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.

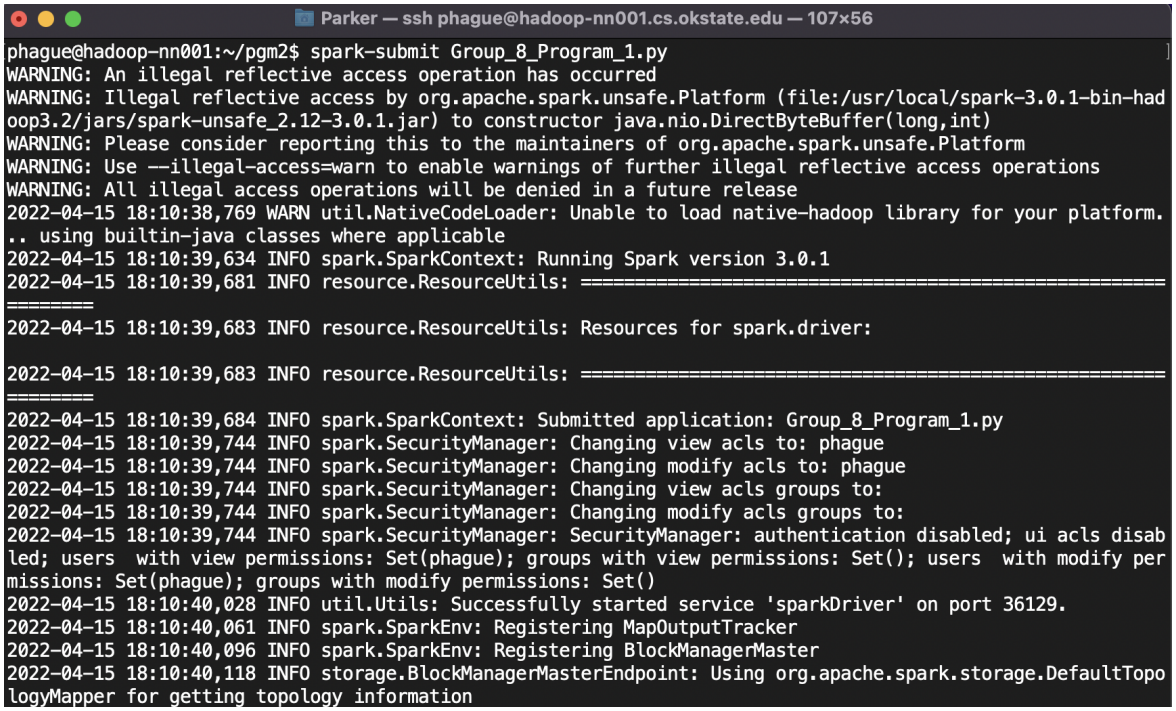
```
house_price_dataset.createOrReplaceTempView("HousePriceDataset")
```

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**.

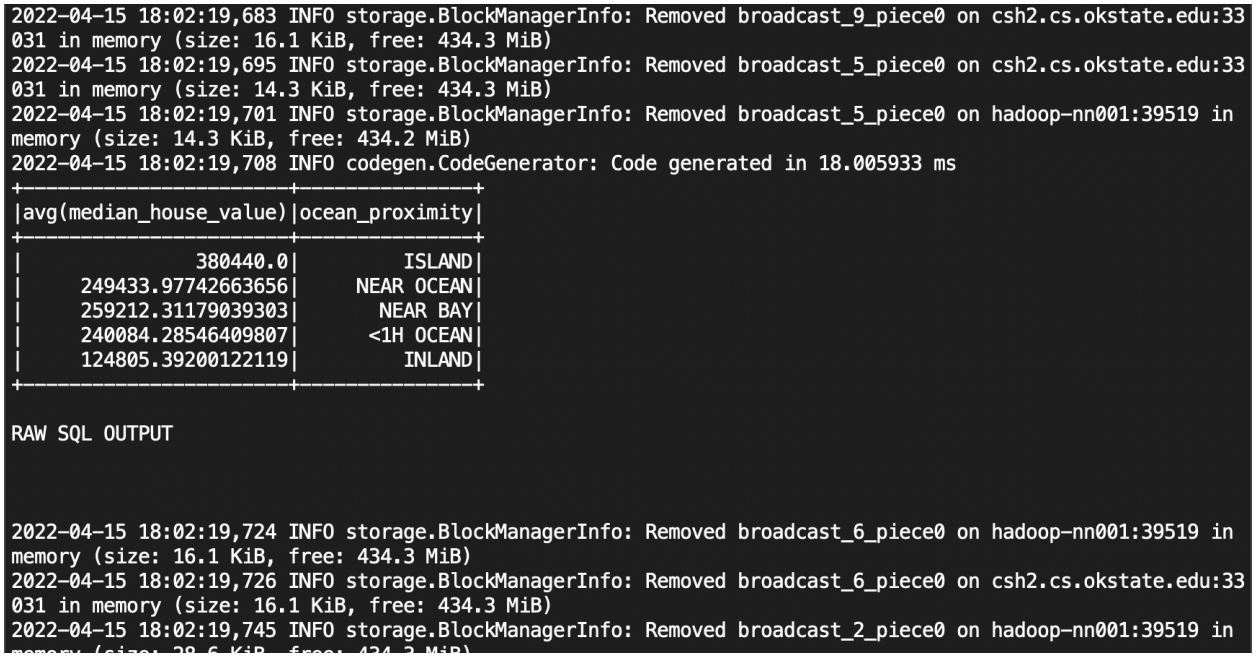
Results:

Program First Starts Executing:



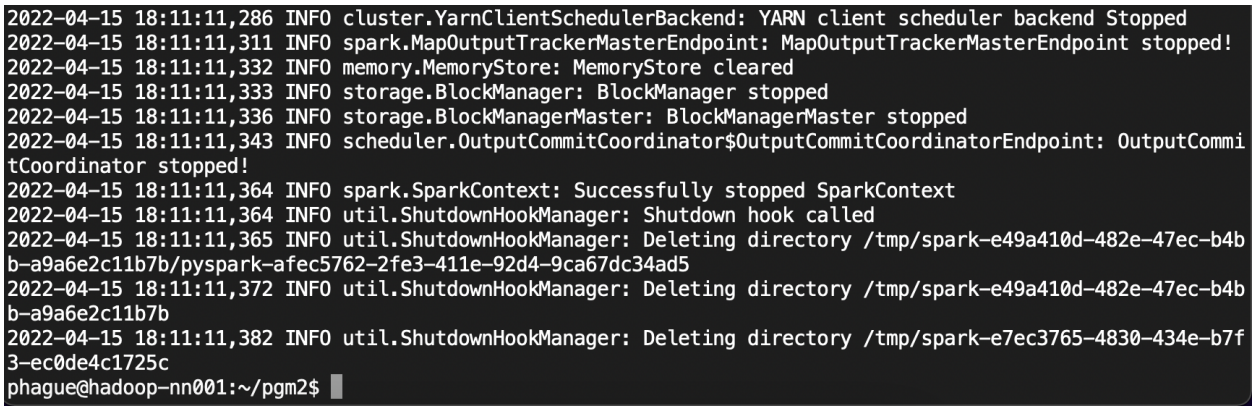
Img1.3

Queried Results:



img1.4

End of Program Execution



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**.

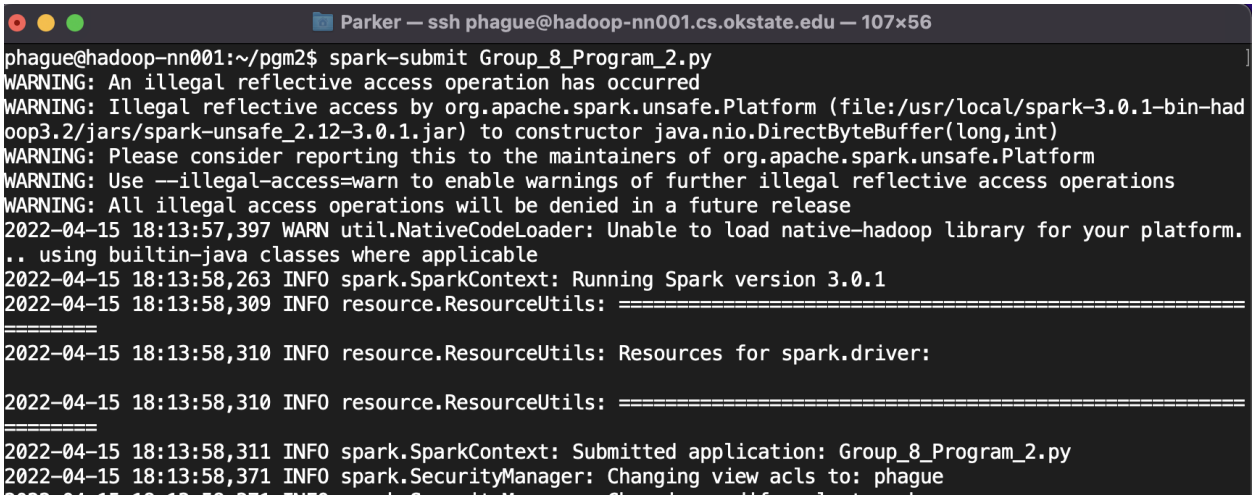
```
spark_query = house_price_dataset.groupBy("ocean_proximity").agg(F.mean("median_house_value")).show()
```

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**.

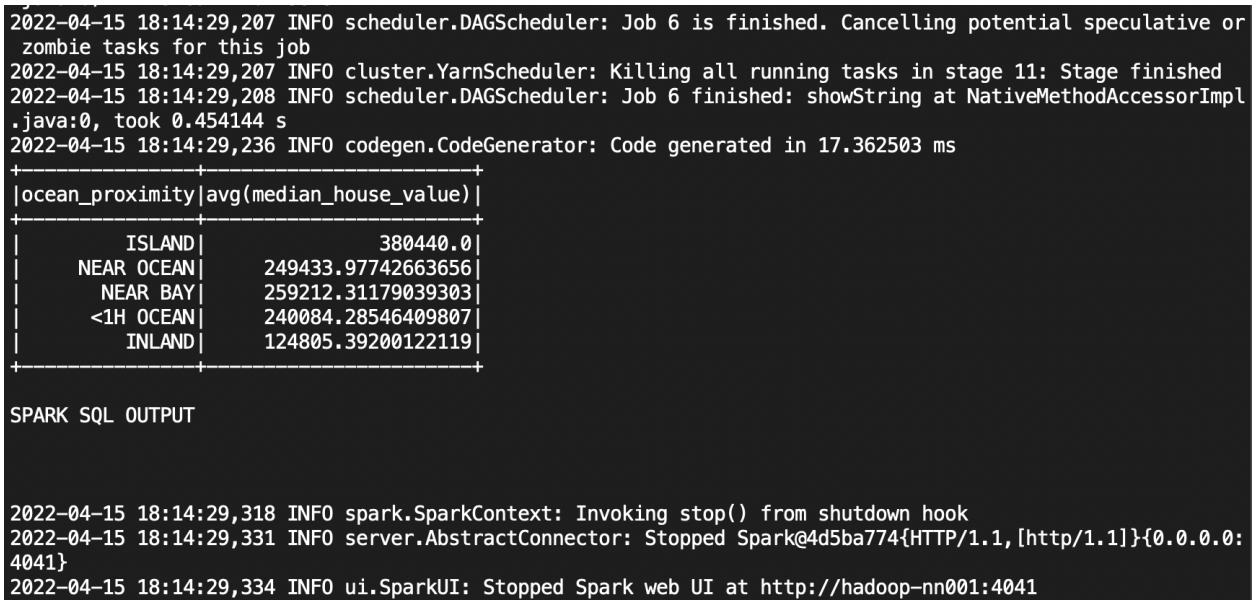
Results:

Program First Starts Executing:



Img2.2

Queried Results:



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.