

# **Program2 Approach & Analysis**

Big Data Management - Group 8

**Parker Hague**

Oklahoma State University

**Yuhan Jin**

Oklahoma State University

**Dax Jones**

Oklahoma State University

## Task 1

Using the SQL 'GROUP' clause to calculate the average prices or median house value of houses based on ocean proximity

### Approach:

- First, before querying the data, we had to view it and make sure it looked correct. We used the following command from **img1.1** to view the dataset from the HDFS server:

```
hdfs dfs -cat /user/kaggle/kaggle_data/california_housing.csv
```

**img1.1**

- Next, we need to get the dataset from the HDFS server and read it into a Spark dataframe. To do this we created an instance to a Spark session to allow us to do Spark operations on the dataset.
- We then created a Spark dataframe and read the dataset into the dataframe to allow us to do queries on it. We were able to set the options to infer the header and data type information.
- The last step before creating the query is to create a table from the dataframe. This gives us the ability to execute raw SQL queries on the table (dataframe). The following picture **img1.2** shows the code that achieves this.

```
house_price_dataset.createOrReplaceTempView("HousePriceDataset")
```

**img1.2**

- Finally, we can now perform the query on the dataset to calculate the average prices of houses based on ocean proximity. In our query, we select the average median house value and ocean\_proximity: **SELECT AVG(median\_house\_value), ocean\_proximity**. We are able to select which table by specifying: **FROM HousePriceDataset**. Lastly, we used the GROUP BY clause to group the selected rows by the ocean proximity: **GROUP BY ocean\_proximity**. After querying, we were able to test and verify the correctness of our results see **img1.4**.

## Results:

### Program First Starts Executing:

```
Parker — ssh phague@hadoop-nn001.cs.okstate.edu — 107x56
phague@hadoop-nn001:~/pgm2$ spark-submit Group_8_Program_1.py
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.spark.unsafe.Platform (file:/usr/local/spark-3.0.1-bin-hadoop3.2/jars/spark-unsafe_2.12-3.0.1.jar) to constructor java.nio.DirectByteBuffer(long,int)
WARNING: Please consider reporting this to the maintainers of org.apache.spark.unsafe.Platform
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
2022-04-15 18:10:38,769 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform.
.. using builtin-java classes where applicable
2022-04-15 18:10:39,634 INFO spark.SparkContext: Running Spark version 3.0.1
2022-04-15 18:10:39,681 INFO resource.ResourceUtils: =====
2022-04-15 18:10:39,683 INFO resource.ResourceUtils: Resources for spark.driver:
2022-04-15 18:10:39,683 INFO resource.ResourceUtils: =====
2022-04-15 18:10:39,684 INFO spark.SparkContext: Submitted application: Group_8_Program_1.py
2022-04-15 18:10:39,744 INFO spark.SecurityManager: Changing view acls to: phague
2022-04-15 18:10:39,744 INFO spark.SecurityManager: Changing modify acls to: phague
2022-04-15 18:10:39,744 INFO spark.SecurityManager: Changing view acls groups to:
2022-04-15 18:10:39,744 INFO spark.SecurityManager: Changing modify acls groups to:
2022-04-15 18:10:39,744 INFO spark.SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view permissions: Set(phague); groups with view permissions: Set(); users with modify permissions: Set(phague); groups with modify permissions: Set()
2022-04-15 18:10:40,028 INFO util.Utils: Successfully started service 'sparkDriver' on port 36129.
2022-04-15 18:10:40,061 INFO spark.SparkEnv: Registering MapOutputTracker
2022-04-15 18:10:40,096 INFO spark.SparkEnv: Registering BlockManagerMaster
2022-04-15 18:10:40,118 INFO storage.BlockManagerMasterEndpoint: Using org.apache.spark.storage.DefaultTopologyMapper for getting topology information
```

Img1.3

### Queried Results:

```
2022-04-15 18:02:19,683 INFO storage.BlockManagerInfo: Removed broadcast_9_piece0 on csh2.cs.okstate.edu:33031 in memory (size: 16.1 KiB, free: 434.3 MiB)
2022-04-15 18:02:19,695 INFO storage.BlockManagerInfo: Removed broadcast_5_piece0 on csh2.cs.okstate.edu:33031 in memory (size: 14.3 KiB, free: 434.3 MiB)
2022-04-15 18:02:19,701 INFO storage.BlockManagerInfo: Removed broadcast_5_piece0 on hadoop-nn001:39519 in memory (size: 14.3 KiB, free: 434.2 MiB)
2022-04-15 18:02:19,708 INFO codegen.CodeGenerator: Code generated in 18.005933 ms

+-----+-----+
|avg(median_house_value)|ocean_proximity|
+-----+-----+
|          380440.0|          ISLAND|
| 249433.97742663656|        NEAR OCEAN|
| 259212.31179039303|        NEAR BAY|
| 240084.28546409807|         <1H OCEAN|
| 124805.39200122119|          INLAND|
+-----+-----+

RAW SQL OUTPUT

2022-04-15 18:02:19,724 INFO storage.BlockManagerInfo: Removed broadcast_6_piece0 on hadoop-nn001:39519 in memory (size: 16.1 KiB, free: 434.3 MiB)
2022-04-15 18:02:19,726 INFO storage.BlockManagerInfo: Removed broadcast_6_piece0 on csh2.cs.okstate.edu:33031 in memory (size: 16.1 KiB, free: 434.3 MiB)
2022-04-15 18:02:19,745 INFO storage.BlockManagerInfo: Removed broadcast_2_piece0 on hadoop-nn001:39519 in memory (size: 16.1 KiB, free: 434.3 MiB)
```

img1.4

## End of Program Execution

```
2022-04-15 18:11:11,286 INFO cluster.YarnClientSchedulerBackend: YARN client scheduler backend Stopped
2022-04-15 18:11:11,311 INFO spark.MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
2022-04-15 18:11:11,332 INFO memory.MemoryStore: MemoryStore cleared
2022-04-15 18:11:11,333 INFO storage.BlockManager: BlockManager stopped
2022-04-15 18:11:11,336 INFO storage.BlockManagerMaster: BlockManagerMaster stopped
2022-04-15 18:11:11,343 INFO scheduler.OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
2022-04-15 18:11:11,364 INFO spark.SparkContext: Successfully stopped SparkContext
2022-04-15 18:11:11,364 INFO util.ShutdownHookManager: Shutdown hook called
2022-04-15 18:11:11,365 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-e49a410d-482e-47ec-b4b
b-a9a6e2c11b7b/pyspark-afec5762-2fe3-411e-92d4-9ca67dc34ad5
2022-04-15 18:11:11,372 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-e49a410d-482e-47ec-b4b
b-a9a6e2c11b7b
2022-04-15 18:11:11,382 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-e7ec3765-4830-434e-b7f
3-ec0de4c1725c
phague@hadoop-nn001:~/pgm2$
```

img1.5

## Task 2

Using Python or the language of our choice in Spark to calculate the average prices or median house value of houses based on ocean proximity

### *Approach:*

- For this part, a lot of the approach was very similar. The main differences were in how the query was executed. We didn't have to view and verify the dataset from the HDFS server since we already did that in part 1.
- Like in part 1, we need to get the dataset from the HDFS server and read it into a Spark dataframe. To do this we created an instance to a Spark session to allow us to do Spark operations on the dataset.
- We then created a Spark dataframe and read the dataset into the dataframe to allow us to do queries on it. We were able to set the options to infer the header and data type information.
- Since this time we are only querying the dataframe directly with spark, we don't have to create a table version of the dataframe like we had to in part 1.
- For the query, it was a relatively simple line of code using Spark's functionality to query through its inherent methods. Using the dataframe method, `groupBy`, we were able to easily group all the unique ocean proximities. Then, we appended the `agg` method to the query so we could apply the averaging function. The averaging was done using another built-in Spark function where the average was calculated by the column we passed to it: median house price. This query can be seen in **img2.1**.

```
spark_query = house_price_dataset.groupBy("ocean_proximity").agg(F.mean("median_house_value")).show()
```

*img2.1*

- The query then returned a dataframe. We were able to view and verify that the results matched the output from the query on part 1. The results from the operation of averaging median house prices that are grouped by ocean proximity can be seen in **img2.3**.

## Results:

### Program First Starts Executing:

```
Parker — ssh phague@hadoop-nn001.cs.okstate.edu — 107x56
phague@hadoop-nn001:~/pgm2$ spark-submit Group_8_Program_2.py
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.spark.unsafe.Platform (file:/usr/local/spark-3.0.1-bin-hadoop3.2/jars/spark-unsafe_2.12-3.0.1.jar) to constructor java.nio.DirectByteBuffer(long,int)
WARNING: Please consider reporting this to the maintainers of org.apache.spark.unsafe.Platform
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
2022-04-15 18:13:57,397 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform.
.. using builtin-java classes where applicable
2022-04-15 18:13:58,263 INFO spark.SparkContext: Running Spark version 3.0.1
2022-04-15 18:13:58,309 INFO resource.ResourceUtils: =====
2022-04-15 18:13:58,310 INFO resource.ResourceUtils: Resources for spark.driver:
2022-04-15 18:13:58,310 INFO resource.ResourceUtils: =====
2022-04-15 18:13:58,311 INFO spark.SparkContext: Submitted application: Group_8_Program_2.py
2022-04-15 18:13:58,371 INFO spark.SecurityManager: Changing view acls to: phague
```

img2.2

### Queried Results:

```
2022-04-15 18:14:29,207 INFO scheduler.DAGScheduler: Job 6 is finished. Cancelling potential speculative or
zombie tasks for this job
2022-04-15 18:14:29,207 INFO cluster.YarnScheduler: Killing all running tasks in stage 11: Stage finished
2022-04-15 18:14:29,208 INFO scheduler.DAGScheduler: Job 6 finished: showString at NativeMethodAccessorImpl
.java:0, took 0.454144 s
2022-04-15 18:14:29,236 INFO codegen.CodeGenerator: Code generated in 17.362503 ms
+-----+
|ocean_proximity|avg(median_house_value)|
+-----+
| ISLAND | 380440.0 |
| NEAR OCEAN | 249433.97742663656 |
| NEAR BAY | 259212.31179039303 |
| <1H OCEAN | 240084.28546409807 |
| INLAND | 124805.39200122119 |
+-----+

SPARK SQL OUTPUT

2022-04-15 18:14:29,318 INFO spark.SparkContext: Invoking stop() from shutdown hook
2022-04-15 18:14:29,331 INFO server.AbstractConnector: Stopped Spark@4d5ba774{HTTP/1.1,[http/1.1]}{0.0.0.0:4041}
2022-04-15 18:14:29,334 INFO ui.SparkUI: Stopped Spark web UI at http://hadoop-nn001:4041
```

img2.3

### ***End of Program Execution:***

```
2022-04-15 18:14:29,459 INFO spark.SparkContext: Successfully stopped SparkContext
2022-04-15 18:14:29,459 INFO util.ShutdownHookManager: Shutdown hook called
2022-04-15 18:14:29,461 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-510dc1a1-1a49-40e3-826
c-32c218b8c403/pyspark-88087680-edae-41ba-94bc-ce1ae3077552
2022-04-15 18:14:29,464 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-510dc1a1-1a49-40e3-826
c-32c218b8c403
2022-04-15 18:14:29,469 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-d1231de7-bb63-47f9-bff
2-b03dd66c5346
phague@hadoop-nn001:~/pgm2$
```

*img2.4*

### **Final Thoughts:**

This was an easy assignment for us. We used two different query methods and they both yielded the same result. We decided that we preferred the way we did it in task 2 because we thought it was easier to use functions to operate on the dataframe directly; rather than creating a table and writing raw SQL for the query.