```
!pip install -q pyathena
```

```
WARNING: Running pip as the 'root' user can result in broken
permissions and conflicting behaviour with the system package manager.
It is recommended to use a virtual environment instead:
https://pip.pypa.io/warnings/venv
```

```
import boto3
import sagemaker
import pandas as pd
from pyathena import connect
```

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

Auth with AWS

```
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)
```

Dropping album_name from dataset

```
raw_df = pd.read_csv('dataset.csv')
display(raw_df.head())
```

```
   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]
```

```
cleaned_df = raw_df.drop(columns=['album_name', 'Unnamed: 0'])
display(cleaned_df.head())
```

```
                 track_id                 artists
track_name  \
0  5SuOikwiRyPMVoIQDJUgSV              Gen Hoshino
Comedy
1  4qPNDBW1i3p13qLCt0Ki3A             Ben Woodward            Ghost -
Acoustic
2  1iJBSr7s7jYXzM8EGcbK5b  Ingrid Michaelson;ZAYN              To
Begin Again
3  6lfxq3CG4xtTiEg7opyCyx             Kina Grannis  Can't Help Falling
In Love
4  5vjLSffimiIP26QG5WcN2K        Chord Overstreet
Hold On

   popularity  duration_ms  explicit  danceability  energy  key
loudness  \
0          73       230666     False         0.676  0.4610    1    -
```

```
6.746
1          55       149610     False          0.420  0.1660    1   -
17.235
2          57       210826     False          0.438  0.3590    0    -
9.734
3          71       201933     False          0.266  0.0596    0   -
18.515
4          82       198853     False          0.618  0.4430    2    -
9.681

   mode  speechiness  acousticness  instrumentalness  liveness
valence  \
0     0       0.1430        0.0322          0.000001    0.3580
0.715
1     1       0.0763        0.9240          0.000006    0.1010
0.267
2     1       0.0557        0.2100          0.000000    0.1170
0.120
3     1       0.0363        0.9050          0.000071    0.1320
0.143
4     1       0.0526        0.4690          0.000000    0.0829
0.167

     tempo  time_signature track_genre
0   87.917               4    acoustic
1   77.489               4    acoustic
2   76.332               4    acoustic
3  181.740               3    acoustic
4  119.949               4    acoustic

# create local version
cleaned_df.to_csv('dataset_clean.csv', index=False)
```

Convert csv to tsv and move to S3

```
s3_private_data_path = "s3://{}/w2-musicData/csv".format(bucket)
print(s3_private_data_path)

s3://sagemaker-us-east-1-106006112223/w2-musicData/csv

!aws s3 cp "dataset_clean.csv" $s3_private_data_path/

upload: ./dataset_clean.csv to
s3://sagemaker-us-east-1-106006112223/w2-musicData/csv/dataset_clean.c
sv

!aws s3 ls $s3_private_data_path/

2024-09-17 05:56:57   16931936 dataset_clean.csv
```

Create DB in Athena for queries

```
# Set S3 staging directory -- this is a temporary directory used for
Athena queries
database_name = "w2_music_db"
table_name_tsv = 'music_ds_tsv10'
s3_staging_dir = "s3://{0}/athena/staging".format(bucket)
print(s3_staging_dir)
conn = connect(region_name=region, s3_staging_dir=s3_staging_dir)
```

```
s3://sagemaker-us-east-1-106006112223/athena/staging
```

```
statement = "CREATE DATABASE IF NOT EXISTS {}".format(database_name)
pd.read_sql(statement, conn)

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

```
/tmp/ipykernel_140/3245868569.py:2: 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.
  pd.read_sql(statement, conn)
/tmp/ipykernel_140/3245868569.py:5: 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)
```

```
  database_name
0       default
1        dsoaws
2   w2_music_db
```

Create tables in DB and schemas

```
# SQL statement to execute
statement = """CREATE EXTERNAL TABLE IF NOT EXISTS {}.{}(
         track_id string,
         artists string,
         track_name string,
         popularity int,
         duration_ms int,
         explicit string,
         danceability float,
         energy float,
         key int,
         loudness float,
         mode int,
         speechiness float,
         acousticness float,
         instrumentalness float,
         liveness float,
         valence float,
         tempo float,
         time_signature int,
         track_genre string
) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LINES TERMINATED BY
'\\n' LOCATION '{}'
TBLPROPERTIES ('compressionType'='gzip',
'skip.header.line.count'='1')""".format(
    database_name, table_name_tsv, s3_private_data_path
)

print(statement)

pd.read_sql(statement, conn)
```

```
CREATE EXTERNAL TABLE IF NOT EXISTS w2_music_db.music_ds_tsv10(
         track_id string,
         artists string,
         track_name string,
         popularity int,
         duration_ms int,
         explicit boolean,
         danceability float,
         energy float,
         key int,
         loudness float,
         mode int,
         speechiness float,
         acousticness float,
         instrumentalness float,
         liveness float,
         valence float,
         tempo float,
         time_signature int,
         track_genre string
) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LINES TERMINATED BY '\
n' LOCATION 's3://sagemaker-us-east-1-106006112223/w2-musicData/csv'
TBLPROPERTIES ('compressionType'='gzip', 'skip.header.line.count'='1')
```

```
/tmp/ipykernel_359/1675701273.py:29: 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.
  pd.read_sql(statement, conn)
```

```
Empty DataFrame
Columns: []
Index: []

statement = "SHOW TABLES IN W2_MUSIC_DB"

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

/tmp/ipykernel_359/1294112312.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)

         tab_name
0  music_ds_tsv10
1   music_ds_tsv8
2   music_ds_tsv9

# first test query to get all data via athena

statement = """SELECT * FROM {}.{} LIMIT 5""".format(
    database_name, table_name_tsv
)
print(statement)
sql_df = pd.read_sql(statement, conn)
display(sql_df.head(5))

SELECT * FROM w2_music_db.music_ds_tsv10 LIMIT 5

/tmp/ipykernel_359/1057200777.py:7: 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.
  sql_df = pd.read_sql(statement, conn)

                 track_id                 artists
track_name  \
0  5SuOikwiRyPMVoIQDJUgSV             Gen Hoshino
Comedy
1  4qPNDBW1i3p13qLCt0Ki3A            Ben Woodward            Ghost -
Acoustic
2  1iJBSr7s7jYXzM8EGcbK5b  Ingrid Michaelson;ZAYN              To
Begin Again
3  6lfxq3CG4xtTiEg7opyCyx            Kina Grannis  Can't Help Falling
In Love
4  5vjLSffimiIP26QG5WcN2K        Chord Overstreet
Hold On

   popularity  duration_ms  explicit  danceability  energy  key
```

```
loudness  \
0          73       230666      False         0.676  0.4610    1    -
6.746
1          55       149610      False         0.420  0.1660    1   -
17.235
2          57       210826      False         0.438  0.3590    0    -
9.734
3          71       201933      False         0.266  0.0596    0   -
18.515
4          82       198853      False         0.618  0.4430    2    -
9.681

   mode  speechiness  acousticness  instrumentalness  liveness
valence  \
0     0       0.1430        0.0322          0.000001    0.3580
0.715
1     1       0.0763        0.9240          0.000006    0.1010
0.267
2     1       0.0557        0.2100          0.000000    0.1170
0.120
3     1       0.0363        0.9050          0.000071    0.1320
0.143
4     1       0.0526        0.4690          0.000000    0.0829
0.167

     tempo  time_signature track_genre
0   87.917               4    acoustic
1   77.489               4    acoustic
2   76.332               4    acoustic
3  181.740               3    acoustic
4  119.949               4    acoustic

# reading local csv file using pandas
full_df = pd.read_csv('dataset_clean.csv')
full_df = full_df.dropna()
display(full_df.head())

                 track_id                 artists
track_name  \
0  5SuOikwiRyPMVoIQDJUgSV              Gen Hoshino
Comedy
1  4qPNDBW1i3p13qLCt0Ki3A             Ben Woodward            Ghost -
Acoustic
2  1iJBSr7s7jYXzM8EGcbK5b  Ingrid Michaelson;ZAYN              To
Begin Again
3  6lfxq3CG4xtTiEg7opyCyx             Kina Grannis  Can't Help Falling
In Love
4  5vjLSffimiIP26QG5WcN2K         Chord Overstreet
Hold On
```

```
   popularity  duration_ms  explicit  danceability  energy  key  
loudness  \
0          73       230666     False         0.676  0.4610    1    -
6.746   
1          55       149610     False         0.420  0.1660    1   -
17.235   
2          57       210826     False         0.438  0.3590    0    -
9.734   
3          71       201933     False         0.266  0.0596    0   -
18.515   
4          82       198853     False         0.618  0.4430    2    -
9.681   

   mode  speechiness  acousticness  instrumentalness  liveness  
valence  \
0     0       0.1430        0.0322          0.000001    0.3580   
0.715   
1     1       0.0763        0.9240          0.000006    0.1010   
0.267   
2     1       0.0557        0.2100          0.000000    0.1170   
0.120   
3     1       0.0363        0.9050          0.000071    0.1320   
0.143   
4     1       0.0526        0.4690          0.000000    0.0829   
0.167   

     tempo  time_signature track_genre  
0   87.917               4    acoustic  
1   77.489               4    acoustic  
2   76.332               4    acoustic  
3  181.740               3    acoustic  
4  119.949               4    acoustic  
```

Homework queries

1. List artist, track_name, and popularity for songs that have a popularity greater than or equal to 99

```
statement = """SELECT artists, track_name, popularity FROM {}.{}
    WHERE popularity >= 99""".format(
    database_name, table_name_tsv
)

# CAST(popularity AS INTEGER) >= 99
print(statement)

df = pd.read_sql(statement, conn)
df.head(10)
```

```
SELECT artists, track_name, popularity FROM w2_music_db.music_ds_tsv10
    WHERE popularity >= 99

/tmp/ipykernel_359/2117349163.py:9: 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 = pd.read_sql(statement, conn)

                artists                 track_name  popularity
0  Sam Smith;Kim Petras  Unholy (feat. Kim Petras)         100
1  Sam Smith;Kim Petras  Unholy (feat. Kim Petras)         100

# pandas
pd_df = full_df[full_df['popularity'] >= 99]
[['artists','track_name','popularity']]
display(pd_df.head())

                    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
```

2. List artists with an average popularity of 92

```
statement = """SELECT artists FROM {}.{}
    GROUP BY artists HAVING AVG(popularity) = 92""".format(
    database_name, table_name_tsv
)

print(statement)

df = pd.read_sql(statement, conn)
df.head(10)

SELECT artists FROM w2_music_db.music_ds_tsv9
    GROUP BY artists HAVING AVG(popularity) = 92

/tmp/ipykernel_245/3210071803.py:14: 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 = pd.read_sql(statement, conn)

             artists
0       Harry Styles
1  Rema;Selena Gomez
```

```
# pandas
artists_avg_popularity = full_df.groupby('artists').filter(lambda x:
x['popularity'].mean() == 92)
display(artists_avg_popularity.head())
artists_avg_popularity_list =
artists_avg_popularity['artists'].unique()
print(artists_avg_popularity_list)
```

```
                     track_id            artists  \
81052  4LRPiXqCikLlN15c3yImP7       Harry Styles
81100  0WtM2NBVQNNJLh6scP13H8  Rema;Selena Gomez
81158  6UelLqGlWMcVH1E5c4H7lY       Harry Styles
81205  4Dvkj6JhhA12EX05fT7y2e       Harry Styles

                            track_name  popularity  duration_ms
explicit  \
81052                        As It Was          95       167303
False
81100  Calm Down (with Selena Gomez)          92       239317
False
81158                 Watermelon Sugar          89       174000
False
81205                        As It Was          92       167303
False

       danceability  energy  key  loudness  mode  speechiness
acousticness  \
81052         0.520   0.731    6    -5.338     0       0.0557
0.342
81100         0.801   0.806   11    -5.206     1       0.0381
0.382
81158         0.548   0.816    0    -4.209     1       0.0465
0.122
81205         0.520   0.731    6    -5.338     0       0.0557
0.342

       instrumentalness  liveness  valence    tempo  time_signature  \
81052          0.001010     0.311    0.662  173.930               4
81100          0.000669     0.114    0.802  106.999               4
81158          0.000000     0.335    0.557   95.390               4
81205          0.001010     0.311    0.662  173.930               4

      track_genre
81052         pop
81100         pop
81158         pop
81205         pop

['Harry Styles' 'Rema;Selena Gomez']
```

3. List the Top 10 most energetic genres

```
statement = """SELECT DISTINCT track_genre
    FROM {}.{}
    LIMIT 10;""".format(
    database_name, table_name_tsv
)

print(statement)
df = pd.read_sql(statement, conn)
display(df.head(10))

# Error in schema / parsing, track_genre is all messed up
```

```
SELECT DISTINCT track_genre
    FROM w2_music_db.music_ds_tsv10
    LIMIT 10;
```

```
/tmp/ipykernel_359/1041569638.py:8: 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 = pd.read_sql(statement, conn)
```

```
   track_genre
0     acoustic
1            4
2            3
3       163.99
4      124.157
5      193.395
6     afrobeat
7     alt-rock
8  alternative
9      ambient
```

```
statement = """SELECT track_genre, AVG(energy) AS avg_energy FROM {}.
{}
    GROUP BY track_genre
    ORDER BY avg_energy DESC""".format(
    database_name, table_name_tsv
)

print(statement)
df = pd.read_sql(statement, conn)
df.head(10)
```

```
SELECT track_genre, AVG(energy) AS avg_energy FROM
w2_music_db.music_ds_tsv10
    GROUP BY track_genre
    ORDER BY avg_energy DESC
```

```
/tmp/ipykernel_359/4047995415.py:14: 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 = pd.read_sql(statement, conn)

  track_genre  avg_energy
0       0.797   1174026.0
1       0.556    691306.0
2       0.492    542000.0
3        0.45    538160.0
4       0.347    526706.0
5      0.0761    502786.0
6      0.0903    449813.0
7       0.035    440310.0
8       0.483    371160.0
9       0.147    355693.0

# pandas
top_energetic_genres = full_df.groupby('track_genre')
['energy'].mean().sort_values(ascending=False).head(10)
display(top_energetic_genres)

track_genre
death-metal      0.931470
grindcore        0.924201
metalcore        0.914485
happy            0.910971
hardstyle        0.901246
drum-and-bass    0.876635
black-metal      0.874897
heavy-metal      0.874003
party            0.871237
j-idol           0.868677
Name: energy, dtype: float64
```

4. How many tracks is Bad Bunny on?

```
# SELECT COUNT(*) AS track_count
# FROM w2_music_db.tracks
# WHERE artists LIKE '%Bad Bunny%';

statement = """SELECT COUNT(*) AS track_count FROM {}.{}
    WHERE artists LIKE '%Bad Bunny%'""".format(
    database_name, table_name_tsv
)

print(statement)
df = pd.read_sql(statement, conn)
print(df)
```

```
/tmp/ipykernel_359/768749324.py:11: 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 = pd.read_sql(statement, conn)

SELECT COUNT(*) AS track_count FROM w2_music_db.music_ds_tsv10
    WHERE artists LIKE '%Bad Bunny%'
   track_count
0          416

bad_bunny_tracks_count = full_df[full_df['artists'].str.contains('Bad
Bunny')].shape[0]
print(bad_bunny_tracks_count)

416
```

5. Show the top 10 genres in terms of popularity sorted by their most popular track

```
# SELECT track_genre, MAX(popularity) AS max_popularity
# FROM w2_music_db.tracks
# GROUP BY track_genre
# ORDER BY max_popularity DESC
# LIMIT 10;

statement = """SELECT track_genre, MAX(popularity) AS max_popularity
FROM {}.{}
    GROUP BY track_genre
    ORDER BY max_popularity DESC
    LIMIT 10""".format(
    database_name, table_name_tsv
)

print(statement)
df = pd.read_sql(statement, conn)
df.head(10)

# noticed slight difference in return... hip hop genre got ereased?

SELECT track_genre, MAX(popularity) AS max_popularity FROM
w2_music_db.music_ds_tsv10
    GROUP BY track_genre
    ORDER BY max_popularity DESC
    LIMIT 10

/tmp/ipykernel_359/3944904740.py:15: 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 = pd.read_sql(statement, conn)
  track_genre  max_popularity
0       dance             100
1         pop             100
2       latin              98
3   reggaeton              98
4      latino              98
5         edm              98
6      reggae              98
7       piano              96
8        rock              96
9       chill              93
```

```
# pandas
top_genres_by_popularity = full_df.groupby('track_genre')
['popularity'].max().sort_values(ascending=False).head(10)
print(top_genres_by_popularity)
```

```
track_genre
dance        100
pop          100
hip-hop       99
latin         98
edm           98
latino        98
reggaeton     98
reggae        98
rock          96
piano         96
Name: popularity, dtype: int64
```

```
%%html

<p><b>Shutting down your kernel for this notebook to release
resources.</b></p>
<button class="sm-command-button" data-commandlinker-
command="kernelmenu:shutdown" style="display:none;">Shutdown
Kernel</button>

<script>
try {
    els = document.getElementsByClassName("sm-command-button");
    els[0].click();
}
catch(err) {
    // NoOp
}
</script>
```

```
%%javascript

try {
    Jupyter.notebook.save_checkpoint();
    Jupyter.notebook.session.delete();
}
catch(err) {
    // NoOp
}
```