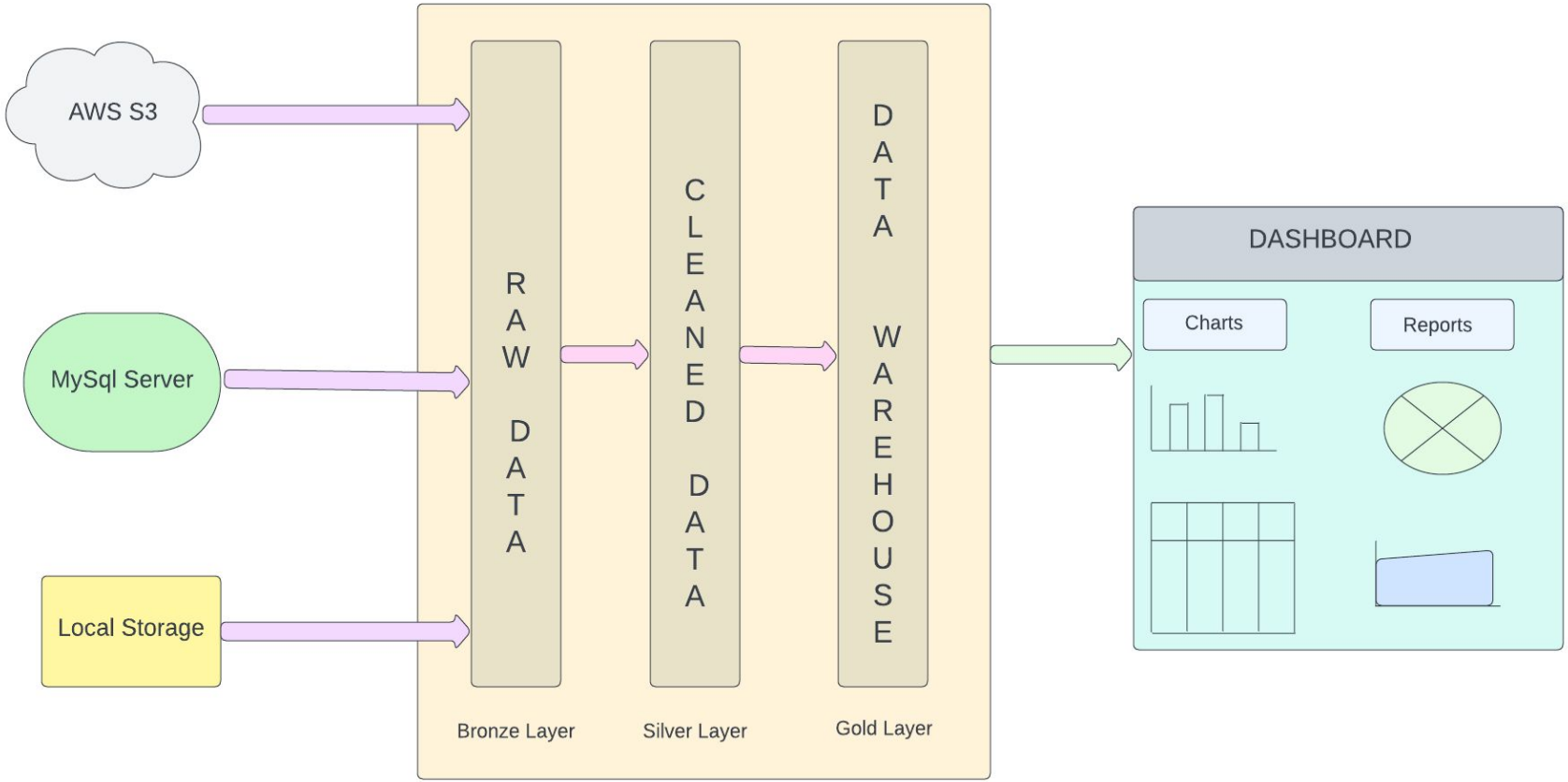
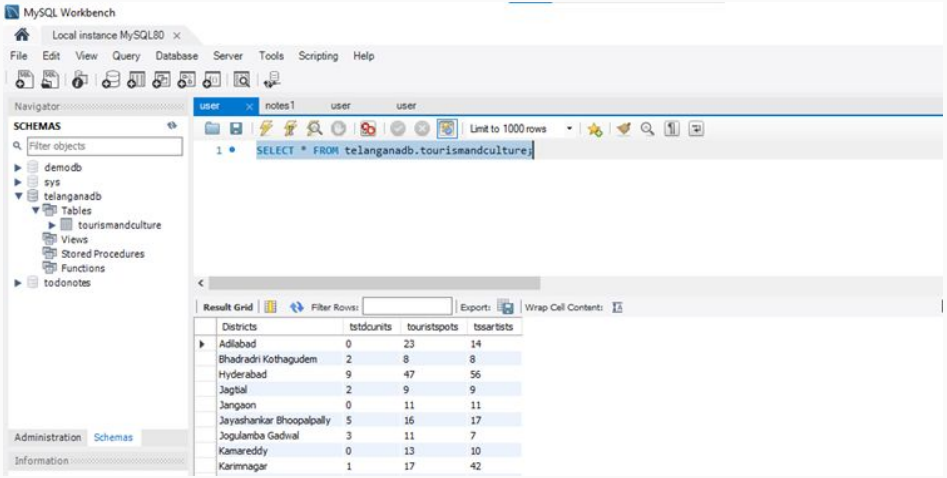
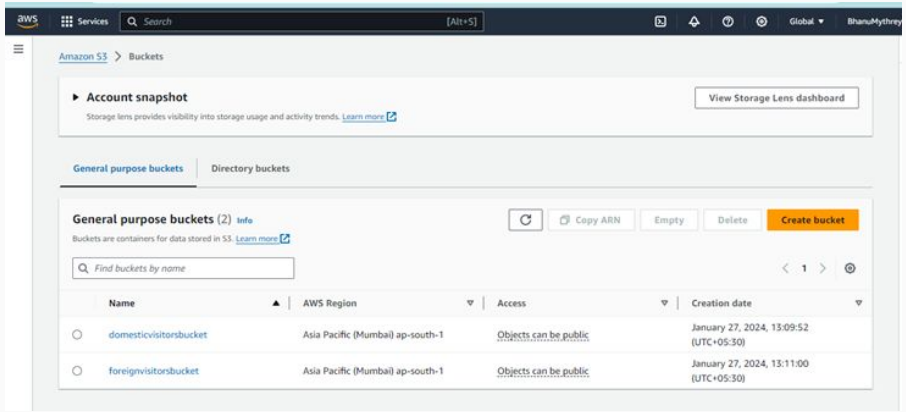
Telangana Tourism Insights Analysis using Data Engineering System

Team 12


AWS S3
MySql Server
Local Storage
RAW DATA
CLEANED DATA
DATA WAREHOUSE
Bronze Layer
Silver Layer
Gold Layer
DASHBOARD
Charts
Reports

Data Storage





Pipeline setup

```
C:\Users\Bhanu Mythreyi\OneDrive\Desktop\Data Engineering Project>docker run -it --rm -p 8888:8080 python:3.8-slim /bin/bash
Unable to find image 'python:3.8-slim' locally
3.8-slim: Pulling from library/python
a803e7c4b030: Already exists
bf3336e84c8e: Pull complete
8d1eed3b8c05: Pull complete
777fa63a9425: Pull complete
6fea7da35dae: Pull complete
Digest: sha256:704ebdc57baf207b20cf93797817ec9c9038aed1e43fe41e99eab2ad37430d62
```

```
C:\Users\Bhanu Mythreyi\OneDrive\Desktop\Data Engineering Project>docker ps
CONTAINER ID   IMAGE            COMMAND       CREATED         STATUS         PORTS                    NAMES
436ee2a9a9e3   python:3.8-slim  "/bin/bash"   4 minutes ago   Up 4 minutes   0.0.0.0:8888->8080/tcp   pensive_boyd
```

```
root@436ee2a9a9e3:/# export AIRFLOW_HOME=/opt/airflow
```

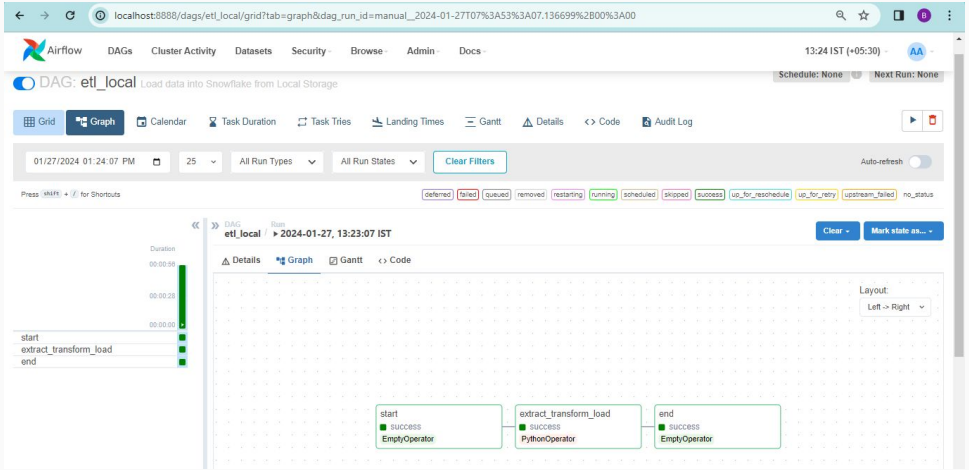
```
(.airflowvirtualenv) airflow@436ee2a9a9e3:~$ pip install "apache-airflow[crypto,celery,postgres,cncf.kubernetes,docker]"==2.5.1 --constraint ./constraints-3.8.txt
```

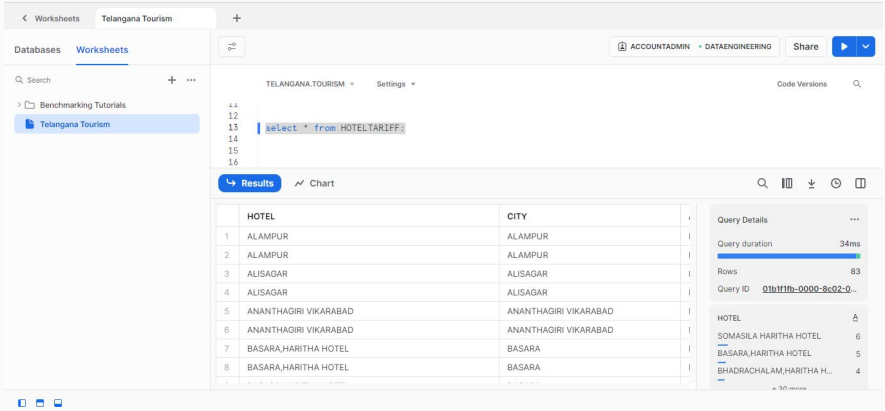
```
(.airflowvirtualenv) airflow@436ee2a9a9e3:~$ airflow db init
```

```
(.airflowvirtualenv) airflow@436ee2a9a9e3:~$ airflow scheduler &
[1] 2923
(.airflowvirtualenv) airflow@436ee2a9a9e3:~$ /opt/airflow/.airflowvirtualenv/lib/python3.8/site-packages/airflow/models/base.py:49 MovedIn20Warning: [31mDeprecated API
features detected! These feature(s) are not compatible with SQLAlchemy 2.0. [32mTo prevent incompatible upgrades prior to updating applications, ensure requirements fil
es are pinned to "sqlalchemy<2.0". [36mSet environment variable SQLALCHEMY_WARN_20=1 to show all deprecation warnings.  Set environment variable SQLALCHEMY_SILENCE_UBER
_WARNING=1 to silence this message.[0m (Background on SQLAlchemy 2.0 at: https://sqlalche.me/e/b8d9)
  _____ |__( )_________  _____/___/_______  _____
  ____ /| |_  /__  ___/_  __ /_  __ /_  __ \_ | /| / /
  ___ ___ |  / _  /   _  _/ _  _/ / / /_/ /_ |/ |/ /
  _/_/  |_/_/  /_/    /_/   /_/   \____/____/|__/
[2023-10-08 05:03:42 +0000] [2925] [INFO] Starting gunicorn 20.1.0
[2023-10-08 05:03:42 +0000] [2925] [INFO] Listening at: http://[::]:8793 (2925)
[2023-10-08 05:03:42 +0000] [2925] [INFO] Using worker: sync
[2023-10-08 05:03:42 +0000] [2926] [INFO] Booting worker with pid: 2926
[2023-10-08 05:03:42,147] {scheduler_job.py:714} INFO - Starting the scheduler
[2023-10-08 05:03:42,147] {scheduler_job.py:719} INFO - Processing each file at most -1 times
[2023-10-08 05:03:42,152] {executor_loader.py:107} INFO - Loaded executor: SequentialExecutor
[2023-10-08 05:03:42,157] {manager.py:163} INFO - Launched DagFileProcessorManager with pid: 2927
[2023-10-08 05:03:42,160] {scheduler_job.py:1408} INFO - Resetting orphaned tasks for active dag runs
[2023-10-08 05:03:42,170] {settings.py:58} INFO - Configured default timezone Timezone('UTC')
[2023-10-08 05:03:42 +0000] [2928] [INFO] Booting worker with pid: 2928
[2023-10-08T05:03:42.224+0000] {manager.py:409} WARNING - Because we cannot use more than 1 thread (parsing_processes = 2) when using sqlite. So we set parallelism to 1
```

```
(.airflowvirtualenv) airflow@436ee2a9a9e3:~$ airflow webserver &
[2] 2979
(.airflowvirtualenv) airflow@436ee2a9a9e3:~$ /opt/airflow/.airflowvirtualenv/lib/python3.8/site-packages/airflow/models/base.py:49 MovedIn20Warning: [31mDeprecated API
Features detected! These feature(s) are not compatible with SQLAlchemy 2.0. [32mTo prevent incompatible upgrades prior to updating applications, ensure requirements fil
es are pinned to "sqlalchemy<2.0". [36mSet environment variable SQLALCHEMY_WARN_20=1 to show all deprecation warnings.  Set environment variable SQLALCHEMY_SILENCE_UBER
_WARNING=1 to silence this message.[0m (Background on SQLAlchemy 2.0 at: https://sqlalche.me/e/b8d9)
  _____ |__( )_________  _____/___/_______  _____
  ____ /| |_  /__  ___/_  __ /_  __ /_  __ \_ | /| / /
  ___ ___ |  / _  /   _  _/ _  _/ / / /_/ /_ |/ |/ /
  _/_/  |_/_/  /_/    /_/   /_/   \____/____/|__/
Running the Gunicorn Server with:
Workers: 4 sync
Host: 0.0.0.0:8080
Timeout: 120
Logfiles: - -
Access Logformat:
=================================================================
[2023-10-08 05:04:56,478] {manager.py:243} INFO - Inserted Role: Admin
[2023-10-08 05:04:56,487] {manager.py:243} INFO - Inserted Role: Public
[2023-10-08 05:04:56,491] {manager.py:824} WARNING - No user yet created, use flask fab command to do it.
[2023-10-08 05:04:56,574] {manager.py:504} INFO - Created Permission View: can edit on Passwords
[2023-10-08 05:04:56,582] {manager.py:563} INFO - Added Permission can edit on Passwords to role Admin
```

ETL pipeline for local data source





Bronze Layer

```python
def extract_data_from_csv():
    local_csv_file = "data/hoteltariff.csv"
    with open(local_csv_file, 'r') as file:
        data = pd.read_csv(file)
    return data
```

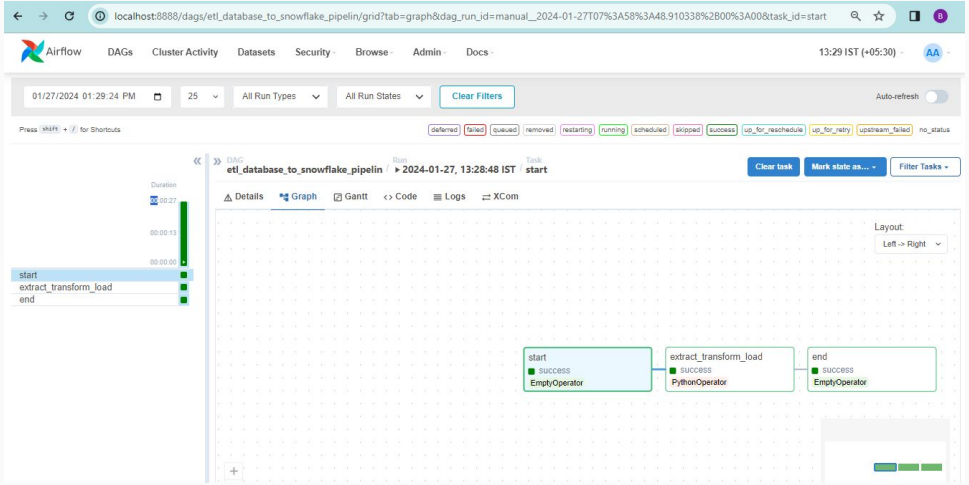
Silver Layer

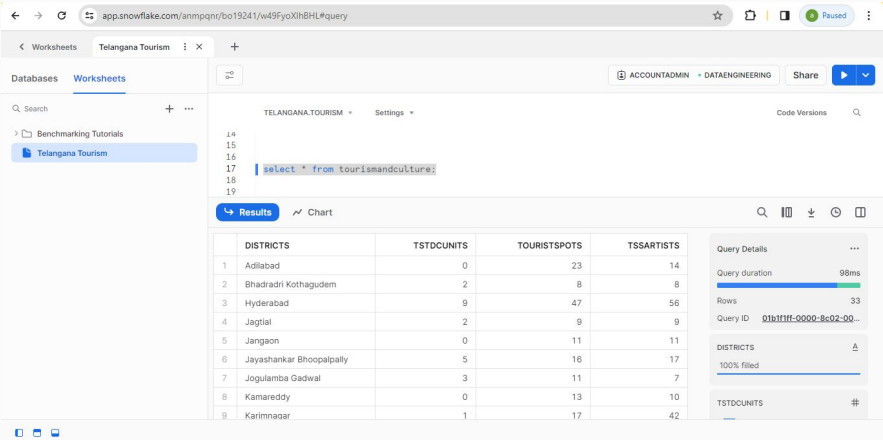
```python
def transform_data(extracted):
    data_frame = extracted.copy()

    tariff_columns = ['SundayTariff', 'MondayTariff', 'TuesdayTariff', 'WednesdayTariff', 'ThursdayTariff', 'FridayTariff', 'SaturdayTariff']
    weekend = ['FridayTariff', 'SaturdayTariff', 'SundayTariff']
    data_frame = data_frame.fillna(0)
    for index, row in data_frame.iterrows():
        row_values = row[tariff_columns]

        zero_count = 0
        non_zero_values = []
        for value in row_values:
            if value == 0:
                zero_count += 1
            elif value != 0:
                non_zero_values.append(value)

        modification_data_frame = data_frame.copy()
        if zero_count < len(row_values):
            for col in tariff_columns:
                if row[col] == 0:
                    if col in weekend:
                        max_value = max(non_zero_values)
                        modification_data_frame.at[index, col] = max_value
                    else:
                        most_repeated_value = find_most_common(non_zero_values)
                        modification_data_frame.at[index, col] = most_repeated_value

    return modification_data_frame
```

```python
def load_data_to_snowflake(transformed):
    snowflake_params = {
        "account": "                        !",
        "user": "               t",
        "password": "Password6",
        "warehouse": "dataengineering",
        "database": "telangana",
        "schema": "tourism",
    }


    conn = connect(**snowflake_params)
    cur = conn.cursor()

    try:
        transformed_data = transformed.copy()
        snowflake_table = 'HotelTariff'
        columns = transformed_data.columns.tolist()

        drop_table_statement = f"DROP TABLE IF EXISTS {snowflake_table};"
        cur.execute(drop_table_statement)

        create_table_statement = f"CREATE TABLE {snowflake_table} ("
        for col in columns:
            if col in ['Hotel', 'City', 'District', 'Roomtype']:
                create_table_statement += f"{col} VARCHAR(255),"
            else:
                create_table_statement += f"{col} INT,"

        create_table_statement = create_table_statement.rstrip(',')
        create_table_statement += ");"

        cur.execute(create_table_statement)
```

```python
    for index, row in transformed_data.iterrows():
        values = []
        for value in row:
            if isinstance(value, str):
                values.append(f"'{value}'")
            else:
                values.append(str(value))

        column_names = ', '.join(columns)
        row_values = ', '.join(values)

        sql_statement = f"INSERT INTO {snowflake_table} ({column_names}) VALUES ({row_values})"

        print(f"SQL statement: {sql_statement}")
        cur.execute(sql_statement)

    conn.commit()

    print("DATA LOADING SUCCESSFUL!!")
except Exception as e:
    print(f"Error: {str(e)}")
    conn.rollback()

finally:
    cur.close()
    conn.close()
```

ETL pipeline for database source





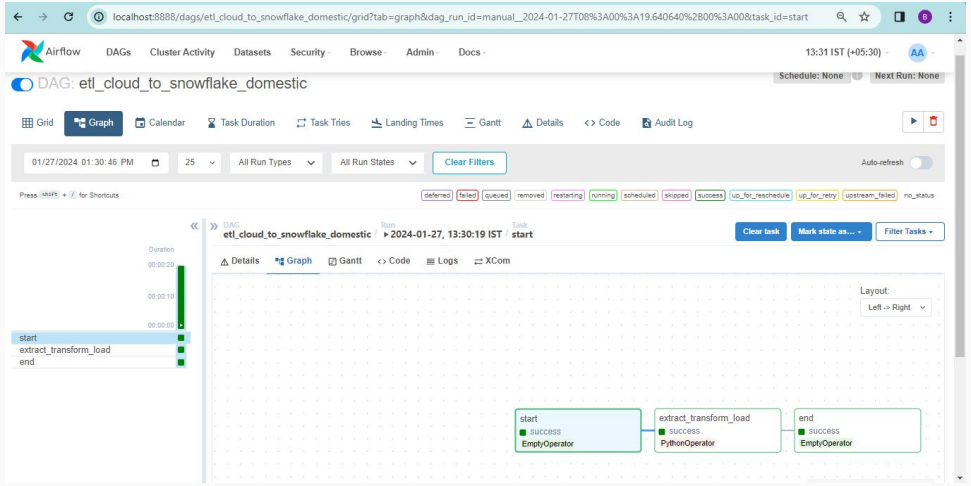
```python
def extract_data_from_csv():
    mysql_params = {
        'host': '            l',
        'user': 'myuser',
        'password': 'mypassword',
        'database': 'telanganadb',
    }

    conn = mysql.connector.connect(**mysql_params)
    cursor = conn.cursor()

    query = 'SELECT * FROM tourismandculture'

    try:
        cursor.execute(query)

        data = cursor.fetchall()

        columns = [desc[0] for desc in cursor.description]
        extracted_data = pd.DataFrame(data, columns=columns)

        return extracted_data

    finally:
        cursor.close()
        conn.close()
```

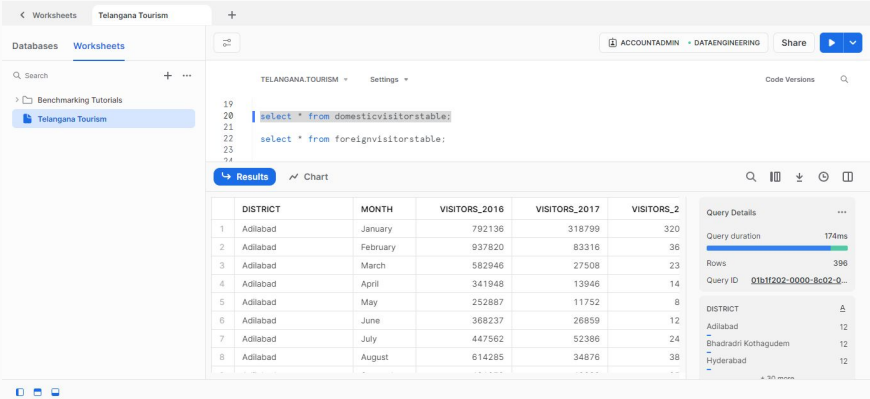
```python
def transform_data(extracted):
    data = extracted.copy()
    my_data = data.fillna(0)
    return my_data
```

```python
def load_data_to_snowflake(transformed):
    snowflake_params = {
        "account": "                    2",
        "user": "                ",
        "password": "Password6",
        "warehouse": "dataengineering",
        "database": "telangana",
        "schema": "tourism",
    }


    conn = connect(**snowflake_params)
    cur = conn.cursor()

    try:
        transformed_data = transformed.copy()


        snowflake_table = 'tourismandculture'
        columns = transformed_data.columns.tolist()

        drop_table_statement = f"DROP TABLE IF EXISTS {snowflake_table};"
        cur.execute(drop_table_statement)

        create_table_statement = f"CREATE TABLE {snowflake_table} ("
        for col in columns:
            if col in ['District']:
                create_table_statement += f"{col} VARCHAR(255),"
            else:
                create_table_statement += f"{col} INT,"

        create_table_statement = create_table_statement.rstrip(',')
        create_table_statement += ");"

        cur.execute(create_table_statement)
```

```python
    for index, row in transformed_data.iterrows():
        values = []
        for value in row:
            if isinstance(value, str):
                values.append(f"'{value}'")
            else:
                values.append(str(value))

        column_names = ', '.join(columns)
        row_values = ', '.join(values)

        sql_statement = f"INSERT INTO {snowflake_table} ({column_names}) VALUES ({row_values})"

        cur.execute(sql_statement)

    conn.commit()

    print("DATA LOADING SUCCESSFUL!!")
except Exception as e:
    print(f"Error: {str(e)}")
    conn.rollback()

finally:
    cur.close()
    conn.close()
```

ETL pipeline for cloud data sources





Bronze Layer

```python
def extract_data_from_s3(bucket_name, aws_conn_id):
    try:
        s3_hook = S3Hook(aws_conn_id)
        s3_objects = s3_hook.list_keys(bucket_name=bucket_name)

        if not s3_objects:
            raise Exception(f"No objects found in the S3 bucket '{bucket_name}'.")

        latest_object = max(s3_objects)
        file_content = s3_hook.read_key(latest_object, bucket_name)

        data_frame = pd.read_csv(io.StringIO(file_content))
        print("EXTRACTION DONE")
        return data_frame

    except Exception as e:
        print(f"Error in extract_data_from_s3: {str(e)}")
        raise
```

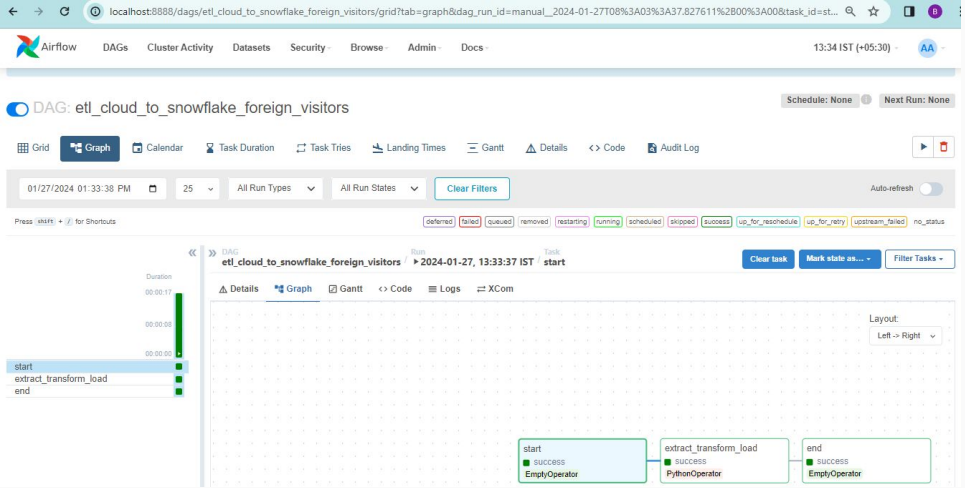
Silver Layer

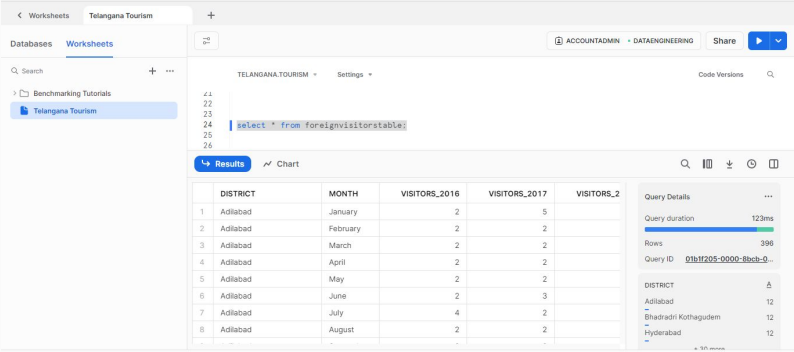
```python
def transform_data(extracted):
    try:
        data_frame = extracted.copy()
        visitor_columns = ['Visitors_2016', 'Visitors_2017', 'Visitors_2018', 'Visitors_2019', 'Visitors_2020']

        for col in visitor_columns:
            new_col_values = []
            for value in data_frame[col]:
                if pd.isna(value) or not value.strip().isdigit():
                    new_col_values.append(0)
                else:
                    new_col_values.append(int(value))

            data_frame[col] = new_col_values

        district_sums = {}
        district_counts = {}

        for index, row in data_frame.iterrows():
            district = row['District']
            for col in visitor_columns:
                if row[col] != 0:
                    key = (district, col)

                    if key not in district_sums:
                        district_sums[key] = 0
                        district_counts[key] = 0

                    district_sums[key] += row[col]
                    district_counts[key] += 1
```

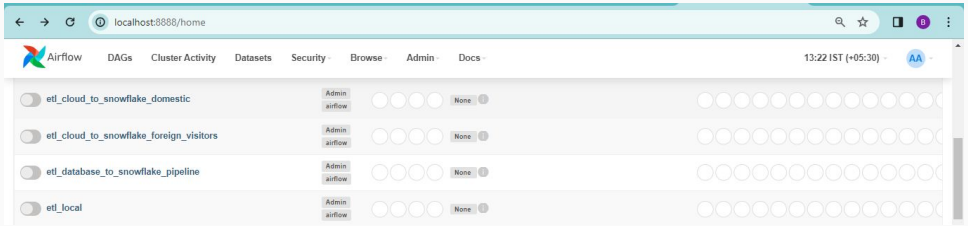
```python
    district_means = {}
    for (district, col), value_sum in district_sums.items():
        count = district_counts[(district, col)]
        if district not in district_means:
            district_means[district] = {}

        if count > 0:
            district_means[district][col] = value_sum // count
        else:
            district_means[district][col] = 0

    modified_data_frame = data_frame.copy()
    for index, row in data_frame.iterrows():
        district = row['District']
        for col in visitor_columns:
            if row[col] == 0:
                if district in district_means and col in district_means[district]:
                    modified_data_frame.at[index, col] = district_means[district][col]
                else:
                    modified_data_frame.at[index, col] = 0

    print("TRANSFORMATION DONE")
    return modified_data_frame

except Exception as e:
    print(f"Error in transform_data: {str(e)}")
    raise
```

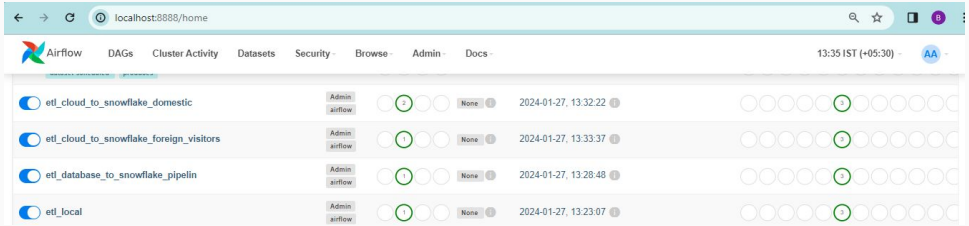
```python
def load_data_to_snowflake(transformed):
    snowflake_params = {
        "account": "                        ",
        "user": "                ",
        "password": "Password6",
        "warehouse": "dataengineering",
        "database": "telangana",
        "schema": "tourism",
    }

    conn = connect(**snowflake_params)
    cur = conn.cursor()

    try:
        snowflake_table = 'domesticvisitorstable'
        columns = transformed.columns.tolist()

        drop_table_statement = f"DROP TABLE IF EXISTS {snowflake_table};"
        cur.execute(drop_table_statement)

        create_table_statement = f"CREATE TABLE {snowflake_table} ("
        for col in columns:
            if col in ['District', 'Month']:
                create_table_statement += f"{col} VARCHAR(255),"
            else:
                create_table_statement += f"{col} INT,"

        create_table_statement = create_table_statement.rstrip(',')
        create_table_statement += ");"


        cur.execute(create_table_statement)
```

```python
    for index, row in transformed.iterrows():
        values = []
        for value in row:
            if isinstance(value, str):
                values.append(f"'{value}'")
            else:
                values.append(str(value))
        column_names = ', '.join(columns)
        row_values = ', '.join(values)

        sql_statement = f"INSERT INTO {snowflake_table} ({column_names}) VALUES ({row_values})"

        print(f"SQL statement: {sql_statement}")
        cur.execute(sql_statement)

    conn.commit()
    print("LOADING DONE")
except Exception as e:
    print(f"Error in load_data_to_snowflake: {str(e)}")
    conn.rollback()
finally:
    cur.close()
    conn.close()
```


localhost:8888/dags/etl_cloud_to_snowflake_foreign_visitors/grid?tab=graph&dag_run_id=manual__2024-01-27T08%3A03%3A37.827611%2B00%3A00&task_id=st...
Airflow
DAGs
Cluster Activity
Datasets
Security
Browse
Admin
Docs
13:34 IST (+05:30)
DAG: etl_cloud_to_snowflake_foreign_visitors
Schedule: None
Next Run: None
Grid
Graph
Calendar
Task Duration
Task Tries
Landing Times
Gantt
Details
Code
Audit Log
01/27/2024 01:33:38 PM
25
All Run Types
All Run States
Clear Filters
Auto-refresh
Press shift + / for Shortcuts
deferred
failed
queued
removed
restarting
running
scheduled
skipped
success
up_for_reschedule
up_for_retry
upstream_failed
no_status
DAG
etl_cloud_to_snowflake_foreign_visitors
Run
2024-01-27, 13:33:37 IST
Task
start
Clear task
Mark state as...
Filter Tasks
Duration
00:00:17
00:00:08
00:00:00
start
extract_transform_load
end
Details
Graph
Gantt
Code
Logs
XCom
Layout:
Left -> Right
start
success
EmptyOperator
extract_transform_load
success
PythonOperator
end
success
EmptyOperator


Worksheets
Telangana Tourism
Databases
Worksheets
Search
Benchmarking Tutorials
Telangana Tourism
ACCOUNTADMIN
DATAENGINEERING
Share
TELANGANA.TOURISM
Settings
Code Versions
select * from foreignvisitorstable;
Results
Chart
DISTRICT
MONTH
VISITORS_2016
VISITORS_2017
VISITORS_2
1 Adilabad January 2 5
2 Adilabad February 2 2
3 Adilabad March 2 2
4 Adilabad April 2 2
5 Adilabad May 2 2
6 Adilabad June 2 3
7 Adilabad July 4 2
8 Adilabad August 2 2
Query Details
Query duration 123ms
Rows 396
Query ID 01b1f205-0000-8bcb-0...
DISTRICT
Adilabad 12
Bhadradri Kothagudem 12
Hyderabad 12

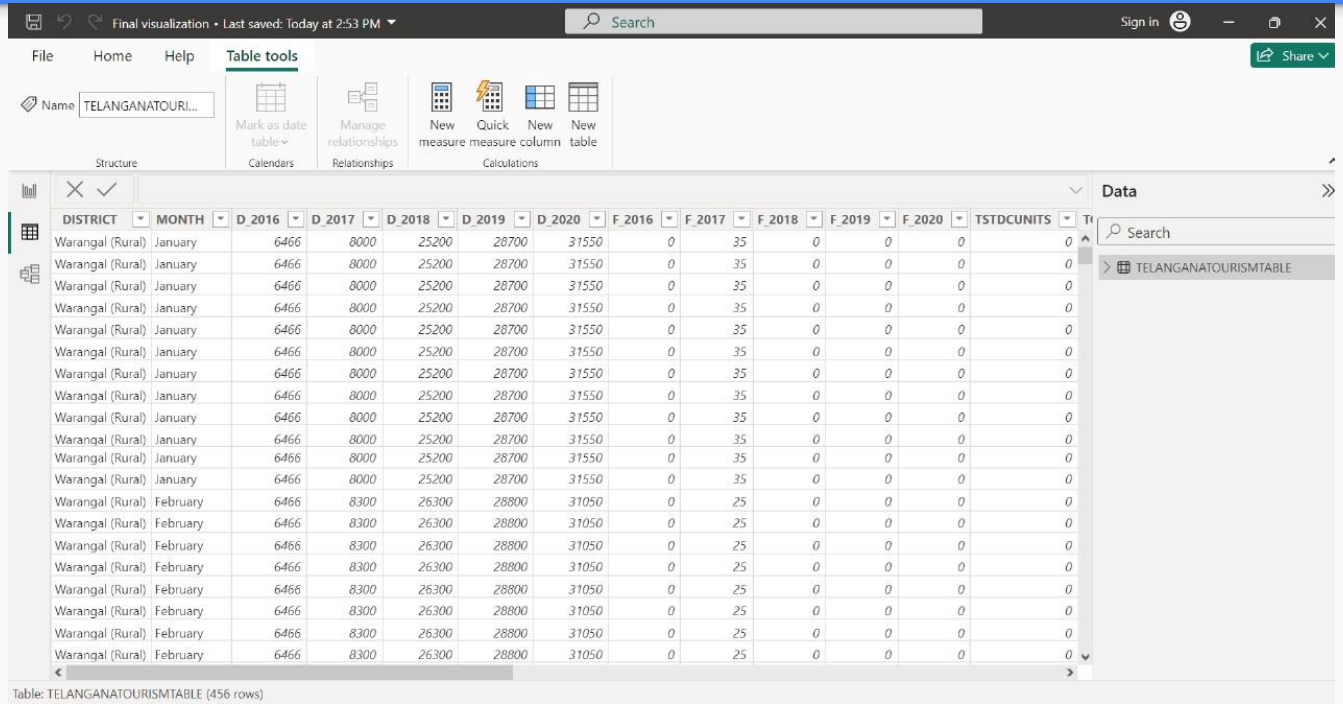

localhost:8888/home
Airflow
DAGs
Cluster Activity
Datasets
Security
Browse
Admin
Docs
13:22 IST (+05:30)
AA
etl_cloud_to_snowflake_domestic
etl_cloud_to_snowflake_foreign_visitors
etl_database_to_snowflake_pipeline
etl_local
Admin
airflow
None


localhost:8888/home
Airflow
DAGs
Cluster Activity
Datasets
Security
Browse
Admin
Docs
13:35 IST (+05:30)
AA
etl_cloud_to_snowflake_domestic
2024-01-27, 13:32:22
etl_cloud_to_snowflake_foreign_visitors
2024-01-27, 13:33:37
etl_database_to_snowflake_pipelin
2024-01-27, 13:28:48
etl_local
2024-01-27, 13:23:07
Admin
airflow
None

Gold layer data

```
create table telanganatourismtable as select d.district as district,d.month as month, d.visitors_2016 as
d_2016,d.visitors_2017 as d_2017,d.visitors_2018 as d_2018,d.visitors_2019 as d_2019,d.visitors_2020 as
d_2020,f.visitors_2016 as f_2016,f.visitors_2017 as f_2017,f.visitors_2018 as f_2018,f.visitors_2019 as
f_2019,f.visitors_2020 as f_2020,tc.tstdcunits as tstdcunits,tc.touristspots as touristspots,
tc.tssartists as tssartists,ht.totalrooms as totalrooms,ht.sundaytariff as sundaytariff,ht.mondaytariff
as mondaytariff,ht.tuesdaytariff as tuesdaytariff,ht.wednesdaytariff as
wednesdaytariff,ht.thursdaytariff as thursdaytariff,ht.fridaytariff as fridaytariff,ht.saturdaytariff as
saturdaytariff  from domesticvisitorstable d, foreignvisitorstable f,tourismandculture as tc,HOTELTARIFF
ht where d.district = f.district and d.district = ht.district and d.district=tc.district and
d.month=f.month;
```

Power bi



Name TELANGANATOURI...

DISTRICT	MONTH	D_2016	D_2017	D_2018	D_2019	D_2020	F_2016	F_2017	F_2018	F_2019	F_2020	TSTDCUNITS
Warangal (Rural)	January	6466	8000	25200	28700	31550	0	35	0	0	0	0
Warangal (Rural)	January	6466	8000	25200	28700	31550	0	35	0	0	0	0
Warangal (Rural)	January	6466	8000	25200	28700	31550	0	35	0	0	0	0
Warangal (Rural)	January	6466	8000	25200	28700	31550	0	35	0	0	0	0
Warangal (Rural)	January	6466	8000	25200	28700	31550	0	35	0	0	0	0
Warangal (Rural)	January	6466	8000	25200	28700	31550	0	35	0	0	0	0
Warangal (Rural)	January	6466	8000	25200	28700	31550	0	35	0	0	0	0
Warangal (Rural)	January	6466	8000	25200	28700	31550	0	35	0	0	0	0
Warangal (Rural)	January	6466	8000	25200	28700	31550	0	35	0	0	0	0
Warangal (Rural)	January	6466	8000	25200	28700	31550	0	35	0	0	0	0
Warangal (Rural)	January	6466	8000	25200	28700	31550	0	35	0	0	0	0
Warangal (Rural)	January	6466	8000	25200	28700	31550	0	35	0	0	0	0
Warangal (Rural)	February	6466	8300	26300	28800	31050	0	25	0	0	0	0
Warangal (Rural)	February	6466	8300	26300	28800	31050	0	25	0	0	0	0
Warangal (Rural)	February	6466	8300	26300	28800	31050	0	25	0	0	0	0
Warangal (Rural)	February	6466	8300	26300	28800	31050	0	25	0	0	0	0
Warangal (Rural)	February	6466	8300	26300	28800	31050	0	25	0	0	0	0
Warangal (Rural)	February	6466	8300	26300	28800	31050	0	25	0	0	0	0
Warangal (Rural)	February	6466	8300	26300	28800	31050	0	25	0	0	0	0
Warangal (Rural)	February	6466	8300	26300	28800	31050	0	25	0	0	0	0

Table: TELANGANATOURISMTABLE (456 rows)

Final visualization • Last saved: Today at 11:19 PM

Search

Sign in

File Home Insert Modeling View Optimize Help

Share

Cut
Copy
Paste
Format painter
Clipboard

Get data
Excel workbook
OneLake data hub
SQL Server
Enter data
Dataverse
Recent sources
Data

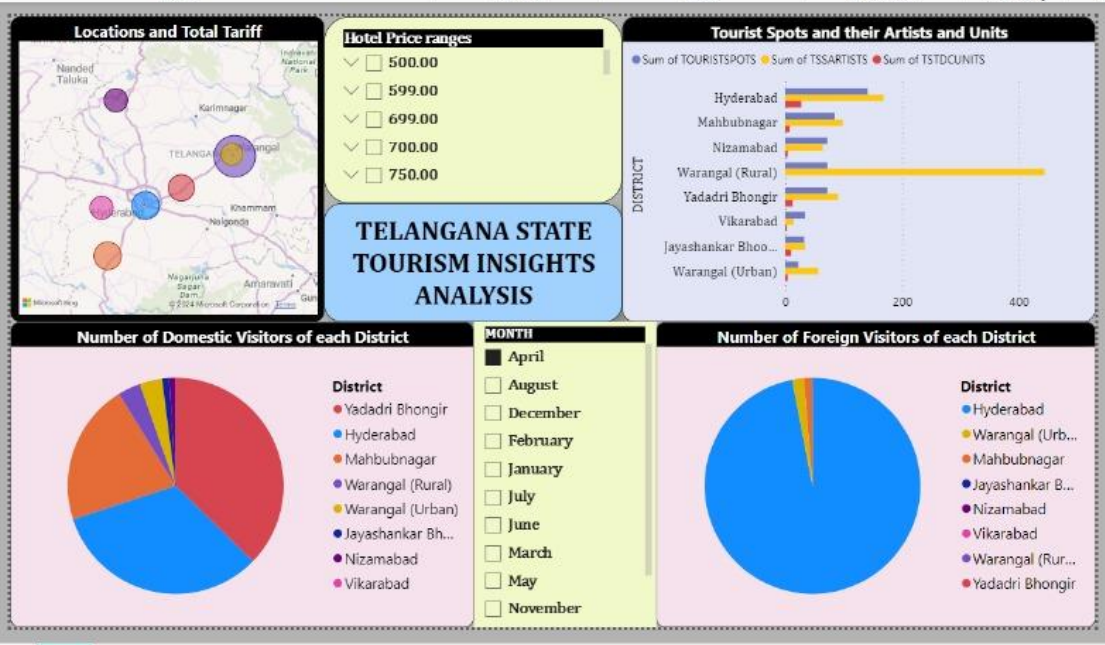
Transform data
Refresh
Queries

New visual
Text box
More visuals
Insert

New measure
Quick measure
Calculations

Sensitivity
Sensitivity

Publish
Share



Filters
Visualizations
Data

Page 1

Page 1 of 1

60%

Search

26°

ENG IN

11:28 PM
21-02-2024

Testing

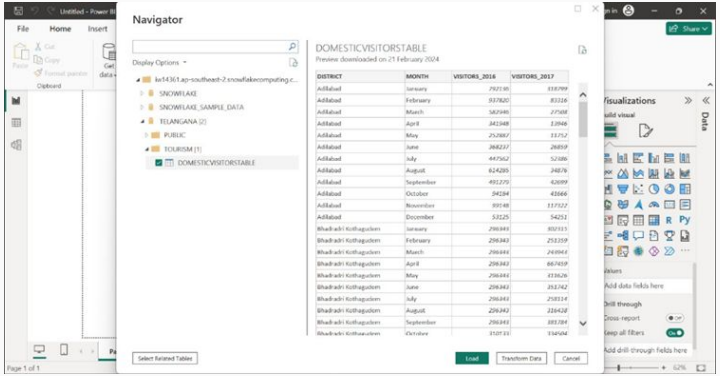
White box testing:

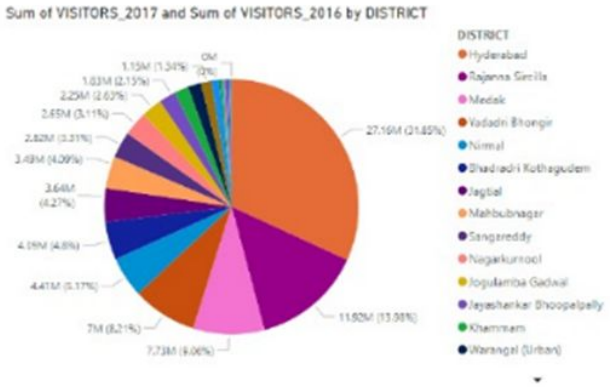
Test Case 1: Domestic visitors data: from year 2016 to 2017

```
14  select * from domesticvisitorstable;
```

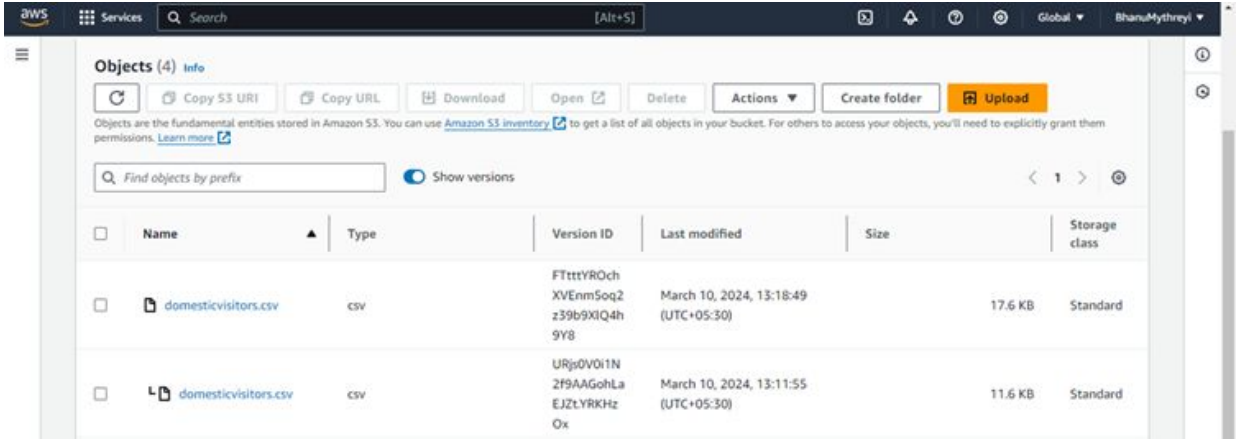
Results Chart

	DISTRICT	MONTH	VISITORS_2016	VISITORS_2017
1	Adilabad	January	792136	318799
2	Adilabad	February	937820	83316
3	Adilabad	March	582946	27508
4	Adilabad	April	341948	13946
5	Adilabad	May	252887	11752
6	Adilabad	June	368237	26859
7	Adilabad	July	447562	52386
8	Adilabad	August	614285	34876
9	Adilabad	September	491279	42699





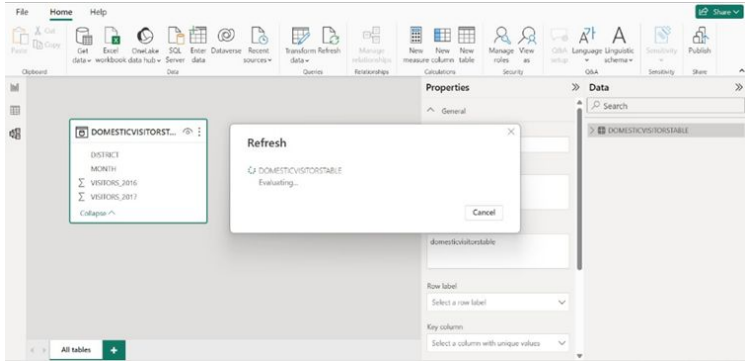
Test Case 2: Domestic visitors data till 2020

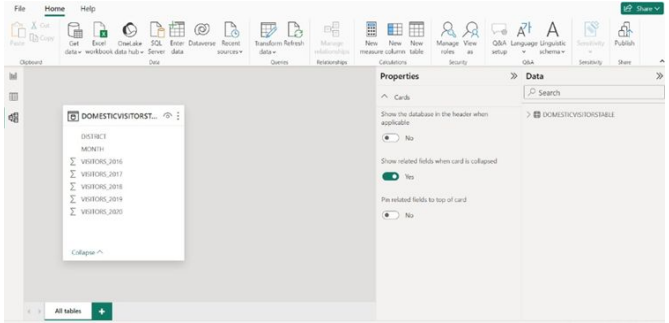


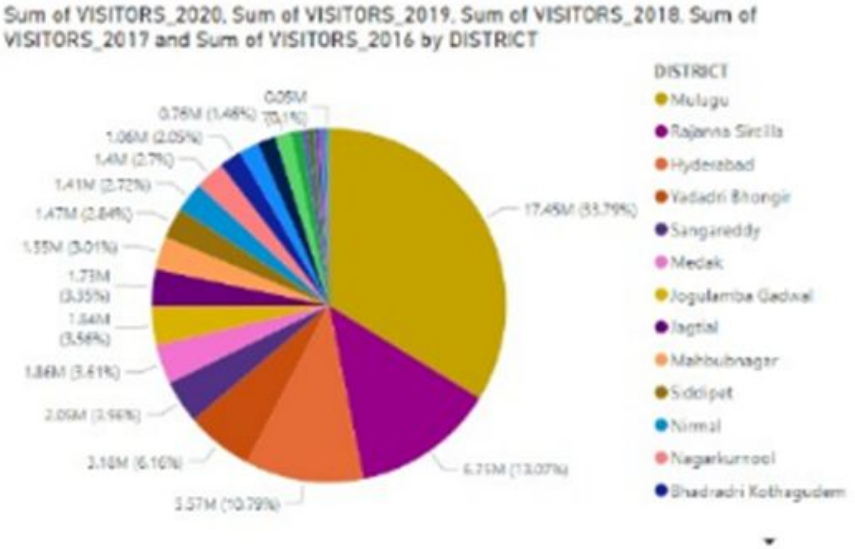
```
14 select * from domesticvisitorstable;
```

Results | Chart

	DISTRICT	MONTH	VISITORS_2016	VISITORS_2017	VISITORS_2018	VISITORS_2019
1	Adilabad	January	792136	318799	320356	25071
2	Adilabad	February	937820	83316	36550	406177
3	Adilabad	March	582946	27508	23011	14347
4	Adilabad	April	341948	13946	14183	9972
5	Adilabad	May	252887	11752	8197	6997
6	Adilabad	June	368237	26859	12052	9161
7	Adilabad	July	447562	52386	24666	11791
8	Adilabad	August	614285	34876	38939	108173







Black box testing:

Test Case 1:

Data consisting of 3 districts

Test Case 2:

Data of all districts

Thank You