Big Data Project

AM	03003148
email	filippou.michail@gmail.com
github	https://github.com/choosen23/Los-Angeles-Crimes-Analysis
Name	Filippou Michail

Requirement 1

Name	Link
Spark	http://83.212.72.191:8080/
Hadoop	http://83.212.72.191:9870/dfshealth.html#tab-overview
passwd	7uXhwN75B9

Requirement 2

The commands to upload the files to hadoop

```
hadoop fs -mkdir -p ~/data

hadoop fds-put la_crimes_1.csv la_crimes_2.csv LA_income_2015.cs
```

```
user@master:~$ hadoop fs -ls ~/data
Found 10 items
-rw-r--r--   2 user supergroup       1502 2024-06-08 22:31 /home/user/data/LAPD_Police_Stations.csv
drwxr-xr-x   - user supergroup          0 2024-06-08 23:14 /home/user/data/LAPD_Police_Stations.parquet
-rw-r--r--   2 user supergroup      12859 2024-06-08 22:31 /home/user/data/LA_income_2015.csv
drwxr-xr-x   - user supergroup          0 2024-06-08 23:14 /home/user/data/LA_income_2015.parquet
-rw-r--r--   2 user supergroup  537190637 2024-06-08 22:31 /home/user/data/la_crimes_1.csv
drwxr-xr-x   - user supergroup          0 2024-06-08 23:15 /home/user/data/la_crimes_1.parquet
-rw-r--r--   2 user supergroup  241898956 2024-06-08 22:31 /home/user/data/la_crimes_2.csv
drwxr-xr-x   - user supergroup          0 2024-06-08 23:15 /home/user/data/la_crimes_2.parquet
-rw-r--r--   2 user supergroup     897062 2024-06-08 22:31 /home/user/data/revgecoding.csv
drwxr-xr-x   - user supergroup          0 2024-06-08 23:15 /home/user/data/revgecoding.parquet
user@master:~$
```

Requirement 3

Results

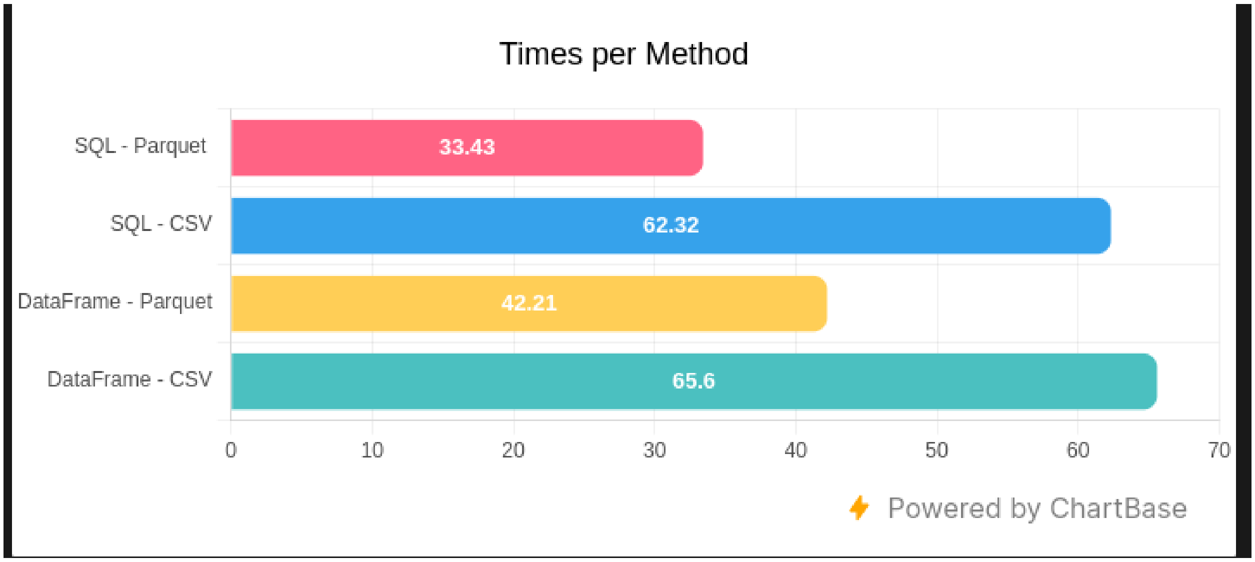
year	month	crime_total	ranking
2010	1	19520	1
2010	3	18131	2
2010	7	17857	3
2011	1	18141	1
2011	7	17283	2
2011	10	17034	3
2012	1	17954	1
2012	8	17661	2
2012	5	17502	3
2013	8	17441	1
2013	1	16828	2
2013	7	16645	3
2014	10	17331	1
2014	7	17258	2
2014	12	17198	3
2015	10	19221	1
2015	8	19011	2

```
|2015|7    |18709      |3      |
|2016|10   |19660      |1      |
|2016|8    |19496      |2      |
|2016|7    |19450      |3      |
|2017|10   |20437      |1      |
|2017|7    |20199      |2      |
|2017|1    |19849      |3      |
|2018|5    |19976      |1      |
|2018|7    |19879      |2      |
|2018|8    |19765      |3      |
|2019|7    |19126      |1      |
|2019|8    |18987      |2      |
|2019|3    |18865      |3      |
|2020|1    |18542      |1      |
|2020|2    |17273      |2      |
|2020|5    |17221      |3      |
|2021|10   |19328      |1      |
|2021|7    |18673      |2      |
|2021|8    |18389      |3      |
|2022|5    |20453      |1      |
|2022|10   |20315      |2      |
|2022|6    |20257      |3      |
|2023|10   |20040      |1      |
|2023|8    |20029      |2      |
|2023|1    |19908      |3      |
|2024|1    |18772      |1      |
|2024|2    |17244      |2      |
|2024|3    |16074      |3      |
+----+-----+-----------+-------+
```

Times per Method

Aa API	# Time
DataFrame - CSV	65.60165596008301

Aa API	# Time
DataFrame - Parquet	42.213536739349365
SQL - CSV	62.3171060085296
SQL - Parquet	33.43423843383789



Requirement 4

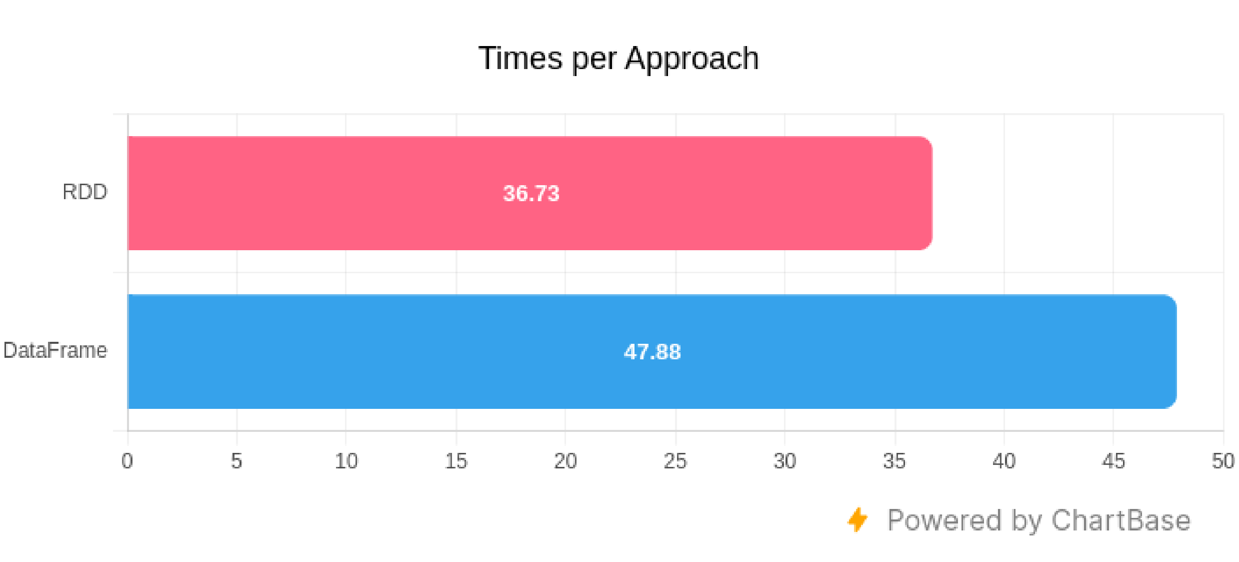
Results

```
+---------+------+
|part_day |count |
+---------+------+
|Night    |243815|
|Evening  |192164|
|Afternoon|151816|
```

```
|Morning  |126865|
+---------+------+
```

Times per Approach

Aa Approach	# Time
DataFrame	47.881619691848755
RDD	36.734586000442505



Requirement 5

Before answering the queries lets understand some things.

Catalyst Optimizer

A novel query optimizer for Spark using advance techniques like pattern matching etc. [1]

Join Strategy.

Based on the factors of the data, like data size, data distribution, if there are and how are joined, the format, the spark selects different methods to do joins the data.

1. Broadcast Join

Used when one of the data set is small enough to fit in memory, then is broadcasted to all the worker nodes.

Cases

- One dataset is significantly smaller than the other.
- Only perform equi-join.
- No bandwidth constraints.

2. Shuffle Join

A common strategy to join two large datasets.

Cases

- Too large data to fit in one node.
- Data are partitioned across multiple nodes.
- Outer or left outer joins in the large data sets.

3. Sort Merge Join

Joins large data sets that are spread across multiple nodes in the cluster.

Cases

- Large data, but not so large to user shuffle join.
- Not evently distributed across platforms.
- Operations that cannot perform using broadcast joins (full outer join, left outer join)

4. Broadcast Nested Loop

One small (to fit in a node) and one larger. The smaller broadcasted to all the nodes.

Cases

- One dataset fits the memory
- Selective join key, small number of matching records.
- Large dataset has a good data distribution (not scattered).

Explanations from here!

Results

For the top 3 incomes:

```
+--------------------+-------------+
|         Vict Descent|total_victims|
+--------------------+-------------+
|Hispanic/Latin/Me...|         1996|
|               White|         1136|
|               Black|          952|
|               Other|          518|
|             Unknown|          287|
|         Other Asian|           93|
|            Filipino|           10|
|              Korean|            9|
```

```
|American Indian/A...|           4|
|           Chinese|          2|
|          Japanese|          2|
|        Vietnamese|          1|
|         Guamanian|          1|
+--------------------+-------------+
```

For the bottom 3 incomes:

```
+--------------------+-------------+
|       Vict Descent|total_victims|
+--------------------+-------------+
|             White|          5|
|Hispanic/Latin/Me...|          4|
|           Unknown|          3|
|             Black|          3|
|             Other|          1|
+--------------------+-------------+
```

Using the command :

```
top_3_crimes.explain(mode="extended")
```

We can have the details about the join.

```
== Physical Plan ==
AdaptiveSparkPlan isFinalPlan=false
+- Project [ZIPcode#115, DR_NO#17, Date Rptd#18, DATE OCC#19, TIME OCC#20, AREA #21, AREA NAME#22, Rpt Dist No#23, Part 1-2#24, Crm Cd#25, Crm Cd Desc#26, Mocodes#27, Vict Age#28
, Vict Sex#29, Vict Descent#30, Premis Cd#31, Premis Desc#32, Weapon Used Cd#33, Weapon Desc#34, Status#35, Status Desc#36, Crm Cd 1#37, Crm Cd 2#38, Crm Cd 3#39, ... 8 more fiel
ds]
   +- BroadcastHashJoin [cast(ZIPcode#115 as int)], [ZIPcode#245], LeftOuter, BuildRight, false
      :- Project [DR_NO#17, Date Rptd#18, DATE OCC#19, TIME OCC#20, AREA #21, AREA NAME#22, Rpt Dist No#23, Part 1-2#24, Crm Cd#25, Crm Cd Desc#26, Mocodes#27, Vict Age#28, Vict
Sex#29, Vict Descent#30, Premis Cd#31, Premis Desc#32, Weapon Used Cd#33, Weapon Desc#34, Status#35, Status Desc#36, Crm Cd 1#37, Crm Cd 2#38, Crm Cd 3#39, Crm Cd 4#40, ... 6 mor
e fields]
```

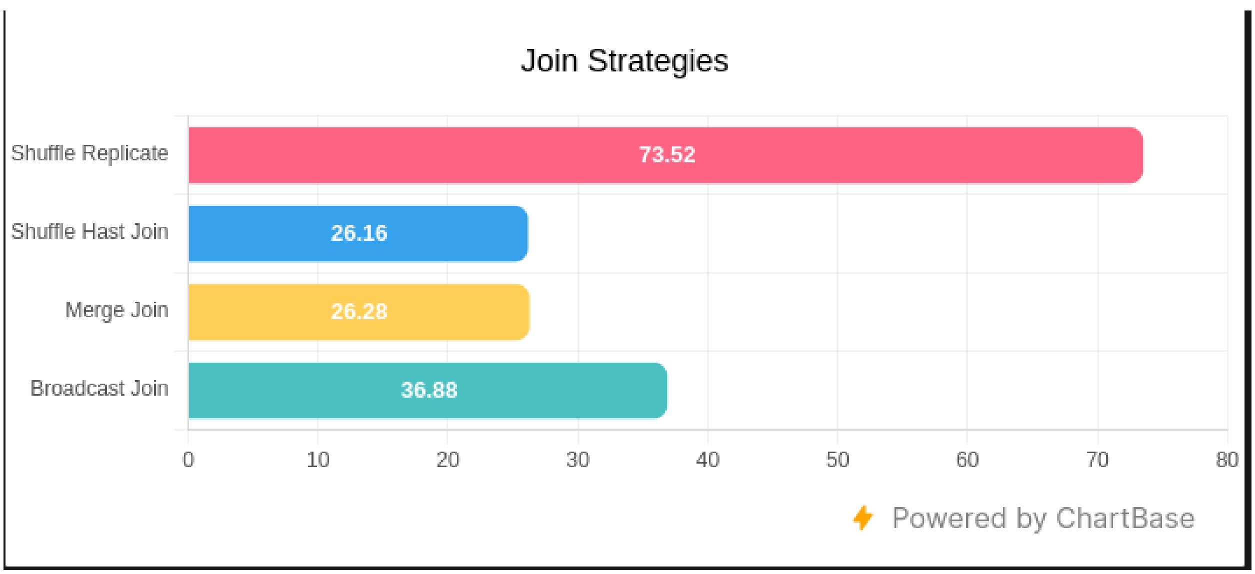
The optimizer uses BroadcastHashJoin.

```
== Physical Plan ==
AdaptiveSparkPlan isFinalPlan=false
+- Project [ZIPcode#115, DR_NO#17, Date Rptd#18, DATE OCC#19, TIME OCC#20, AREA #21, AREA NAME#22, Rpt Dist No#23, Part 1-2#24, Crm Cd#25, Crm Cd Desc#26, Mocodes#27, Vict Age#28
, Vict Sex#29, Vict Descent#30, Premis Cd#31, Premis Desc#32, Weapon Used Cd#33, Weapon Desc#34, Status#35, Status Desc#36, Crm Cd 1#37, Crm Cd 2#38, Crm Cd 3#39, ... 9 more fiel
ds]
   +- SortMergeJoin [ZIPcode#115], [ZIPcode#387], Inner
      :- Sort [ZIPcode#115 ASC NULLS FIRST], false, 0
      :  +- Exchange hashpartitioning(ZIPcode#115, 200), ENSURE_REQUIREMENTS, [plan_id=286]
      :     +- Project [ZIPcode#115, DR_NO#17, Date Rptd#18, DATE OCC#19, TIME OCC#20, AREA #21, AREA NAME#22, Rpt Dist No#23, Part 1-2#24, Crm Cd#25, Crm Cd Desc#26, Mocodes#27,
 Vict Age#28, Vict Sex#29, Vict Descent#30, Premis Cd#31, Premis Desc#32, Weapon Used Cd#33, Weapon Desc#34, Status#35, Status Desc#36, Crm Cd 1#37, Crm Cd 2#38, Crm Cd 3#39, ...
 8 more fields]
```

And for the last query uses SortMergeJoin

Join Strategies

Aa Strategy	# Time
Broadcast Join	36.87837076187
Merge Join	26.277905
Shuffle Hast Join	26.164289
Shuffle Replicate	73.52139



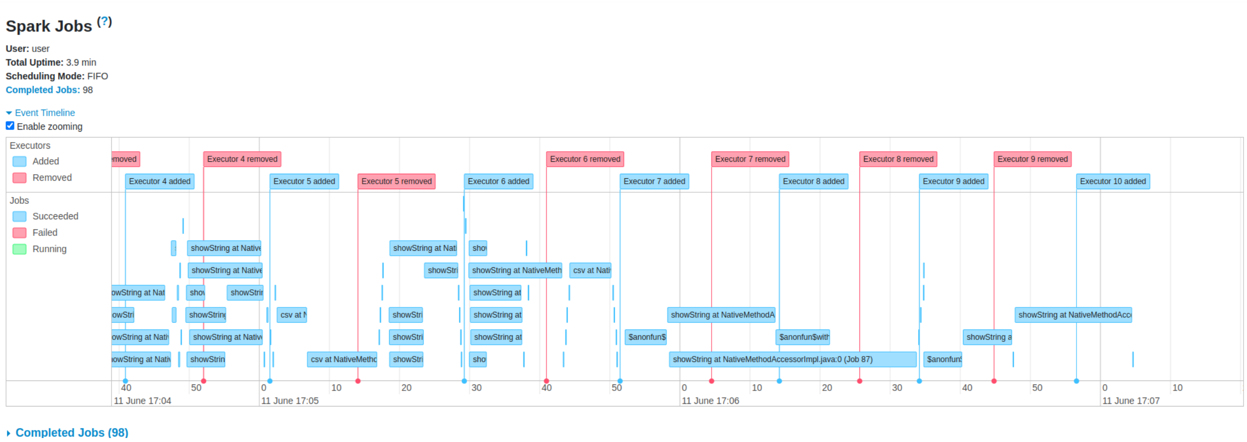
As expected, Merge Join performed very well, due to the fact that our datasets are easily sorted.

The reason the shuffle hast performed so well, is due to the fact that the distribution of the join keys, suits this method.

We expected the broadcast join to perform better due to the fact that we joined a small and a large datasets, and as it mentioned this is the ideal join strategy.

Shuffle Replicate, needs enormous resources in order to work with large files, so it is expected to perform worst.

Screenshot from the history



Requirement 6

Result

```
+-----------+------------------+---------------+
|division   |average_distance  |incidents_total|
+-----------+------------------+---------------+
|77th Street|8341.568622849469 |17021          |
```

```
|Southeast  |12937.589629958506|12948         |
|Newton     |6213.464524257617 |9844          |
|Southwest  |15851.752251094427|8912          |
|Hollenbeck |13553.628977447854|6202          |
|Harbor     |15769.635432110937|5622          |
|Rampart    |5640.65311302953  |5116          |
|Mission    |14802.533072226992|4503          |
|Olympic    |16365.801498962444|4424          |
|Northeast  |6857.946290753058 |3920          |
|Foothill   |10373.145862441319|3775          |
|Hollywood  |13279.880232264879|3643          |
|Central    |13782.2106740909  |3615          |
|Wilshire   |7613.386922339974 |3525          |
|N Hollywood|11774.735196819262|3465          |
|West Valley|10043.938742117367|2903          |
|Van Nuys   |10873.477689439418|2733          |
|Pacific    |7889.32894337054  |2709          |
|Devonshire |17242.142869989486|2472          |
|Topanga    |7494.3508454559   |2285          |
+-----------+------------------+--------------+
```

Analysis

We will provide a pseudocode for each one of the approaches.

Repetition Join

1. In the map phase, each map task processes a split of either Crime Data (R), or (S) from Stations.
2. Union the datasets.
3. Repartitioned by the ‘AREA’ key, preparing for the join operation.
4. Sorting each partition by ‘AREA’
5. Grouping the by the ‘AREA’ key.

6. Separate the records based on the ‘S’ or ‘R’.
    a. We have for each AREA the police stations coordinates and all the crimes that are marked with the same area.
7. Cross-Product Join. We find for each crime the distance from the police station and the count of the crime.

So we have for each area the total crimes and the distance from the station in the format of

<AREA, distance, 1>

Broadcast Join

1. Create a Map of the station data.
2. Broadcast it to all Spark Nodes.
3. Define the Join Function.
    - For each row get the broadcasted area.
    - If there is the station in the broadcasted data,
    - Calculate the distance between this station and the crime geolocation, and return it with the counter. <AREA, DISTANCE, 1>
4. Apply this to each record to each Node.
5. Return the result.

Requirement 7

Results

Look Requirement 6.

We implement the hole feature in the previous Requirement.