ArupChakraborty_Assignment2.1

May 19, 2025

1 AWS Data Lake Query Notebook

This notebook sets up the environment, configures Athena, and executes various queries to analyze the ingested data from the AWS S3 Data Lake.

[2]:
```python
import boto3
import sagemaker
import pandas as pd

sess = sagemaker.Session()
bucket = sess.default_bucket()
role = sagemaker.get_execution_role()
region = boto3.Session().region_name
account_id = boto3.client("sts").get_caller_identity().get("Account")

sm = boto3.Session().client(service_name="sagemaker", region_name=region)
```

```
sagemaker.config INFO - Not applying SDK defaults from location:
/etc/xdg/sagemaker/config.yaml
sagemaker.config INFO - Not applying SDK defaults from location:
/home/sagemaker-user/.config/sagemaker/config.yaml
```

[3]:
```python
s3_private_path_tsv = "s3://{}/amazon-reviews-pds/tsv".format(bucket)
print(s3_private_path_tsv)
```

```
s3://sagemaker-us-east-1-672518276407/amazon-reviews-pds/tsv
```

[4]:
```python
!aws s3 cp "dataset.csv" $s3_private_path_tsv/
```

```
upload: ./dataset.csv to s3://sagemaker-us-east-1-672518276407/amazon-reviews-
pds/tsv/dataset.csv
```

[5]:
```python
print(s3_private_path_tsv)
```

```
s3://sagemaker-us-east-1-672518276407/amazon-reviews-pds/tsv
```

[41]:
```python
import boto3
import pandas as pd
from io import StringIO
```

```python
# S3 bucket and file path
bucket_name = "sagemaker-us-east-1-672518276407"
file_key = "amazon-reviews-pds/tsv/dataset.csv"

# Create an S3 client
s3 = boto3.client("s3")

try:
    # Fetch the CSV file from S3
    response = s3.get_object(Bucket=bucket_name, Key=file_key)

    # Read the content of the file
    content = response["Body"].read().decode("utf-8")

    # Load the content into a DataFrame
    data = pd.read_csv(StringIO(content))

    # Display the first few rows
    print(data.head())
except Exception as e:
    print(f"Error occurred: {e}")
```

```
   Unnamed: 0                track_id                 artists  \
0           0  5SuOikwiRyPMVoIQDJUgSV              Gen Hoshino
1           1  4qPNDBW1i3p13qLCt0Ki3A             Ben Woodward
2           2  1iJBSr7s7jYXzM8EGcbK5b  Ingrid Michaelson;ZAYN
3           3  6lfxq3CG4xtTiEg7opyCyx             Kina Grannis
4           4  5vjLSffimiIP26QG5WcN2K        Chord Overstreet

                                          album_name  \
0                                             Comedy
1                                   Ghost (Acoustic)
2                                     To Begin Again
3  Crazy Rich Asians (Original Motion Picture Sou…
4                                            Hold On

                   track_name  popularity  duration_ms  explicit  \
0                      Comedy          73       230666     False
1            Ghost - Acoustic          55       149610     False
2              To Begin Again          57       210826     False
3  Can't Help Falling In Love          71       201933     False
4                     Hold On          82       198853     False

   danceability  energy  …  loudness  mode  speechiness  acousticness  \
0         0.676  0.4610  …    -6.746     0       0.1430        0.0322
1         0.420  0.1660  …   -17.235     1       0.0763        0.9240
```

```
2          0.438  0.3590  …    -9.734     1       0.0557        0.2100
3          0.266  0.0596  …   -18.515     1       0.0363        0.9050
4          0.618  0.4430  …    -9.681     1       0.0526        0.4690

   instrumentalness  liveness  valence    tempo  time_signature  track_genre
0          0.000001    0.3580    0.715   87.917               4     acoustic
1          0.000006    0.1010    0.267   77.489               4     acoustic
2          0.000000    0.1170    0.120   76.332               4     acoustic
3          0.000071    0.1320    0.143  181.740               3     acoustic
4          0.000000    0.0829    0.167  119.949               4     acoustic

[5 rows x 21 columns]
```

[43]:
```
# Filter songs with popularity greater than or equal to 99
df_high_popularity = data[data['popularity'] >= 99]
df_high_popularity = df_high_popularity[['artists', 'track_name', 'popularity']]
print("Songs with popularity  99:")
print(df_high_popularity)
```

```
Songs with popularity  99:
                   artists                            track_name  popularity
20001  Sam Smith;Kim Petras             Unholy (feat. Kim Petras)         100
51664      Bizarrap;Quevedo  Quevedo: Bzrp Music Sessions, Vol. 52          99
81051  Sam Smith;Kim Petras             Unholy (feat. Kim Petras)         100
```

[44]:
```
# Group by artist and calculate the average popularity
df_avg_popularity = data.groupby('artists')['popularity'].mean().reset_index()

# Filter artists with an average popularity of 92
df_avg_popularity_92 = df_avg_popularity[df_avg_popularity['popularity'] == 92]
print("Artists with an average popularity of 92:")
print(df_avg_popularity_92)
```

```
Artists with an average popularity of 92:
                 artists  popularity
11491       Harry Styles        92.0
22845  Rema;Selena Gomez        92.0
```

[46]:
```
# Group by genre and calculate the average energy
df_avg_energy = data.groupby('track_genre')['energy'].mean().reset_index()

# Get the top 10 genres by average energy
df_top10_genres = df_avg_energy.sort_values(by='energy', ascending=False).
  ↪head(10)
print("Top 10 genres with the highest average energy:")
print(df_top10_genres)
```

```
Top 10 genres with the highest average energy:
      track_genre    energy
22    death-metal  0.931470
42      grindcore  0.924201
72      metalcore  0.914485
46          happy  0.910971
49      hardstyle  0.901246
27  drum-and-bass  0.876635
6     black-metal  0.874897
50    heavy-metal  0.874003
78          party  0.871237
61         j-idol  0.868677
```

[47]:
```python
# Filter tracks by the artist 'Bad Bunny'
bad_bunny_count = data[data['artists'] == 'Bad Bunny'].shape[0]
print(f"Number of tracks featuring Bad Bunny: {bad_bunny_count}")
```

```
Number of tracks featuring Bad Bunny: 48
```

[48]:
```python
# Group by genre and find the maximum popularity within each genre
df_genre_popularity = data.groupby('track_genre')['popularity'].max().
 ↪reset_index()

# Sort by popularity and get the top 10 genres
df_top10_popular_genres = df_genre_popularity.sort_values(by='popularity',␣
 ↪ascending=False).head(10)
print("Top 10 genres by their most popular track:")
print(df_top10_popular_genres)
```

```
Top 10 genres by their most popular track:
   track_genre  popularity
80         pop         100
20       dance         100
51     hip-hop          99
68      latino          98
89   reggaeton          98
30         edm          98
67       latin          98
88      reggae          98
79       piano          96
90        rock          96
```

[39]:
```python
statement = "SHOW TABLES in {}".format('dsoaws')

df_show = pd.read_sql(statement, conn)
df_show.head(5)
```

```
/tmp/ipykernel_3285/2205842015.py:3: UserWarning: pandas only supports
SQLAlchemy connectable (engine/connection) or database string URI or sqlite3
```

```
DBAPI2 connection. Other DBAPI2 objects are not tested. Please consider using
SQLAlchemy.
  df_show = pd.read_sql(statement, conn)
```

[39]:
```
            tab_name
0  amazon_reviews_tsv
```

[34]:
```python
# SageMaker session setup
sess = sagemaker.Session()
bucket = sess.default_bucket()
role = sagemaker.get_execution_role()
region = boto3.Session().region_name

# Athena connection setup
database_name = "dsoaws"
table_name_tsv = "amazon_reviews_tsv"

conn = connect(region_name=region, s3_staging_dir=s3_staging_dir)
print("Setup completed.")
```

```
Setup completed.
```

[61]:
```python
# Athena connection setup
conn = connect(region_name=region, s3_staging_dir=s3_staging_dir)

# Improved execute_query function
def execute_query(query, description="Query Result"):
    """
    Executes a Pandas query or calculation and prints the result.

    Parameters:
        query (pd.DataFrame or int): The result of the query.
        description (str): A brief description of the query result.
    """
    try:
        print(f"\n{description}")
        print("=" * len(description))

        # Check if the query returned a DataFrame
        if isinstance(query, pd.DataFrame):
            if not query.empty:
                display(query.head(100))  # Display the first 100 rows
                print(f"Total Records: {len(query)}")
            else:
                print("No matching records found.")
        # Check if the query returned a single value (like a count)
        elif isinstance(query, int):
            print(f"Result: {query}")
        else:
            print("Unexpected query result type.")
    except Exception as e:
        print(f"Error executing query: {e}")
```

1.1 List artist, track_name, and popularity for songs with popularity 99

[62]:
```
import pandasql as psql

# SQL-like query on the DataFrame
query = """
SELECT artists, track_name, popularity
FROM data
WHERE popularity >= 99
"""
result = psql.sqldf(query, locals())
execute_query(result, "Songs with Popularity  99 (SQL-like)")
```

```
Songs with Popularity  99 (SQL-like)
=====================================

                artists                          track_name  popularity
0  Sam Smith;Kim Petras            Unholy (feat. Kim Petras)         100
1      Bizarrap;Quevedo  Quevedo: Bzrp Music Sessions, Vol. 52          99
2  Sam Smith;Kim Petras            Unholy (feat. Kim Petras)         100

Total Records: 3
```

1.2 List artists with an average popularity of 92

[65]:
```
query = """
SELECT artists, AVG(popularity) AS avg_popularity
FROM data
GROUP BY artists
HAVING AVG(popularity) = 92
"""
result = psql.sqldf(query, locals())
execute_query(result, "List artists with an average popularity of 92")
```

```
List artists with an average popularity of 92
=============================================

             artists  avg_popularity
0       Harry Styles            92.0
1  Rema;Selena Gomez            92.0

Total Records: 2
```

1.3 List the Top 10 genres with the highest average energy

[66]:
```
query = """
SELECT track_genre, AVG(energy) AS avg_energy
FROM data
GROUP BY track_genre
ORDER BY avg_energy DESC
LIMIT 10
"""
result = psql.sqldf(query, locals())
execute_query(result, "List the Top 10 genres with the highest average energy")
```

```
List the Top 10 genres with the highest average energy
======================================================

     track_genre  avg_energy
0    death-metal    0.931470
1      grindcore    0.924201
2      metalcore    0.914485
3          happy    0.910971
4      hardstyle    0.901246
5  drum-and-bass    0.876635
6    black-metal    0.874897
7    heavy-metal    0.874003
8          party    0.871237
9         j-idol    0.868677

Total Records: 10
```

1.4 How many tracks is Bad Bunny on?

[67]:
```
query = """
SELECT COUNT(*)
FROM data
WHERE artists = 'Bad Bunny'
"""
result = psql.sqldf(query, locals())
execute_query(result, "How many tracks is Bad Bunny on?")
```

```
How many tracks is Bad Bunny on?
================================

   COUNT(*)
0        48

Total Records: 1
```

1.5 Show the top 10 genres in terms of popularity, sorted by their most popular track

```
[68]: query = """
SELECT track_genre, MAX(popularity) AS max_popularity
FROM data
GROUP BY track_genre
ORDER BY max_popularity DESC
LIMIT 10
"""
result = psql.sqldf(query, locals())
execute_query(result, "Show the top 10 genres in terms of popularity, sorted by␣
↪their most popular track")
```

```
Show the top 10 genres in terms of popularity, sorted by their most popular
track
================================================================================
=

  track_genre  max_popularity
0         pop             100
1       dance             100
2     hip-hop              99
3   reggaeton              98
4      reggae              98
5      latino              98
6       latin              98
7         edm              98
8        rock              96
9       piano              96

Total Records: 10
```

```
[ ]:
```