

BOSTON HOUSING DATA ENGINEERING PROJECT

Group 18

- Chetan Harshal Tote - 22631977
- Joshil Fernandes - 86334288
- Soham Sanjay Vaidya - 94169353
- Surabhi Kailas Sangore - 63101300
- Arbaaz Khaja Qutubuddin - 26428351
- Carolyn Gundimi - 48091939
- Aju Thomas - 48329426
- Devarsh Rajesh Bende - 80060804
- Omama Mashhood Ur Rahman - 48260972
- Kiran Kumar Pinjare - 46858007

INTRODUCTION

The Boston Housing Data Engineering Project is a comprehensive initiative that leverages cutting-edge technologies and tools to transform, analyze, and visualize housing data for the city of Boston. Employing a combination of Apache Airflow, Terraform, Google Cloud Platform (GCP), Docker, BigQuery, and Looker Studio, this project aims to streamline the end-to-end data engineering process for improved accessibility and insights into Boston's housing market.

METHODOLOGY

Dataset Selection:

Our objective is to conduct a thorough assessment of the uniformity of housing schemes in Boston. We Choose a dataset of interest from sources such as <https://www.kaggle.com/code/tolgahancepel/boston-housing-regression-analysis/notebook>

Data Ingestion:

Raw housing data is ingested from Kaggle and stored in Google Cloud Storage for further processing.

Data Transformation and Warehousing:

Apache Airflow orchestrates data transformation tasks, utilizing Terraform to provision necessary infrastructure. Processed data is stored in Big Query for efficient querying and analytics.

Containerized Deployment:

Docker ensures that the entire project, including dependencies, can be deployed consistently across different environments, promoting reproducibility and minimizing deployment issues.

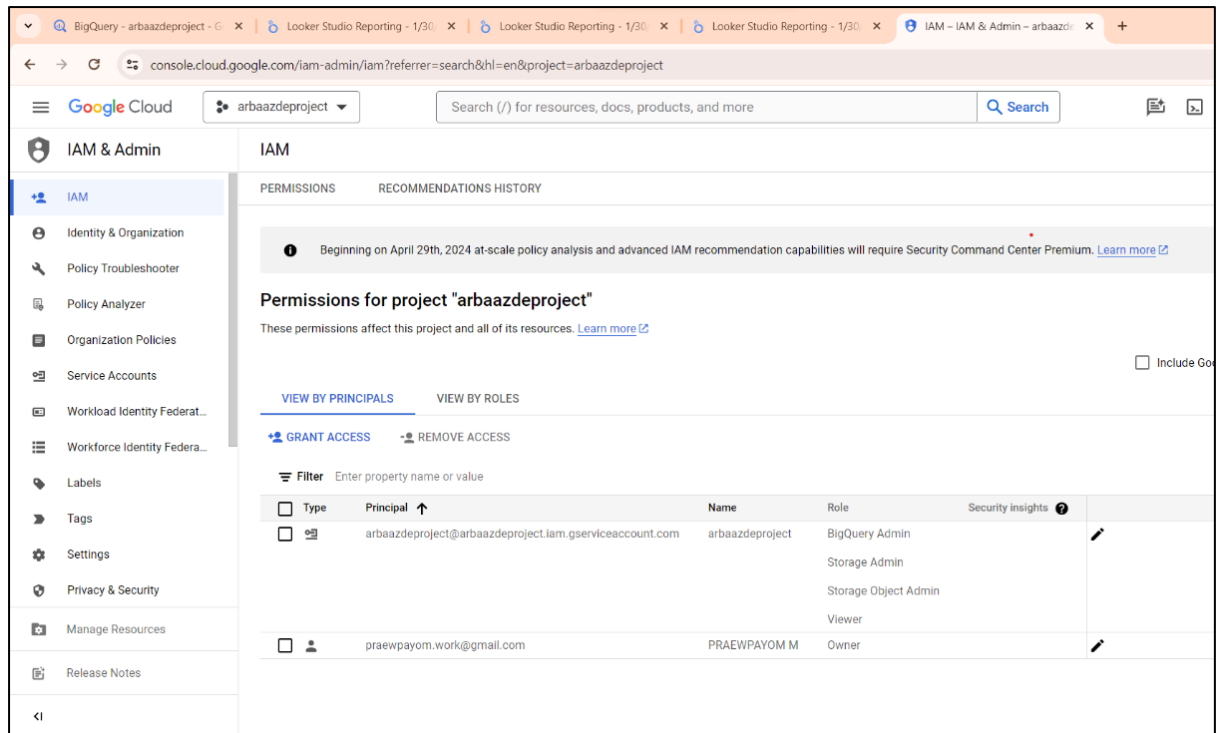
Interactive Data Exploration:

Looker Studio is employed to create interactive dashboards and reports, providing stakeholders with a user-friendly interface for exploring and gaining insights from the housing data.

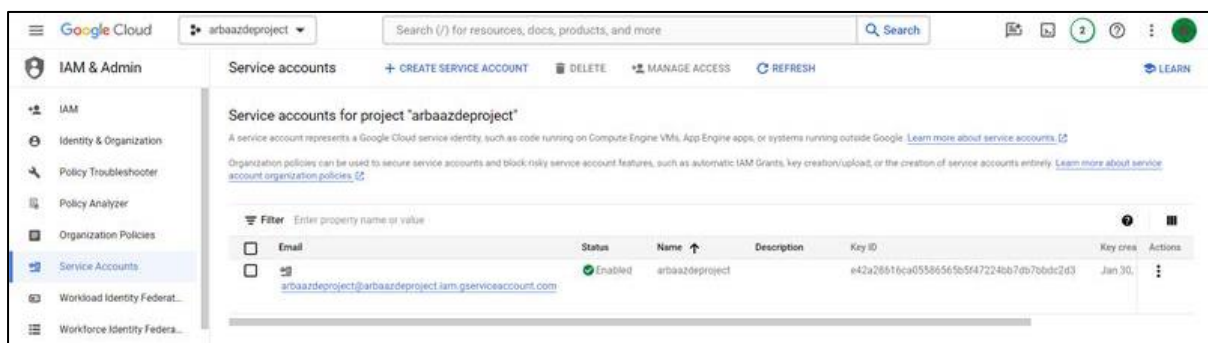
WORKFLOW:

Google Cloud Setup:

- Logged into the Google Cloud account and accessed the Cloud Console for managing project resources.
- Created a project ID tailored to the specific goals and scope of our project to maintain organizational clarity.
- Established a service account for the project, ensuring it was granted the necessary permissions, including roles pertinent to BigQuery and other relevant services.
- Enabled the required APIs directly within the Google Cloud Console to ensure seamless access to functionalities critical for our project.
- Configured a service account, generating a key JSON file associated with it to facilitate secure authentication for our interactions with Google Cloud services.



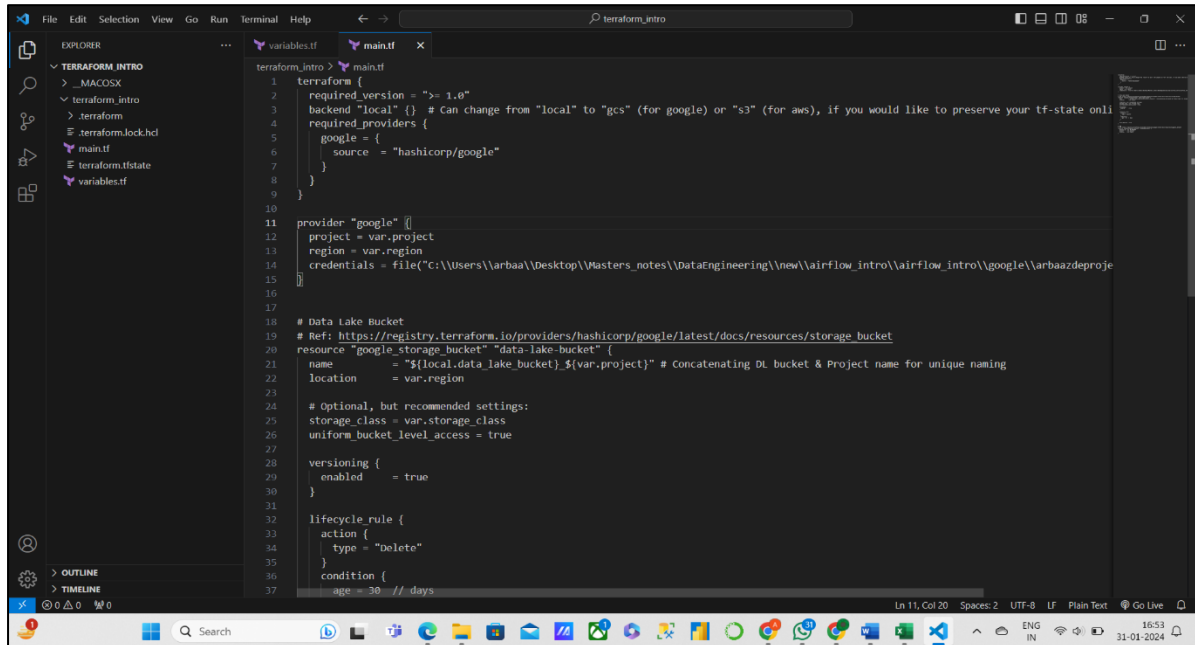
- Saved the generated JSON file in the "google" folder within the "airflow_intro" directory for streamlined organization and reference.
- Duplicated the same JSON file into the ".google" folder to ensure consistent and convenient access across various components of our project.
- Utilized the key file as our primary authentication mechanism, securing and authorizing interactions with Google Cloud services, particularly within the Apache Airflow environment.



Terraform Setup:

- Confirmed the presence of Terraform on our laptop before proceeding with the project tasks.
- Accessed the dedicated "terraform" folder housing our Terraform configuration files.
- Successfully updated the "main.tf" and "variables.tf" files as per the provided instructions, ensuring precise configuration tailored to our project's needs.
- Opened a command prompt within the "terraform" folder to seamlessly execute the necessary commands.
- Initialized the Terraform project using the `terraform init` command, which automatically fetched essential plugins and established the working directory.

- Reviewed the anticipated changes in the infrastructure by executing `terraform plan`, providing a comprehensive preview before implementing modifications.
- Finally, executed `terraform apply` and confirmed the action by typing "yes" when prompted. This command effectively applied the planned changes, provisioning or updating resources based on the specified configurations in our Terraform files.

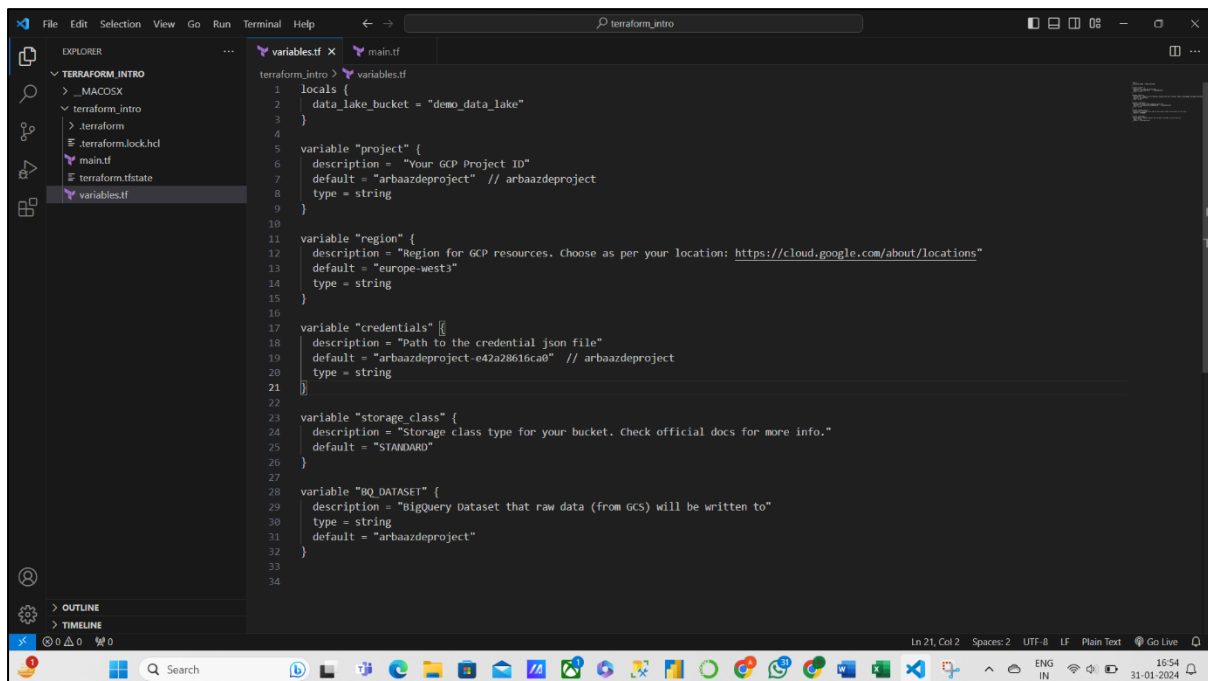


```

1 terraform {
2   required_version = ">= 1.0"
3   backend "local" {} # Can change from "local" to "gcs" (for google) or "s3" (for aws), if you would like to preserve your tf-state onli
4   required_providers {
5     google = {
6       source = "hashicorp/google"
7     }
8   }
9 }
10
11 provider "google" {
12   project = var.project
13   region = var.region
14   credentials = file("c:\\Users\\arbaa\\Desktop\\Masters_notes\\DataEngineering\\new\\airflow_intro\\airflow_intro\\google\\arbaazdeproje
15 }
16
17
18 # Data Lake Bucket
19 # Ref: https://registry.terraform.io/providers/hashicorp/google/latest/docs/resources/storage_bucket
20 resource "google_storage_bucket" "data-lake-bucket" {
21   name = "${local.data_lake_bucket}${var.project}" # Concatenating DL bucket & Project name for unique naming
22   location = var.region
23
24   # Optional, but recommended settings:
25   storage_class = var.storage_class
26   uniform_bucket_level_access = true
27
28   versioning {
29     enabled = true
30   }
31
32   lifecycle_rule {
33     action {
34       type = "Delete"
35     }
36     condition {
37       age = 30 // days

```

Main.tf File : Updated the path of the JSON file



```

1 locals {
2   data_lake_bucket = "demo_data_lake"
3 }
4
5 variable "project" {
6   description = "Your GCP Project ID"
7   default = "arbaazdeproject" // arbaazdeproject
8   type = string
9 }
10
11 variable "region" {
12   description = "Region for GCP resources. Choose as per your location: https://cloud.google.com/about/locations"
13   default = "europe-west3"
14   type = string
15 }
16
17 variable "credentials" {
18   description = "Path to the credential json file"
19   default = "arbaazdeproject-e42a28616ca0" // arbaazdeproject
20   type = string
21 }
22
23 variable "storage_class" {
24   description = "Storage class type for your bucket. Check official docs for more info."
25   default = "STANDARD"
26 }
27
28 variable "BQ_DATASET" {
29   description = "BigQuery Dataset that raw data (from GCS) will be written to"
30   type = string
31   default = "arbaazdeproject"
32 }
33
34

```

Variable.tf File : Updated the project, credentials, big query dataset variable.

Cloud Storage

Buckets

Monitoring

Einstellungen

ERSTELLEN

AKTUALISIEREN

LERNEN

Ab dem 29. April 2024 ist für größere Richtlinienanalysen und fortgeschrittene IAM-Funktionen zur Empfehlungserstellung Security Command Center Premium erforderlich. [Weitere Informationen](#)

SCHLIESSEN

Filter

Buckets filtern

<input type="checkbox"/>	Name ↑	Erstellt	Standorttyp	Speicherort	Standard-Speicherklasse ?	Zuletzt geändert	Öffentlicher Zugriff ?	Zu
<input type="checkbox"/>	demo_data_lake_arbaazdeproject	30.01.2024, 17:44:17	Region	europe-west3	Standard	30.01.2024, 17:44:17	Nicht öffentlich	El ⋮

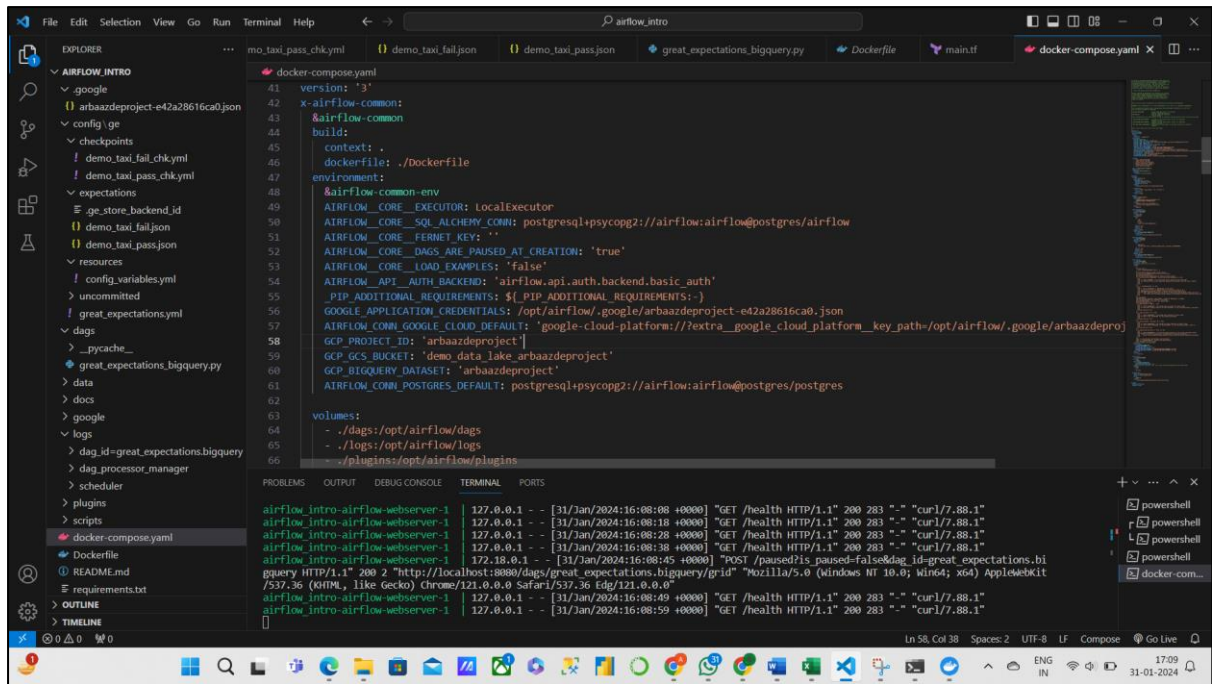
Bucket created using terraform

Resources are displayed.	SHOW ONLY CHECKED ITEMS
<ul style="list-style-type: none"> arbaazdeproject <ul style="list-style-type: none"> Saved queries (1) External connections arbaazdeproject <ul style="list-style-type: none"> Lstat wheel arbaazdeproject 	<ul style="list-style-type: none"> ☆ ☆ ☆ ☆ ☆

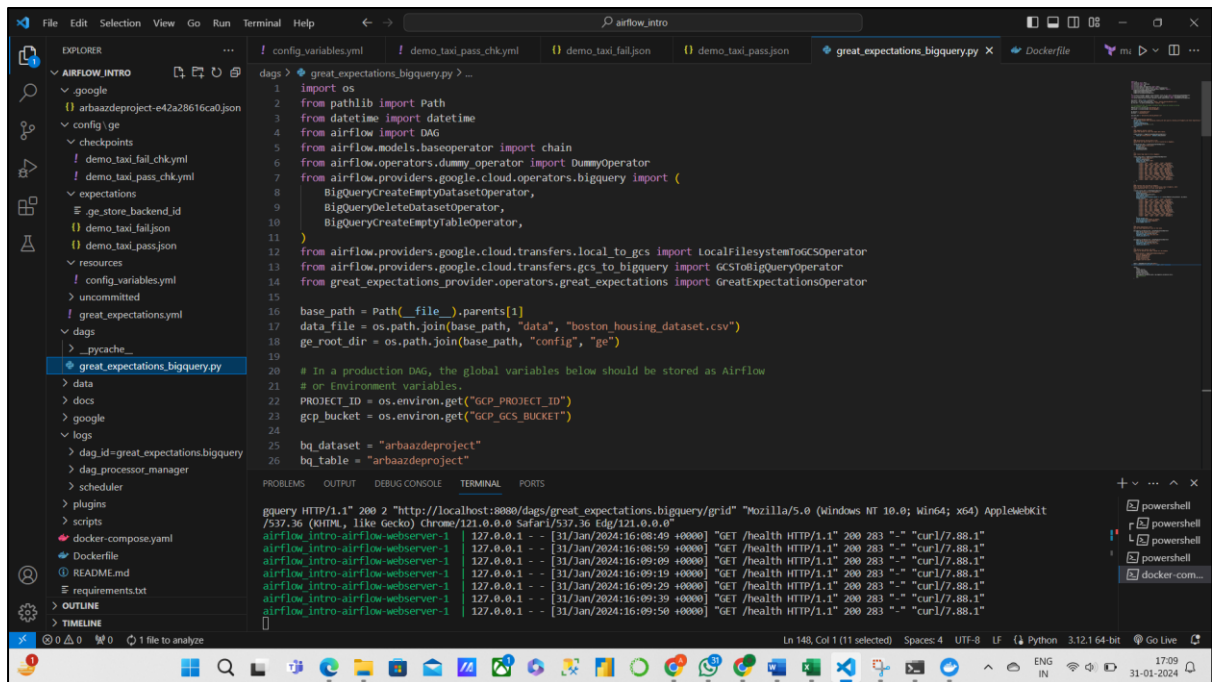
Folder created in bigquery using terraform

Airflow Setup:

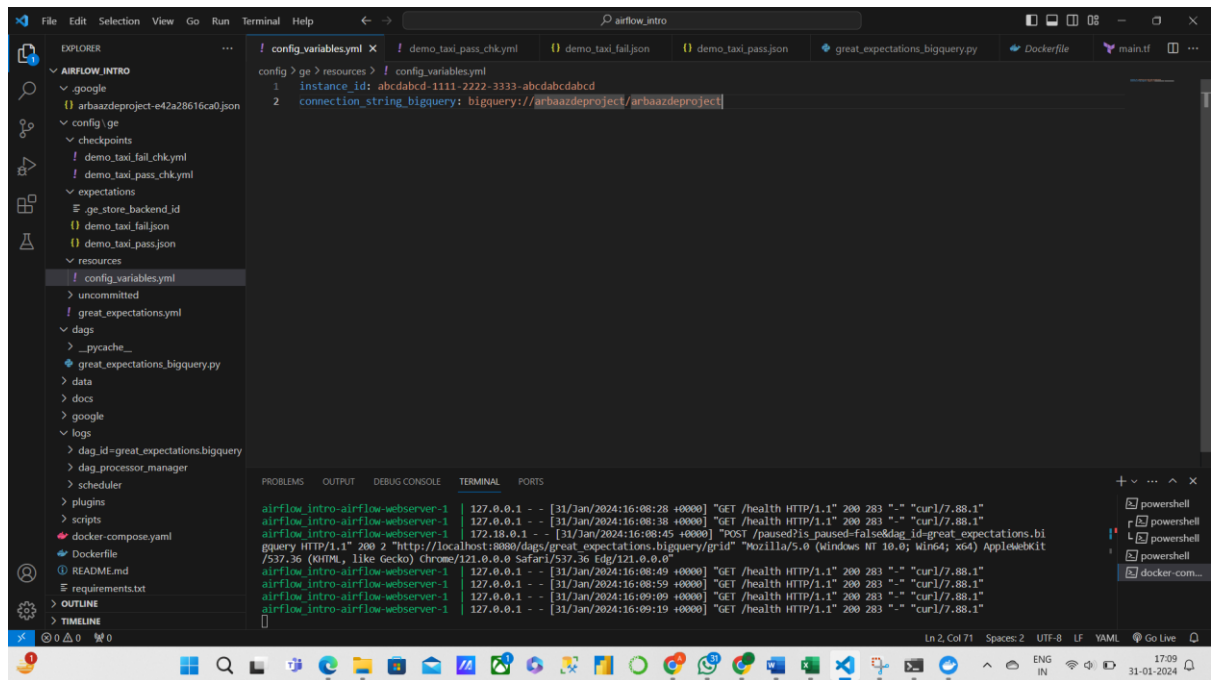
- Adjusted credentials and configurations in various files for the Airflow setup to align with our project specifics.
- Edited the `docker-compose.yaml` file, making accurate replacements for project IDs, key names, bucket names, and dataset names as specified.



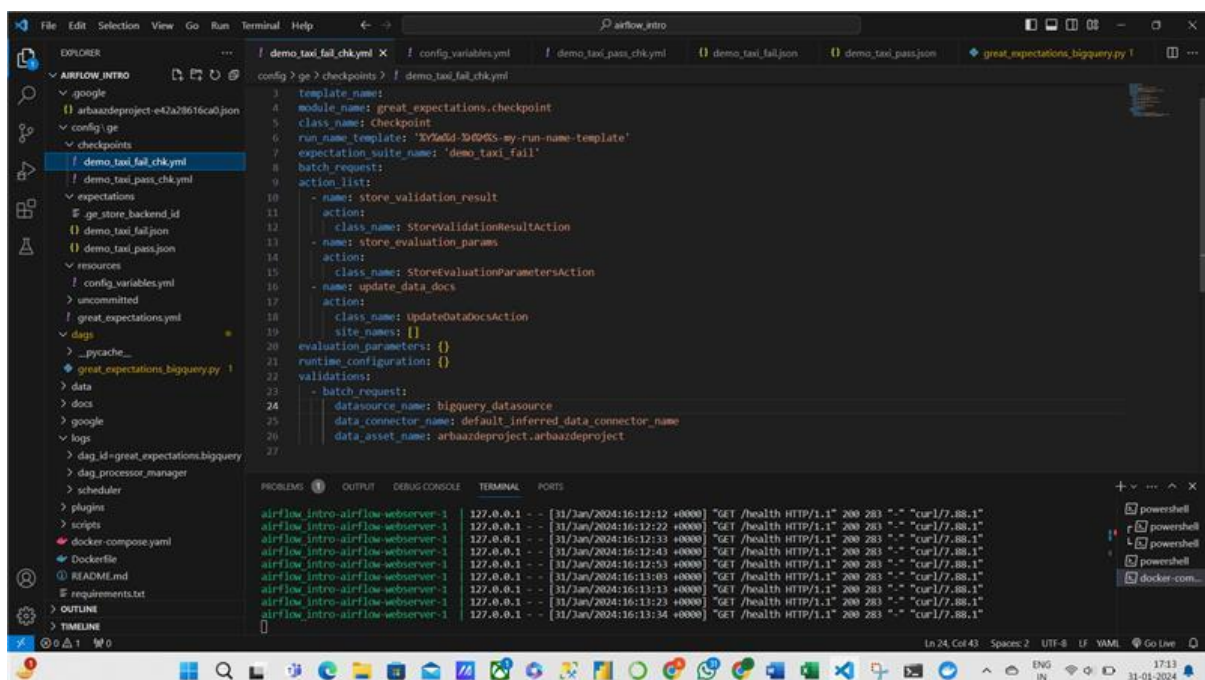
- Modified the `great_expectations_bigquery.py` file to reflect our project's requirements, ensuring accurate replacement of project IDs, key names, bucket names, and dataset names.



- Updated the `config_variables.yml` file with the specific project IDs, key names, bucket names, and dataset names, ensuring consistency with our project.



- Made the necessary modifications to `demo_taxi_fail_chk.yml` and `demo_taxi_pass_chk.yml` files, replacing project IDs, key names, bucket names, and dataset names to accurately represent our project's configurations.

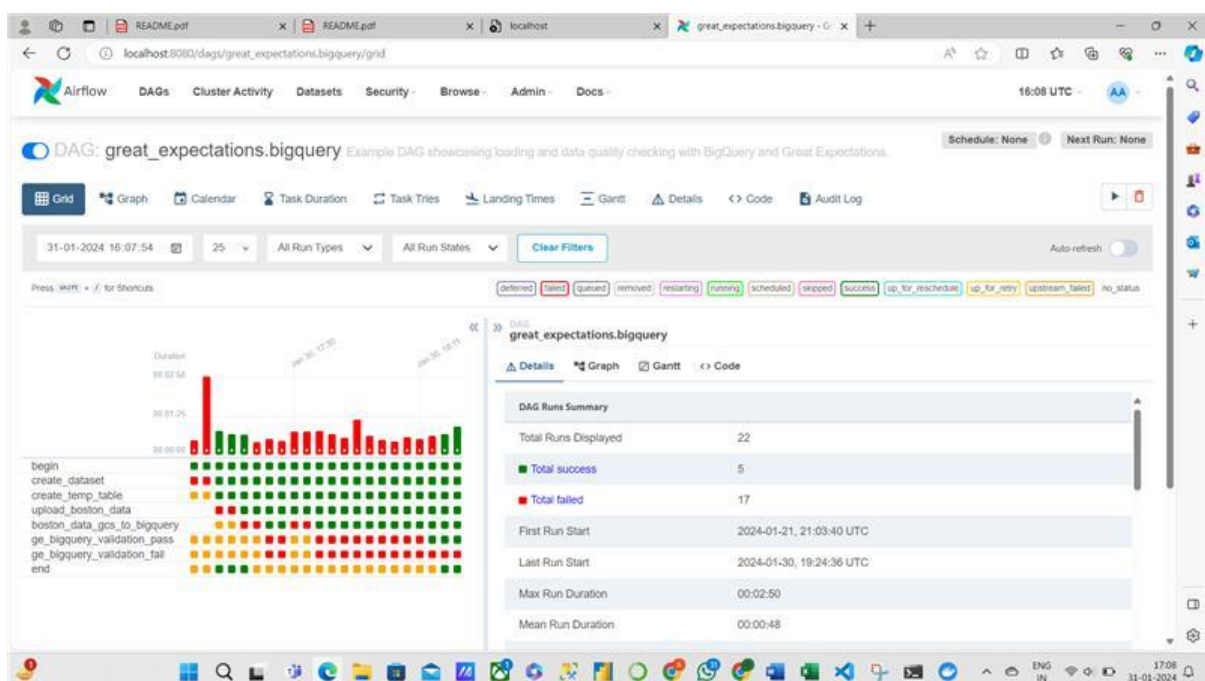


- Verified that each replacement in the files was accurate and aligned with our project's unique identifiers and settings. These adjustments are crucial to ensure that Airflow operates seamlessly with the correct credentials and configurations in our specific environment.

Execution:

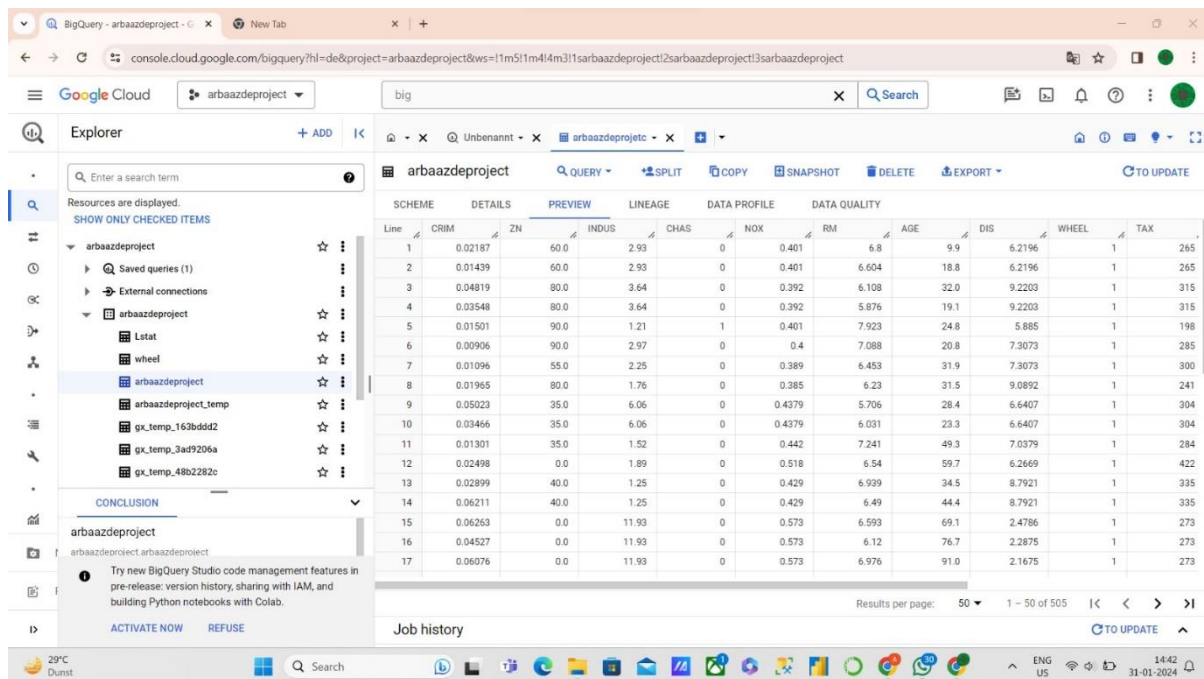
Executed the steps detailed in the README.MD file within the "airflow_intro" folder for a seamless setup of Apache Airflow:

- Initiated the Docker build process using the command ``docker-compose build`` to create the necessary Docker containers for our Airflow environment.
- Initialized Airflow by running ``docker-compose up airflow-init``. This step ensured that Airflow's essential components were set up, allowing for a functional environment.
- Started Airflow by executing ``docker-compose up``. This command launched the Airflow web server and scheduler within the Docker containers, enabling the orchestration of our data pipelines.
- Accessed the Airflow Web UI at the local host, logging in with the provided credentials (username: airflow, password: airflow) to interact with and manage our workflows visually.
- Executed the "great_expectations_bigquery.py" pipeline within the Airflow environment. This pipeline leverages Great Expectations to validate and assess data in BigQuery, providing valuable insights and ensuring data quality.
- Verified the successful execution of each step to ensure that our Airflow environment was correctly set up and the specified pipeline executed without issues.



Further Steps:

- Confirmed the existence and structure of the dataset table created in the Google Cloud BigQuery dashboard, ensuring successful deployment of the table in our designated project.
- Executed queries to generate the required supplementary tables, enhancing the dataset's depth and scope.
- Leveraged the newly generated tables along with the main dataset table as data sources for crafting visualizations and dashboards in Looker Studio. This involved utilizing Looker's intuitive interface to design insightful and interactive visual representations of the data.
- Ensured that the Looker Studio dashboards provided a comprehensive view of the dataset, allowing stakeholders to explore and derive meaningful insights from the combined data sources. This step marked a crucial aspect of the data engineering project, as it enabled effective data visualization and interpretation for informed decision-making.



Line	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	WHEEL	TAX
1	0.02187	60.0	2.93	0	0.401	6.8	9.9	6.2196	1	265
2	0.01439	60.0	2.93	0	0.401	6.604	18.8	6.2196	1	265
3	0.04819	80.0	3.64	0	0.392	6.108	32.0	9.2203	1	315
4	0.03548	80.0	3.64	0	0.392	5.876	19.1	9.2203	1	315
5	0.01501	90.0	1.21	1	0.401	7.923	24.8	5.885	1	198
6	0.00906	90.0	2.97	0	0.4	7.088	20.8	7.3073	1	285
7	0.01096	55.0	2.25	0	0.389	6.453	31.9	7.3073	1	300
8	0.01965	80.0	1.76	0	0.385	6.23	31.5	9.0892	1	241
9	0.05023	35.0	6.06	0	0.4379	5.706	28.4	6.6407	1	304
10	0.03466	35.0	6.06	0	0.4379	6.031	23.3	6.6407	1	304
11	0.01301	35.0	1.52	0	0.442	7.241	49.3	7.0379	1	284
12	0.02498	0.0	1.89	0	0.518	6.54	59.7	6.2669	1	422
13	0.02899	40.0	1.25	0	0.429	6.939	34.5	8.7921	1	335
14	0.06211	40.0	1.25	0	0.429	6.49	44.4	8.7921	1	335
15	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273
16	0.04527	0.0	11.93	0	0.573	6.12	76.7	2.2875	1	273
17	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273

DAG used in the project:

Environment Setup:

- Defined the **base_path** variable, determining the project's base directory path, derived from the current file's path.
- Constructed the full path for the Boston housing dataset CSV file using the **data_file** variable, combining the base path, "data" directory, and the file name.
- Specified the **ge_root_dir** variable, indicating the directory path where Great Expectations configurations are stored, formed by joining the base path, "config" directory, and "ge" subdirectory.

DAG Configuration:

- Retrieved GCP project ID and GCS bucket from environment variables (**PROJECT_ID** and **gcp_bucket**).
- Defined names for the BigQuery dataset and table (**bq_dataset** and **bq_table**).
- Specified the destination path for the data file on Google Cloud Storage (GCS) using the **gcp_data_dest** variable.

DAG Definition:

- Defined an Airflow DAG named "great_expectations.bigquery" with metadata such as description, documentation, schedule interval (set to None for manual triggering), start date, and catch-up policy.

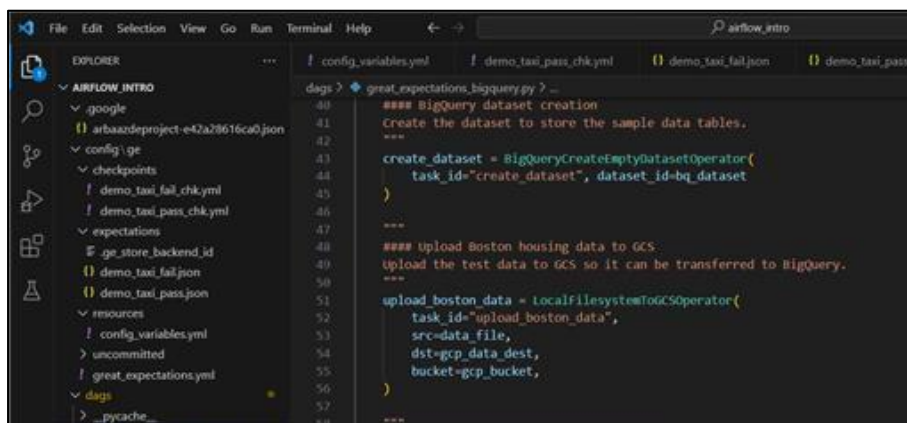
Tasks:

- **Task 1 (create_dataset):**

Used **BigQueryCreateEmptyDatasetOperator** to create an empty BigQuery dataset with the specified **bq_dataset** name.

- **Task 2 (upload_boston_data):**

Utilized **LocalFilesystemToGCSOperator** to upload the Boston housing dataset CSV file to GCS, ensuring data availability for subsequent processing.



```
File Edit Selection View Go Run Terminal Help
airflow_intro

EXPLORER
  AIRFLOW_INTRO
    google
      arbaazdep@project-e42a28616ca0.json
    config
      ge
        checkpoints
          demo_taxi_fall.chk.yml
          demo_taxi_pass.chk.yml
        expectations
          ge_store_backend_id
          demo_taxi_fall.json
          demo_taxi_pass.json
    resources
      config_variables.yml
    uncommitted
      great_expectations.yml
    dags
      __pycache__

dags > great_expectations.bigquery.py > ...
40  """ BigQuery dataset creation
41  Create the dataset to store the sample data tables.
42  """
43  create_dataset = BigQueryCreateEmptyDatasetOperator(
44      task_id="create_dataset", dataset_id=bq_dataset
45  )
46
47  """
48  """ Upload Boston housing data to GCS.
49  Upload the test data to GCS so it can be transferred to BigQuery.
50  """
51  upload_boston_data = LocalFilesystemToGCSOperator(
52      task_id="upload_boston_data",
53      src=data_file,
54      dst=gcp_data_dest,
55      bucket=gcp_bucket,
56  )
57
58  """
```

- **Task 3 (create_temp_table):**

Used **BigQueryCreateEmptyTableOperator** to create an empty temporary BigQuery table (**{bq_table}_temp**) within the specified dataset.

Defined the table schema with various fields such as "CRIM," "ZN," etc.

```

dag > great_expectations_biqquery.py ? ...
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82

```

```

""" Create Temp Table for GE in BigQuery """
create_temp_table = BigQueryCreateEmptyTableOperator(
    task_id="create_temp_table",
    dataset_id=bq_dataset,
    table_id=f"{bq_table}_temp",
    schema_fields=[
        {"name": "CRIM", "type": "FLOAT", "mode": "NULLABLE"},
        {"name": "ZN", "type": "FLOAT", "mode": "NULLABLE"},
        {"name": "INDUS", "type": "FLOAT", "mode": "NULLABLE"},
        {"name": "CHAS", "type": "INTEGER", "mode": "NULLABLE"},
        {"name": "NOX", "type": "FLOAT", "mode": "NULLABLE"},
        {"name": "RM", "type": "FLOAT", "mode": "NULLABLE"},
        {"name": "AGE", "type": "FLOAT", "mode": "NULLABLE"},
        {"name": "DIS", "type": "FLOAT", "mode": "NULLABLE"},
        {"name": "RAD", "type": "INTEGER", "mode": "NULLABLE"},
        {"name": "TAX", "type": "INTEGER", "mode": "NULLABLE"},
        {"name": "PTRATIO", "type": "FLOAT", "mode": "NULLABLE"},
        {"name": "B", "type": "FLOAT", "mode": "NULLABLE"},
        {"name": "LSTAT", "type": "FLOAT", "mode": "NULLABLE"},
        {"name": "MEDV", "type": "FLOAT", "mode": "NULLABLE"},
    ],
)

```

- **Task 4 (transfer_boston_data):**

Employed **GCSToBigQueryOperator** to move the data uploaded to GCS to BigQuery.

Specified details like source format (CSV), schema fields, and destination table in BigQuery.

```

alp.
_variables.yml
demo_taxi_pass.chk.yml
demo_taxi_fail.json
demo_taxi_pass.json
great_expectations_biqquery.py ? ...

```

```

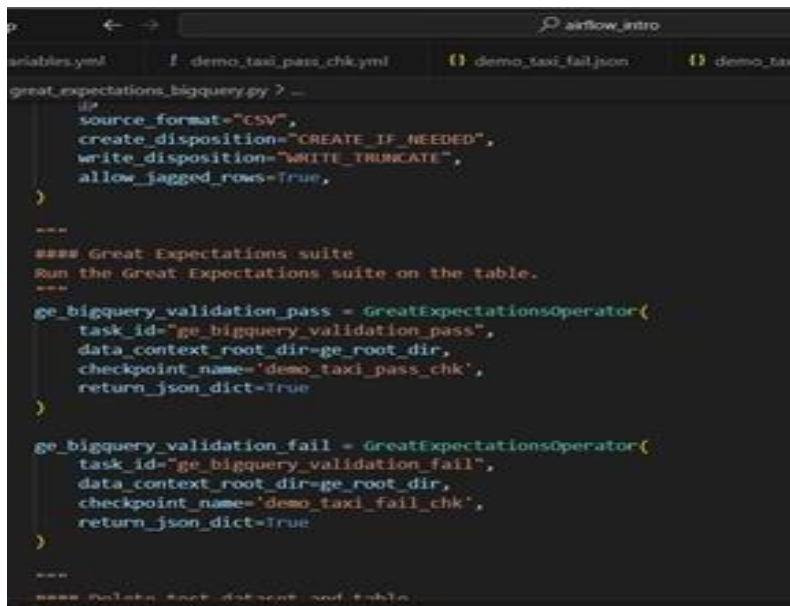
transfer_boston_data = GCSToBigQueryOperator(
    task_id="boston_data_gcs_to_bigquery",
    bucket=gcp_bucket,
    source_objects=[gcp_data_dest],
    skip_leading_rows=1,
    destination_project_dataset_table="{}.{}.format(PROJECT_ID, bq_dataset, bq_table)",
    schema_fields=[
        {"name": "CRIM", "type": "FLOAT", "mode": "NULLABLE"},
        {"name": "ZN", "type": "FLOAT", "mode": "NULLABLE"},
        {"name": "INDUS", "type": "FLOAT", "mode": "NULLABLE"},
        {"name": "CHAS", "type": "INTEGER", "mode": "NULLABLE"},
        {"name": "NOX", "type": "FLOAT", "mode": "NULLABLE"},
        {"name": "RM", "type": "FLOAT", "mode": "NULLABLE"},
        {"name": "AGE", "type": "FLOAT", "mode": "NULLABLE"},
        {"name": "DIS", "type": "FLOAT", "mode": "NULLABLE"},
        {"name": "RAD", "type": "INTEGER", "mode": "NULLABLE"},
        {"name": "TAX", "type": "INTEGER", "mode": "NULLABLE"},
        {"name": "PTRATIO", "type": "FLOAT", "mode": "NULLABLE"},
        {"name": "B", "type": "FLOAT", "mode": "NULLABLE"},
        {"name": "LSTAT", "type": "FLOAT", "mode": "NULLABLE"},
        {"name": "MEDV", "type": "FLOAT", "mode": "NULLABLE"},
    ],
    source_format="CSV",
    create_disposition="CREATE_IF_NEEDED",
    write_disposition="WRITE_TRUNCATE",
    allow_large_files=True,
)

```

- **Tasks 5 and 6 (ge_biqquery_validation_pass and ge_biqquery_validation_fail):**

Utilized the Great Expectations Operator to run validation tests on the specified BigQuery table.

Two tasks were created for both pass and fail scenarios, each associated with a different Great Expectations checkpoint.



```
great_expectations_bigquery.py > ...
    if
        source_format="CSV",
        create_disposition="CREATE_IF_NEEDED",
        write_disposition="WRITE_TRUNCATE",
        allow_jagged_rows=True,
    )

    """
    """ Great Expectations suite
    Run the Great Expectations suite on the table.
    """
    ge_bigquery_validation_pass = GreatExpectationsOperator(
        task_id="ge_bigquery_validation_pass",
        data_context_root_dir=ge_root_dir,
        checkpoint_name='demo_taxi_pass_chk',
        return_json_dict=True
    )

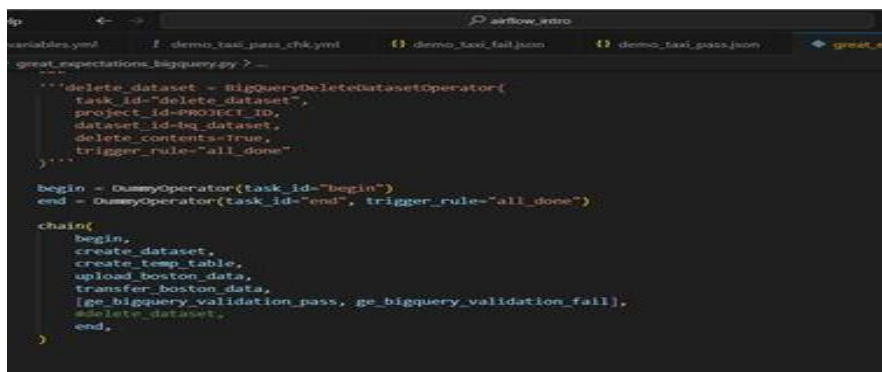
    ge_bigquery_validation_fail = GreatExpectationsOperator(
        task_id="ge_bigquery_validation_fail",
        data_context_root_dir=ge_root_dir,
        checkpoint_name='demo_taxi_fail_chk',
        return_json_dict=True
    )

    """
    """ Delete test dataset and table
    """
```

- Tasks 7 and 8 (begin and end):

begin and **end** were dummy operators used as placeholders.

begin marked the start of the DAG and **end** signified its completion.



```
great_expectations_bigquery.py > ...
    """delete_dataset = BigQueryDeleteDatasetOperator(
        task_id="delete_dataset",
        project_id=project_id,
        dataset_id=bq_dataset,
        delete_contents=True,
        trigger_rule="all_done"
    )"""

    begin = DummyOperator(task_id="begin")
    end = DummyOperator(task_id="end", trigger_rule="all_done")

    chain(
        begin,
        create_dataset,
        create_temp_table,
        upload_boston_data,
        transfer_boston_data,
        [ge_bigquery_validation_pass, ge_bigquery_validation_fail],
        delete_dataset,
        end,
    )
```

- Task Execution Flow:

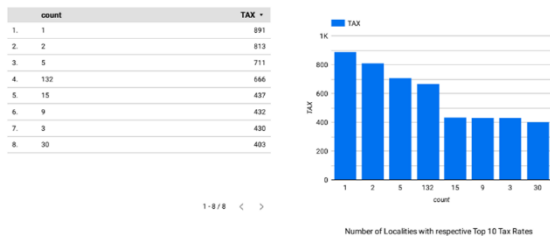
Utilized the **chain** function to define the execution order of tasks in the DAG.

The sequence was: **begin** -> **create_dataset** -> **create_temp_table** -> **upload_boston_data** -> **transfer_boston_data** -> [**ge_bigquery_validation_pass**, **ge_bigquery_validation_fail**] -> **end**.

This chaining ensured tasks were executed in a logical order, orchestrating the entire data processing workflow from dataset creation to data validation.

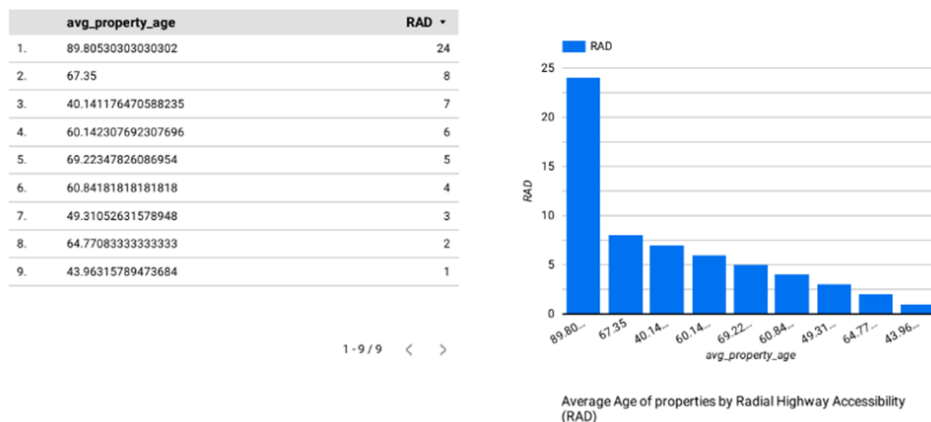
Analysing Visualisation

top10taxrates



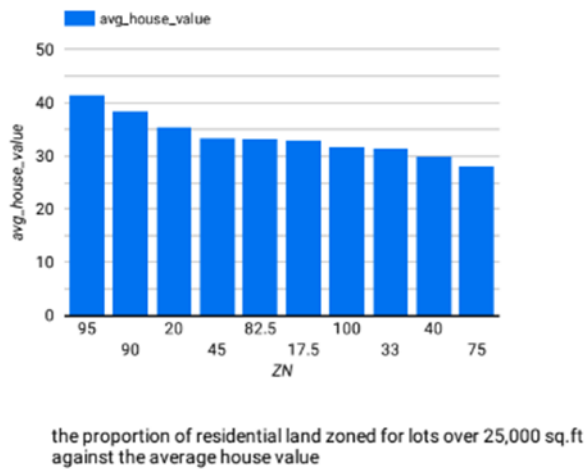
- The image depicts the number of localities with the top 10 tax rates. Only 1 locality has the highest tax rate, amounting to 891, while the 30 localities have the lowest rate, making up to a total tax rate of 403. Hence, it is feasible to rent a home in these localities due to the low interest rates. In the locality having the highest interest rate, it would not be feasible for most of the population to sustain a living there, hence, the housing pattern in Boston is quite uneven.

AVG_PROPERTY_AGE vs RAD



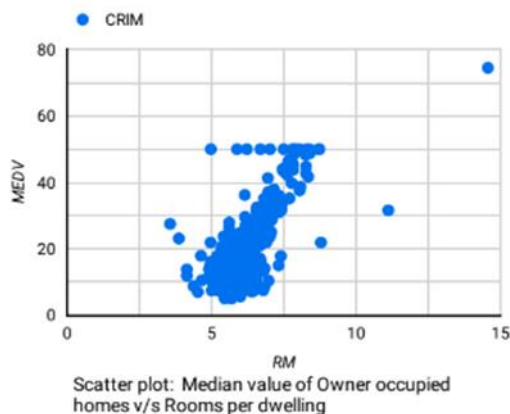
The graph above shows the average of the proportion of owner-occupied units built prior to 1940 in accordance with the access to radial highways. 89.8 (approximately 90) houses built prior to 1940 have the highest index of accessibility to the radial highways whereas 43 houses have the lowest index e.g. only 1 access to the highways. On a generic build, the rise in the access to the highway is quite sharp, descending from 1 to 8, and then a peak rise to 24. This depicts how 90 houses have a access well supported to the highways, making the area well suited for most of the population, connecting it to the other regions; however, the other localities don't enjoy this luxury. Hence, this again concludes that the housing schemes in Boston are quite uneven.

AVG_HOUSE_VALUE against ZN



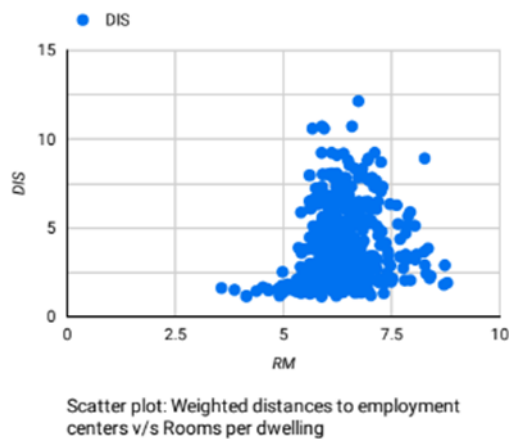
The graph runs via the relation between the average house value of the homes and the proportion of residential land zoned. The relationship is linearly decreasing between both the variables. 95 homes worth 42,000\$ had been restricted for use, making the highest loss of 3990000\$. Only 75 homes worth 28,000\$, which is the lowest cost until now for the land, had been restricted, incurring a loss of 2100000\$. The lowest loss is far less than the greatest loss. Hence, the housing schemes is built on a unequal grounds the land should be used more efficiently, minimizing the differences between both the losses.

DISTRIBUTION OF THE PER CAPITA CRIME RATE BY TOWN (MEDV vs RM)



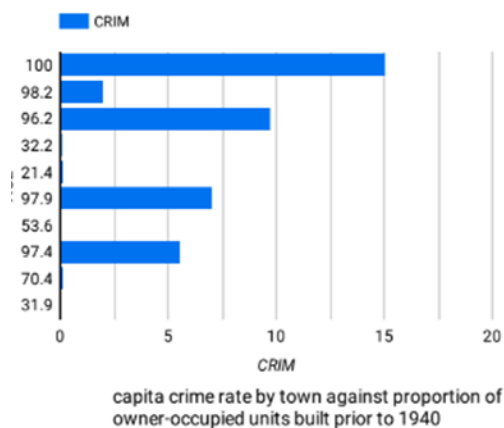
The following outlines the distribution of the per capita crime rate by town based on the relationship of the average number of the rooms and the median value of the homes. Most of the crimes are concentrated where the rooms are 4-7 and the value lies between 5,000\$-45,000\$. Whereas the houses having 15 rooms and the value of over 72,000\$ have the least or rare crimes. This could be evidence of the kind of localities and the security provided to them. The houses with a higher value indicating the posher locality, would be having a higher security in contrast to the average housing schemes. Hence, the housing scheme in Boston proves to be uneven throughout.

WEIGHTED DISTANCES TO EMPLOYMENT CENTERS



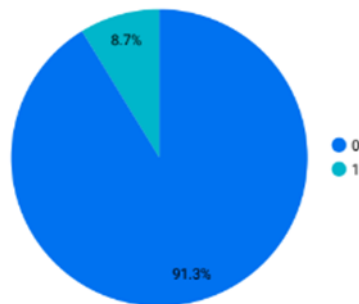
The figure below shows the relationship between the weighted distances to five Boston employment centers and the Average number of rooms per dwelling. The Most of the employment centers lie within 1-9 km for the houses having the rooms in the range of 3-8. This means, most of the average localities having houses with these rooms have employment centers nearby as most of the population would be residing here. Therefore, the housing scheme seems to be even in this matter of fact.

CAPITA CRIME RATE AGAINST PROPORTION OF HOUSES



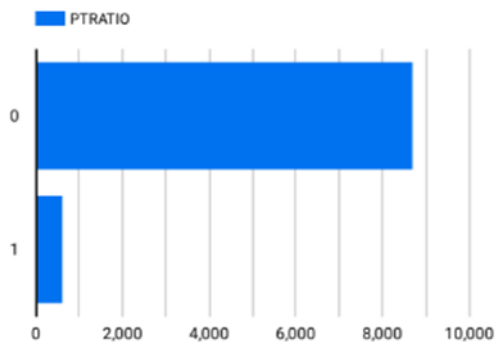
Here, the crime rates per capita are compared with the proportion of houses built before 1940. 100 houses that were built prior to 1940 have the highest crime rate, that amounts to 15%. Whereas 32 houses built have a crime rate of 0.1%. Most of the houses in Boston face crime in most of the localities, when compared to the average crime rate that amounts to 10%. Making most of the housing schemes out of the league as the localities don't happen to be safe as most of the houses have had the highest crime rates.

PROPORTION OF THE HOUSES NEAR/AWAY TO THE CHARLES RIVER



Here, the proportion of the houses near and away to the Charles River is proposed. 1 depicts the house being near to the river, while 0 depicts the house being away from the river. Only 8.7% of houses are near to the river, making them an ideal place for the residents. One more time when we see how the housing schemes are not even throughout Boston.

PUPIL TEACHING RATION AGAINST CHARLES RIVER [CHAS]



This figure shows how the pupil-teacher ratio is moving in regards with the Charles River dummy variable: e.g. the house being near to the river or not. The houses not being near to the river have a higher pupil teacher ratio e.g. 88,000. The reason being that, out of 100%, 91.3% are the houses away from the river (as shows in the previous visualization). Hence, the localities having a house near the river would find the place ideal but would not be able to reside for a long term due to the low pupil teacher ratio. Hence, housing schemes should be made and built keeping in regards with all these factors.

Visualisation Links:

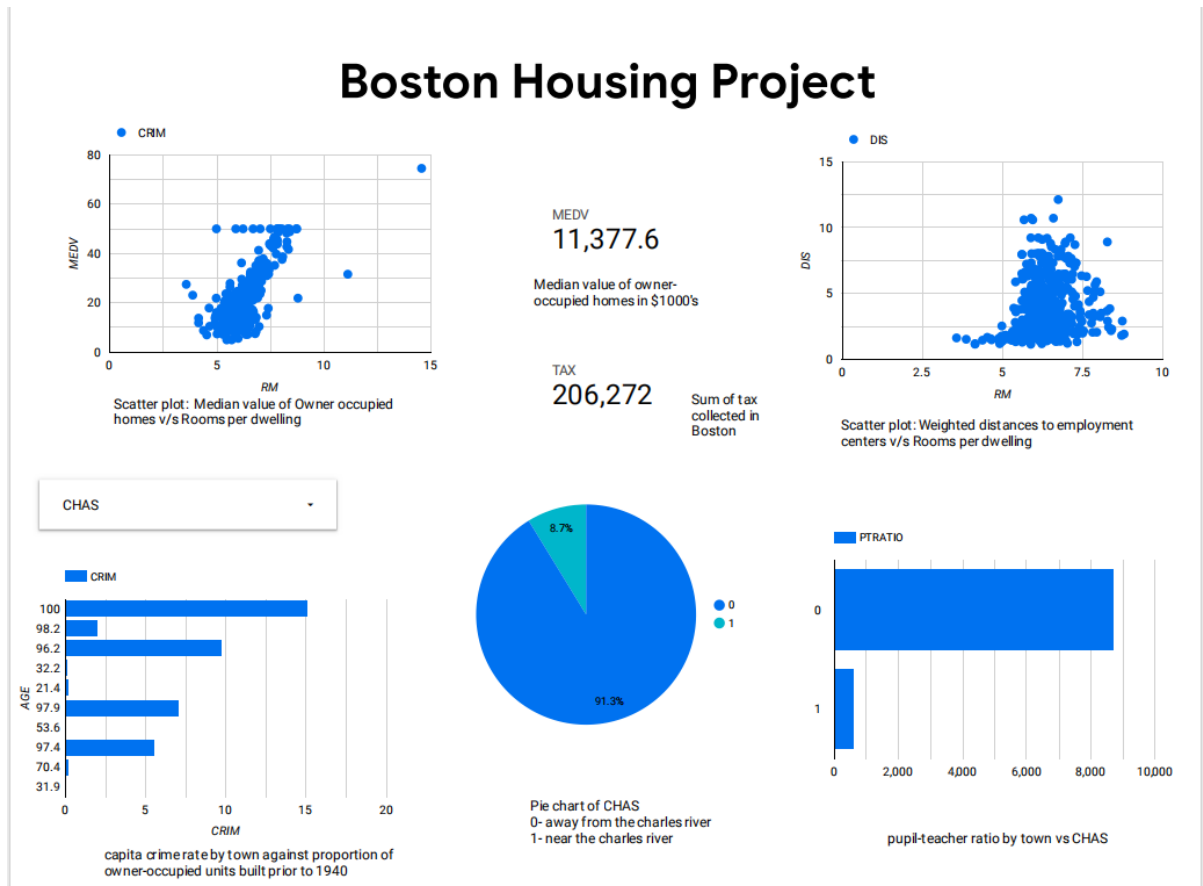
<https://lookerstudio.google.com/u/0/reporting/d2dac397-3588-4edb-b409-4f3195eae49c/page/tEnnC/edit>

<https://lookerstudio.google.com/u/0/reporting/ffc9a61c-c279-4e9a-92d0-a985aedd08ae/page/tEnnC/edit>

<https://lookerstudio.google.com/u/0/reporting/6c3f9e97-b3b3-41d1-8f6e-f2cfae8b43a4/page/tEnnC/edit>

<https://lookerstudio.google.com/u/0/reporting/5ed96117-f19a-4f5d-8a6b-a8e0997e8c19/page/tEnnC/edit>

FINAL DASHBOARD



Dashboard links-

<https://lookerstudio.google.com/reporting/5ed96117-f19a-4f5d-8a6b-a8e0997e8c19>

NOTE : For viewing the visualization on the looker studio, we need to provide permission access. Therefore, please contact us over teams by sharing your email ID in order to view the dashboard and the other visualizations.

CONCLUSION

The project's successful completion establishes a robust data engineering pipeline for Boston housing data. Utilizing technologies such as Apache Airflow, Terraform, Google Cloud Platform, Docker, BigQuery, and Looker Studio, the project encompasses data ingestion, processing, validation, and visualization. Notable achievements include precise environment setup, workflow configuration in Airflow, seamless integration with GCP, data quality checks with Great Expectations, and insightful visualizations in Looker Studio. The orchestrated flow covers dataset creation, data uploading to GCS, transfer to BigQuery, and thorough validations. This comprehensive solution streamlines complex data processes, ensures data quality, and provides a user-friendly interface for exploring and understanding Boston housing data, making it an asset for data engineering and analysis in real estate and housing markets.