PROJECT REPORT

SPRING 2023 CPSC 531-03

HOUSE RENTAL ANALYSIS

TEAM MEMBERS

NAME :MANIKRISHNA SANGANABATLA
EMAIL: msanganabatla@csu.fullerton.edu
CWID:885638767

NAME: KRISHNA ABHISHITH PURIHELLA
EMAIL: purihella.abhi@csu.fullerton.edu
CWID:885211615

CONTENTS

1. INTRODUCTION
2. PROJECT OBJECTIVES
3. SCOPE AND LIMITATIONS
4. FUNCTIONALITIES
5. ARCHITECTURE AND DESIGN
6. GITHUB LOCATION OF CODE
7. DEPLOYMENT INSTRUCTIONS
8. STEPS TO RUN THE APPLICATION
9. TEST RESULTS.
10. REFERENCES.

INTRODUCTION:

This Project is centered on analyzing various aspects linked to a house. Prediction of house rent is vital among those aspects. Let us deep dive into some attributes that link us in analyzing the house rent. The price of a house depends on the region, area of the house, the facilities, etc., and prediction is carried out by using some of the existing machine learning algorithms. This Report focuses on the above part and how the data is collected and driven to do the analysis. Since this project requires a vast amount of data to analyze, the data is retrieved from the cloud and is then subjected to pre-processing and partitioning using AWS (Amazon Web Service) EMR clusters and is then analyzed by a tool (Athena). The analysis is then visualized by integrating Athena to one of the significant visualization tools called Tableau.

PROJECT OBJECTIVES:

GENERAL OBJECTIVE: The general objective of this project is to develop a visualization view of the rent that was predicted using various aspects.

SPECIFIC OBJECTIVE: Some of the specific objectives of this project are

1) Pre-process the dataset with several defects in the data—partitioning, and analyze the data by creating an EMR cluster that runs on AWS EC2 instances.
2) Integration of cluster instances with the system through SSH (Secure Socket Shell). A total of 3 instances are created – 1 Master Node and 2 Worker Nodes.
3) Create a PySpark program to query various scenarios associated with the dataset.
4) Submit a spark job and run the program
5) Upload the queried results into S3
bucket of AWS, where it can be again used to generate visualization charts.

6) Predict the House Rents using the Random-Forest method; here, we use a machine learning algorithm to predict various aspects using the dataset.
7) Application of big queries to generate tables of required attributes using AWS Athena.
8) Integration of AWS Athena with a visualization tool called Tableau.

SCOPE AND LIMITATIONS:

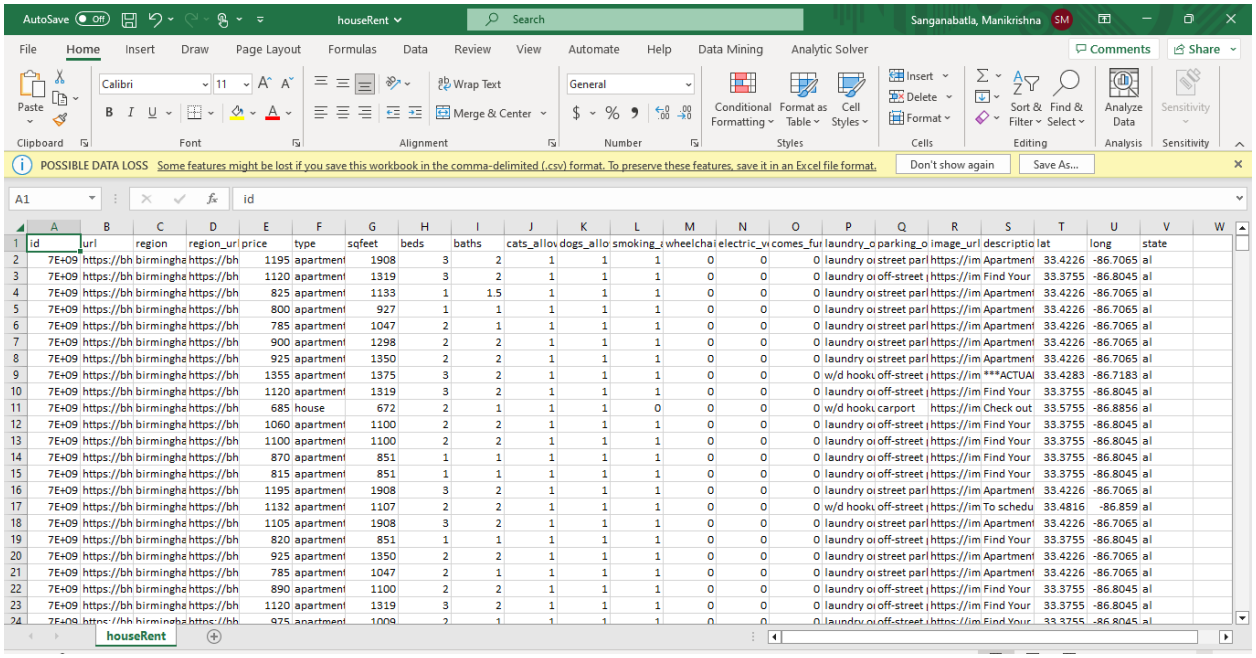
The scope of a house rental analysis project can differ depending on the project's objectives, accessible data, and anticipated findings. Examining rental costs can reveal the typical rentals charged for various types of homes in a given area. It can also detect trends over time and assist landlords, property managers, and renters make informed decisions.
Limitations of this project can be

1. Data Availability: One of the most significant restrictions of a house rental study project is data availability and quality. Rental data may be difficult to collect or incomplete, affecting the analysis's accuracy.
2. Lack of Standardization: Another constraint of house rental analysis is the lack of market standardization. Each market may have its distinct traits and elements influencing rental costs, making it difficult to compare statistics across regions.
3. The sample size of rental data may be too small to produce valid results or too vast to handle properly, limiting the scope of a rental analysis project.

FUNCTIONALITIES:

1)<u>DATASET</u> :

The Dataset is obtained from one of the open-source websites called Kaggle. Our dataset is of 1.2 Gb in total size. It is comprised of different attributes. There are a total of
Our Dataset consists of a total of 22 columns like price, region, URL, sqfeet, type, region, state, etc.,



2)Data Cleaning:

Data cleaning is a critical process to ensure data is correct and consistent. It involves removing duplicates, fixing errors, and removing irrelevant data. To do this, PySpark was installed, created a spark session, and created a custom dataset as a data frame. To clean the data, PySpark used commands like dropna() and fillna() to replace null values, handle invalid rows and columns, and replace null values with 0's .dropna() used to remove invalid/ unnecessary rows and columns and fillna() used to fill the those rows/columns with the custom values.

AWS EMR Cluster: Amazon EMR (Elastic MapReduce) is a web service provided by Amazon Web Services (AWS) that allows customers to process massive volumes of data in a distributed computing environment. An EMR cluster is a collection of Amazon Elastic Compute Cloud (EC2) machines running Hadoop and other large data processing frameworks collaborating to accomplish data processing tasks. Here are some of the important features of an AWS EMR cluster:

Distributed Processing: The fundamental role of an EMR cluster is to enable distributed processing of huge datasets. This means data can be broken into smaller bits and processed in parallel across numerous EC2 instances.

EMR clusters support a variety of large data processing frameworks, including Apache Hadoop, Spark, and Hive.

Scalability: EMR clusters are extremely scalable, allowing customers to add or delete EC2 instances as needed to fulfill processing requirements. This makes it simple to scale up or down based on changing data processing requirements.

Automation: EMR clusters automate many of the procedures associated with constructing and operating Hadoop clusters, such as installation, configuration, and scaling.
Integration with other AWS Services: EMR clusters interface with other AWS services such as S3, EC2, and CloudWatch, allowing users to move data effortlessly across services and monitor cluster performance.

APACHE SPARK: Apache Spark is a distributed computing system that is open source and used for big data processing. It has many features to help with data processing, analysis, and machine learning activities. The key functionalities are In-Memory Computing, Distributed Computing, Data Processing, Machine Learning, and Integration with Other Tools like Hadoop, Cassandra, and Amazon S3, making working with data from multiple sources easy.

AWS S3 BUCKET: Amazon Simple Storage Service (S3) is an object storage service that is extremely scalable and provided by Amazon Web Services (AWS). It offers a wide range of features to aid in the storage, management, and access of data objects in the cloud. Here are some of the key features of an AWS S3 bucket:
Scalability: S3 offers limitless scalability, allowing customers to store and retrieve massive volumes of data objects in the cloud.
Storage: S3 provides a durable storage solution designed to withstand data loss due to hardware failures, natural disasters, and other occurrences.
Availability: S3 delivers high availability for data objects by storing several redundant copies of data across different availability zones.
Security: S3 offers a variety of security capabilities to safeguard data objects, including encryption of data at rest and in transit, access control via AWS Identity and Access Management (IAM) policies, and versioning to keep track of changes to data objects.
Integration with Other AWS Services: S3 integrates with other AWS services, such as EC2, Lambda, and EMR, enabling users to easily store and access data objects from different services.

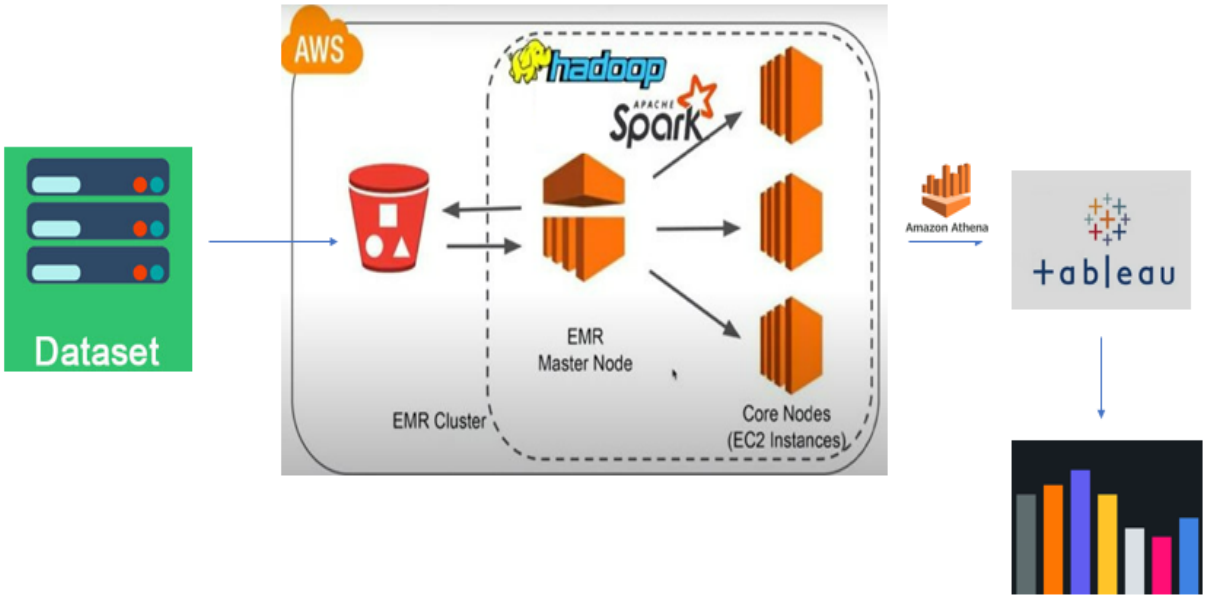
AWS Athena:
Amazon Athena is an interactive query service that allows users to run conventional SQL queries to explore data in Amazon S3. It has a wide range of functions to help with data analysis and reporting. Some of the important aspects are as follows:
Scalability: Because Athena is highly scalable and capable of handling enormous datasets, users can examine data in S3 at any scale.
Compatibility: Because Athena supports normal SQL queries, customers can easily analyze their data using familiar SQL syntax. It also supports various data formats such as CSV, JSON, ORC, Parquet, and Avro.
Athena interfaces with other AWS services, such as AWS Glue and Amazon QuickSight, allowing customers to easily extract, manipulate, and visualize their data.

Performance: Depending on the complexity of the query, Athena provides quick query performance, with results given in seconds or minutes.

TABLEAU: Tableau is a powerful data visualization and business intelligence software that businesses and organizations use to analyze, interpret, and exchange data. Tableau's primary features are as follows:
Data Connectivity: Tableau can link to a variety of data sources, including spreadsheets, databases, and cloud-based data warehouses. It also supports several file types, including CSV, PARQUET,Excel, and JSON.
Data Preparation: Tableau provides strong data preparation features that enable users to transform, clean, and blend data from many sources to generate a cohesive dataset. This allows users to arrange their data for analysis and visualization without the need for complicated code.
Data Visualization: Tableau provides a wide range of visualization tools to create interactive and engaging visualizations like charts, graphs, and maps. Users can also create dashboards and stories to share their insights with others.
These are some of the detailed functionalities of the tools and services used to design this project.

APPROACH: Below is the following approach for our project :

DESIGN AND ARCHITECTURE:



GITHUB LOCATION OF CODE :

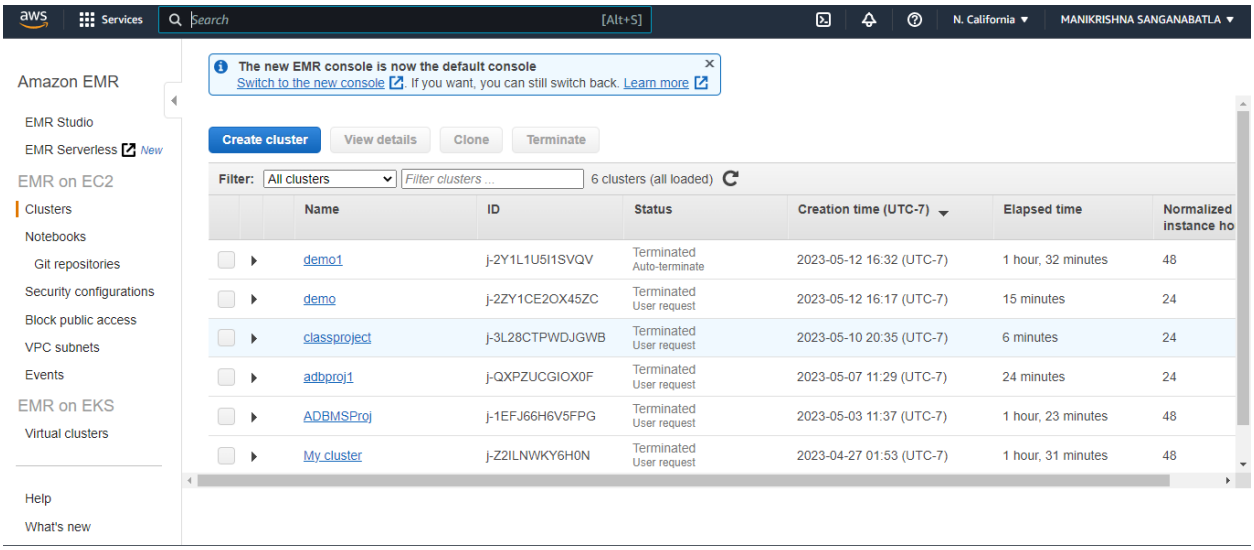
https://github.com/manikittu810/ADBMS_PROJECT

TECHNOLOGY USED :
OS: Windows , macOS
Framework : Amazon Web Service (AWS) , Spark , for BigQuery used SQL
Languages Used : Python, SQL
Libraries : Pyspark
IDE : Jupyter Notebook ,VS code

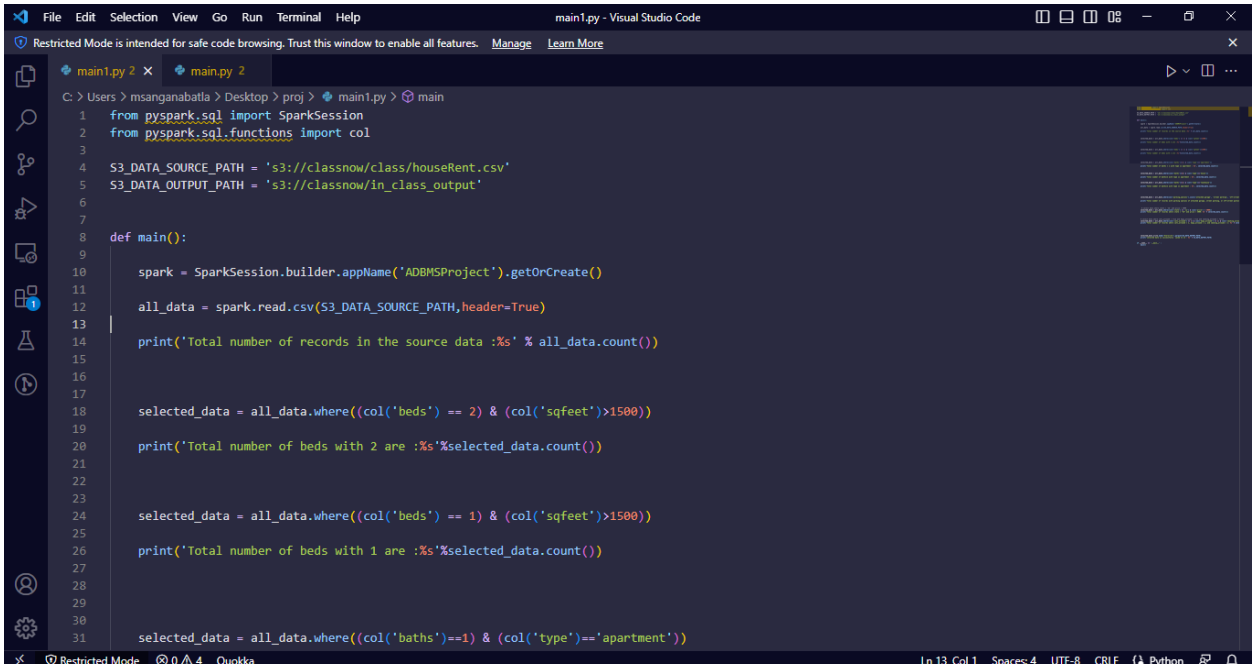
Below is a design diagram of this project that represents the flow and usage of the tools and applications.

DESIGN :

- A cleaned Dataset is taken and is uploaded into the S3 Bucket of the AWS. This S3 bucket is the cloud data storage provided by AWS.

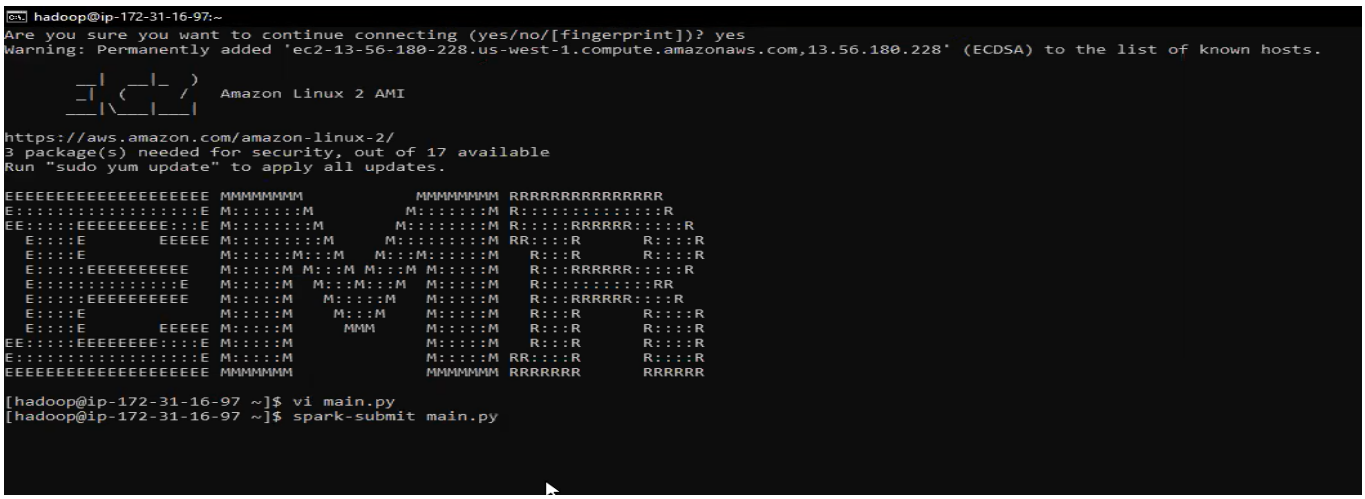
- Then comes the creation of EMR Clusters(Elastic Map Reduce Cluster). EMR cluster is a hadoop cluster that helps in generating instances to run the data on distributed file systems.
- The EMR cluster is generated by giving necessary parameters, here we have used the Apache Hadoop yarn version ……. and this cluster consists of total 3 instances (1 master and 2 slave nodes). **A cluster with the name of <any_custom_name> is created in this project.**
- 

- A python code is designed locally that queries various aspects; the data stored in the S3 bucket is retrieved in this code to provide the output. You can find this code inside the GitHub link provided in this document with the **Filename <main.py> and <main1.py>**

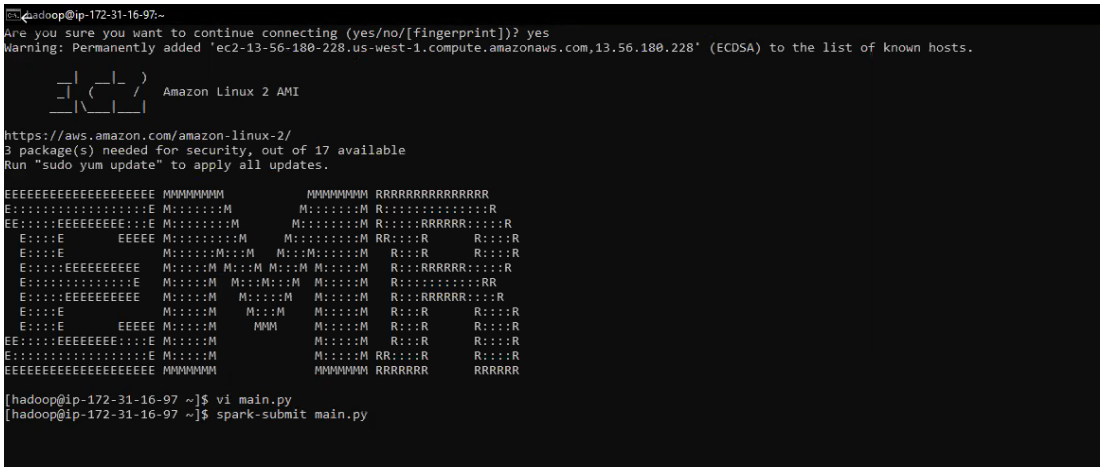


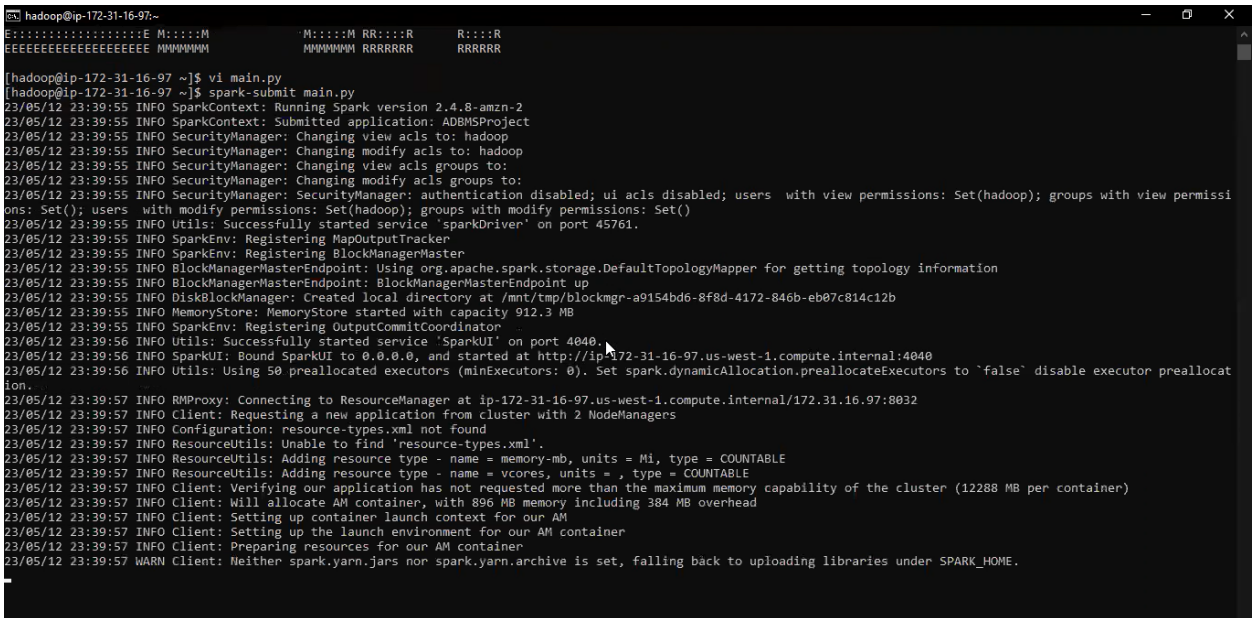
You can find the same code in the github using the above provided link.

- Integration of the EMR cluster with the local using SSH is carried out. This is done submitting the following commands on the command prompt.
- Ssh -i <location (or) path of the aws key pair> <cluster provided command for integrating master node with local >

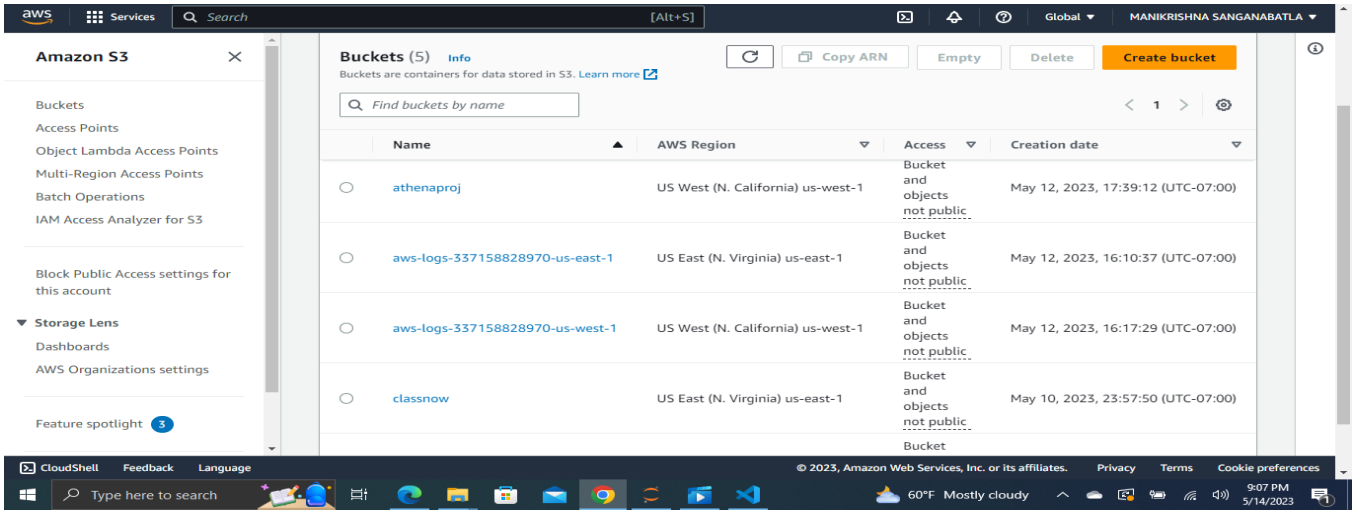


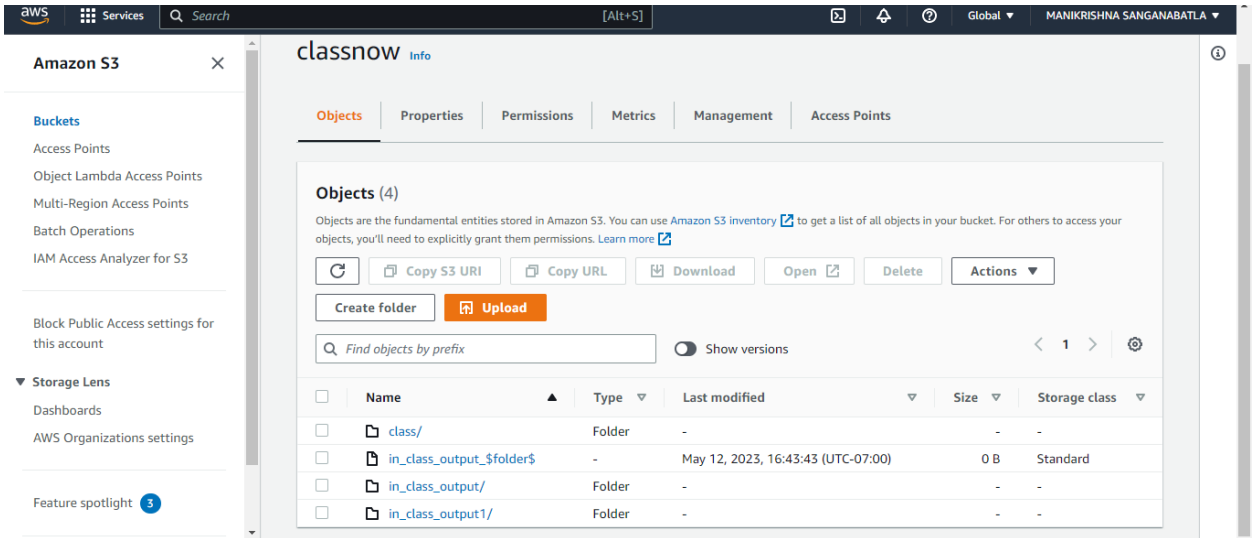
- A spark job is then submitted on the local console by using the command **"spark-submit <filename.py>"**. After submission of the spark job, the code is then run and the results are displayed on the console.





- The output results are stored back to S3 bucket into a different folder , whose path is already defined in the code that was ran by submitting the spark job. These files are again used in the later stages to generate visual reports.



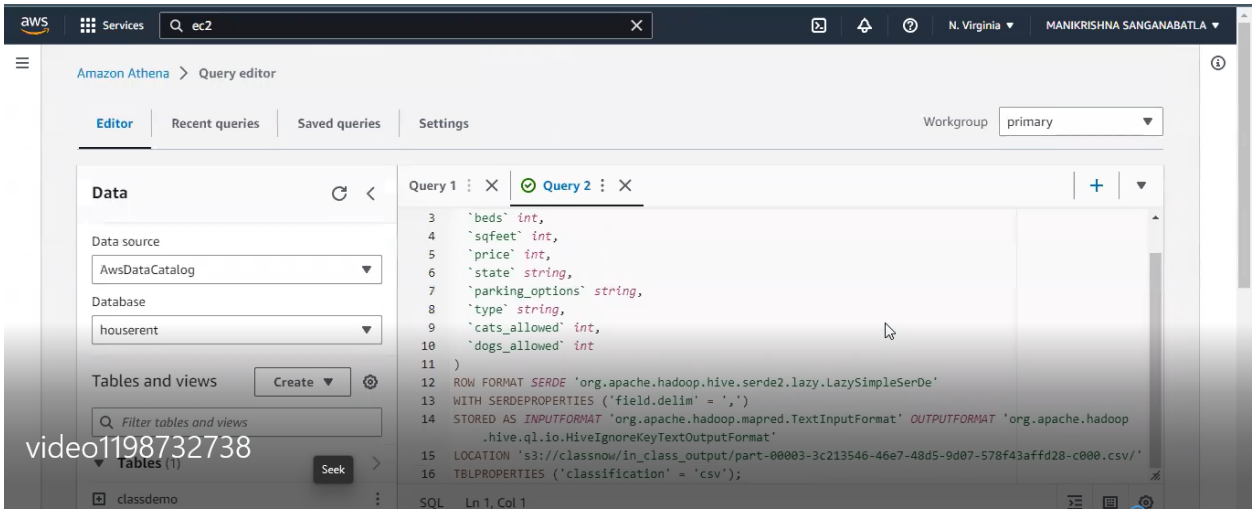


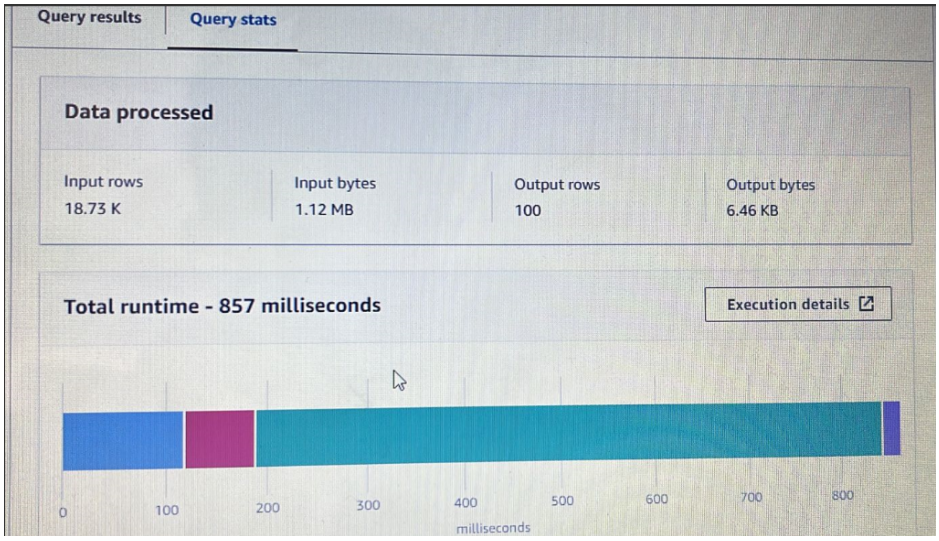
ML Algorithm for prediction : For predictions we have used random forest algorithm which gave us the accurate results. We have also used many algorithms but random forest gave us the best results. We many attributes in our dataset , so, we have only taken four attributes to predict the rent of a type of a house in the state of california. Below is the code and results.

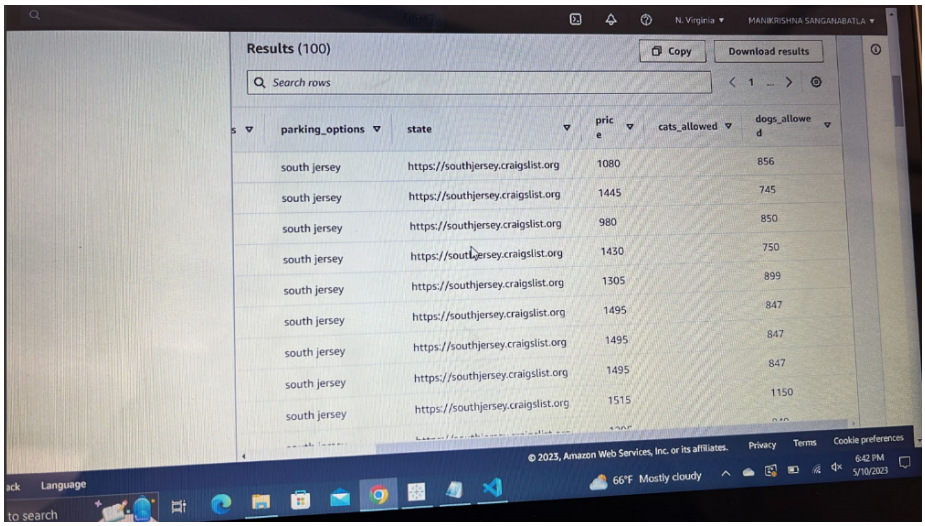
(If the code is not clearly visible here , you can find the same code in the github repo, Link for the git hub repo is alo provided).

```
new_data = pd.DataFrame({
    'type': ['house'],
    'state': ['ca'],
    'baths': [1],
    'beds': [2],
    'sqfeet': [1000]
})
new_data_cat = ohe.transform(new_data[['type', 'state']])
new_data_cat = pd.DataFrame(new_data_cat.toarray(), columns=ohe.get_feature_names_out(['type', 'state']))
new_data_num = new_data[['baths', 'beds', 'sqfeet']]
new_data_num = scaler.transform(new_data_num)
new_data_num = pd.DataFrame(new_data_num, columns=['baths', 'beds', 'sqfeet'])
new_data = pd.concat([new_data_cat, new_data_num], axis=1)
prediction = rf.predict(new_data)
print('Predicted rent:', prediction)
```

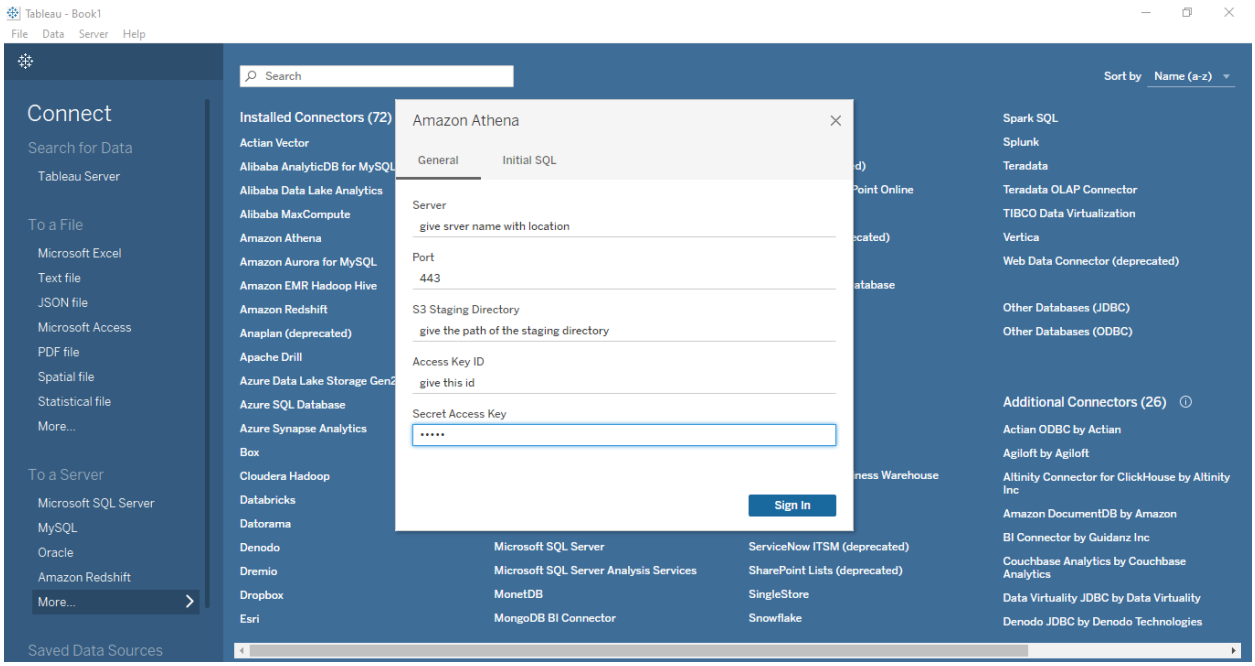
The output files stored are subjected to Bigquery using Amazon Athena to display the tables depending upon the desired output.







- The Athena is then connected to Tableau, a visualization tool to generate charts or reports depending upon the tables generated in the Athena**.**

- 

The process of running this project is briefly explained in Steps to Run the Application.

TABLEAU LINK for public access :
https://public.tableau.com/app/profile/manikrishna.sanganabatla

INSTRUCTIONS FOR DEPLOYMENT:

1. Create an EMR Cluster:
    - Log in to the AWS Management Console and navigate to the Amazon EMR service.
    - Click on "Create cluster" and configure the cluster settings according to the requirements. Make sure to select the appropriate version of EMR and Spark.
    - Specify the number of instances, instance types, and other cluster details. You can also customize advanced options if necessary.
    - Once the configuration is complete, click on "Create cluster" to launch the EMR cluster.
2. Upload Input Data to S3:
    - Log in to the AWS Management Console and navigate to the Amazon S3 service.
    - Create a new S3 bucket or select an existing one to store the input data.

- Upload the input data files (e.g., CSV, JSON) to the S3 bucket you created. Take note of the S3 bucket path for future reference.

3. Prepare the Spark Job:
   - Package the Spark job code and dependencies into a Python file. Make sure all the necessary dependencies are included.
   - If you're using a Python script, ensure it has the necessary import statements and handles the input and output paths correctly.
   - Test the Spark job code locally to ensure it's working as expected before deploying it to the EMR cluster.
4. Connect to the EMR Cluster Master Node:
   - Connect to the EMR cluster's master node using SSH. You can use the EC2 key pair associated with the cluster.
   - Once connected, navigate to the Spark directory where the application code is stored.
   - Use this command :
     Ssh -i <location (or) path of the aws key pair> <cluster provided command for integrating master node with local >
5. Run the Spark Job:
   - Submit the Spark job using the spark-submit command, specifying the path to the application code and any necessary arguments. For example:
   - Bash
   - Spark-submit <filename.py>
   - Monitor the job execution logs to ensure it completes successfully.
6. Store Output Files in S3:
   - After the Spark job completes, verify that the output files are generated and available in the specified S3 output path. You can check the S3 bucket you specified for the output files.

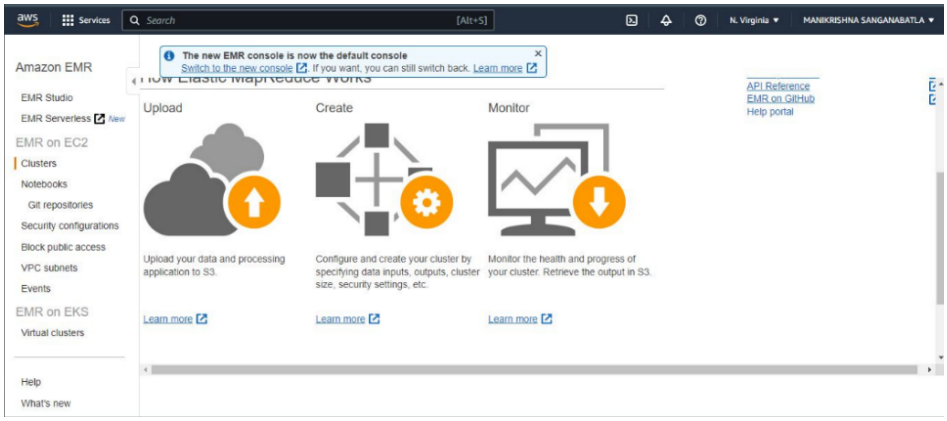
     (6.1). ML prediction : For this project random forest works better, goto the jupyter notebook or google collab and execute the ml algorithm to get mse, accuracy, give four unique attributes to predict the rent of the house types in various regions. (shift + enter to excute the code).

7. Set Up Athena and Create a Table:
   - Go to the AWS Management Console and navigate to the Amazon Athena service.
   - Click on "Query Editor" to open the Athena query editor.
   - Create a new database if required, and then create a table in Athena that matches the schema of the output data.

- Use the "Create Table" wizard or write SQL statements to define the table structure and specify the S3 location of the output data files.

8. Write SQL Queries for Visualizations in Tableau:
    - Open Tableau and establish a connection to Athena.
    - Create a new worksheet or dashboard and drag the required dimensions and measures onto the canvas.
    - Write SQL queries using Tableau's interface to retrieve the data from Athena for the visualizations.
    - Customize the visualizations, apply filters, and create interactive elements as needed.
    - Save and publish the visualizations to share with others.

STEPS TO RUN THE APPLICATION:



1.Create an EMR Cluster :

- Sign in to the AWS Management Console and navigate to the Amazon EMR service.
- Click on "Create cluster" and configure the cluster settings according to the requirements. Select the appropriate version of EMR and Spark.
- Specify the number of instances, instance types, and other cluster details. You can also customize advanced options if necessary.
- Click on "Create cluster" to launch the EMR cluster.

2.Upload Data to S3:

- Sign in to the AWS Management Console and go to the Amazon S3 service.
- Create a new S3 bucket or choose an existing one to store the input data.
- Upload the input data files (e.g., CSV, JSON) to the S3 bucket you created. Remember the S3 bucket path for future reference.

3. Connect to the EMR cluster master node :

- Establish an SSH connection to the master node of the EMR cluster using the EC2 key pair associated with the cluster.

4.Transfer Spark Job Code to the Master Node:

- Copy the Spark job code (e.g., my_spark_job.py) to the master node using the SCP or SFTP command. Alternatively, you can clone a Git repository containing the Spark job code directly on the master node.

5.RUN THE SPARK JOB :

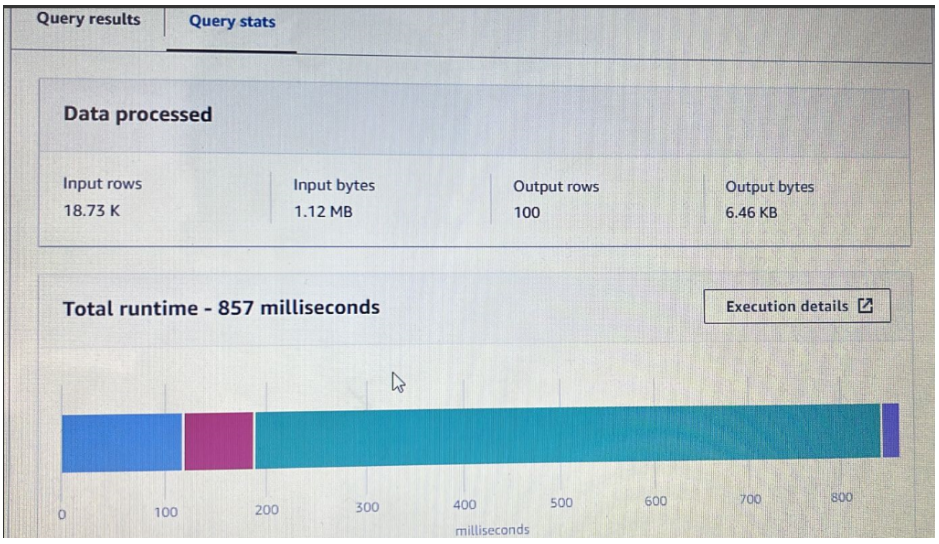
- Execute the Spark job on the master node using the spark-submit command. Adjust the command according to the specific job requirements:
- Copy code
- Ssh -i <location (or) path of the aws key pair> <cluster provided command for integrating master node with local >
- Spark-submit <filename.py>
- Monitor the job execution logs to ensure it runs successfully.

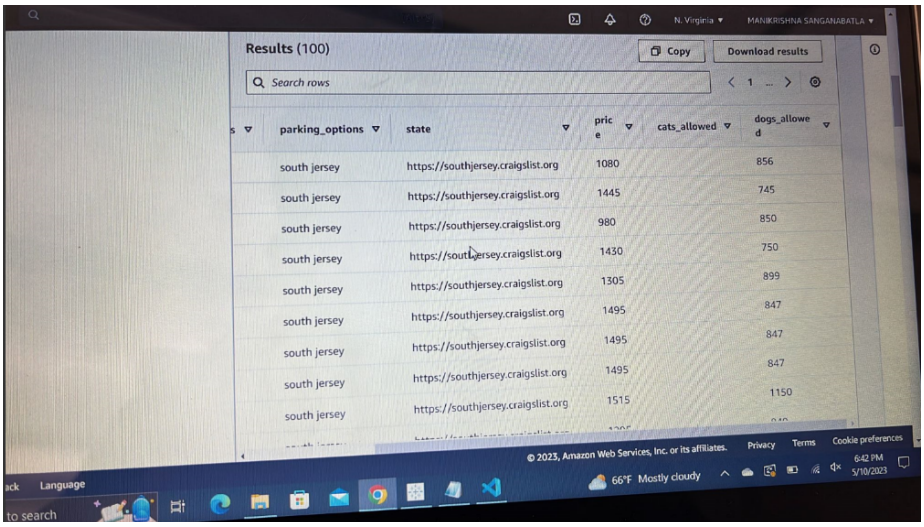
6.Store Output Files in S3:

- Once the Spark job completes, verify that the output files are generated.
- Upload the output files to the designated S3 output path or create a new S3 bucket for the output if needed.
- (6.1). ML prediction : For this project random forest works better, goto the jupyter notebook or google collab and execute the ml algorithm to get mse, accuracy, give four unique attributes to predict the rent of the house types in various regions. (shift + enter to excute the code).

7.Set Up Athena and Create a Table:

- Access the AWS Management Console and navigate to the Amazon Athena service.
- Open the Athena query editor.
- Create a new database if necessary and define a table that matches the schema of the output data.
- Use the "Create Table" wizard or write SQL statements to specify the table structure and provide the S3 location of the output data files.





8.Write SQL Queries for Visualizations in Tableau:

- Launch Tableau and establish a connection to Athena.
- Create a new workbook and add a new worksheet or dashboard.
- Drag and drop the desired dimensions and measures onto the canvas.
- Compose SQL queries within Tableau to retrieve the data from Athena for the visualizations.

- Customize the visualizations, apply filters, and create interactive elements to suit the analysis requirements.

TEST RESULTS :

1.EMR Cluster Creation:

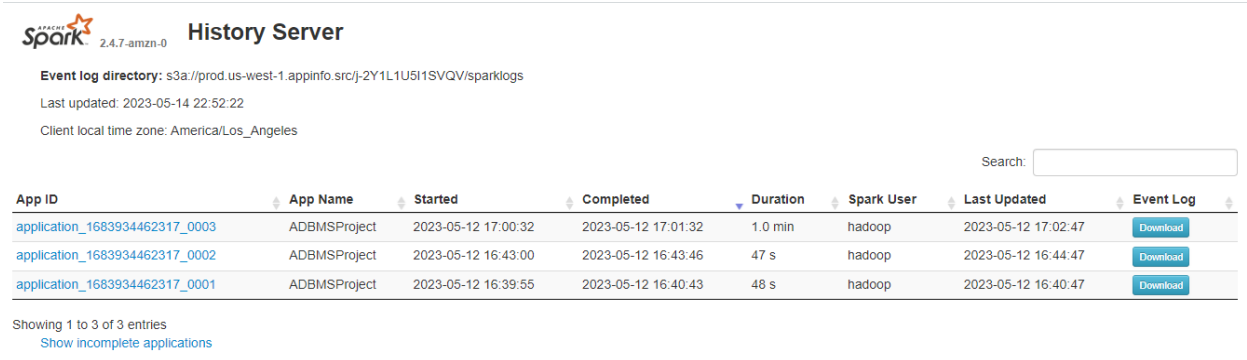
- Verify that the EMR cluster is successfully created without any errors or failures.
- Check the EMR cluster's status and ensure it transitions to the "Waiting" state.

2.Uploading Input Data to S3:

- Confirm that the input data files are uploaded to the specified S3 bucket.
- Validate the data files' integrity and ensure they are accessible.

3.Running the Spark Job:

- Connect to the EMR cluster's master node using SSH.
- Submit the Spark job and monitor the logs for any exceptions or errors.
- Check the output files generated by the Spark job and confirm they are saved to the specified S3 output path.
- Also check with the “Application User Interfaces”, such as , Spark History server,Yarn Timeline server:

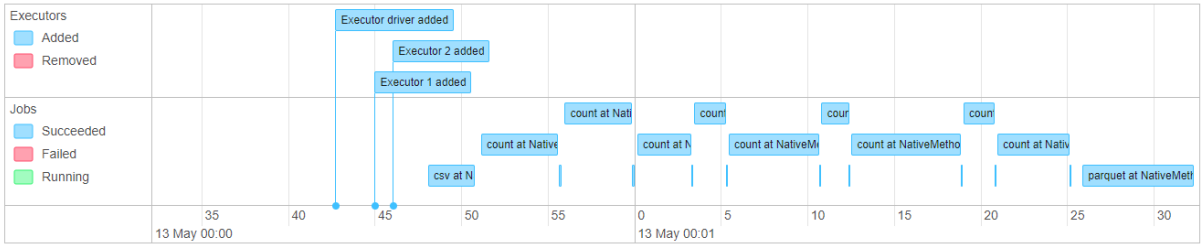


Spark Jobs (?)

User: hadoop
Total Uptime: 1.0 min
Scheduling Mode: FIFO
Completed Jobs: 20

Event Timeline

Enable zooming

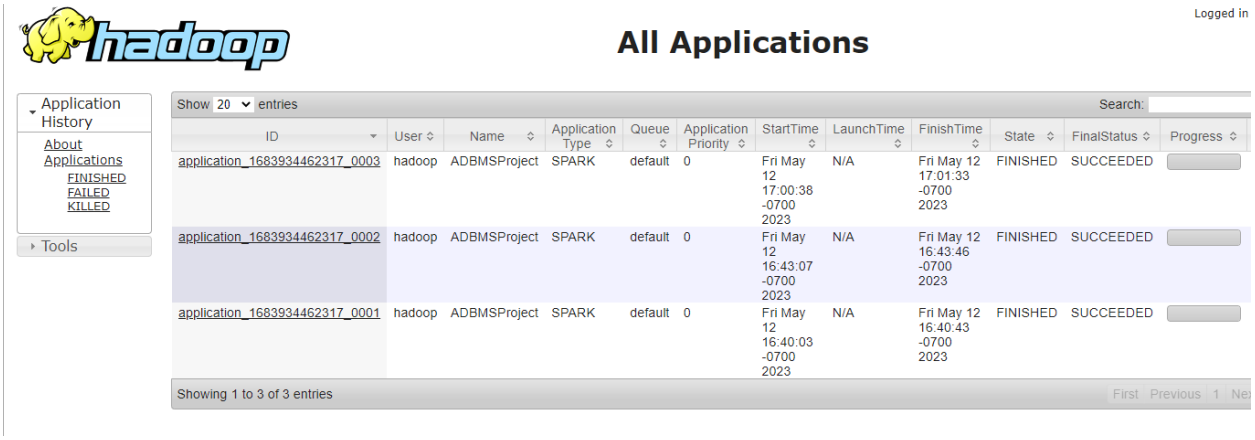


Completed Jobs (20)

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
19	parquet at NativeMethodAccessorImpl.java:0 parquet at NativeMethodAccessorImpl.java:0	2023/05/13 00:01:25	6 s	1/1	8/8
18	count at NativeMethodAccessorImpl.java:0	2023/05/13 00:01:25	26 ms	1/1 (1 skipped)	1/1 (8 skipped)

Completed Jobs (20)

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
19	parquet at NativeMethodAccessorImpl.java:0 parquet at NativeMethodAccessorImpl.java:0	2023/05/13 00:01:25	6 s	1/1	8/8
18	count at NativeMethodAccessorImpl.java:0 count at NativeMethodAccessorImpl.java:0	2023/05/13 00:01:25	26 ms	1/1 (1 skipped)	1/1 (8 skipped)
17	count at NativeMethodAccessorImpl.java:0 count at NativeMethodAccessorImpl.java:0	2023/05/13 00:01:20	4 s	1/1	8/8
16	count at NativeMethodAccessorImpl.java:0 count at NativeMethodAccessorImpl.java:0	2023/05/13 00:01:20	25 ms	1/1 (1 skipped)	1/1 (8 skipped)
15	count at NativeMethodAccessorImpl.java:0 count at NativeMethodAccessorImpl.java:0	2023/05/13 00:01:18	2 s	1/1	8/8
14	count at NativeMethodAccessorImpl.java:0 count at NativeMethodAccessorImpl.java:0	2023/05/13 00:01:18	30 ms	1/1 (1 skipped)	1/1 (8 skipped)
13	count at NativeMethodAccessorImpl.java:0 count at NativeMethodAccessorImpl.java:0	2023/05/13 00:01:12	6 s	1/1	8/8
12	count at NativeMethodAccessorImpl.java:0 count at NativeMethodAccessorImpl.java:0	2023/05/13 00:01:12	26 ms	1/1 (1 skipped)	1/1 (8 skipped)
11	count at NativeMethodAccessorImpl.java:0 count at NativeMethodAccessorImpl.java:0	2023/05/13 00:01:10	2 s	1/1	8/8
10	count at NativeMethodAccessorImpl.java:0 count at NativeMethodAccessorImpl.java:0	2023/05/13 00:01:10	27 ms	1/1 (1 skipped)	1/1 (8 skipped)
9	count at NativeMethodAccessorImpl.java:0 count at NativeMethodAccessorImpl.java:0	2023/05/13 00:01:05	5 s	1/1	8/8



4.Storing Output Files in S3:

- Verify that the output files are present in the designated S3 output path.
- Ensure the output files contain the expected data and have the correct file format.
- ML prediction results are displayed as below :

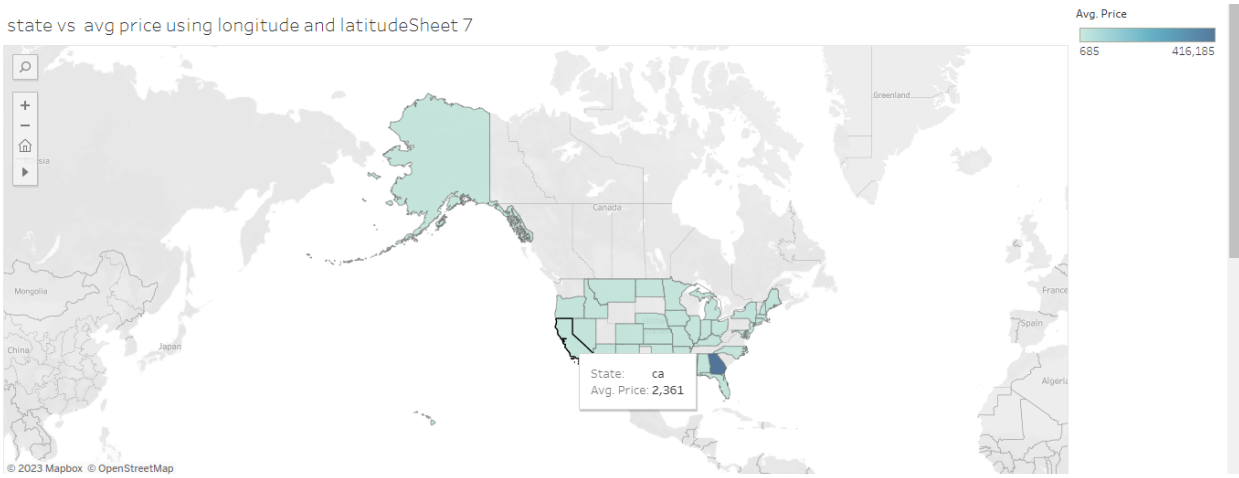
```
rf.fit(X_train, y_train)

# Make predictions on the test set
rf_pred = rf.predict(X_test)

# Evaluate the model using Mean Squared Error
rf_mse = mean_squared_error(y_test, rf_pred)
print('Random Forest MSE:', rf_mse)

# Make predictions on new data
new_data = pd.DataFrame({
    'type': ['house'],
    'state': ['ca'],
    'baths': [1],
    'beds': [2],
    'sqfeet': [1000]
})
new_data_cat = ohe.transform(new_data[['type', 'state']])
new_data_cat = pd.DataFrame(new_data_cat.toarray(), columns=ohe.get_feature_names_out(['type', 'state']))
new_data_num = new_data[['baths', 'beds', 'sqfeet']]
new_data_num = scaler.transform(new_data_num)
new_data_num = pd.DataFrame(new_data_num, columns=['baths', 'beds', 'sqfeet'])
new_data = pd.concat([new_data_cat, new_data_num], axis=1)
prediction = rf.predict(new_data)
print('Predicted rent:', prediction)

Fitting 5 folds for each of 27 candidates, totalling 135 fits
Best Parameters: {'max_depth': 5, 'min_samples_split': 2, 'n_estimators': 200}

Predicted rent: [1929.28690387]
```

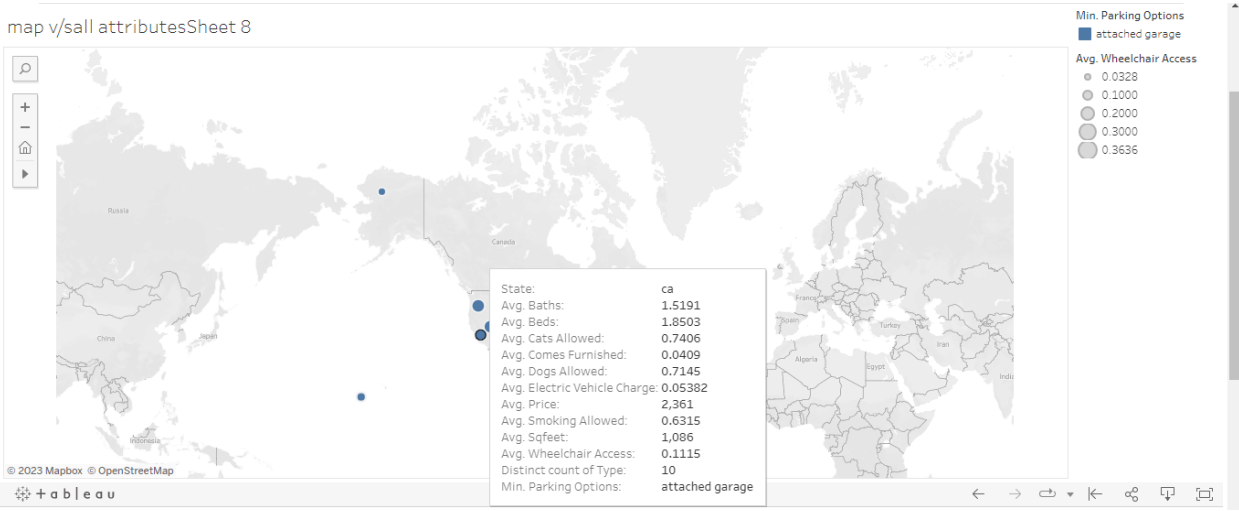
5. Using Output Data in Athena:

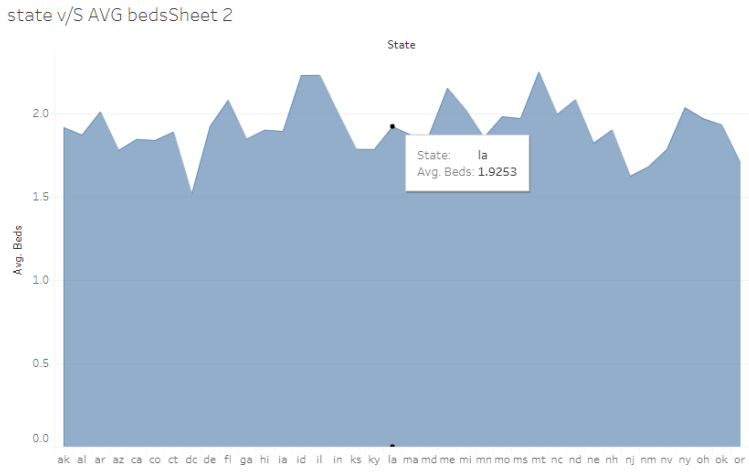
- Create a table in Athena using the output data files stored in S3.
- Run sample SQL queries against the table to retrieve and validate the data.
- Ensure the data matches the expected format and aligns with the original input.

6.Writing SQL Queries for Visualizations in Tableau:

- Connect Tableau to Athena and establish a successful connection.
- Create visualizations using the SQL queries against the Athena table.
- Verify that the visualizations accurately represent the analyzed data and provide the desired insights.







These are general steps to consider for testing the house rental analysis project. It's important to customize and expand upon these tests based on the specific requirements and project functionalities. Additionally, conducting end-to-end integration tests that cover the entire

workflow, from data ingestion to visualization, is recommended to ensure seamless execution and accurate results.

Note : Major part of the above execution of this project can be found in the Zip file which is uploaded in the canvas.

REFERENCES :

https://www.kaggle.com/datasets/rkb0023/houserentpredictiondataset

(All the information mentioned in the last deployment and steps to run methods also can be found in AWS Educate website)

https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-gs.html

(same above doc for S3 bucket creation, Athena)

https://help.tableau.com/current/pro/desktop/en-us/examples_amazonathena.htm