In [1]:

```
print("Welcome to my EMR Notebook!")
```

```
Starting Spark application
```

ID	YARN Application ID	Kind	State	
2	application_1584804639642_0003	pyspark	idle	Link (http://i 230.ec2.internal:20888/proxy/application_158480463

```
SparkSession available as 'spark'.

Welcome to my EMR Notebook!
```

In [2]:

```
%%info
```

Current session configs: `{'conf': {'spark.pyspark.python': 'python3', 'spark.pyspark.virtualenv.enabled': 'true', 'spark.pyspark.virtualenv.type': 'native', 'spark.pyspark.virtualenv.bin.path': '/usr/bin/virtualenv'}, 'kind': 'pyspark'}`

ID	YARN Application ID	Kind	State	
2	application_1584804639642_0003	pyspark	idle	Link (http://i 230.ec2.internal:20888/proxy/application_158480463

In [3]:

```
sc.list_packages()
```

```
Package                    Version
-------------------------- -------
beautifulsoup4             4.8.1
boto                       2.49.0
jmespath                   0.9.4
lxml                       4.4.2
mysqlclient                1.4.6
nltk                       3.4.5
nose                       1.3.4
numpy                      1.14.5
pip                        20.0.2
py-dateutil                2.2
python36-sagemaker-pyspark 1.2.6
pytz                       2019.3
PyYAML                     3.11
setuptools                 46.1.0
six                        1.13.0
soupsieve                  1.9.5
wheel                      0.34.2
windmill                   1.6
```

In [69]:

```
sc.install_pypi_package("boto3")
```

```
Collecting boto3
  Downloading boto3-1.12.26-py2.py3-none-any.whl (128 kB)
Collecting s3transfer<0.4.0,>=0.3.0
  Downloading s3transfer-0.3.3-py2.py3-none-any.whl (69 kB)
Requirement already satisfied: jmespath<1.0.0,>=0.7.1 in /usr/local/lib/pytho
n3.6/site-packages (from boto3) (0.9.4)
Collecting botocore<1.16.0,>=1.15.26
  Downloading botocore-1.15.26-py2.py3-none-any.whl (6.0 MB)
Collecting docutils<0.16,>=0.10
  Downloading docutils-0.15.2-py3-none-any.whl (547 kB)
Collecting urllib3<1.26,>=1.20; python_version != "3.4"
  Downloading urllib3-1.25.8-py2.py3-none-any.whl (125 kB)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /mnt/tmp/158481
7234280-0/lib/python3.6/site-packages (from botocore<1.16.0,>=1.15.26->boto3)
(2.8.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.6/site-pack
ages (from python-dateutil<3.0.0,>=2.1->botocore<1.16.0,>=1.15.26->boto3) (1.
13.0)
Installing collected packages: docutils, urllib3, botocore, s3transfer, boto3
Successfully installed boto3-1.12.26 botocore-1.15.26 docutils-0.15.2 s3trans
fer-0.3.3 urllib3-1.25.8
```

In [4]:

```
sc.install_pypi_package("pandas") #Install pandas version 0.25.1
sc.install_pypi_package("matplotlib", "https://pypi.org/simple")
sc.install_pypi_package("IPython", "https://pypi.org/simple")
```

```
Collecting pandas
  Using cached pandas-1.0.3-cp36-cp36m-manylinux1_x86_64.whl (10.0 MB)
Collecting python-dateutil>=2.6.1
  Using cached python_dateutil-2.8.1-py2.py3-none-any.whl (227 kB)
Requirement already satisfied: numpy>=1.13.3 in /usr/local/lib64/python3.6/si
te-packages (from pandas) (1.14.5)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.6/site-
packages (from pandas) (2019.3)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.6/site-pack
ages (from python-dateutil>=2.6.1->pandas) (1.13.0)
Installing collected packages: python-dateutil, pandas
Successfully installed pandas-1.0.3 python-dateutil-2.8.1

Collecting matplotlib
  Using cached matplotlib-3.2.1-cp36-cp36m-manylinux1_x86_64.whl (12.4 MB)
Collecting pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1
  Using cached pyparsing-2.4.6-py2.py3-none-any.whl (67 kB)
Requirement already satisfied: numpy>=1.11 in /usr/local/lib64/python3.6/site
-packages (from matplotlib) (1.14.5)
Requirement already satisfied: python-dateutil>=2.1 in /mnt/tmp/1584817234280
-0/lib/python3.6/site-packages (from matplotlib) (2.8.1)
Collecting kiwisolver>=1.0.1
  Using cached kiwisolver-1.1.0-cp36-cp36m-manylinux1_x86_64.whl (90 kB)
Collecting cycler>=0.10
  Using cached cycler-0.10.0-py2.py3-none-any.whl (6.5 kB)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.6/site-pack
ages (from python-dateutil>=2.1->matplotlib) (1.13.0)
Requirement already satisfied: setuptools in /mnt/tmp/1584817234280-0/lib/pyt
hon3.6/site-packages (from kiwisolver>=1.0.1->matplotlib) (46.1.0)
Installing collected packages: pyparsing, kiwisolver, cycler, matplotlib
Successfully installed cycler-0.10.0 kiwisolver-1.1.0 matplotlib-3.2.1 pypars
ing-2.4.6
```

In [70]:

```
import pandas as pd
import matplotlib.pyplot as plt
from IPython.display import display
import boto3
from io import StringIO
```

Definir el bucket

In [ ]:

```
client = boto3.client('s3', region_name='us-east-1')
bucket = 'recommendation-system-md' # already created on S3
```

Leyendo los datos

In [6]:

```
triplets_file = 'https://static.turi.com/datasets/millionsong/10000.txt'
song_df_1 = pd.read_table(triplets_file,header=None)
song_df_1.columns = ['user_id', 'song_id', 'listen_count']
```

In [7]:

```
songs_metadata_file = 'https://static.turi.com/datasets/millionsong/song_data.csv'
```

In [8]:

```
song_df_2 =  pd.read_csv(songs_metadata_file)
song_df_2['title']=song_df_2['title'].str.strip()
song_df = pd.merge(song_df_1, song_df_2.drop_duplicates(['song_id']), on="song_id", how="l
eft")
```

In [9]:

```
song_df.columns
```

```
Index(['user_id', 'song_id', 'listen_count', 'title', 'release', 'artist_nam
e',
       'year'],
      dtype='object')
```

In [87]:

```
print(f'El total de registros es de  {len(song_df)}')
```

```
El total de registros es de  2000000
```

In [10]:

```
print(f'El total de canciones únicas es {len(song_df["song_id"].drop_duplicates())}')
```

```
El total de canciones únicas es 10000
```

In [11]:

```
print(f'El total de usuarios únicos es {len(song_df["user_id"].drop_duplicates())}')
```

```
El total de usuarios únicos es 76353
```

In [12]:

```
print(f'El total de artistas únicos es {len(song_df["artist_name"].drop_duplicates())}')
```

```
El total de artistas únicos es 3375
```

Ejemplo de una Playlist

In [13]:

```
#Un ejemplo de una playlist
playlist = song_df[song_df["user_id"]==song_df["user_id"].loc[0]][['title', 'release', 'ar
tist_name']]
playlist.head()
```

```
             title                        release    artist_name
0         The Cove              Thicker Than Water  Jack Johnson
1   Entre Dos Aguas             Flamenco Para Niños  Paco De Lucia
2         Stronger                     Graduation     Kanye West
3   Constellations              In Between Dreams   Jack Johnson
4     Learn To Fly  There Is Nothing Left To Lose   Foo Fighters
```

In [14]:

```
print(f"El número de canciones que tiene la playlist del usuario {song_df['user_id'].loc
[0]} es de {len(playlist['title'].unique())}")
```

```
El número de canciones que tiene la playlist del usuario b80344d063b5ccb3212f
76538f3d9e43d87dca9e es de 45
```

Calculado las canciones más escuchadas

In [79]:

```
listen_total_by_song = song_df.groupby(['song_id','title', 'release', 'artist_name']).agg
({"listen_count":"sum","user_id":"nunique"})
```

In [80]:

```
csv_buffer = StringIO()
listen_total_by_song.to_csv(csv_buffer)
client.put_object(Bucket=bucket, Key='listen_total_by_song.csv',Body=csv_buffer.getvalue
())
```

```
{'ResponseMetadata': {'RequestId': '894E9B6F04CD3434', 'HostId': '/r9ypT+W/72
hC9qs6IuFujAgyCL2SRteWXWC6YxS71sYhpe9Mk79YoFeLZyoX6zg57ZwzT/jmik=', 'HTTPStat
usCode': 200, 'HTTPHeaders': {'x-amz-id-2': '/r9ypT+W/72hC9qs6IuFujAgyCL2SRte
WXWC6YxS71sYhpe9Mk79YoFeLZyoX6zg57ZwzT/jmik=', 'x-amz-request-id': '894E9B6F0
4CD3434', 'date': 'Sat, 21 Mar 2020 19:49:58 GMT', 'etag': '"4805fc17214cb273
b934898cc03759b2"', 'content-length': '0', 'server': 'AmazonS3'}, 'RetryAttem
pts': 0}, 'ETag': '"4805fc17214cb273b934898cc03759b2"'}
```

Calculando los artistas más escuchados

In [81]:

```
listen_total_by_artist = song_df.groupby(['artist_name']).agg({"listen_count":["sum","coun
t"],'user_id':'nunique','song_id':'nunique','release':'nunique'})
# top100_artist = listen_total_by_artist.loc[0:100]
```

In [82]:

```
cols = [f"{l0}_{l1}" for (l0, l1) in zip(listen_total_by_artist.columns.get_level_values(0
), listen_total_by_artist.columns.get_level_values(1))]
listen_total_by_artist.columns = cols
```

In [83]:

```
csv_buffer = StringIO()
listen_total_by_artist.to_csv(csv_buffer)
client.put_object(Bucket=bucket, Key='artistas_listen_user.csv',Body=csv_buffer.getvalue
())
```

```
{'ResponseMetadata': {'RequestId': 'F1DBBC43E13D00CA', 'HostId': 'ROJAzWd4txn
trYeLZ0BbQ8FqOTnc8YxajZnUzGZa42DKXIar/ZA9/qDpEG7tIg1vXRTuqnsPF38=', 'HTTPStat
usCode': 200, 'HTTPHeaders': {'x-amz-id-2': 'ROJAzWd4txntrYeLZ0BbQ8FqOTnc8Yxa
jZnUzGZa42DKXIar/ZA9/qDpEG7tIg1vXRTuqnsPF38=', 'x-amz-request-id': 'F1DBBC43E
13D00CA', 'date': 'Sat, 21 Mar 2020 19:56:43 GMT', 'etag': '"fb8129e73bce241f
64e1b2403a2559fd"', 'content-length': '0', 'server': 'AmazonS3'}, 'RetryAttem
pts': 0}, 'ETag': '"fb8129e73bce241f64e1b2403a2559fd"'}
```

Calculado el album más escuchado

In [84]:

```
listen_total_by_album = song_df.groupby(['release','artist_name']).agg({"listen_count":["s
um","count"],'user_id':'nunique','song_id':'nunique'})
```

In [85]:

```
cols = [f"{l0}_{l1}" for (l0, l1) in zip(listen_total_by_album.columns.get_level_values(0
), listen_total_by_album.columns.get_level_values(1))]
listen_total_by_album.columns = cols
```

In [86]:

```
csv_buffer = StringIO()
listen_total_by_album.to_csv(csv_buffer)
client.put_object(Bucket=bucket, Key='listen_total_by_album.csv',Body=csv_buffer.getvalue
())
```

```
{'ResponseMetadata': {'RequestId': '94955707D5873886', 'HostId': 'HESde/PXefk
8I9Q+2OzdeshHQJ/SNN/v97kfAKpv0TWFARHiCqDIPiACwdnv53xBEBKJbz7WO50=', 'HTTPStat
usCode': 200, 'HTTPHeaders': {'x-amz-id-2': 'HESde/PXefk8I9Q+2OzdeshHQJ/SNN/v
97kfAKpv0TWFARHiCqDIPiACwdnv53xBEBKJbz7WO50=', 'x-amz-request-id': '94955707D
5873886', 'date': 'Sat, 21 Mar 2020 20:00:37 GMT', 'etag': '"0d142e86f4a3ee12
0af96331eea9fcbb"', 'content-length': '0', 'server': 'AmazonS3'}, 'RetryAttem
pts': 0}, 'ETag': '"0d142e86f4a3ee120af96331eea9fcbb"'}
```

In [ ]:

```

```