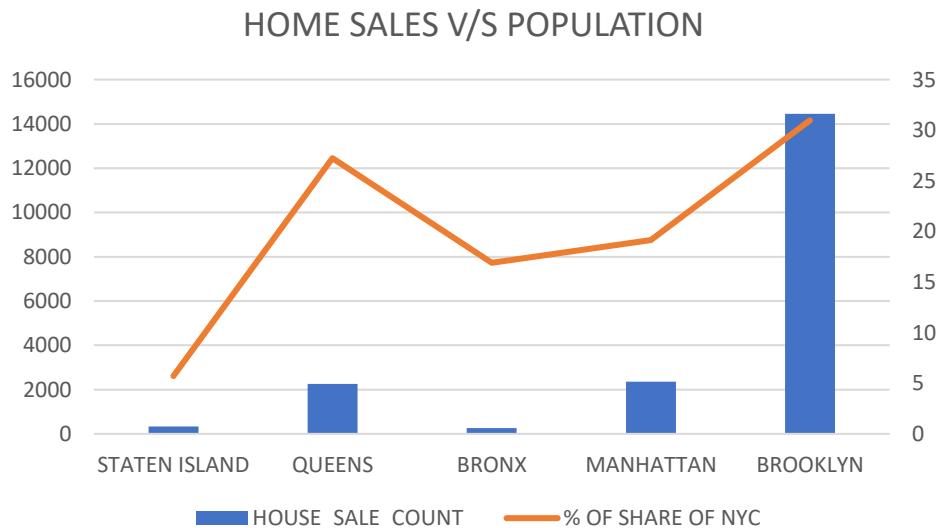
A	B	C	D	E	F
1	BOROUGH	NEIGHBORHOOD	sum(NUMBER_OF_SALES)		
2	BRONX	BAYCHESTER	372		
3	BROOKLYN	BEDFORD STUYVESANT	671		
4	MANHATTAN	HARLEM-CENTRAL	59		
5	QUEENS	FLUSHING-NORTH	1083		
6	STATEN ISLAND	GREAT KILLS	802		



6. Is there any correlation between the population of a neighborhood and the average sale price of homes in that neighborhood?

The relationship between neighborhood population and home sale prices can impact the decisions of businesses in the real estate and retail industries. Real estate investors and developers can focus on neighborhoods with a growing population and potential for increased property values. Real estate agents can tailor marketing strategies to areas with a specific population size. Retail businesses can identify areas with a growing population to attract customers.

A	B	C
1	BORO_NM	HOUSE_SALE_COUNT
2	STATEN ISLAND	330
3	QUEENS	2265
4	BRONX	261
5	MANHATTAN	2360
6	BROOKLYN	14454



REFERENCES

- **NEW YORK HOME SALES DATA:** <https://data.cityofnewyork.us/City-Government/DOF-Summary-of-Neighborhood-Sales-by-Neighborhood-/5ebm-myj7> (0.00578GB)
- **NYPD COMPLAINT DATA:** <https://data.cityofnewyork.us/Public-Safety/NYPD-Complaint-Data-Historic/qgea-i56i> (2.45GB)
- **MOTOR VEHICAL COLLISIONS DATA:** <https://data.cityofnewyork.us/Public-Safety/Motor-Vehicle-Collisions-Crashes/h9gi-nx95> (0.21GB)
- **NEW YORK POPULATION BY BOROUGH:** <https://data.cityofnewyork.us/City-Government/New-York-City-Population-by-Borough-1950-2040/xywu-7bv9> (2KB)

Total Dataset size = 2.7GB (approx..)



Thank you