TIme to Query Data From Athena

In [ ]:
```
%store -r ingest_create_athena_table_csv_passed
```

In [ ]:
```
try:
    ingest_create_athena_table_csv_passed
except NameError:
    print("++++++++++++++++++++++++++++++++++++++++++++++++")
    print("[ERROR] YOU HAVE TO RUN ALL PREVIOUS NOTEBOOKS.  You did not register th
    print("++++++++++++++++++++++++++++++++++++++++++++++++")
```

In [ ]:
```
print(ingest_create_athena_table_csv_passed)
```

```
True
```

In [ ]:
```
if not ingest_create_athena_table_csv_passed:
    print("++++++++++++++++++++++++++++++++++++++++++++++++")
    print("[ERROR] YOU HAVE TO RUN ALL PREVIOUS NOTEBOOKS.  You did not register th
    print("++++++++++++++++++++++++++++++++++++++++++++++++")
else:
    print("[OK]")
```

```
[OK]
```

In [ ]:
```
%store
```

```
Stored variables and their in-db values:
data_path                                    -> '/root/AAI-540/Module2/csv'
ingest_create_athena_db_mod2_passed          -> True
ingest_create_athena_table_csv_passed        -> True
s3_private_path_csv                          -> 's3://sagemaker-us-east-1-00460
8622582/module2_dat
setup_dependencies_mod2_passed               -> True
setup_s3_bucket_passed                       -> True
```

Setup

In [ ]:
```
import sagemaker
import boto3

sess = sagemaker.Session()
bucket = sess.default_bucket()
role = sagemaker.get_execution_role()
region = boto3.Session().region_name

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

In [ ]:
```
import awswrangler as wr
```

Query From Glue Catalog

In [ ]:
```
database_name = "mod2_db"
table_name_csv = "music"
```

In [ ]:
```
for table in wr.catalog.get_tables(database="mod2_db"):
    print(table["Name"])
```

```
music
```

Query From Athena

In [ ]:
```
%%time
df = wr.athena.read_sql_query(sql="SELECT * FROM {}.{} LIMIT 5000".format(database_
```

```
CPU times: user 456 ms, sys: 48.9 ms, total: 505 ms
Wall time: 2.82 s
```

In [ ]:
```
df.head()
```

Out[ ]:

	track_id	artists	popularity	duration_ms	explicit	danceabili
0	5SuOikwiRyPMVoIQDJUgSV	Gen Hoshino	73	230666	False	0.6
1	4qPNDBW1i3p13qLCt0Ki3A	Ben Woodward	55	149610	False	0.4
2	1iJBSr7s7jYXzM8EGcbK5b	Ingrid Michaelson;ZAYN	57	210826	False	0.4
3	6lfxq3CG4xtTiEg7opyCyx	Kina Grannis	71	201933	False	0.2
4	5vjLSffimiIP26QG5WcN2K	Chord Overstreet	82	198853	False	0.6

QUERY Tasks

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

In [ ]:
```
%%time
# For the sake of the table that I am using, I took out the artist_name and track_n
df = wr.athena.read_sql_query(sql="SELECT artists, popularity FROM {} WHERE popular
```

```
CPU times: user 584 ms, sys: 32.3 ms, total: 616 ms
Wall time: 2.74 s
```

In [ ]:
```
df.head()
```

Out[ ]:

	artists	popularity
0	Sam Smith;Kim Petras	100
1	Bizarrap;Quevedo	99
2	Sam Smith;Kim Petras	100

2. List artists with an average popularity of 92

In [ ]:
```
%%time
df = wr.athena.read_sql_query(sql="SELECT artists, AVG(popularity) AS avg_popularit
```

```
CPU times: user 465 ms, sys: 18 ms, total: 483 ms
Wall time: 3.29 s
```

In [ ]:
```
df.head()
```

Out[ ]:

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

3.List the Top 10 most energetic genres

In [ ]:
```
%%time
df = wr.athena.read_sql_query(sql="SELECT AVG(energy) as avg_energy, track_genre FR
```

```
CPU times: user 600 ms, sys: 49.3 ms, total: 649 ms
Wall time: 2.96 s
```

In [ ]:
```
df.head(10)
```

Out[ ]:

	avg_energy	track_genre
0	0.931470	death-metal
1	0.924201	grindcore
2	0.914220	metalcore
3	0.910971	happy
4	0.901246	hardstyle
5	0.876617	drum-and-bass
6	0.874897	black-metal
7	0.874003	heavy-metal
8	0.871237	party
9	0.868677	j-idol

4. How many tracks is Bad Bunny On?

In [ ]:
```
%%time
df = wr.athena.read_sql_query(sql="SELECT COUNT(*) AS bb_count FROM {}.{} WHERE art
# For this particular scnenario, I used the artists feature column since I removed
```

```
CPU times: user 555 ms, sys: 52.8 ms, total: 608 ms
Wall time: 3.48 s
```

In [ ]:
```
df.head()
```

Out[ ]:

	bb_count
0	48

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

In [ ]:
```
%%time
df = wr.athena.read_sql_query(sql="SELECT MAX(popularity) as max_popularity, track_
```

```
CPU times: user 615 ms, sys: 47.2 ms, total: 663 ms
Wall time: 2.98 s
```

In [ ]:
```
df.head(10)
```

Out[ ]:

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

Rewritting this in Pandas

In [ ]:
```
%store
```

```
Stored variables and their in-db values:
data_path                                       -> '/root/AAI-540/Module2/csv'
ingest_create_athena_db_mod2_passed             -> True
ingest_create_athena_table_csv_passed           -> True
s3_private_path_csv                             -> 's3://sagemaker-us-east-1-00460
8622582/module2_dat
setup_dependencies_mod2_passed                  -> True
setup_s3_bucket_passed                          -> True
```

In [ ]:
```
%store -r data_path
```

In [ ]:
```
# Reading in the CSV
import pandas as pd
```

```
df_pd = pd.read_csv(f"{data_path}/new_dataset.csv")
df_pd.head()
```

Out[ ]:

	track_id	artists	popularity	duration_ms	explicit	danceabili
0	5SuOikwiRyPMVoIQDJUgSV	Gen Hoshino	73	230666	False	0.6
1	4qPNDBW1i3p13qLCt0Ki3A	Ben Woodward	55	149610	False	0.4
2	1iJBSr7s7jYXzM8EGcbK5b	Ingrid Michaelson;ZAYN	57	210826	False	0.4
3	6lfxq3CG4xtTiEg7opyCyx	Kina Grannis	71	201933	False	0.2
4	5vjLSffimiIP26QG5WcN2K	Chord Overstreet	82	198853	False	0.6

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

In [ ]:
```
query_1 = df_pd[df_pd['popularity']>=99][['artists','popularity']]
print(query_1)
```

```
                  artists  popularity
20001  Sam Smith;Kim Petras         100
51664      Bizarrap;Quevedo          99
81051  Sam Smith;Kim Petras         100
```

2. List artists with an average popularity of 92

In [ ]:
```
avg = df_pd.groupby('artists')['popularity'].mean().reset_index()
query_2 = avg[avg['popularity']==92]
print(query_2)
```

```
                artists  popularity
11491       Harry Styles        92.0
22845  Rema;Selena Gomez        92.0
```

3.List the Top 10 most energetic genres

In [ ]:
```
avg = df_pd.groupby('track_genre')['energy'].mean().reset_index()
query_3 = avg.sort_values(by='energy', ascending=False).head(10)
print(query_3)
```

```
      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
```

4. How many tracks is Bad Bunny On?

In [ ]:
```
# For this particular scnenario, I used the artists feature column since I removed
query_4 = sum(df_pd[df_pd['artists']=='Bad Bunny'].value_counts())
print(f"There are {query_4} Bad Bunny Tracks")
```

```
There are 48 Bad Bunny Tracks
```

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

In [ ]:
```
avg = df_pd.groupby('track_genre')['popularity'].max().reset_index()
query_3 = avg.sort_values(by='popularity', ascending=False).head(10)
print(query_3)
```

```
   track_genre  popularity
20       dance         100
80         pop         100
51     hip-hop          99
67       latin          98
30         edm          98
68      latino          98
89   reggaeton          98
88      reggae          98
90        rock          96
79       piano          96
```

Some of the values here are in slightly different order, but the contents match. Thank you!

Release Resources

In [ ]:
```
%%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"

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

Shutting down your kernel for this notebook to release resources.