Big Data Analytics

SparkSql vs Apache Drill vs HiveQL (Performance Comparison)

Submitted to:

Dr. Tariq Mehmood

Submitted by:

Shiza Azam

25557

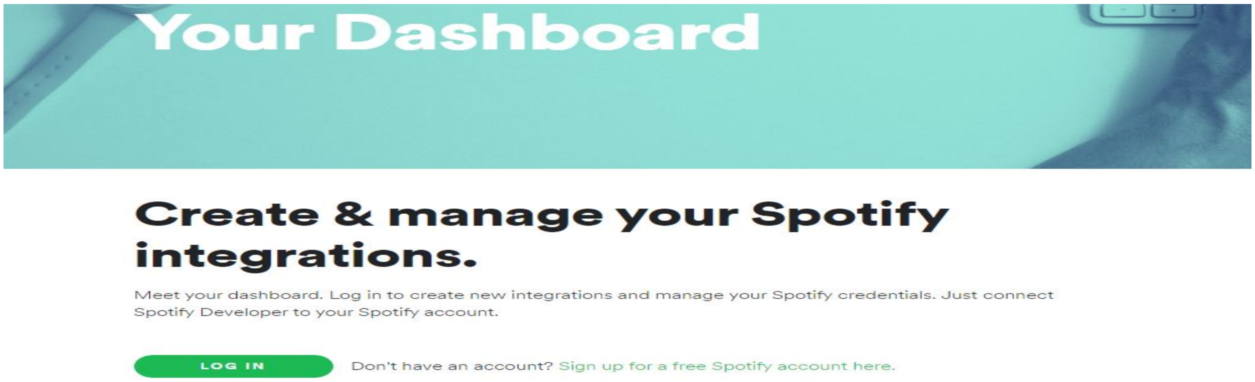
Use Case:

Imagine wanting to get into a music industry and believe that we might have a skill to spot great talent and promote them to become out next star, and for that we need to be familiar with the real music industry facts:

We might want to know:

1- Which tracks are the most popular among users?
2- Which genre's artist has higher no of followers?
3- What kind of songs are getting popular in particular year?
4- How the song trend is being changed over the years?
5- Which artists gained popularity in a 2021 over a trending genre?

For this purpose, we have used real dataset from the industry Spotify. The dataset has been extracted from their website using Spotify API. Spotify is one of the most popular music streaming platform. The have provided an API for developers to use their database to analyze data and building applications. The developer side can be access through this website [https://developer.spotify.com/].



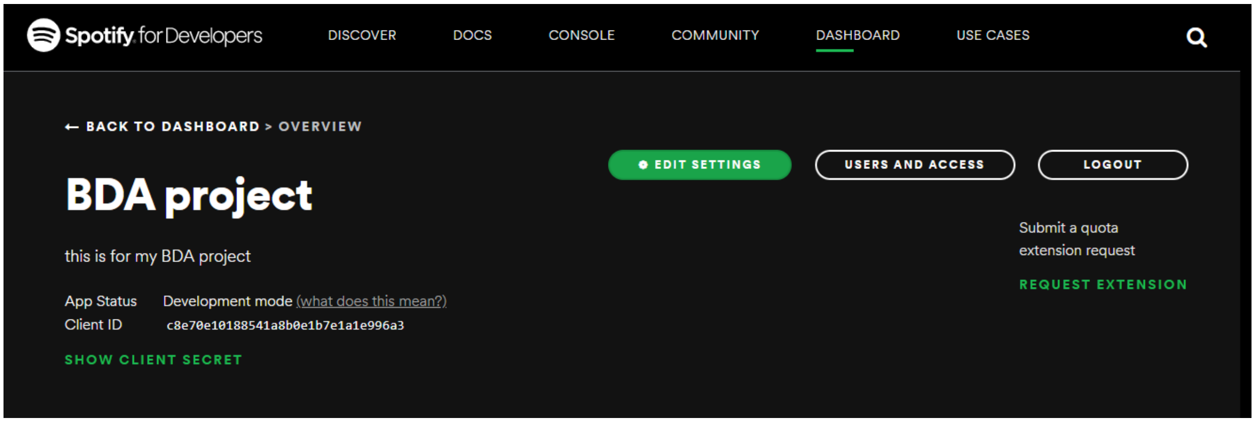
Your Dashboard

Create & manage your Spotify integrations.

Meet your dashboard. Log in to create new integrations and manage your Spotify credentials. Just connect Spotify Developer to your Spotify account.

Don't have an account? Sign up for a free Spotify account here.

Login to the dashboard of "developer.spotify.com". Once logged in, select into create client id and follow the instructions given in the website.



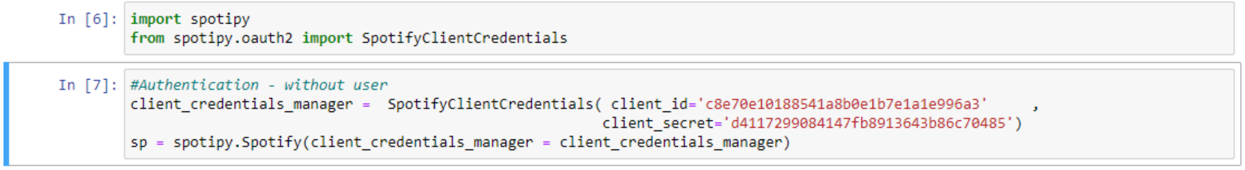
BDA project

this is for my BDA project

App Status Development mode (what does this mean?)
Client ID c8e70e10188541a8b0e1b7e1a1e996a3

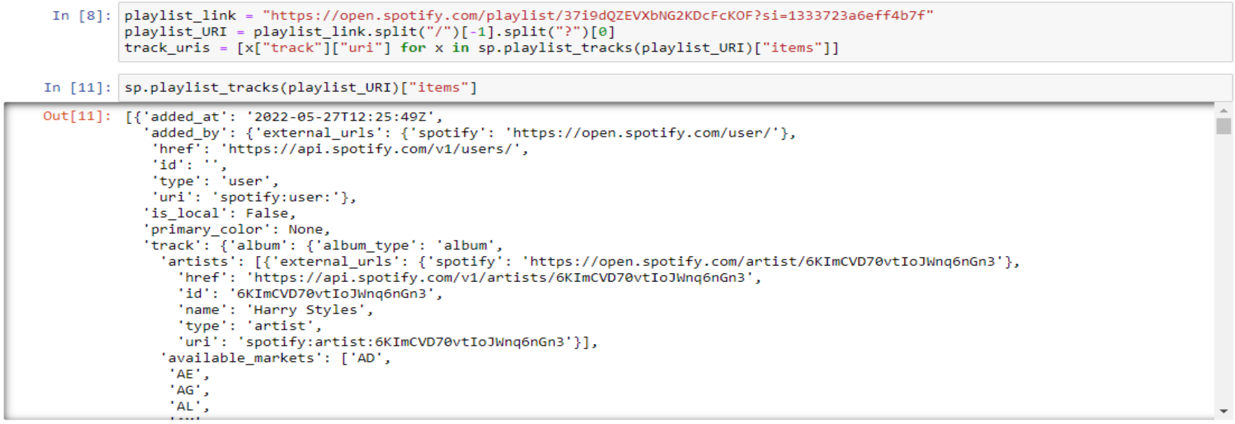
To Access the Spotify's authorized data, we need to create client ID, client secret and Spotify object to access the API.

First install Spotify library in your python environment via **pip install Spotify** command, and run the below command in your python notebook. Use your client id, and secret id from developer credentials website.



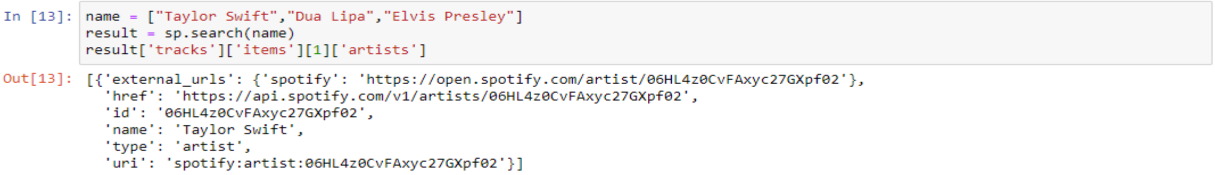
```
In [6]: import spotipy
        from spotipy.oauth2 import SpotifyClientCredentials

In [7]: #Authentication - without user
        client_credentials_manager =  SpotifyClientCredentials( client_id='c8e70e10188541a8b0e1b7e1a1e996a3'     ,
                                                                 client_secret='d4117299084147fb8913643b86c70485')
        sp = spotipy.Spotify(client_credentials_manager = client_credentials_manager)
```

Now search for a particular albums to see what songs are available in it,



```
In [8]: playlist_link = "https://open.spotify.com/playlist/37i9dQZEVXbNG2KDcFcKOF?si=1333723a6eff4b7f"
        playlist_URI = playlist_link.split("/")[-1].split("?")[0]
        track_uris = [x["track"]["uri"] for x in sp.playlist_tracks(playlist_URI)["items"]]

In [11]: sp.playlist_tracks(playlist_URI)["items"]

Out[11]: [{'added_at': '2022-05-27T12:25:49Z',
           'added_by': {'external_urls': {'spotify': 'https://open.spotify.com/user/'},
            'href': 'https://api.spotify.com/v1/users/',
            'id': '',
            'type': 'user',
            'uri': 'spotify:user:'},
           'is_local': False,
           'primary_color': None,
           'track': {'album': {'album_type': 'album',
             'artists': [{'external_urls': {'spotify': 'https://open.spotify.com/artist/6KImCVD70vtIoJWnq6nGn3'},
               'href': 'https://api.spotify.com/v1/artists/6KImCVD70vtIoJWnq6nGn3',
               'id': '6KImCVD70vtIoJWnq6nGn3',
               'name': 'Harry Styles',
               'type': 'artist',
               'uri': 'spotify:artist:6KImCVD70vtIoJWnq6nGn3'}],
             'available_markets': ['AD',
              'AE',
              'AG',
              'AL',
```

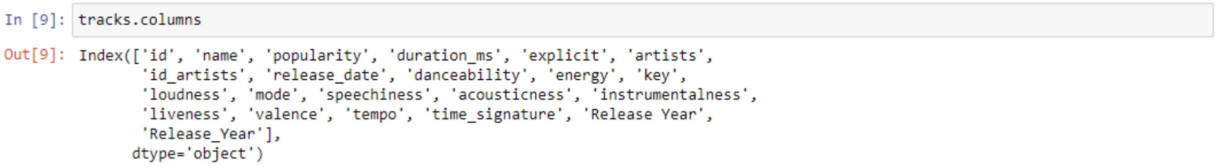
We can also search via the artist name:



```
In [13]: name = ["Taylor Swift","Dua Lipa","Elvis Presley"]
         result = sp.search(name)
         result['tracks']['items'][1]['artists']

Out[13]: [{'external_urls': {'spotify': 'https://open.spotify.com/artist/06HL4z0CvFAxyc27GXpf02'},
           'href': 'https://api.spotify.com/v1/artists/06HL4z0CvFAxyc27GXpf02',
           'id': '06HL4z0CvFAxyc27GXpf02',
           'name': 'Taylor Swift',
           'type': 'artist',
           'uri': 'spotify:artist:06HL4z0CvFAxyc27GXpf02'}]
```

In our Project, we have extracted data through two ways, by artist and tracks.

In our track data, we have these columns:



```
In [9]: tracks.columns

Out[9]: Index(['id', 'name', 'popularity', 'duration_ms', 'explicit', 'artists',
               'id_artists', 'release_date', 'danceability', 'energy', 'key',
               'loudness', 'mode', 'speechiness', 'acousticness', 'instrumentalness',
               'liveness', 'valence', 'tempo', 'time_signature', 'Release Year',
               'Release_Year'],
              dtype='object')
```

This dataset contains 681895 streamed songs from 1995 to 2022. It also contains the metadata about each song and artist

The columns in the tracks included are:

track_id: unique identifier for each track

artists: name of the main artist

name: track name

explicit: true or false if contains explicit content

artist_id : unique identifier for each artist

all_artists : a list of all artist names that appeared on the track

all_artists_ids : a list of all artist ids that appeared on the track

acousticness: confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic

danceability: describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. A value of 0.0 is least danceable and 1.0 is most danceable.

duration: duration of track in milliseconds

energy: measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy.

instrumentalness: Predicts whether a track contains no vocals. “Ooh” and “aah” sounds are treated as instrumental in this context.

key: The estimated overall key of the track. Integers map to pitches using standard Pitch Class notation . E.g. 0 = C, 1 = C♯/D♭, 2 = D, and so on. If no key was detected, the value is -1.

liveness: Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live.

loudness: The overall loudness of a track in decibels (dB). Values typical range between -60 and 0 db.

mode: Mode indicates the modality (major or minor) of a track, the type of scale from which its melodic content is derived. Major is represented by 1 and minor is 0.

speechiness: Speechiness detects the presence of spoken words in a track.

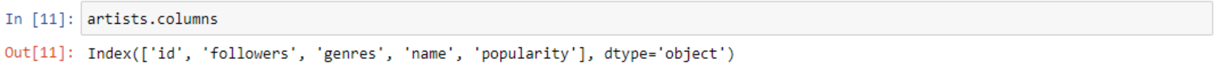
tempo: The overall estimated tempo of a track in beats per minute (BPM)

time_signature: An estimated overall time signature of a track. The time signature (meter) is a notational convention to specify how many beats are in each bar (or measure).

valence: A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track.

release_date : when the album was released

And for artist dataset, it contains the metadata for each artist in track. The dataset includes 1134430 rows and 5 columns included in the dataset are:



```
In [11]: artists.columns
Out[11]: Index(['id', 'followers', 'genres', 'name', 'popularity'], dtype='object')
```

id : unique identifier for each artist

name : artist name

popularity : [0-100] where 100 is the most popular. Score as of updated_at date.

followers : total number of spotify followers as of the updated_at date.

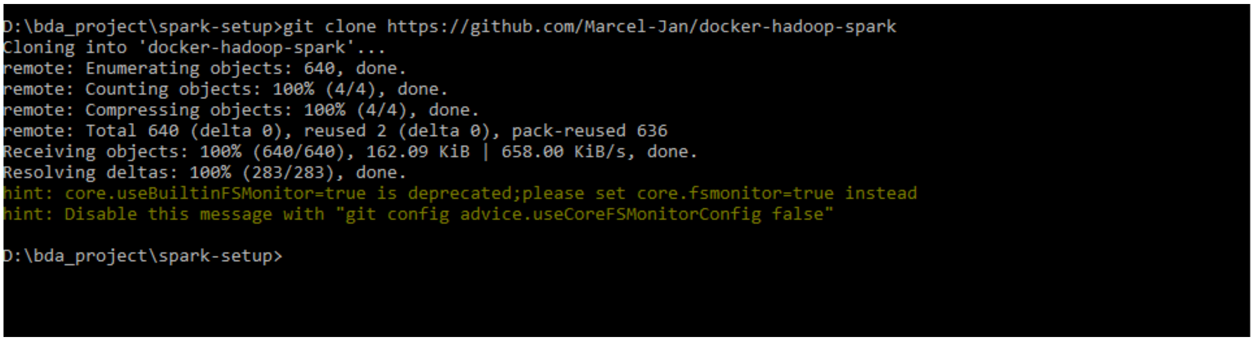
genres : artist all genres

The purpose of this project is to compare the performance of all SparkSQL- HiveQL and Apache Drill. For the purpose, we have used Docker multi container environment to setup SparkSQL and HiveQL, and a separate Docker environment for Drill. The dataset file was stored and retrieved from HDFS for analyzing purpose.

Spark-SQL

SparkSQL is a module of spark that is used for structure processing and acts as a distributed SQL query engine.

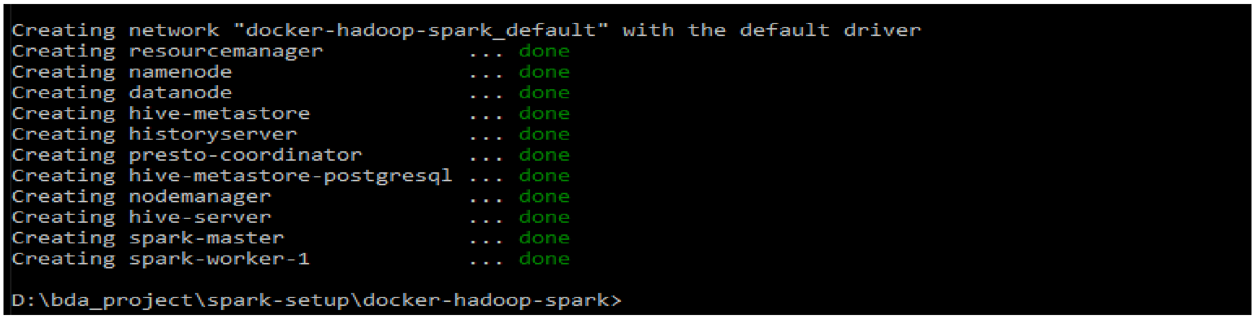
The step by step process has been shown to configure the Spark and Hive multi container environment.



```
D:\bda_project\spark-setup>git clone https://github.com/Marcel-Jan/docker-hadoop-spark
Cloning into 'docker-hadoop-spark'...
remote: Enumerating objects: 640, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 640 (delta 0), reused 2 (delta 0), pack-reused 636
Receiving objects: 100% (640/640), 162.09 KiB | 658.00 KiB/s, done.
Resolving deltas: 100% (283/283), done.
hint: core.useBuiltinFSMonitor=true is deprecated;please set core.fsmonitor=true instead
hint: Disable this message with "git config advice.useCoreFSMonitorConfig false"

D:\bda_project\spark-setup>
```

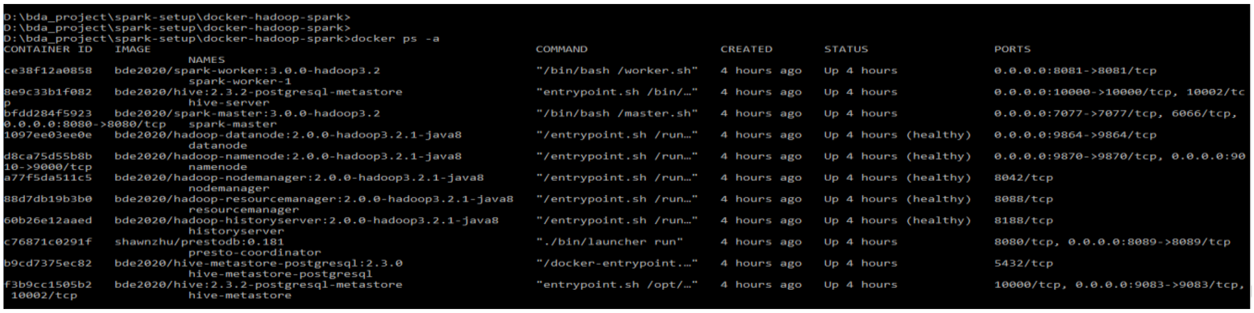
First clone the project using the above command, and after cloning go into directory and run command,

docker-compose up –d ,it will start installing all the required images for this environment. After installing all the images the output should look like this:



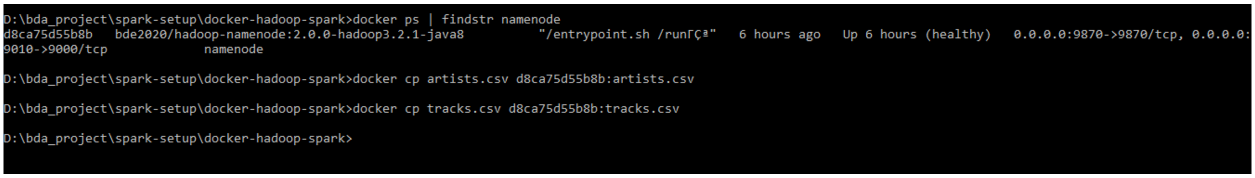
```
Creating network "docker-hadoop-spark_default" with the default driver
Creating resourcemanager            ... done
Creating namenode                   ... done
Creating datanode                   ... done
Creating hive-metastore             ... done
Creating historyserver              ... done
Creating presto-coordinator         ... done
Creating hive-metastore-postgresql  ... done
Creating nodemanager                ... done
Creating hive-server                ... done
Creating spark-master               ... done
Creating spark-worker-1             ... done

D:\bda_project\spark-setup\docker-hadoop-spark>
```

Run the **docker ps –a**, and the output should look like this. Meaning all the containers are running in a detached mode.



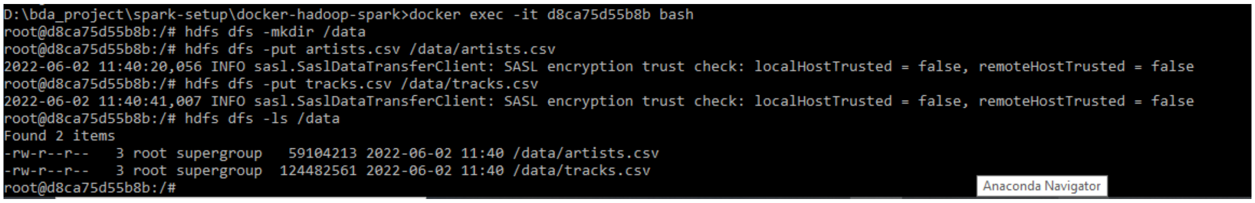
```
D:\bda_project\spark-setup\docker-hadoop-spark>
D:\bda_project\spark-setup\docker-hadoop-spark>
D:\bda_project\spark-setup\docker-hadoop-spark>docker ps -a
CONTAINER ID   IMAGE                                                    COMMAND                   CREATED         STATUS                    PORTS
                         NAMES
ce38f12a0858   bde2020/spark-worker:3.0.0-hadoop3.2                     "/bin/bash /worker.sh"    4 hours ago     Up 4 hours                0.0.0.0:8081->8081/tcp
                         spark-worker-1
8e9c33b1f082   bde2020/hive:2.3.2-postgresql-metastore                  "entrypoint.sh /bin/…"    4 hours ago     Up 4 hours                0.0.0.0:10000->10000/tcp, 10002/tc
p                        hive-server
bfdd284f5923   bde2020/spark-master:3.0.0-hadoop3.2                     "/bin/bash /master.sh"    4 hours ago     Up 4 hours                0.0.0.0:7077->7077/tcp, 6066/tcp,
0.0.0.0:8080->8080/tcp   spark-master
1097ee03ee0e   bde2020/hadoop-datanode:2.0.0-hadoop3.2.1-java8          "/entrypoint.sh /run…"    4 hours ago     Up 4 hours (healthy)      0.0.0.0:9864->9864/tcp
                         datanode
d8ca75d55b8b   bde2020/hadoop-namenode:2.0.0-hadoop3.2.1-java8          "/entrypoint.sh /run…"    4 hours ago     Up 4 hours (healthy)      0.0.0.0:9870->9870/tcp, 0.0.0.0:90
10->9000/tcp             namenode
a77f5da511c5   bde2020/hadoop-nodemanager:2.0.0-hadoop3.2.1-java8       "/entrypoint.sh /run…"    4 hours ago     Up 4 hours (healthy)      8042/tcp
                         nodemanager
88d7db19b3b0   bde2020/hadoop-resourcemanager:2.0.0-hadoop3.2.1-java8   "/entrypoint.sh /run…"    4 hours ago     Up 4 hours (healthy)      8088/tcp
                         resourcemanager
60b26e12aaed   bde2020/hadoop-historyserver:2.0.0-hadoop3.2.1-java8     "/entrypoint.sh /run…"    4 hours ago     Up 4 hours (healthy)      8188/tcp
                         historyserver
c76871c0291f   shawnzhu/prestodb:0.181                                  "./bin/launcher run"      4 hours ago     Up 4 hours                8080/tcp, 0.0.0.0:8089->8089/tcp
                         presto-coordinator
b9cd7375ec82   bde2020/hive-metastore-postgresql:2.3.0                  "/docker-entrypoint.…"    4 hours ago     Up 4 hours                5432/tcp
                         hive-metastore-postgresql
f3b9cc1505b2   bde2020/hive:2.3.2-postgresql-metastore                  "entrypoint.sh /opt/…"    4 hours ago     Up 4 hours                10000/tcp, 0.0.0.0:9083->9083/tcp,
 10002/tcp               hive-metastore
```

Next step is to store the data into Hadoop file system, for that we first need to copy all the required files into the namenode. This can be achieved via the **docker cp** command shown below



```
D:\bda_project\spark-setup\docker-hadoop-spark>docker ps | findstr namenode
d8ca75d55b8b   bde2020/hadoop-namenode:2.0.0-hadoop3.2.1-java8          "/entrypoint.sh /runΓÇª"   6 hours ago   Up 6 hours (healthy)   0.0.0.0:9870->9870/tcp, 0.0.0.0:
9010->9000/tcp             namenode

D:\bda_project\spark-setup\docker-hadoop-spark>docker cp artists.csv d8ca75d55b8b:artists.csv

D:\bda_project\spark-setup\docker-hadoop-spark>docker cp tracks.csv d8ca75d55b8b:tracks.csv

D:\bda_project\spark-setup\docker-hadoop-spark>
```

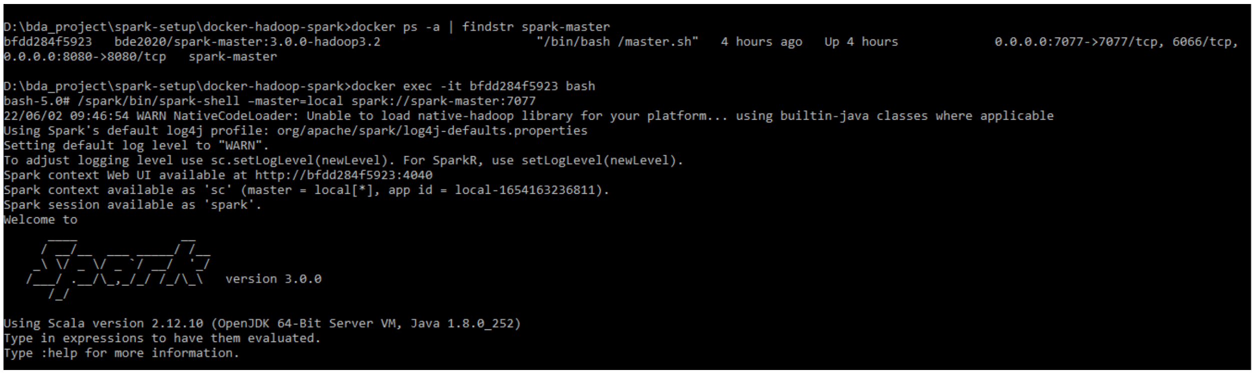
Now to put the files into hdfs, we need to go into the name-node container, by the **docker execute command** and put the files into hdfs via tha **hdfs dfs –put** command shown below.



```
D:\bda_project\spark-setup\docker-hadoop-spark>docker exec -it d8ca75d55b8b bash
root@d8ca75d55b8b:/# hdfs dfs -mkdir /data
root@d8ca75d55b8b:/# hdfs dfs -put artists.csv /data/artists.csv
2022-06-02 11:40:20,056 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
root@d8ca75d55b8b:/# hdfs dfs -put tracks.csv /data/tracks.csv
2022-06-02 11:40:41,007 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
root@d8ca75d55b8b:/# hdfs dfs -ls /data
Found 2 items
-rw-r--r--   3 root supergroup   59104213 2022-06-02 11:40 /data/artists.csv
-rw-r--r--   3 root supergroup  124482561 2022-06-02 11:40 /data/tracks.csv
root@d8ca75d55b8b:/#
```

Once the file is loaded into the hdfs, now we can access in hdfs in the spark container. To go into the spark bash, run the command **docker execute –it spark-containerID bash** and then run the command to launch the spark:

/spark/bin/spark-shell –master=local spark://spark-master:7077

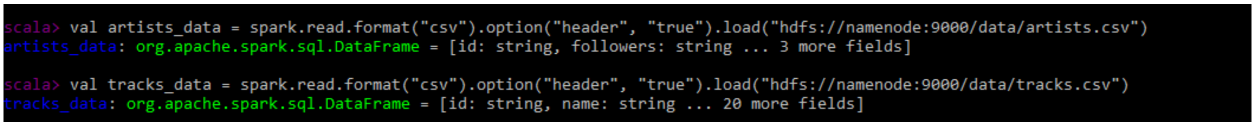


```
D:\bda_project\spark-setup\docker-hadoop-spark>docker ps -a | findstr spark-master
bfdd284f5923   bde2020/spark-master:3.0.0-hadoop3.2                      "/bin/bash /master.sh"   4 hours ago   Up 4 hours             0.0.0.0:7077->7077/tcp, 6066/tcp,
0.0.0.0:8080->8080/tcp   spark-master

D:\bda_project\spark-setup\docker-hadoop-spark>docker exec -it bfdd284f5923 bash
bash-5.0# /spark/bin/spark-shell -master=local spark://spark-master:7077
22/06/02 09:46:54 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Spark context Web UI available at http://bfdd284f5923:4040
Spark context available as 'sc' (master = local[*], app id = local-1654163236811).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.0.0
      /_/

Using Scala version 2.12.10 (OpenJDK 64-Bit Server VM, Java 1.8.0_252)
Type in expressions to have them evaluated.
Type :help for more information.
```

Now we can successfully run our spark queries in Scala.

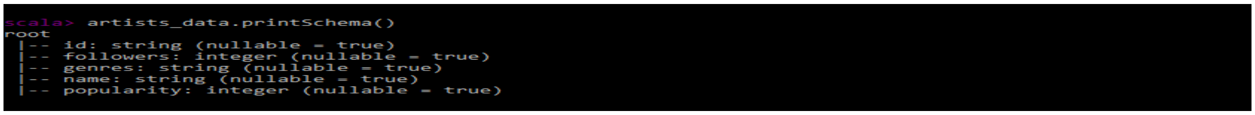
Spark-Queries:

First we need to load the data from the hdfs that can be accessed through **hdfs://namenode:9000/filepath**

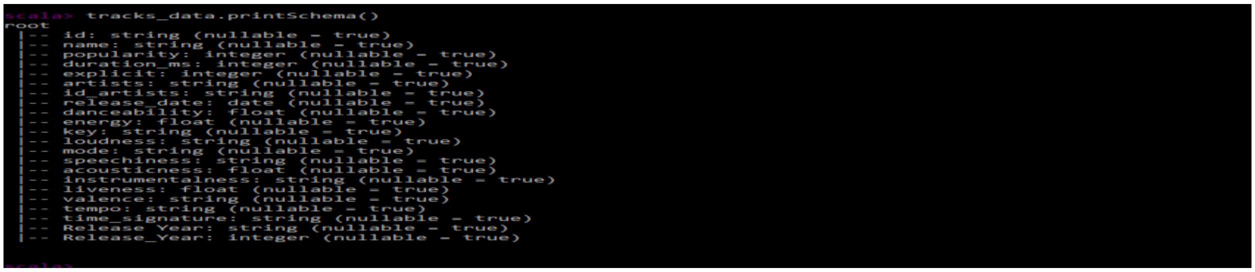


```
scala> val artists_data = spark.read.format("csv").option("header", "true").load("hdfs://namenode:9000/data/artists.csv")
artists_data: org.apache.spark.sql.DataFrame = [id: string, followers: string ... 3 more fields]

scala> val tracks_data = spark.read.format("csv").option("header", "true").load("hdfs://namenode:9000/data/tracks.csv")
tracks_data: org.apache.spark.sql.DataFrame = [id: string, name: string ... 20 more fields]
```

After loading the file, printing the schema of both files.

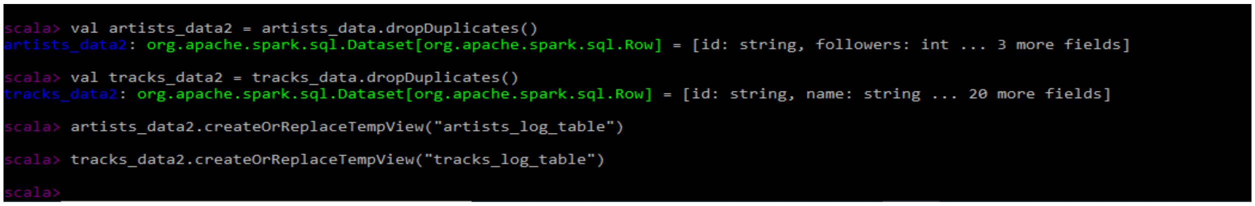


```
scala> artists_data.printSchema()
root
 |-- id: string (nullable = true)
 |-- followers: integer (nullable = true)
 |-- genres: string (nullable = true)
 |-- name: string (nullable = true)
 |-- popularity: integer (nullable = true)
```



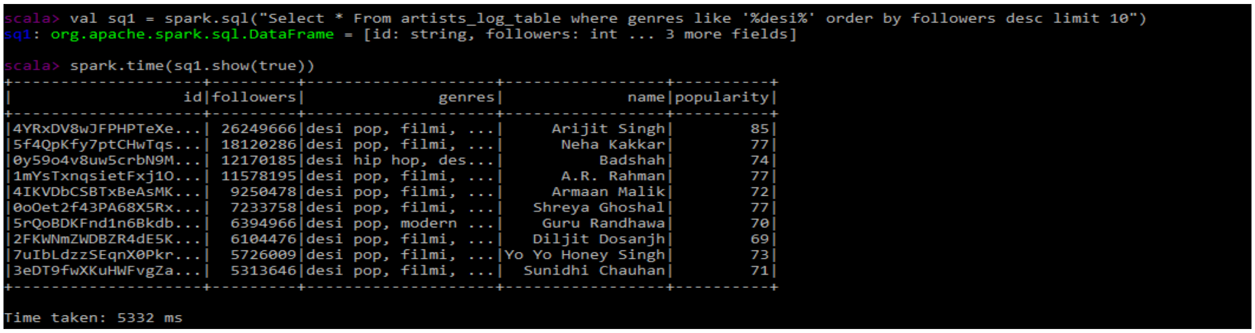
```
scala> tracks_data.printSchema()
root
 |-- id: string (nullable = true)
 |-- name: string (nullable = true)
 |-- popularity: integer (nullable = true)
 |-- duration_ms: integer (nullable = true)
 |-- explicit: integer (nullable = true)
 |-- artists: string (nullable = true)
 |-- id_artists: string (nullable = true)
 |-- release_date: date (nullable = true)
 |-- danceability: float (nullable = true)
 |-- energy: float (nullable = true)
 |-- key: string (nullable = true)
 |-- loudness: string (nullable = true)
 |-- mode: string (nullable = true)
 |-- speechiness: string (nullable = true)
 |-- acousticness: float (nullable = true)
 |-- instrumentalness: string (nullable = true)
 |-- liveness: float (nullable = true)
 |-- valence: string (nullable = true)
 |-- tempo: string (nullable = true)
 |-- time_signature: string (nullable = true)
 |-- Release Year: string (nullable = true)
 |-- Release_Year: integer (nullable = true)

scala>
```

To drop duplicate rows in spark, we can use dropDuplicates() function. In Spark, we would have to create a temporary view of the dataframe, this effectively creates a temporary SQL table from the dataframe, enabling you to make the same queries that you write, if you are using the database like Postgres or MySQL. The below command will create the view if does not already exist or will update the existing view if the underlying data is changed.



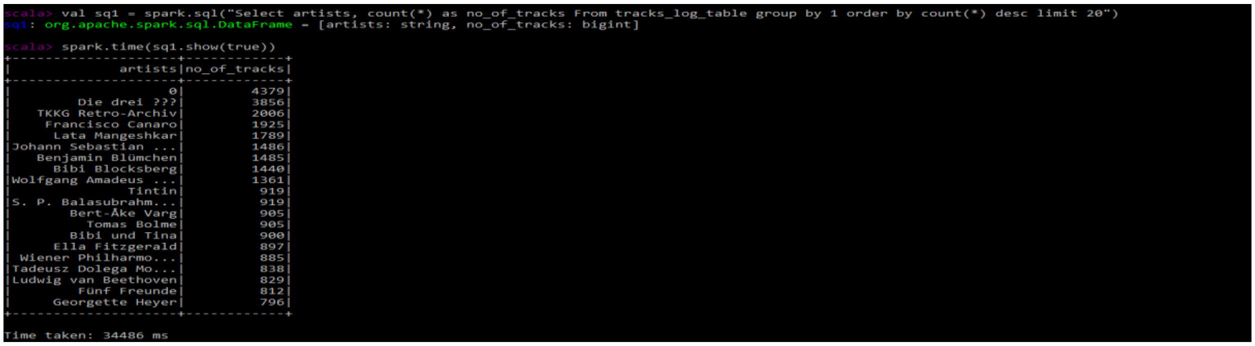
```
scala> val artists_data2 = artists_data.dropDuplicates()
artists_data2: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [id: string, followers: int ... 3 more fields]

scala> val tracks_data2 = tracks_data.dropDuplicates()
tracks_data2: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [id: string, name: string ... 20 more fields]

scala> artists_data2.createOrReplaceTempView("artists_log_table")

scala> tracks_data2.createOrReplaceTempView("tracks_log_table")

scala>
```

1- Select the top 10 artists who has the highest following in desi genre



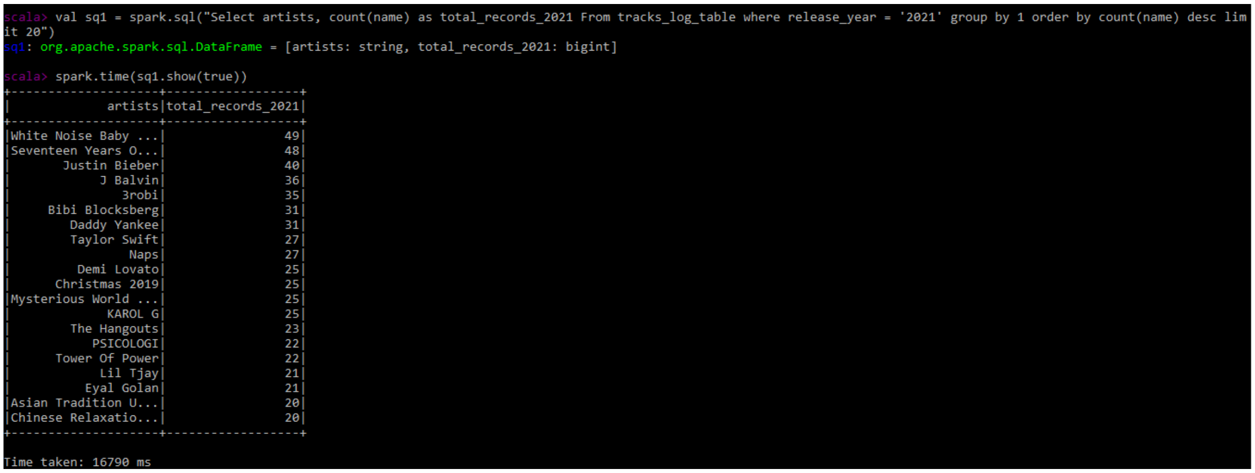
```
scala> val sq1 = spark.sql("Select * From artists_log_table where genres like '%desi%' order by followers desc limit 10")
sq1: org.apache.spark.sql.DataFrame = [id: string, followers: int ... 3 more fields]

scala> spark.time(sq1.show(true))
+--------------------+---------+--------------------+-----------------+----------+
|                  id|followers|              genres|             name|popularity|
+--------------------+---------+--------------------+-----------------+----------+
|4YRxDV8wJFPHPTeXe...| 26249666|desi pop, filmi, ...|     Arijit Singh|        85|
|5f4QpKfy7ptCHwTqs...| 18120286|desi pop, filmi, ...|      Neha Kakkar|        77|
|0y59o4v8uw5crbN9M...| 12170185|desi hip hop, des...|          Badshah|        74|
|1mYsTxnqsietFxj1O...| 11578195|desi pop, filmi, ...|      A.R. Rahman|        77|
|4IKVDbCSBTxBeAsMK...|  9250478|desi pop, filmi, ...|     Armaan Malik|        72|
|0oOet2f43PA68X5Rx...|  7233758|desi pop, filmi, ...|   Shreya Ghoshal|        77|
|5rQoBDKFnd1n6Bkdb...|  6394966|desi pop, modern ...|    Guru Randhawa|        70|
|2FKWNmZWDBZR4dE5K...|  6104476|desi pop, filmi, ...|   Diljit Dosanjh|        69|
|7uIbLdzzSEqnX0Pkr...|  5726009|desi pop, filmi, ...|Yo Yo Honey Singh|        73|
|3eDT9fwXKuHWFvgZa...|  5313646|desi pop, filmi, ...|  Sunidhi Chauhan|        71|
+--------------------+---------+--------------------+-----------------+----------+

Time taken: 5332 ms
```

2- Select the top 20 artists who has the highest no of tracks.



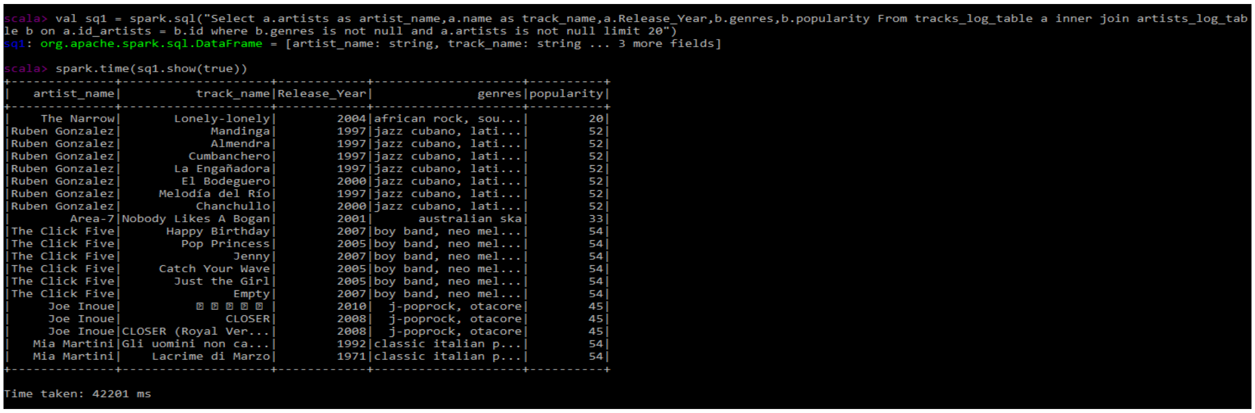
```
scala> val sq1 = spark.sql("Select artists, count(*) as no_of_tracks From tracks_log_table group by 1 order by count(*) desc limit 20")
sq1: org.apache.spark.sql.DataFrame = [artists: string, no_of_tracks: bigint]

scala> spark.time(sq1.show(true))
+--------------------+------------+
|             artists|no_of_tracks|
+--------------------+------------+
|                   0|        4379|
|        Die drei ???|        3856|
|   TKKG Retro-Archiv|        2006|
|    Francisco Canaro|        1925|
|     Lata Mangeshkar|        1789|
|Johann Sebastian ...|        1486|
|   Benjamin Blümchen|        1485|
|     Bibi Blocksberg|        1440|
|Wolfgang Amadeus ...|        1361|
|              Tintin|         919|
|S. P. Balasubrahm...|         919|
|      Bert-Åke Varg|         905|
|         Tomas Bolme|         905|
|       Bibi und Tina|         900|
|     Ella Fitzgerald|         897|
| Wiener Philharmo...|         885|
|Tadeusz Dolega Mo...|         838|
|Ludwig van Beethoven|         829|
|        Fünf Freunde|         812|
|     Georgette Heyer|         796|
+--------------------+------------+

Time taken: 34486 ms
```

3- Select the top 20 artists who has the highest no of track in 2021



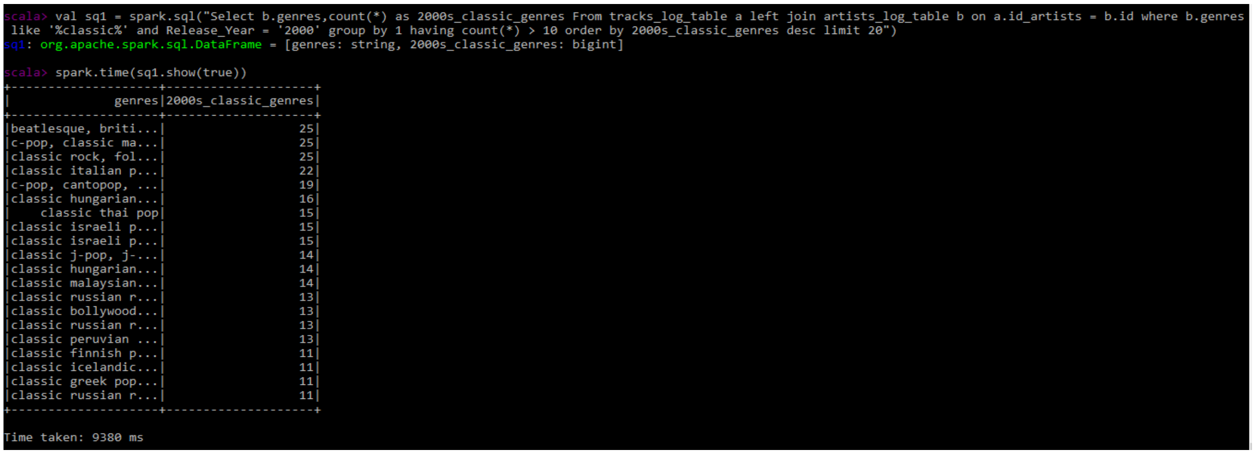
```
scala> val sq1 = spark.sql("Select artists, count(name) as total_records_2021 From tracks_log_table where release_year = '2021' group by 1 order by count(name) desc lim
it 20")
sq1: org.apache.spark.sql.DataFrame = [artists: string, total_records_2021: bigint]

scala> spark.time(sq1.show(true))
+--------------------+------------------+
|             artists|total_records_2021|
+--------------------+------------------+
|White Noise Baby ...|                49|
|Seventeen Years O...|                48|
|       Justin Bieber|                40|
|            J Balvin|                36|
|               3robi|                35|
|     Bibi Blocksberg|                31|
|        Daddy Yankee|                31|
|        Taylor Swift|                27|
|                Naps|                27|
|         Demi Lovato|                25|
|      Christmas 2019|                25|
|Mysterious World ...|                25|
|             KAROL G|                25|
|        The Hangouts|                23|
|           PSICOLOGI|                22|
|      Tower Of Power|                22|
|            Lil Tjay|                21|
|          Eyal Golan|                21|
|Asian Tradition U...|                20|
|Chinese Relaxatio...|                20|
+--------------------+------------------+

Time taken: 16790 ms
```

4- Show the artist track with their respective genres and its popularity.



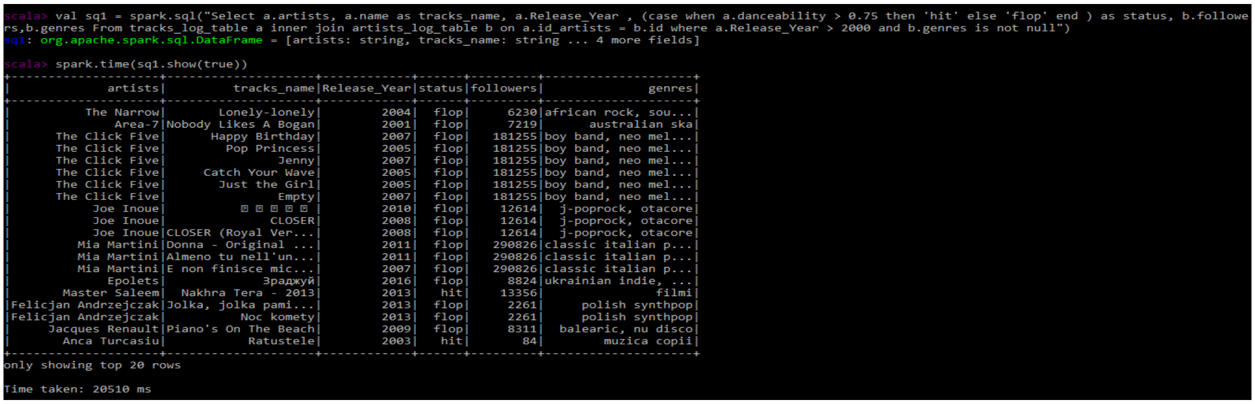
```
scala> val sq1 = spark.sql("Select a.artists as artist_name,a.name as track_name,a.Release_Year,b.genres,b.popularity From tracks_log_table a inner join artists_log_tab
le b on a.id_artists = b.id where b.genres is not null and a.artists is not null limit 20")
sq1: org.apache.spark.sql.DataFrame = [artist_name: string, track_name: string ... 3 more fields]

scala> spark.time(sq1.show(true))
+--------------+--------------------+------------+--------------------+----------+
|   artist_name|          track_name|Release_Year|              genres|popularity|
+--------------+--------------------+------------+--------------------+----------+
|    The Narrow|       Lonely-lonely|        2004|african rock, sou...|        20|
|Ruben Gonzalez|            Mandinga|        1997|jazz cubano, lati...|        52|
|Ruben Gonzalez|            Almendra|        1997|jazz cubano, lati...|        52|
|Ruben Gonzalez|         Cumbanchero|        1997|jazz cubano, lati...|        52|
|Ruben Gonzalez|        La Engañadora|        1997|jazz cubano, lati...|        52|
|Ruben Gonzalez|         El Bodeguero|        2000|jazz cubano, lati...|        52|
|Ruben Gonzalez|      Melodía del Río|        1997|jazz cubano, lati...|        52|
|Ruben Gonzalez|          Chanchullo|        2000|jazz cubano, lati...|        52|
|        Area-7|Nobody Likes A Bogan|        2001|      australian ska|        33|
|The Click Five|      Happy Birthday|        2007|boy band, neo mel...|        54|
|The Click Five|        Pop Princess|        2005|boy band, neo mel...|        54|
|The Click Five|               Jenny|        2007|boy band, neo mel...|        54|
|The Click Five|      Catch Your Wave|        2005|boy band, neo mel...|        54|
|The Click Five|        Just the Girl|        2005|boy band, neo mel...|        54|
|The Click Five|               Empty|        2007|boy band, neo mel...|        54|
|     Joe Inoue|          ⊠ ⊠ ⊠ ⊠ ⊠ |        2010|  j-poprock, otacore|        45|
|     Joe Inoue|              CLOSER|        2008|  j-poprock, otacore|        45|
|     Joe Inoue|CLOSER (Royal Ver...|        2008|  j-poprock, otacore|        45|
|   Mia Martini|Gli uomini non ca...|        1992|classic italian p...|        54|
|   Mia Martini|     Lacrime di Marzo|        1971|classic italian p...|        54|
+--------------+--------------------+------------+--------------------+----------+

Time taken: 42201 ms
```

5- Show the top 20 genres which has the highest no of songs recorded in 2000.



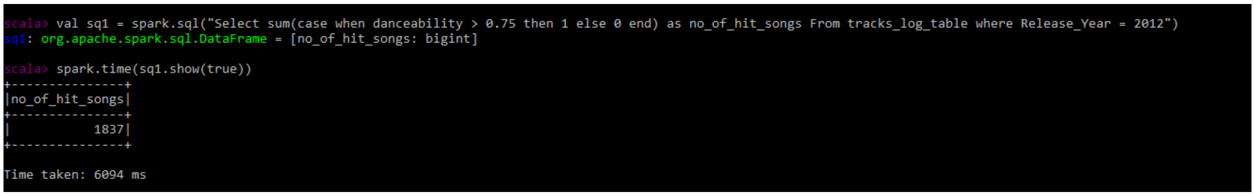
```
scala> val sq1 = spark.sql("Select b.genres,count(*) as 2000s_classic_genres From tracks_log_table a left join artists_log_table b on a.id_artists = b.id where b.genres
 like '%classic%' and Release_Year = '2000' group by 1 having count(*) > 10 order by 2000s_classic_genres desc limit 20")
sq1: org.apache.spark.sql.DataFrame = [genres: string, 2000s_classic_genres: bigint]

scala> spark.time(sq1.show(true))
+--------------------+--------------------+
|              genres|2000s_classic_genres|
+--------------------+--------------------+
|beatlesque, briti...|                  25|
|c-pop, classic ma...|                  25|
|classic rock, fol...|                  25|
|classic italian p...|                  22|
|c-pop, cantopop, ...|                  19|
|classic hungarian...|                  16|
|    classic thai pop|                  15|
|classic israeli p...|                  15|
|classic israeli p...|                  15|
|classic j-pop, j-...|                  14|
|classic hungarian...|                  14|
|classic malaysian...|                  14|
|classic russian r...|                  13|
|classic bollywood...|                  13|
|classic russian r...|                  13|
|classic peruvian ...|                  13|
|classic finnish p...|                  11|
|classic icelandic...|                  11|
|classic greek pop...|                  11|
|classic russian r...|                  11|
+--------------------+--------------------+

Time taken: 9380 ms
```

6- On the basis of danceability, show the top 20 hit songs from 20001.



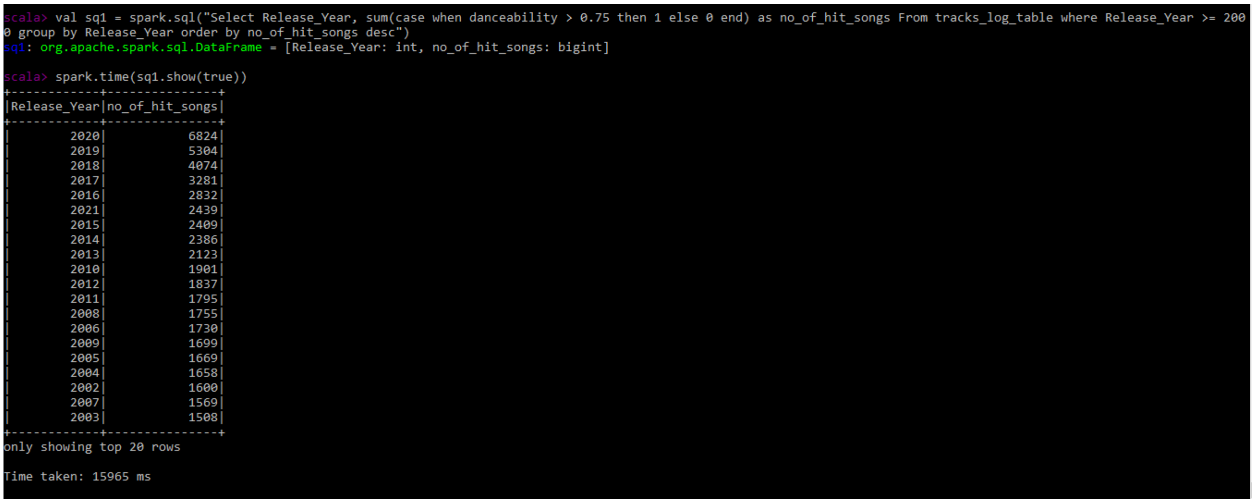
```
scala> val sq1 = spark.sql("Select a.artists, a.name as tracks_name, a.Release_Year , (case when a.danceability > 0.75 then 'hit' else 'flop' end ) as status, b.followe
rs,b.genres From tracks_log_table a inner join artists_log_table b on a.id_artists = b.id where a.Release_Year > 2000 and b.genres is not null")
sq1: org.apache.spark.sql.DataFrame = [artists: string, tracks_name: string ... 4 more fields]

scala> spark.time(sq1.show(true))
+--------------------+--------------------+------------+------+---------+--------------------+
|             artists|         tracks_name|Release_Year|status|followers|              genres|
+--------------------+--------------------+------------+------+---------+--------------------+
|          The Narrow|       Lonely-lonely|        2004|  flop|     6230|african rock, sou...|
|              Area-7|Nobody Likes A Bogan|        2001|  flop|     7219|      australian ska|
|      The Click Five|      Happy Birthday|        2007|  flop|   181255|boy band, neo mel...|
|      The Click Five|        Pop Princess|        2005|  flop|   181255|boy band, neo mel...|
|      The Click Five|               Jenny|        2007|  flop|   181255|boy band, neo mel...|
|      The Click Five|     Catch Your Wave|        2005|  flop|   181255|boy band, neo mel...|
|      The Click Five|       Just the Girl|        2005|  flop|   181255|boy band, neo mel...|
|      The Click Five|               Empty|        2007|  flop|   181255|boy band, neo mel...|
|           Joe Inoue|          □ □ □ □ □ |        2010|  flop|    12614|  j-poprock, otacore|
|           Joe Inoue|              CLOSER|        2008|  flop|    12614|  j-poprock, otacore|
|           Joe Inoue|CLOSER (Royal Ver...|        2008|  flop|    12614|  j-poprock, otacore|
|         Mia Martini|Donna - Original ...|        2011|  flop|   290826|classic italian p...|
|         Mia Martini|Almeno tu nell'un...|        2011|  flop|   290826|classic italian p...|
|         Mia Martini|E non finisce mic...|        2007|  flop|   290826|classic italian p...|
|             Epolets|             Зраджуй|        2016|  flop|     8824|ukrainian indie, ...|
|       Master Saleem|  Nakhra Tera - 2013|        2013|   hit|    13356|               filmi|
|Felicjan Andrzejczak|Jolka, jolka pami...|        2013|  flop|     2261|     polish synthpop|
|Felicjan Andrzejczak|          Noc komety|        2013|  flop|     2261|     polish synthpop|
|     Jacques Renault|Piano's On The Beach|        2009|  flop|     8311|  balearic, nu disco|
|       Anca Turcasiu|           Ratustele|        2003|   hit|       84|        muzica copii|
+--------------------+--------------------+------------+------+---------+--------------------+
only showing top 20 rows

Time taken: 20510 ms
```

7- On the basis of danceability, show the total hit songs in 2012.



```
scala> val sq1 = spark.sql("Select sum(case when danceability > 0.75 then 1 else 0 end) as no_of_hit_songs From tracks_log_table where Release_Year = 2012")
sq1: org.apache.spark.sql.DataFrame = [no_of_hit_songs: bigint]

scala> spark.time(sq1.show(true))
+---------------+
|no_of_hit_songs|
+---------------+
|           1837|
+---------------+

Time taken: 6094 ms
```

8- On the basis of danceability, show the total hit songs for years greater than 2000.



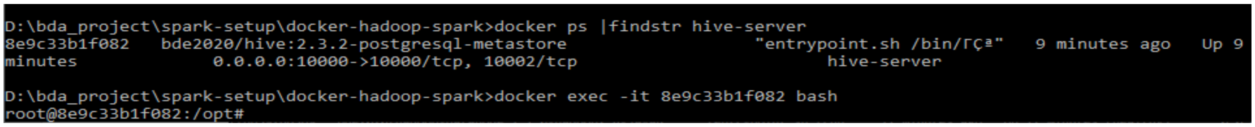
```
scala> val sq1 = spark.sql("Select Release_Year, sum(case when danceability > 0.75 then 1 else 0 end) as no_of_hit_songs From tracks_log_table where Release_Year >= 200
0 group by Release_Year order by no_of_hit_songs desc")
sq1: org.apache.spark.sql.DataFrame = [Release_Year: int, no_of_hit_songs: bigint]

scala> spark.time(sq1.show(true))
+------------+---------------+
|Release_Year|no_of_hit_songs|
+------------+---------------+
|        2020|           6824|
|        2019|           5304|
|        2018|           4074|
|        2017|           3281|
|        2016|           2832|
|        2021|           2439|
|        2015|           2409|
|        2014|           2386|
|        2013|           2123|
|        2010|           1901|
|        2012|           1837|
|        2011|           1795|
|        2008|           1755|
|        2006|           1730|
|        2009|           1699|
|        2005|           1669|
|        2004|           1658|
|        2002|           1600|
|        2007|           1569|
|        2003|           1508|
+------------+---------------+
only showing top 20 rows

Time taken: 15965 ms
```

HiveQL:

HiveQL is a SQL like Query language for Hive to analyze and process structured data in a meta-store.

Since, our hive container is already up we can execute the container by running the command

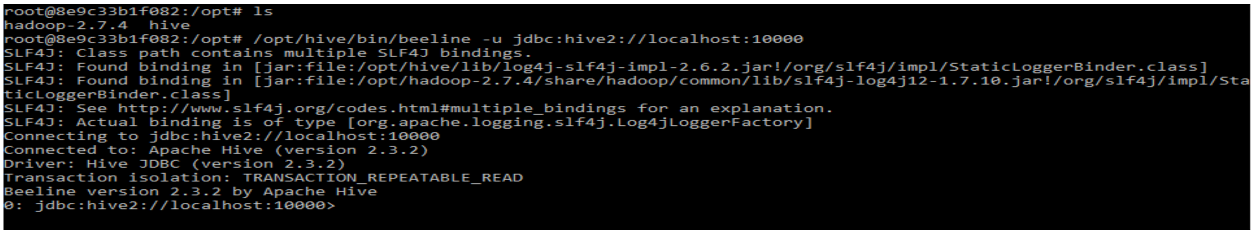
docker execute –it hive-container-image bash



```
D:\bda_project\spark-setup\docker-hadoop-spark>docker ps |findstr hive-server
8e9c33b1f082   bde2020/hive:2.3.2-postgresql-metastore                 "entrypoint.sh /bin/ΓÇª"   9 minutes ago   Up 9
minutes             0.0.0.0:10000->10000/tcp, 10002/tcp                           hive-server

D:\bda_project\spark-setup\docker-hadoop-spark>docker exec -it 8e9c33b1f082 bash
root@8e9c33b1f082:/opt#
```

To launch the hive, run the below give command:

/opt/hive/bin/beeline -u jdbc:hive2://localhost:10000



```
root@8e9c33b1f082:/opt# ls
hadoop-2.7.4  hive
root@8e9c33b1f082:/opt# /opt/hive/bin/beeline -u jdbc:hive2://localhost:10000
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hadoop-2.7.4/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/Sta
ticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Connecting to jdbc:hive2://localhost:10000
Connected to: Apache Hive (version 2.3.2)
Driver: Hive JDBC (version 2.3.2)
Transaction isolation: TRANSACTION_REPEATABLE_READ
Beeline version 2.3.2 by Apache Hive
0: jdbc:hive2://localhost:10000>
```

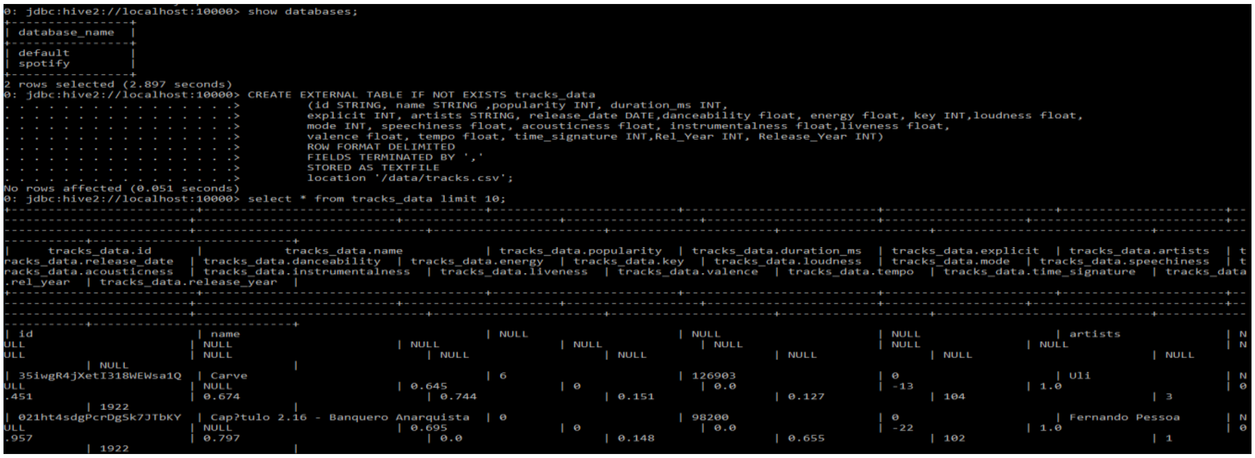
HiveQL Queries:

For hive, before we load the data from hdfs, we would first have to create the table schema.

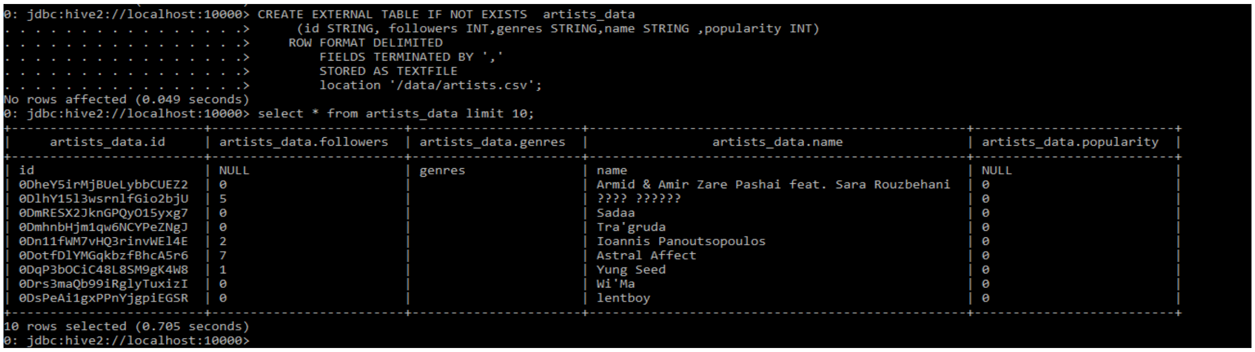
In Hive, we have two kind of tables,

- For Internal tables, hives stores data into its warehouse directory of HDFS and it’s mainly managed by hive itself.
- For External tables, they are stored outside the warehouse directory of hive.

In this project, we would be creating external tables. Since we have two files, we have created schema structure for both table artists and tracks.

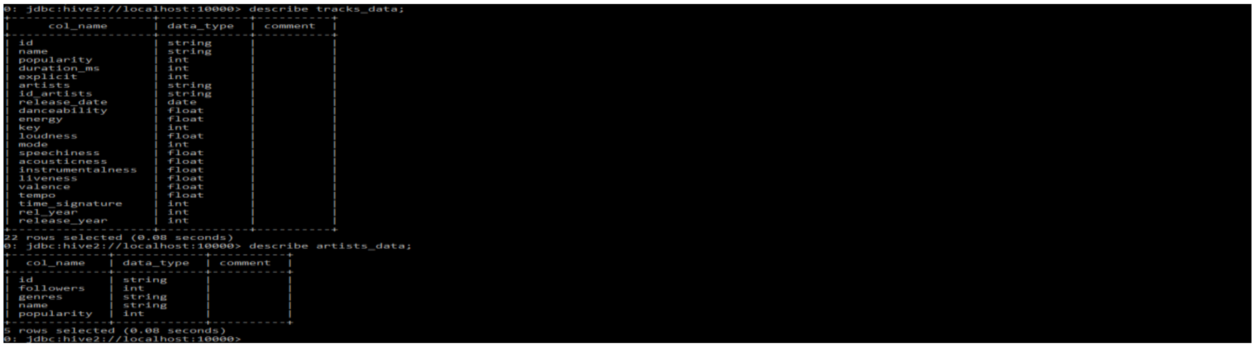


```
0: jdbc:hive2://localhost:10000> show databases;
+----------------+
| database_name  |
+----------------+
| default        |
| spotify        |
+----------------+
2 rows selected (2.897 seconds)
0: jdbc:hive2://localhost:10000> CREATE EXTERNAL TABLE IF NOT EXISTS tracks_data
. . . . . . . . . . . . . . . .>         (id STRING, name STRING ,popularity INT, duration_ms INT,
. . . . . . . . . . . . . . . .>         explicit INT, artists STRING, release_date DATE,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,Rel_Year INT, Release_Year INT)
. . . . . . . . . . . . . . . .>         ROW FORMAT DELIMITED
. . . . . . . . . . . . . . . .>         FIELDS TERMINATED BY ','
. . . . . . . . . . . . . . . .>         STORED AS TEXTFILE
. . . . . . . . . . . . . . . .>         location '/data/tracks.csv';
No rows affected (0.051 seconds)
0: jdbc:hive2://localhost:10000> select * from tracks_data limit 10;
+-------------------------+--------------------------------------+-------------------------+------------------------------+--------------------------+---------------------------+--
-----------------------+---------------------------+-------------------------+--------------------+--------------------------+--------------------------------+--
-----------------------+-------------------------------+-------------------------+--------------------------+-----------------------+----------------------------+-------------
-----------+------------------------------+
|     tracks_data.id      |            tracks_data.name          | tracks_data.popularity  | tracks_data.duration_ms  | tracks_data.explicit  | tracks_data.artists  | t
racks_data.release_date  | tracks_data.danceability  | tracks_data.energy  | tracks_data.key  | tracks_data.loudness  | tracks_data.mode  | tracks_data.speechiness  | t
racks_data.acousticness  | tracks_data.instrumentalness  | tracks_data.liveness  | tracks_data.valence  | tracks_data.tempo  | tracks_data.time_signature  | tracks_data
.rel_year  | tracks_data.release_year  |
+-------------------------+--------------------------------------+-------------------------+------------------------------+--------------------------+---------------------------+--
-----------------------+---------------------------+-------------------------+--------------------+--------------------------+--------------------------------+--
-----------------------+-------------------------------+-------------------------+--------------------------+-----------------------+----------------------------+-------------
-----------+------------------------------+
| id                      | name                                 | NULL                    | NULL                     | NULL                     | artists              | N
ULL                      | NULL                                 | NULL                     | NULL                | NULL                     | NULL                 | N
ULL                      | NULL                                     | NULL                 | NULL                  | NULL                 | NULL                 | NULL
           | NULL                      |
| 35iwgR4jXetI318WEWsa1Q  | Carve                                | 6                       | 126903                   | 0                        | Uli                  | N
ULL                      | NULL                                 | 0.645                    | 0                   | 0.0                      | -13                  | 1.0                   | 0
.451                     | 0.674                                    | 0.744                | 0.151                 | 0.127                | 104                  | 3
           | 1922                      |
| 021ht4sdgPcrDgSk7JTbKY  | Cap?tulo 2.16 - Banquero Anarquista  | 0                       | 98200                    | 0                        | Fernando Pessoa      | N
ULL                      | NULL                                 | 0.695                    | 0                   | 0.0                      | -22                  | 1.0                   | 0
.957                     | 0.797                                    | 0.0                  | 0.148                 | 0.655                | 102                  | 1
           | 1922                      |
```



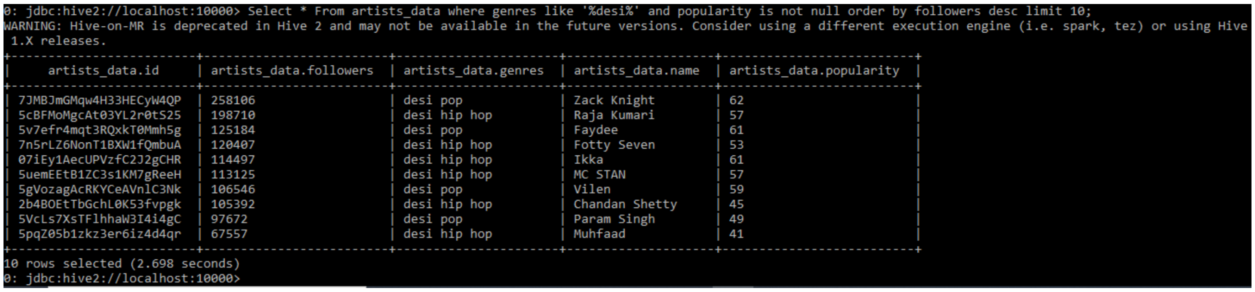
```
0: jdbc:hive2://localhost:10000> CREATE EXTERNAL TABLE IF NOT EXISTS  artists_data
. . . . . . . . . . . . . . . .>      (id STRING, followers INT,genres STRING,name STRING ,popularity INT)
. . . . . . . . . . . . . . . .>     ROW FORMAT DELIMITED
. . . . . . . . . . . . . . . .>         FIELDS TERMINATED BY ','
. . . . . . . . . . . . . . . .>         STORED AS TEXTFILE
. . . . . . . . . . . . . . . .>         location '/data/artists.csv';
No rows affected (0.049 seconds)
0: jdbc:hive2://localhost:10000> select * from artists_data limit 10;
+-------------------------+--------------------------+-----------------------+-------------------------------------------------+--------------------------+
|     artists_data.id     | artists_data.followers   | artists_data.genres   |                artists_data.name                | artists_data.popularity  |
+-------------------------+--------------------------+-----------------------+-------------------------------------------------+--------------------------+
| id                      | NULL                     | genres                | name                                            | NULL                     |
| 0DheY5irMjBUeLybbCUEZ2  | 0                        |                       | Armid & Amir Zare Pashai feat. Sara Rouzbehani  | 0                        |
| 0DlhY15l3wsrnlfGio2bjU  | 5                        |                       | ???? ??????                                     | 0                        |
| 0DmRESX2JknGPQyO15yxg7  | 0                        |                       | Sadaa                                           | 0                        |
| 0DmhnbHjm1qw6NCYPeZNgJ  | 0                        |                       | Tra'gruda                                       | 0                        |
| 0Dn11fWM7vHQ3rinvWEl4E  | 2                        |                       | Ioannis Panoutsopoulos                          | 0                        |
| 0DotfDlYMGqkbzfBhcA5r6  | 7                        |                       | Astral Affect                                   | 0                        |
| 0DqP3bOCiC48L8SM9gK4W8  | 1                        |                       | Yung Seed                                       | 0                        |
| 0Drs3maQb99iRglyTuxizI  | 0                        |                       | Wi'Ma                                           | 0                        |
| 0DsPeAi1gxPPnYjgpiEGSR  | 0                        |                       | lentboy                                         | 0                        |
+-------------------------+--------------------------+-----------------------+-------------------------------------------------+--------------------------+
10 rows selected (0.705 seconds)
0: jdbc:hive2://localhost:10000>
```

To get the schema of the tables, we use describe commands



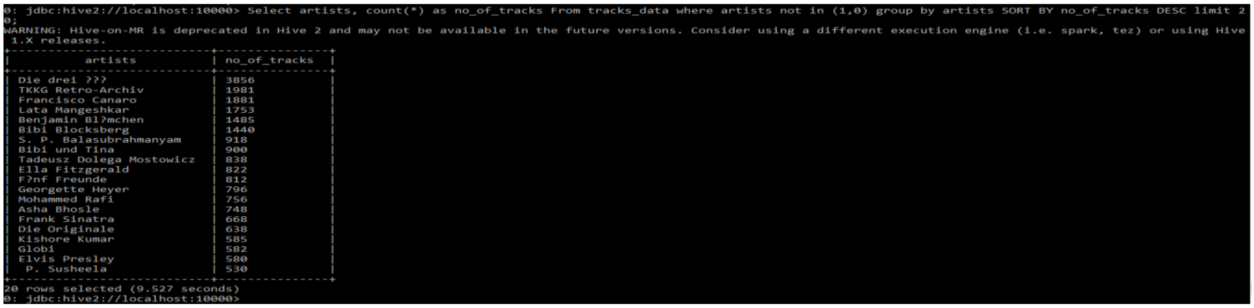
```
0: jdbc:hive2://localhost:10000> describe tracks_data;
+---------------------+-------------+----------+
|      col_name       |  data_type  | comment  |
+---------------------+-------------+----------+
| id                  | string      |          |
| name                | string      |          |
| popularity          | int         |          |
| duration_ms         | int         |          |
| explicit            | int         |          |
| artists             | string      |          |
| id_artists          | string      |          |
| release_date        | date        |          |
| 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         |          |
| rel_year            | int         |          |
| release_year        | int         |          |
+---------------------+-------------+----------+
22 rows selected (0.08 seconds)
0: jdbc:hive2://localhost:10000> describe artists_data;
+-------------+------------+----------+
|  col_name   | data_type  | comment  |
+-------------+------------+----------+
| id          | string     |          |
| followers   | int        |          |
| genres      | string     |          |
| name        | string     |          |
| popularity  | int        |          |
+-------------+------------+----------+
5 rows selected (0.08 seconds)
0: jdbc:hive2://localhost:10000>
```

1- Show the top 10 artists who has the following in desi genre.



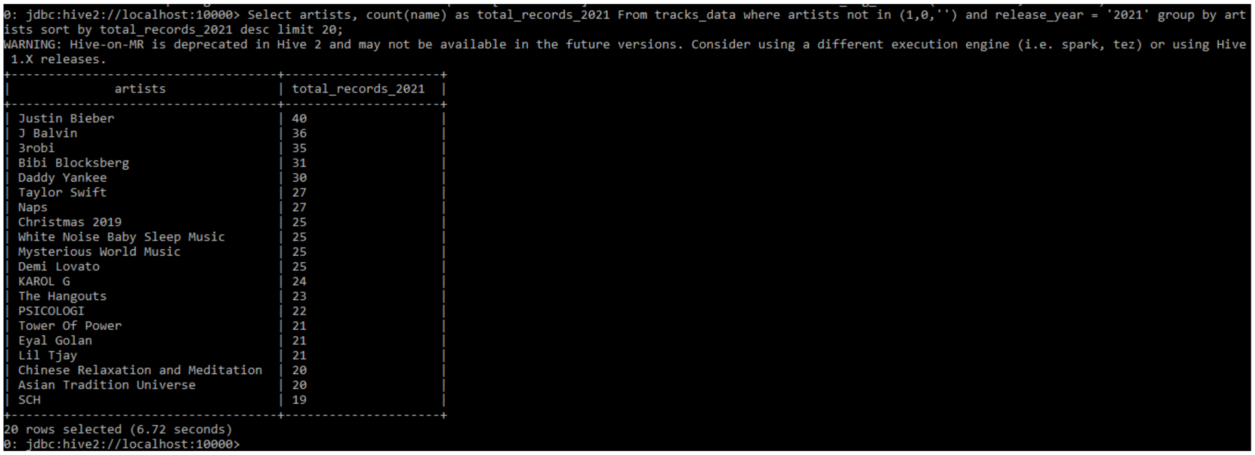
```
0: jdbc:hive2://localhost:10000> Select * From artists_data where genres like '%desi%' and popularity is not null order by followers desc limit 10;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive
 1.X releases.
+-------------------------+-------------------------+----------------------+--------------------+--------------------------+
|     artists_data.id     | artists_data.followers  | artists_data.genres  | artists_data.name  | artists_data.popularity  |
+-------------------------+-------------------------+----------------------+--------------------+--------------------------+
| 7JMBJmGMqw4H33HECyW4QP  | 258106                  | desi pop             | Zack Knight        | 62                       |
| 5cBFMoMgcAt03YL2r0tS25  | 198710                  | desi hip hop         | Raja Kumari        | 57                       |
| 5v7efr4mqt3RQxkT0Mmh5g  | 125184                  | desi pop             | Faydee             | 61                       |
| 7n5rLZ6NonT1BXW1fQmbuA  | 120407                  | desi hip hop         | Fotty Seven        | 53                       |
| 07iEy1AecUPVzfC2J2gCHR  | 114497                  | desi hip hop         | Ikka               | 61                       |
| 5uemEEtB1ZC3s1KM7gReeH  | 113125                  | desi hip hop         | MC STAN            | 57                       |
| 5gVozagAcRKYCeAVnlC3Nk  | 106546                  | desi pop             | Vilen              | 59                       |
| 2b4BOEtTbGchL0K53fvpgk  | 105392                  | desi hip hop         | Chandan Shetty     | 45                       |
| 5VcLs7XsTFlhhaW3I4i4gC  | 97672                   | desi pop             | Param Singh        | 49                       |
| 5pqZ05b1zkz3er6iz4d4qr  | 67557                   | desi hip hop         | Muhfaad            | 41                       |
+-------------------------+-------------------------+----------------------+--------------------+--------------------------+
10 rows selected (2.698 seconds)
0: jdbc:hive2://localhost:10000>
```

2- Show all the top 20 artists who has the highest no of tracks.



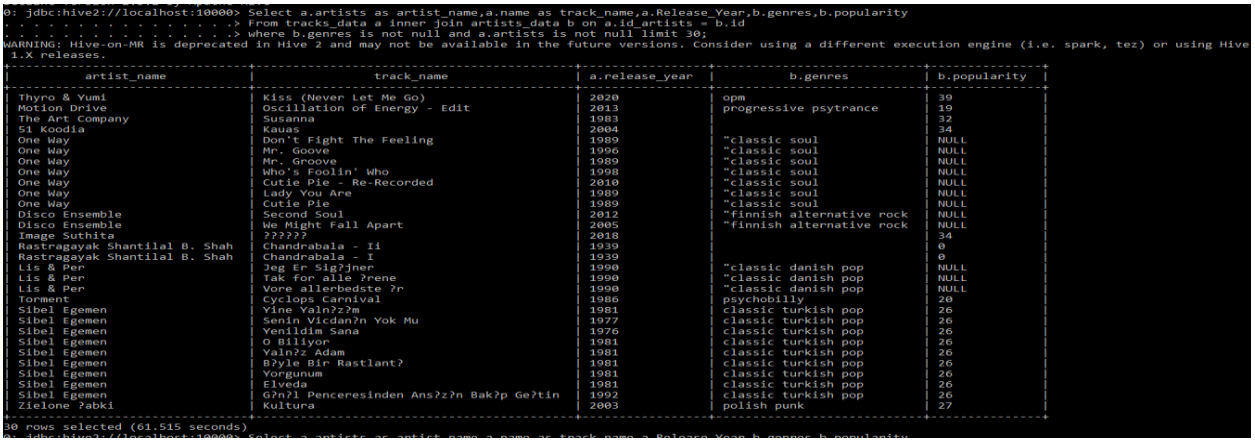
```
0: jdbc:hive2://localhost:10000> Select artists, count(*) as no_of_tracks From tracks_data where artists not in (1,0) group by artists SORT BY no_of_tracks DESC limit 2
0;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive
 1.X releases.
+-----------------------------+---------------+
|           artists           | no_of_tracks  |
+-----------------------------+---------------+
| Die drei ???                | 3856          |
| TKKG Retro-Archiv           | 1981          |
| Francisco Canaro            | 1881          |
| Lata Mangeshkar             | 1753          |
| Benjamin Bl?mchen           | 1485          |
| Bibi Blocksberg             | 1440          |
| S. P. Balasubrahmanyam      | 918           |
| Bibi und Tina               | 900           |
| Tadeusz Dolega Mostowicz    | 838           |
| Ella Fitzgerald             | 822           |
| F?nf Freunde                | 812           |
| Georgette Heyer             | 796           |
| Mohammed Rafi               | 756           |
| Asha Bhosle                 | 748           |
| Frank Sinatra               | 668           |
| Die Originale               | 638           |
| Kishore Kumar               | 585           |
| Globi                       | 582           |
| Elvis Presley               | 580           |
|  P. Susheela                | 530           |
+-----------------------------+---------------+
20 rows selected (9.527 seconds)
0: jdbc:hive2://localhost:10000>
```

3- Show all the artists who has the highest no of recorded songs in 2021.



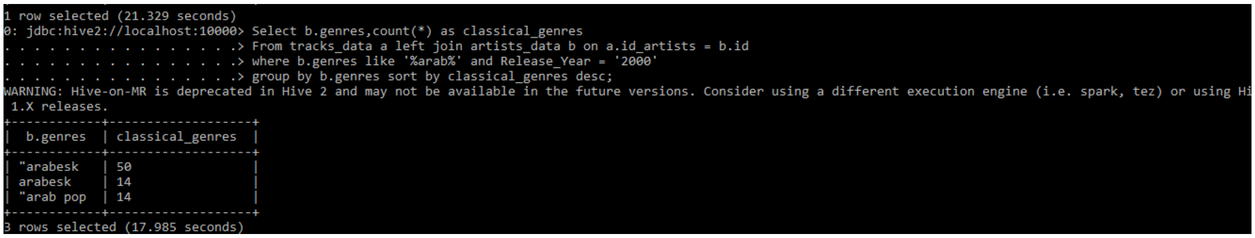
```
0: jdbc:hive2://localhost:10000> Select artists, count(name) as total_records_2021 From tracks_data where artists not in (1,0,'') and release_year = '2021' group by art
ists sort by total_records_2021 desc limit 20;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive
 1.X releases.
+-----------------------------------+--------------------+
|              artists              | total_records_2021 |
+-----------------------------------+--------------------+
| Justin Bieber                     | 40                 |
| J Balvin                          | 36                 |
| 3robi                             | 35                 |
| Bibi Blocksberg                   | 31                 |
| Daddy Yankee                      | 30                 |
| Taylor Swift                      | 27                 |
| Naps                              | 27                 |
| Christmas 2019                    | 25                 |
| White Noise Baby Sleep Music      | 25                 |
| Mysterious World Music            | 25                 |
| Demi Lovato                       | 25                 |
| KAROL G                           | 24                 |
| The Hangouts                      | 23                 |
| PSICOLOGI                         | 22                 |
| Tower Of Power                    | 21                 |
| Eyal Golan                        | 21                 |
| Lil Tjay                          | 21                 |
| Chinese Relaxation and Meditation | 20                 |
| Asian Tradition Universe          | 20                 |
| SCH                               | 19                 |
+-----------------------------------+--------------------+
20 rows selected (6.72 seconds)
0: jdbc:hive2://localhost:10000>
```

4- Show all the artist by their track_name, genres, and its popularity.



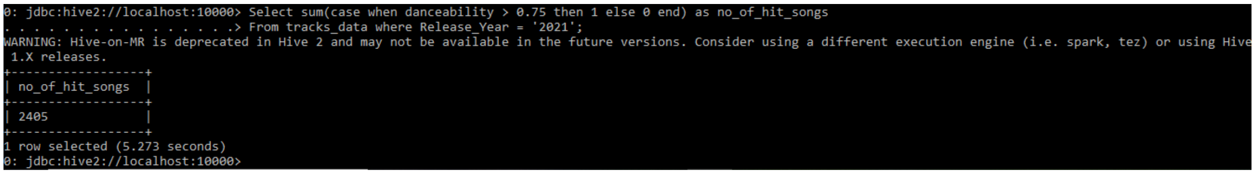
```
0: jdbc:hive2://localhost:10000> Select a.artists as artist_name,a.name as track_name,a.Release_Year,b.genres,b.popularity
. . . . . . . . . . . . . . . .> From tracks_data a inner join artists_data b on a.id_artists = b.id
. . . . . . . . . . . . . . . .> where b.genres is not null and a.artists is not null limit 30;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive
 1.X releases.
+-------------------------------+-------------------------------------------+------------------+------------------------------+----------------+
|          artist_name          |                track_name                 | a.release_year   |           b.genres           | b.popularity   |
+-------------------------------+-------------------------------------------+------------------+------------------------------+----------------+
| Thyro & Yumi                  | Kiss (Never Let Me Go)                    | 2020             | opm                          | 39             |
| Motion Drive                  | Oscillation of Energy - Edit              | 2013             | progressive psytrance        | 19             |
| The Art Company               | Susanna                                   | 1983             |                              | 32             |
| 51 Koodia                     | Kauas                                     | 2004             |                              | 34             |
| One Way                       | Don't Fight The Feeling                   | 1989             | "classic soul                | NULL           |
| One Way                       | Mr. Goove                                 | 1996             | "classic soul                | NULL           |
| One Way                       | Mr. Groove                                | 1989             | "classic soul                | NULL           |
| One Way                       | Who's Foolin' Who                         | 1998             | "classic soul                | NULL           |
| One Way                       | Cutie Pie - Re-Recorded                   | 2010             | "classic soul                | NULL           |
| One Way                       | Lady You Are                              | 1989             | "classic soul                | NULL           |
| One Way                       | Cutie Pie                                 | 1989             | "classic soul                | NULL           |
| Disco Ensemble                | Second Soul                               | 2012             | "finnish alternative rock    | NULL           |
| Disco Ensemble                | We Might Fall Apart                       | 2005             | "finnish alternative rock    | NULL           |
| Image Suthita                 | ??????                                    | 2018             |                              | 34             |
| Rastragayak Shantilal B. Shah | Chandrabala - Ii                          | 1939             |                              | 0              |
| Rastragayak Shantilal B. Shah | Chandrabala - I                           | 1939             |                              | 0              |
| Lis & Per                     | Jeg Er Sig?jner                           | 1990             | "classic danish pop          | NULL           |
| Lis & Per                     | Tak for alle ?rene                        | 1990             | "classic danish pop          | NULL           |
| Lis & Per                     | Vore allerbedste ?r                       | 1990             | "classic danish pop          | NULL           |
| Torment                       | Cyclops Carnival                          | 1986             | psychobilly                  | 20             |
| Sibel Egemen                  | Yine Yaln?z?m                             | 1981             | classic turkish pop          | 26             |
| Sibel Egemen                  | Senin Vicdan?n Yok Mu                     | 1977             | classic turkish pop          | 26             |
| Sibel Egemen                  | Yenildim Sana                             | 1976             | classic turkish pop          | 26             |
| Sibel Egemen                  | O Biliyor                                 | 1981             | classic turkish pop          | 26             |
| Sibel Egemen                  | Yaln?z Adam                               | 1981             | classic turkish pop          | 26             |
| Sibel Egemen                  | B?yle Bir Rastlant?                       | 1981             | classic turkish pop          | 26             |
| Sibel Egemen                  | Yorgunum                                  | 1981             | classic turkish pop          | 26             |
| Sibel Egemen                  | Elveda                                    | 1981             | classic turkish pop          | 26             |
| Sibel Egemen                  | G?n?l Penceresinden Ans?z?n Bak?p Ge?tin  | 1992             | classic turkish pop          | 26             |
| Zielone ?abki                 | Kultura                                   | 2003             | polish punk                  | 27             |
+-------------------------------+-------------------------------------------+------------------+------------------------------+----------------+
30 rows selected (61.515 seconds)
0: jdbc:hive2://localhost:10000> Select a.artists as artist_name,a.name as track_name,a.Release_Year,b.genres,b.popularity
```

5- Show all the records of 2000 in the Arabic genre.



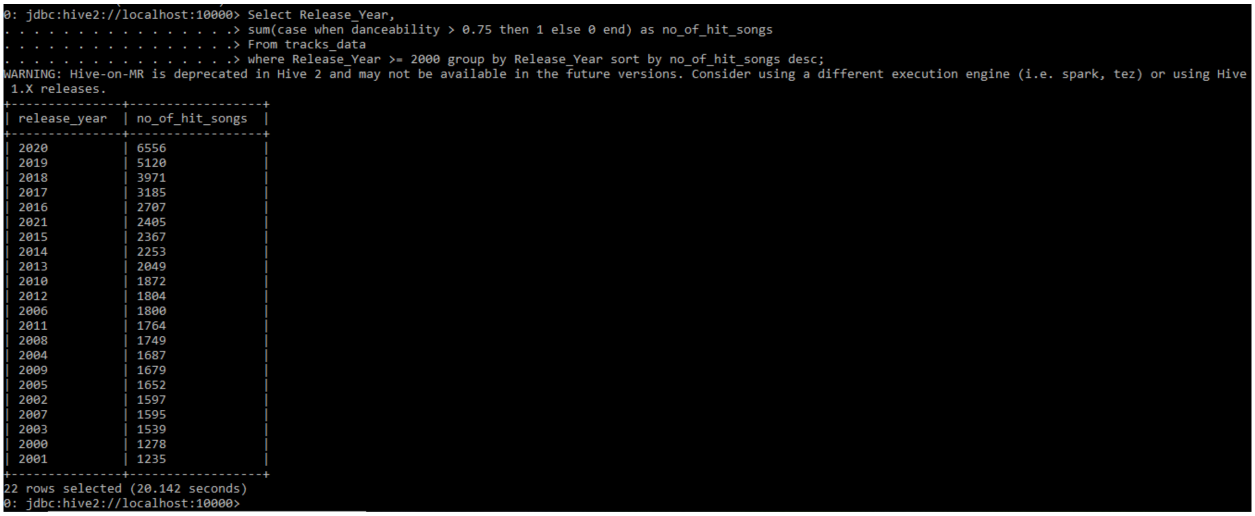
```
1 row selected (21.329 seconds)
0: jdbc:hive2://localhost:10000> Select b.genres,count(*) as classical_genres
. . . . . . . . . . . . . . . .> From tracks_data a left join artists_data b on a.id_artists = b.id
. . . . . . . . . . . . . . . .> where b.genres like '%arab%' and Release_Year = '2000'
. . . . . . . . . . . . . . . .> group by b.genres sort by classical_genres desc;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hi
 1.X releases.
+-----------+-------------------+
| b.genres  | classical_genres  |
+-----------+-------------------+
| "arabesk  | 50                |
| arabesk   | 14                |
| "arab pop | 14                |
+-----------+-------------------+
3 rows selected (17.985 seconds)
```

6- On the basis of danceability, show the total no of hit songs of 2021.



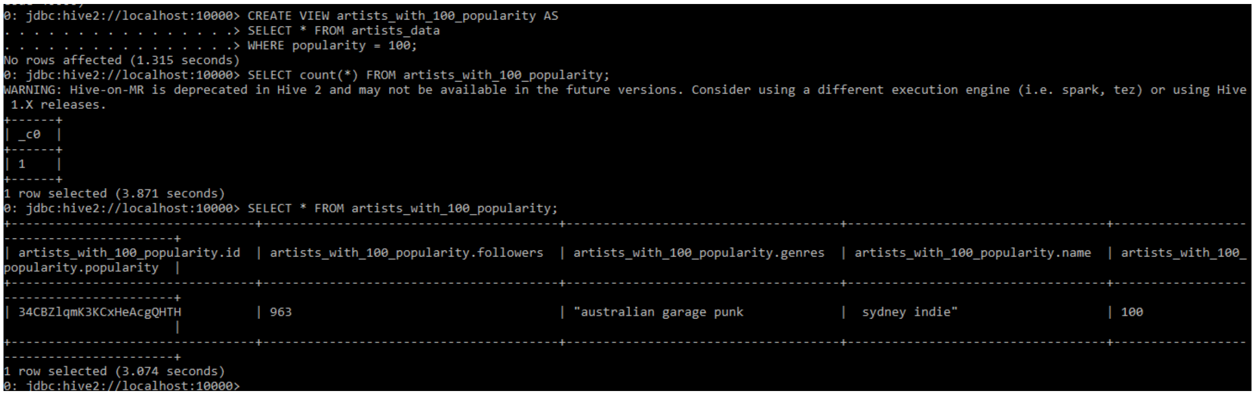
```
0: jdbc:hive2://localhost:10000> Select sum(case when danceability > 0.75 then 1 else 0 end) as no_of_hit_songs
. . . . . . . . . . . . . . . . .> From tracks_data where Release_Year = '2021';
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive
 1.X releases.
+------------------+
| no_of_hit_songs  |
+------------------+
| 2405             |
+------------------+
1 row selected (5.273 seconds)
0: jdbc:hive2://localhost:10000>
```

7- Show the no of hit songs after 2000.



```
0: jdbc:hive2://localhost:10000> Select Release_Year,
. . . . . . . . . . . . . . . . .> sum(case when danceability > 0.75 then 1 else 0 end) as no_of_hit_songs
. . . . . . . . . . . . . . . . .> From tracks_data
. . . . . . . . . . . . . . . . .> where Release_Year >= 2000 group by Release_Year sort by no_of_hit_songs desc;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive
 1.X releases.
+---------------+------------------+
| release_year  | no_of_hit_songs  |
+---------------+------------------+
| 2020          | 6556             |
| 2019          | 5120             |
| 2018          | 3971             |
| 2017          | 3185             |
| 2016          | 2707             |
| 2021          | 2405             |
| 2015          | 2367             |
| 2014          | 2253             |
| 2013          | 2049             |
| 2010          | 1872             |
| 2012          | 1804             |
| 2006          | 1800             |
| 2011          | 1764             |
| 2008          | 1749             |
| 2004          | 1687             |
| 2009          | 1679             |
| 2005          | 1652             |
| 2002          | 1597             |
| 2007          | 1595             |
| 2003          | 1539             |
| 2000          | 1278             |
| 2001          | 1235             |
+---------------+------------------+
22 rows selected (20.142 seconds)
0: jdbc:hive2://localhost:10000>
```

8- We can also create a view on hive table with a specific condition, and use it afterwards whenever required.

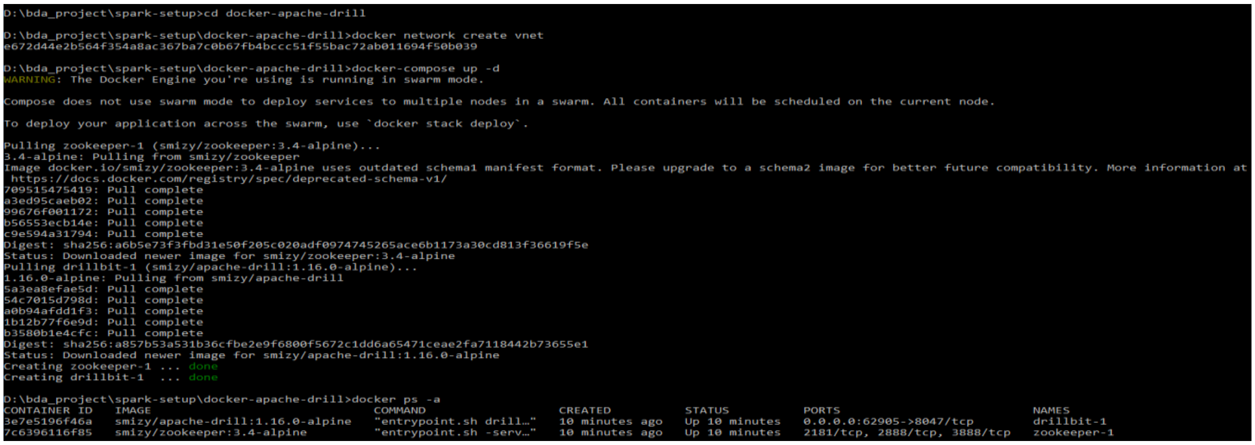


```
0: jdbc:hive2://localhost:10000> CREATE VIEW artists_with_100_popularity AS
. . . . . . . . . . . . . . . . .> SELECT * FROM artists_data
. . . . . . . . . . . . . . . . .> WHERE popularity = 100;
No rows affected (1.315 seconds)
0: jdbc:hive2://localhost:10000> SELECT count(*) FROM artists_with_100_popularity;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive
 1.X releases.
+------+
| _c0  |
+------+
| 1    |
+------+
1 row selected (3.871 seconds)
0: jdbc:hive2://localhost:10000> SELECT * FROM artists_with_100_popularity;
+---------------------------------+----------------------------------------+-------------------------------------+-----------------------------------+------------------
----------------------+
| artists_with_100_popularity.id  | artists_with_100_popularity.followers  | artists_with_100_popularity.genres  | artists_with_100_popularity.name  | artists_with_100_
popularity.popularity  |
+---------------------------------+----------------------------------------+-------------------------------------+-----------------------------------+------------------
----------------------+
| 34CBZlqmK3KCxHeAcgQHTH          | 963                                    | "australian garage punk             |  sydney indie"                    | 100
                      |
+---------------------------------+----------------------------------------+-------------------------------------+-----------------------------------+------------------
----------------------+
1 row selected (3.074 seconds)
0: jdbc:hive2://localhost:10000>
```

ApacheDrill:

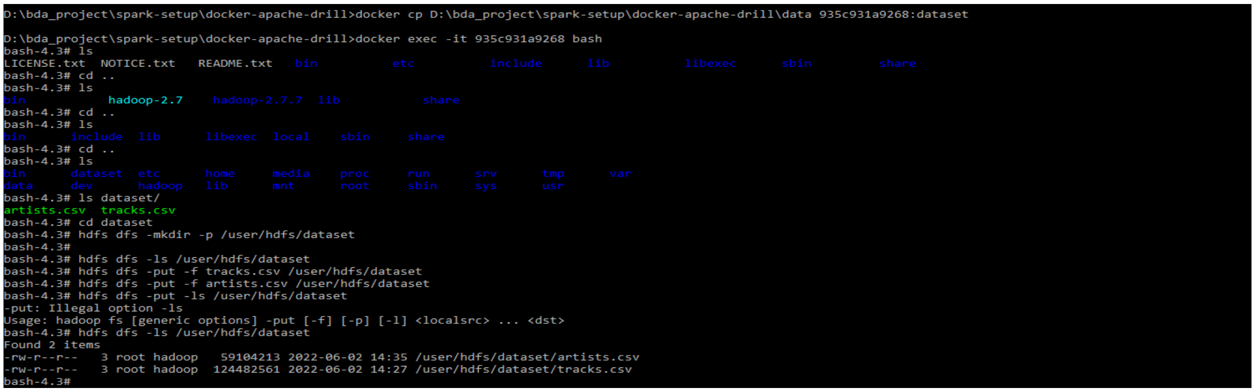
Drill is the columnar based schema-free SQL query engine that supports complex data queries.

For drill we have a separate compose file that we would compose. First we have created the network of vnet. Then run the docker compose command to install the required images and containers.



```
D:\bda_project\spark-setup>cd docker-apache-drill

D:\bda_project\spark-setup\docker-apache-drill>docker network create vnet
e672d44e2b564f354a8ac367ba7c0b67fb4bccc51f55bac72ab011694f50b039

D:\bda_project\spark-setup\docker-apache-drill>docker-compose up -d
WARNING: The Docker Engine you're using is running in swarm mode.

Compose does not use swarm mode to deploy services to multiple nodes in a swarm. All containers will be scheduled on the current node.

To deploy your application across the swarm, use `docker stack deploy`.

Pulling zookeeper-1 (smizy/zookeeper:3.4-alpine)...
3.4-alpine: Pulling from smizy/zookeeper
Image docker.io/smizy/zookeeper:3.4-alpine uses outdated schema1 manifest format. Please upgrade to a schema2 image for better future compatibility. More information at
 https://docs.docker.com/registry/spec/deprecated-schema-v1/
709515475419: Pull complete
a3ed95caeb02: Pull complete
99676f001172: Pull complete
b56553ecb14e: Pull complete
c9e594a31794: Pull complete
Digest: sha256:a6b5e73f3fbd31e50f205c020adf0974745265ace6b1173a30cd813f36619f5e
Status: Downloaded newer image for smizy/zookeeper:3.4-alpine
Pulling drillbit-1 (smizy/apache-drill:1.16.0-alpine)...
1.16.0-alpine: Pulling from smizy/apache-drill
5a3ea8efae5d: Pull complete
54c7015d798d: Pull complete
a0b94afdd1f3: Pull complete
1b12b77f6e9d: Pull complete
b3580b1e4cfc: Pull complete
Digest: sha256:a857b53a531b36cfbe2e9f6800f5672c1dd6a65471ceae2fa7118442b73655e1
Status: Downloaded newer image for smizy/apache-drill:1.16.0-alpine
Creating zookeeper-1 ... done
Creating drillbit-1  ... done

D:\bda_project\spark-setup\docker-apache-drill>docker ps -a
CONTAINER ID   IMAGE                              COMMAND                  CREATED          STATUS          PORTS                                NAMES
3e7e5196f46a   smizy/apache-drill:1.16.0-alpine   "entrypoint.sh drill…"   10 minutes ago   Up 10 minutes   0.0.0.0:62905->8047/tcp              drillbit-1
7c6396116f85   smizy/zookeeper:3.4-alpine         "entrypoint.sh -serv…"   10 minutes ago   Up 10 minutes   2181/tcp, 2888/tcp, 3888/tcp         zookeeper-1
```

Then we have load the data into hdfs.



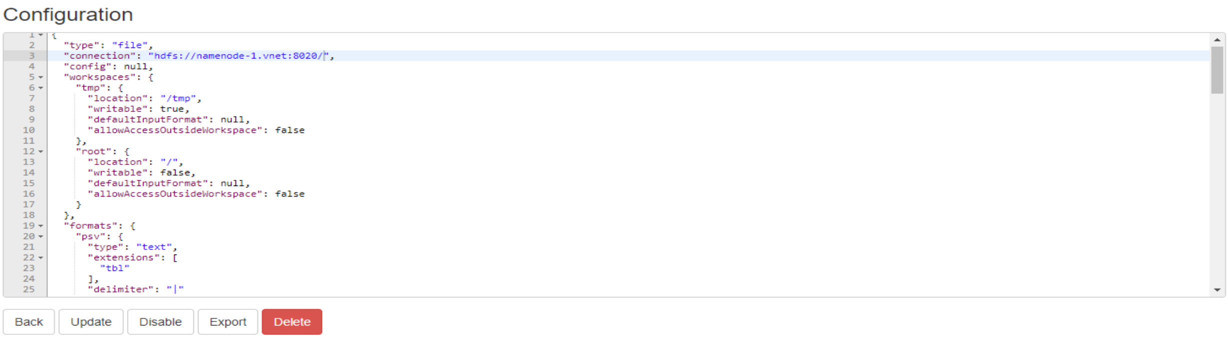
```
D:\bda_project\spark-setup\docker-apache-drill>docker cp D:\bda_project\spark-setup\docker-apache-drill\data 935c931a9268:dataset

D:\bda_project\spark-setup\docker-apache-drill>docker exec -it 935c931a9268 bash
bash-4.3# ls
LICENSE.txt  NOTICE.txt   README.txt  bin         etc         include     lib         libexec     sbin        share
bash-4.3# cd ..
bash-4.3# ls
bin          hadoop-2.7    hadoop-2.7.7  lib          share
bash-4.3# cd ..
bash-4.3# ls
bin      include  lib      libexec  local    sbin     share
bash-4.3# cd ..
bash-4.3# ls
bin      dataset  etc      home     media    proc     run      srv      tmp      var
data     dev      hadoop   lib      mnt      root     sbin     sys      usr
bash-4.3# ls dataset/
artists.csv  tracks.csv
bash-4.3# cd dataset
bash-4.3# hdfs dfs -mkdir -p /user/hdfs/dataset
bash-4.3#
bash-4.3# hdfs dfs -ls /user/hdfs/dataset
bash-4.3# hdfs dfs -put -f tracks.csv /user/hdfs/dataset
bash-4.3# hdfs dfs -put -f artists.csv /user/hdfs/dataset
bash-4.3# hdfs dfs -put -ls /user/hdfs/dataset
-put: Illegal option -ls
Usage: hadoop fs [generic options] -put [-f] [-p] [-l] <localsrc> ... <dst>
bash-4.3# hdfs dfs -ls /user/hdfs/dataset
Found 2 items
-rw-r--r--   3 root hadoop   59104213 2022-06-02 14:35 /user/hdfs/dataset/artists.csv
-rw-r--r--   3 root hadoop  124482561 2022-06-02 14:27 /user/hdfs/dataset/tracks.csv
bash-4.3#
```

After loading the file into hdfs, and before going into drill, we need to set it up some configuration in drill browser.

To access the drill browser Web UI and set the configurations, goto https://localhost:59617/storage/dfs and set parameters:

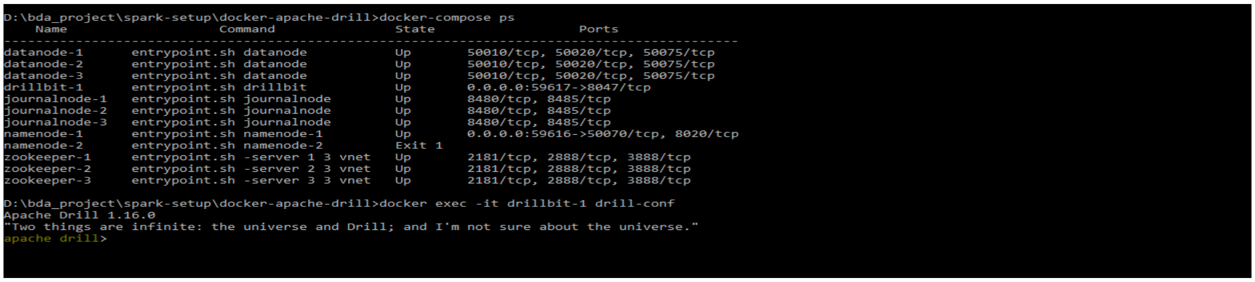
- connection = “hdfs://namenode-1.vnet:8020/”
- enabled = true
- root.location = /user/hdfs

After that we can successfully load the data from hdfs to our drill bash.



```
{
  "type": "file",
  "connection": "hdfs://namenode-1.vnet:8020/",
  "config": null,
  "workspaces": {
    "tmp": {
      "location": "/tmp",
      "writable": true,
      "defaultInputFormat": null,
      "allowAccessOutsideWorkspace": false
    },
    "root": {
      "location": "/",
      "writable": false,
      "defaultInputFormat": null,
      "allowAccessOutsideWorkspace": false
    }
  },
  "formats": {
    "psv": {
      "type": "text",
      "extensions": [
        "tbl"
      ],
      "delimiter": "|"
```

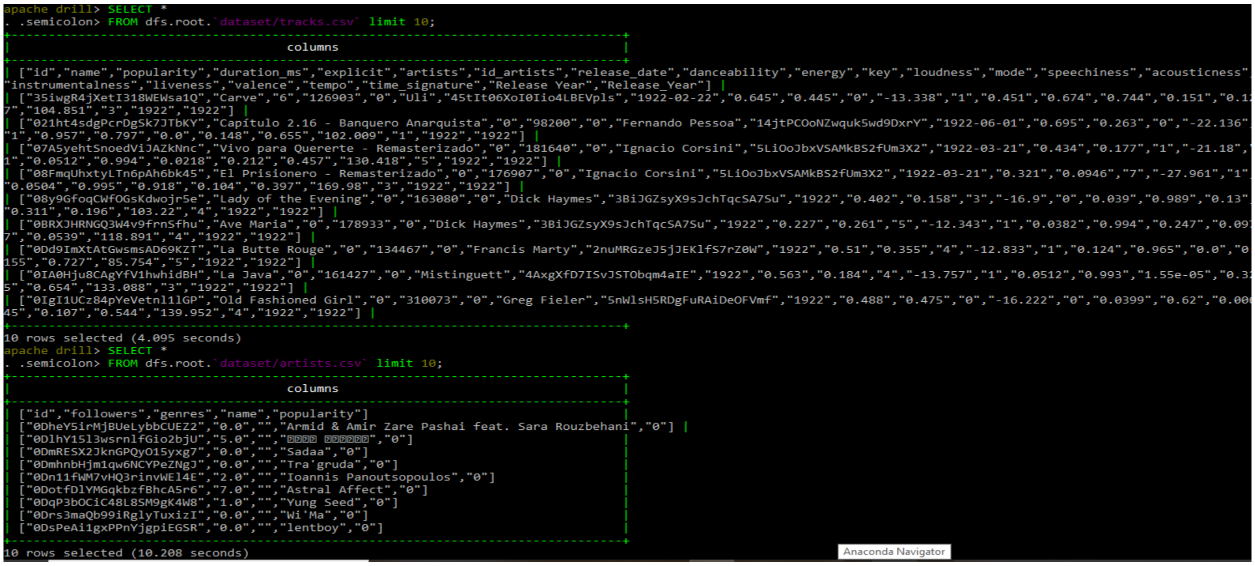
To launch the drill bash, use command **docker exec –it drillbit-1 drill-conf**



```
D:\bda_project\spark-setup\docker-apache-drill>docker-compose ps
    Name                   Command                State                 Ports
--------------------------------------------------------------------------------------------
datanode-1       entrypoint.sh datanode           Up      50010/tcp, 50020/tcp, 50075/tcp
datanode-2       entrypoint.sh datanode           Up      50010/tcp, 50020/tcp, 50075/tcp
datanode-3       entrypoint.sh datanode           Up      50010/tcp, 50020/tcp, 50075/tcp
drillbit-1       entrypoint.sh drillbit           Up      0.0.0.0:59617->8047/tcp
journalnode-1    entrypoint.sh journalnode        Up      8480/tcp, 8485/tcp
journalnode-2    entrypoint.sh journalnode        Up      8480/tcp, 8485/tcp
journalnode-3    entrypoint.sh journalnode        Up      8480/tcp, 8485/tcp
namenode-1       entrypoint.sh namenode-1         Up      0.0.0.0:59616->50070/tcp, 8020/tcp
namenode-2       entrypoint.sh namenode-2         Exit 1
zookeeper-1      entrypoint.sh -server 1 3 vnet   Up      2181/tcp, 2888/tcp, 3888/tcp
zookeeper-2      entrypoint.sh -server 2 3 vnet   Up      2181/tcp, 2888/tcp, 3888/tcp
zookeeper-3      entrypoint.sh -server 3 3 vnet   Up      2181/tcp, 2888/tcp, 3888/tcp

D:\bda_project\spark-setup\docker-apache-drill>docker exec -it drillbit-1 drill-conf
Apache Drill 1.16.0
"Two things are infinite: the universe and Drill; and I'm not sure about the universe."
apache drill>
```

Drill Queries:

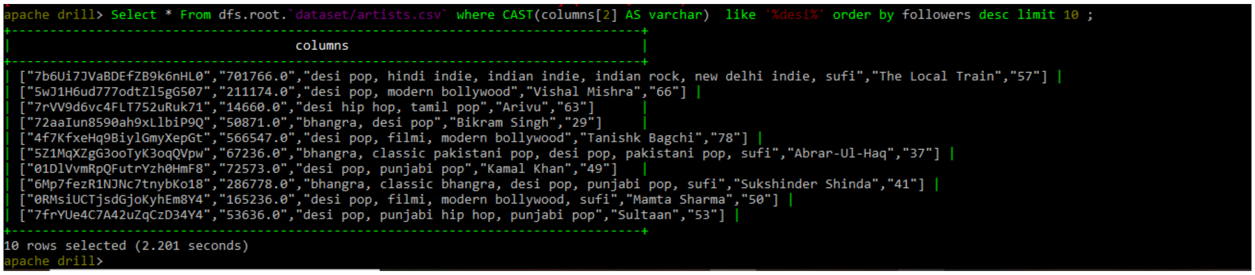
1- To load the data from hdfs use **dfs.root** before the table path.



```
apache drill> SELECT *
. .semicolon> FROM dfs.root.`dataset/tracks.csv` limit 10;
+------------------------------------------------------------------------------------+
|                                      columns                                       |
+------------------------------------------------------------------------------------+
| ["id","name","popularity","duration_ms","explicit","artists","id_artists","release_date","danceability","energy","key","loudness","mode","speechiness","acousticness",
"instrumentalness","liveness","valence","tempo","time_signature","Release Year","Release_Year"] |
| ["35iwgR4jXetI318WEWsa1Q","Carve","6","126903","0","Uli","45tIt06XoI0Iio4LBEVpls","1922-02-22","0.645","0.445","0","-13.338","1","0.451","0.674","0.744","0.151","0.1
7","104.851","3","1922","1922"] |
| ["021ht4sdgPcrDgSk7JTbKY","Capítulo 2.16 - Banquero Anarquista","0","98200","0","Fernando Pessoa","14jtPCOoNZwquk5wd9DxrY","1922-06-01","0.695","0.263","0","-22.136",
"1","0.957","0.797","0.0","0.148","0.655","102.009","1","1922","1922"] |
| ["07A5yehtSnoedViJAZkNnc","Vivo para Quererte - Remasterizado","0","181640","0","Ignacio Corsini","5LiOoJbxVSAMkBS2fUm3X2","1922-03-21","0.434","0.177","1","-21.18",
1","0.0512","0.994","0.0218","0.212","0.457","130.418","5","1922","1922"] |
| ["08FmqUhxtyLTn6pAh6bk45","El Prisionero - Remasterizado","0","176907","0","Ignacio Corsini","5LiOoJbxVSAMkBS2fUm3X2","1922-03-21","0.321","0.0946","7","-27.961","1",
"0.0504","0.995","0.918","0.104","0.397","169.98","3","1922","1922"] |
| ["08y9GfoqCWfOGsKdwojr5e","Lady of the Evening","0","163080","0","Dick Haymes","3BiJGZsyX9sJchTqcSA7Su","1922","0.402","0.158","3","-16.9","0","0.039","0.989","0.13",
"0.311","0.196","103.22","4","1922","1922"] |
| ["0BRXJHRNGQ3W4v9frnSfhu","Ave Maria","0","178933","0","Dick Haymes","3BiJGZsyX9sJchTqcSA7Su","1922","0.227","0.261","5","-12.343","1","0.0382","0.994","0.247","0.09
7","0.0539","118.891","4","1922","1922"] |
| ["0Dd9ImXtAtGwsmsAD69KZT","La Butte Rouge","0","134467","0","Francis Marty","2nuMRGzeJ5jJEKlfS7rZ0W","1922","0.51","0.355","4","-12.833","1","0.124","0.965","0.0","0
155","0.727","85.754","5","1922","1922"] |
| ["0IA0Hju8CAgYfV1hwhidBH","La Java","0","161427","0","Mistinguett","4AxgXfD7ISvJSTObqm4aIE","1922","0.563","0.184","4","-13.757","1","0.0512","0.993","1.55e-05","0.3
5","0.654","133.088","3","1922","1922"] |
| ["0IgI1UCz84pYeVetnl1lGP","Old Fashioned Girl","0","310073","0","Greg Fieler","5nWlsH5RDgFuRAiDeOFVmf","1922","0.488","0.475","0","-16.222","0","0.0399","0.62","0.00
45","0.107","0.544","139.952","4","1922","1922"] |
+------------------------------------------------------------------------------------+
10 rows selected (4.095 seconds)
apache drill> SELECT *
. .semicolon> FROM dfs.root.`dataset/artists.csv` limit 10;
+------------------------------------------------------------------------------------+
|                                      columns                                       |
+------------------------------------------------------------------------------------+
| ["id","followers","genres","name","popularity"]                                    |
| ["0DheY5irMjBUeLybbCUEZ2","0.0","","Armid & Amir Zare Pashai feat. Sara Rouzbehani","0"] |
| ["0DlhY15l3wsrnlfGio2bjU","5.0","","□□□□ □□□□□□","0"]                              |
| ["0DmRESX2JknGPQyO15yxg7","0.0","","Sadaa","0"]                                    |
| ["0DmhnbHjm1qw6NCYPeZNgJ","0.0","","Tra'gruda","0"]                                |
| ["0Dn11fWM7vHQ3rinvWEl4E","2.0","","Ioannis Panoutsopoulos","0"]                   |
| ["0DotfDlYMGqkbzfBhcA5r6","7.0","","Astral Affect","0"]                            |
| ["0DqP3bOCiC48L8SM9gK4W8","1.0","","Yung Seed","0"]                                |
| ["0Drs3maQb99iRglyTuxizI","0.0","","Wi'Ma","0"]                                    |
| ["0DsPeAi1gxPPnYjgpiEGSR","0.0","","lentboy","0"]                                  |
+------------------------------------------------------------------------------------+
10 rows selected (10.208 seconds)
```

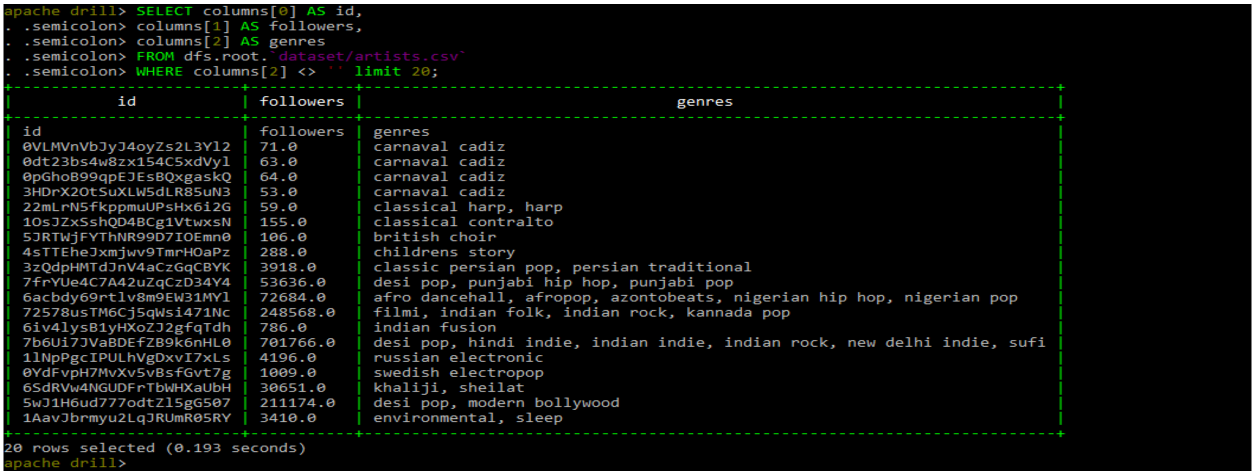
To access the column fields in the table in drill, we would have to use column no rather than a column name.

2- Show the top 10 artist of desi genre by their highest following.



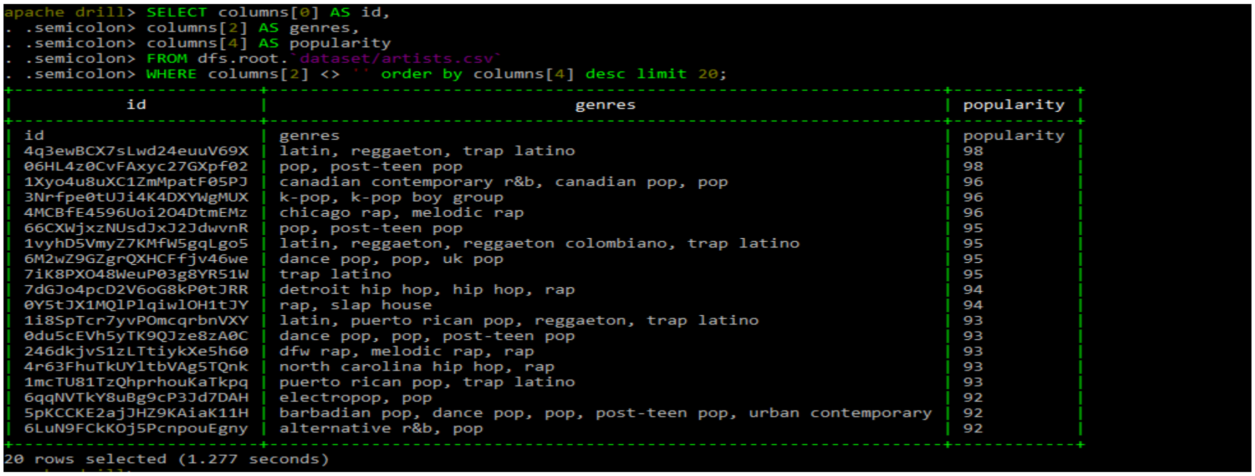
```
apache drill> Select * From dfs.root.`dataset/artists.csv` where CAST(columns[2] AS varchar)  like '%desi%' order by followers desc limit 10 ;
+---------------------------------------------------------------------------------+
|                                     columns                                     |
+---------------------------------------------------------------------------------+
| ["7b6Ui7JVaBDEfZB9k6nHL0","701766.0","desi pop, hindi indie, indian indie, indian rock, new delhi indie, sufi","The Local Train","57"] |
| ["5wJ1H6ud777odtZl5gG507","211174.0","desi pop, modern bollywood","Vishal Mishra","66"] |
| ["7rVV9d6vc4FLT752uRuk71","14660.0","desi hip hop, tamil pop","Arivu","63"]     |
| ["72aaIun8590ah9xLlbiP9Q","50871.0","bhangra, desi pop","Bikram Singh","29"]    |
| ["4f7KfxeHq9BiylGmyXepGt","566547.0","desi pop, filmi, modern bollywood","Tanishk Bagchi","78"] |
| ["5Z1MqXZgG3ooTyK3oqQVpw","67236.0","bhangra, classic pakistani pop, desi pop, pakistani pop, sufi","Abrar-Ul-Haq","37"] |
| ["01DlVvmRpQFutrYzh0HmF8","72573.0","desi pop, punjabi pop","Kamal Khan","49"]  |
| ["6Mp7fezR1NJNc7tnybKo18","286778.0","bhangra, classic bhangra, desi pop, punjabi pop, sufi","Sukshinder Shinda","41"] |
| ["0RMsiUCTjsdGjoKyhEm8Y4","165236.0","desi pop, filmi, modern bollywood, sufi","Mamta Sharma","50"] |
| ["7frYUe4C7A42uZqCzD34Y4","53636.0","desi pop, punjabi hip hop, punjabi pop","Sultaan","53"] |
+---------------------------------------------------------------------------------+
10 rows selected (2.201 seconds)
apache drill>
```

3- Show the artist followers by their genres.



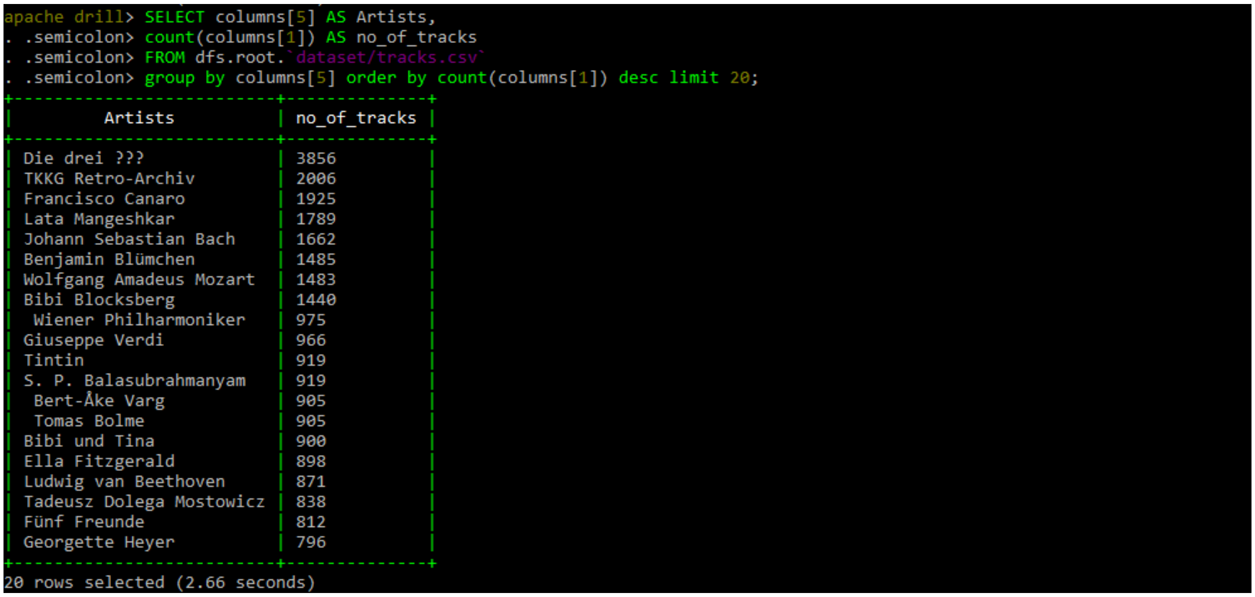
```
apache drill> SELECT columns[0] AS id,
. .semicolon> columns[1] AS followers,
. .semicolon> columns[2] AS genres
. .semicolon> FROM dfs.root.`dataset/artists.csv`
. .semicolon> WHERE columns[2] <> '' limit 20;
+------------------------+-----------+-------------------------------------------------------------------------+
|           id           | followers |                                 genres                                  |
+------------------------+-----------+-------------------------------------------------------------------------+
| id                     | followers | genres                                                                  |
| 0VLMVnVbJyJ4oyZs2L3Yl2 | 71.0      | carnaval cadiz                                                          |
| 0dt23bs4w8zx154C5xdVyl | 63.0      | carnaval cadiz                                                          |
| 0pGhoB99qpEJEsBQxgaskQ | 64.0      | carnaval cadiz                                                          |
| 3HDrX2OtSuXLW5dLR85uN3 | 53.0      | carnaval cadiz                                                          |
| 22mLrN5fkppmuUPsHx6i2G | 59.0      | classical harp, harp                                                    |
| 1OsJZxSshQD4BCg1VtwxsN | 155.0     | classical contralto                                                     |
| 5JRTWjFYThNR99D7IOEmn0 | 106.0     | british choir                                                           |
| 4sTTEheJxmjwv9TmrHOaPz | 288.0     | childrens story                                                         |
| 3zQdpHMTdJnV4aCzGqCBYK | 3918.0    | classic persian pop, persian traditional                                |
| 7frYUe4C7A42uZqCzD34Y4 | 53636.0   | desi pop, punjabi hip hop, punjabi pop                                  |
| 6acbdy69rtlv8m9EW31MYl | 72684.0   | afro dancehall, afropop, azontobeats, nigerian hip hop, nigerian pop    |
| 72578usTM6Cj5qWsi471Nc | 248568.0  | filmi, indian folk, indian rock, kannada pop                            |
| 6iv4lysB1yHXoZJ2gfqTdh | 786.0     | indian fusion                                                           |
| 7b6Ui7JVaBDEfZB9k6nHL0 | 701766.0  | desi pop, hindi indie, indian indie, indian rock, new delhi indie, sufi |
| 1lNpPgcIPULhVgDxvI7xLs | 4196.0    | russian electronic                                                      |
| 0YdFvpH7MvXv5vBsfGvt7g | 1009.0    | swedish electropop                                                      |
| 6SdRVw4NGUDFrTbWHXaUbH | 30651.0   | khaliji, sheilat                                                        |
| 5wJ1H6ud777odtZl5gG507 | 211174.0  | desi pop, modern bollywood                                              |
| 1AavJbrmyu2LqJRUmR05RY | 3410.0    | environmental, sleep                                                    |
+------------------------+-----------+-------------------------------------------------------------------------+
20 rows selected (0.193 seconds)
apache drill>
```

4- Show the artist’s genre by their popularity.



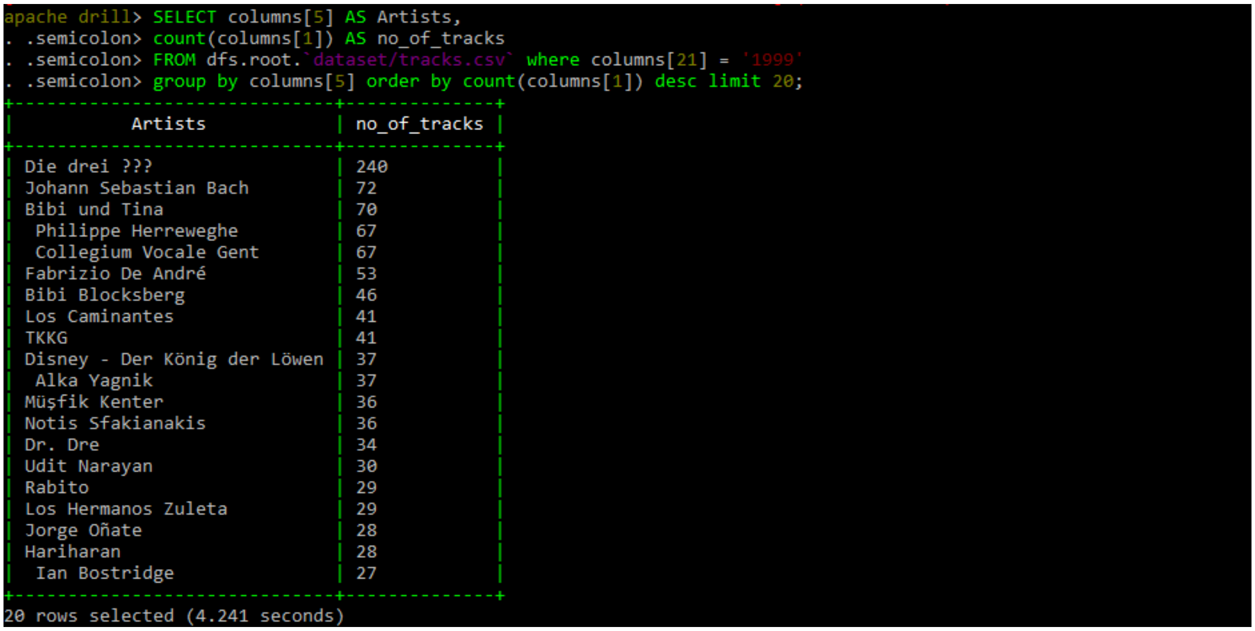
```
apache drill> SELECT columns[0] AS id,
. .semicolon> columns[2] AS genres,
. .semicolon> columns[4] AS popularity
. .semicolon> FROM dfs.root.`dataset/artists.csv`
. .semicolon> WHERE columns[2] <> '' order by columns[4] desc limit 20;
+------------------------+------------------------------------------------------------------+------------+
|           id           |                              genres                              | popularity |
+------------------------+------------------------------------------------------------------+------------+
| id                     | genres                                                           | popularity |
| 4q3ewBCX7sLwd24euuV69X | latin, reggaeton, trap latino                                    | 98         |
| 06HL4z0CvFAxyc27GXpf02 | pop, post-teen pop                                               | 98         |
| 1Xyo4u8uXC1ZmMpatF05PJ | canadian contemporary r&b, canadian pop, pop                     | 96         |
| 3Nrfpe0tUJi4K4DXYWgMUX | k-pop, k-pop boy group                                           | 96         |
| 4MCBfE4596Uoi2O4DtmEMz | chicago rap, melodic rap                                         | 96         |
| 66CXWjxzNUsdJxJ2JdwvnR | pop, post-teen pop                                               | 95         |
| 1vyhD5VmyZ7KMfW5gqLgo5 | latin, reggaeton, reggaeton colombiano, trap latino              | 95         |
| 6M2wZ9GZgrQXHCFfjv46we | dance pop, pop, uk pop                                           | 95         |
| 7iK8PXO48WeuP03g8YR51W | trap latino                                                      | 95         |
| 7dGJo4pcD2V6oG8kP0tJRR | detroit hip hop, hip hop, rap                                    | 94         |
| 0Y5tJX1MQlPlqiwlOH1tJY | rap, slap house                                                  | 94         |
| 1i8SpTcr7yvPOmcqrbnVXY | latin, puerto rican pop, reggaeton, trap latino                  | 93         |
| 0du5cEVh5yTK9QJze8zA0C | dance pop, pop, post-teen pop                                    | 93         |
| 246dkjvS1zLTtiykXe5h60 | dfw rap, melodic rap, rap                                        | 93         |
| 4r63FhuTkUYltbVAg5TQnk | north carolina hip hop, rap                                      | 93         |
| 1mcTU81TzQhprhouKaTkpq | puerto rican pop, trap latino                                    | 93         |
| 6qqNVTkY8uBg9cP3Jd7DAH | electropop, pop                                                  | 92         |
| 5pKCCKE2ajJHZ9KAiaK11H | barbadian pop, dance pop, pop, post-teen pop, urban contemporary | 92         |
| 6LuN9FCkKOj5PcnpouEgny | alternative r&b, pop                                             | 92         |
+------------------------+------------------------------------------------------------------+------------+
20 rows selected (1.277 seconds)
```

5- Shown the top 20 artists who has the higher no of tracks recorded.



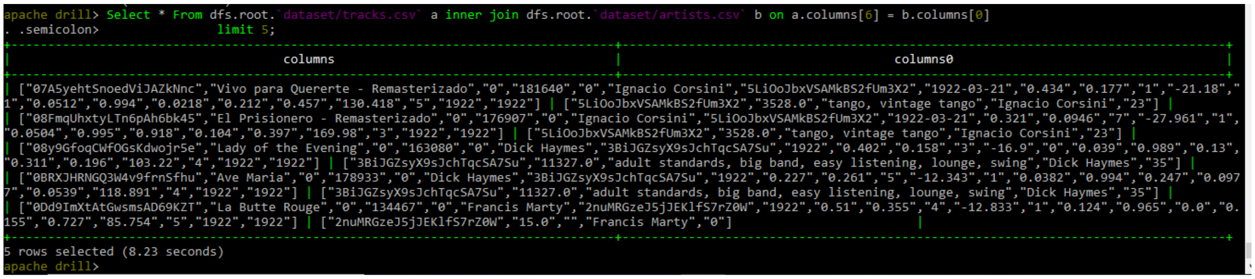
```
apache drill> SELECT columns[5] AS Artists,
. .semicolon> count(columns[1]) AS no_of_tracks
. .semicolon> FROM dfs.root.`dataset/tracks.csv`
. .semicolon> group by columns[5] order by count(columns[1]) desc limit 20;
+--------------------------+--------------+
|         Artists          | no_of_tracks |
+--------------------------+--------------+
| Die drei ???             | 3856         |
| TKKG Retro-Archiv        | 2006         |
| Francisco Canaro         | 1925         |
| Lata Mangeshkar          | 1789         |
| Johann Sebastian Bach    | 1662         |
| Benjamin Blümchen        | 1485         |
| Wolfgang Amadeus Mozart  | 1483         |
| Bibi Blocksberg          | 1440         |
|  Wiener Philharmoniker   | 975          |
| Giuseppe Verdi           | 966          |
| Tintin                   | 919          |
| S. P. Balasubrahmanyam   | 919          |
|  Bert-Åke Varg           | 905          |
|  Tomas Bolme             | 905          |
| Bibi und Tina            | 900          |
| Ella Fitzgerald          | 898          |
| Ludwig van Beethoven     | 871          |
| Tadeusz Dolega Mostowicz | 838          |
| Fünf Freunde             | 812          |
| Georgette Heyer          | 796          |
+--------------------------+--------------+
20 rows selected (2.66 seconds)
```

6- Shown the top 20 artists of 1999 who has the higher no of tracks recorded.



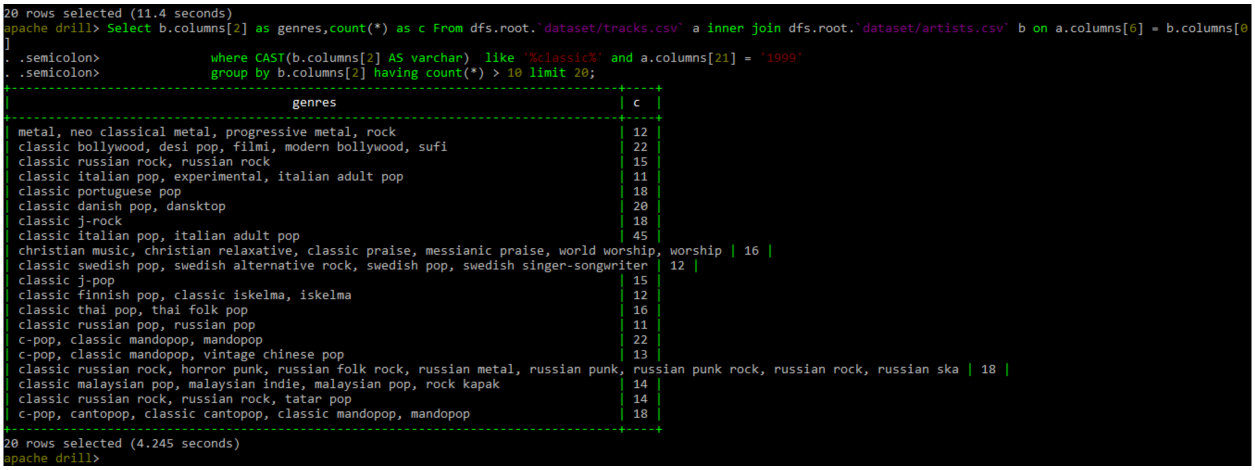
```
apache drill> SELECT columns[5] AS Artists,
. .semicolon> count(columns[1]) AS no_of_tracks
. .semicolon> FROM dfs.root.`dataset/tracks.csv` where columns[21] = '1999'
. .semicolon> group by columns[5] order by count(columns[1]) desc limit 20;
+------------------------------+--------------+
|           Artists            | no_of_tracks |
+------------------------------+--------------+
| Die drei ???                 | 240          |
| Johann Sebastian Bach        | 72           |
| Bibi und Tina                | 70           |
|  Philippe Herreweghe         | 67           |
|  Collegium Vocale Gent       | 67           |
| Fabrizio De André            | 53           |
| Bibi Blocksberg              | 46           |
| Los Caminantes               | 41           |
| TKKG                         | 41           |
| Disney - Der König der Löwen | 37           |
|  Alka Yagnik                 | 37           |
| Müşfik Kenter                | 36           |
| Notis Sfakianakis            | 36           |
| Dr. Dre                      | 34           |
| Udit Narayan                 | 30           |
| Rabito                       | 29           |
| Los Hermanos Zuleta          | 29           |
| Jorge Oñate                  | 28           |
| Hariharan                    | 28           |
|  Ian Bostridge               | 27           |
+------------------------------+--------------+
20 rows selected (4.241 seconds)
```

This is an example of inner join in Drill.



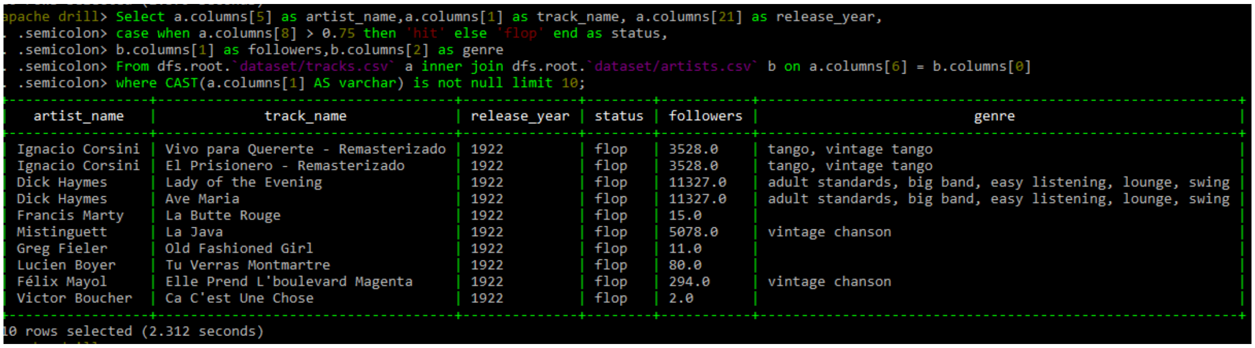
```
apache drill> Select * From dfs.root.`dataset/tracks.csv` a inner join dfs.root.`dataset/artists.csv` b on a.columns[6] = b.columns[0]
. .semicolon>                  limit 5;
+----------------------------------------------------------------------------------------------------+----------------------------------------------------------------------------------------------------+
|                                              columns                                               |                                              columns0                                              |
+----------------------------------------------------------------------------------------------------+----------------------------------------------------------------------------------------------------+
| ["07A5yehtSnoedViJAZkNnc","Vivo para Quererte - Remasterizado","0","181640","0","Ignacio Corsini","5LiOoJbxVSAMkBS2fUm3X2","1922-03-21","0.434","0.177","1","-21.18","
1","0.0512","0.994","0.0218","0.212","0.457","130.418","5","1922","1922"] | ["5LiOoJbxVSAMkBS2fUm3X2","3528.0","tango, vintage tango","Ignacio Corsini","23"] |
| ["08FmqUhxtyLTn6pAh6bk45","El Prisionero - Remasterizado","0","176907","0","Ignacio Corsini","5LiOoJbxVSAMkBS2fUm3X2","1922-03-21","0.321","0.0946","7","-27.961","1",
"0.0504","0.995","0.918","0.104","0.397","169.98","3","1922","1922"] | ["5LiOoJbxVSAMkBS2fUm3X2","3528.0","tango, vintage tango","Ignacio Corsini","23"] |
| ["08y9GfoqCWfOGsKdwojr5e","Lady of the Evening","0","163080","0","Dick Haymes","3BiJGZsyX9sJchTqcSA7Su","1922","0.402","0.158","3","-16.9","0","0.039","0.989","0.13",
"0.311","0.196","103.22","4","1922","1922"] | ["3BiJGZsyX9sJchTqcSA7Su","11327.0","adult standards, big band, easy listening, lounge, swing","Dick Haymes","35"] |
| ["0BRXJHRNGQ3W4v9frnSfhu","Ave Maria","0","178933","0","Dick Haymes","3BiJGZsyX9sJchTqcSA7Su","1922","0.227","0.261","5","-12.343","1","0.0382","0.994","0.247","0.097
7","0.0539","118.891","4","1922","1922"] | ["3BiJGZsyX9sJchTqcSA7Su","11327.0","adult standards, big band, easy listening, lounge, swing","Dick Haymes","35"] |
| ["0Dd9ImXtAtGwsmsAD69KZT","La Butte Rouge","0","134467","0","Francis Marty","2nuMRGzeJ5jJEKlfS7rZ0W","1922","0.51","0.355","4","-12.833","1","0.124","0.965","0.0","0.
155","0.727","85.754","5","1922","1922"] | ["2nuMRGzeJ5jJEKlfS7rZ0W","15.0","","Francis Marty","0"]                                   |
+----------------------------------------------------------------------------------------------------+----------------------------------------------------------------------------------------------------+
5 rows selected (8.23 seconds)
apache drill>
```

7- Show the top 20 classical genres of 1999 which has the highest not of records.



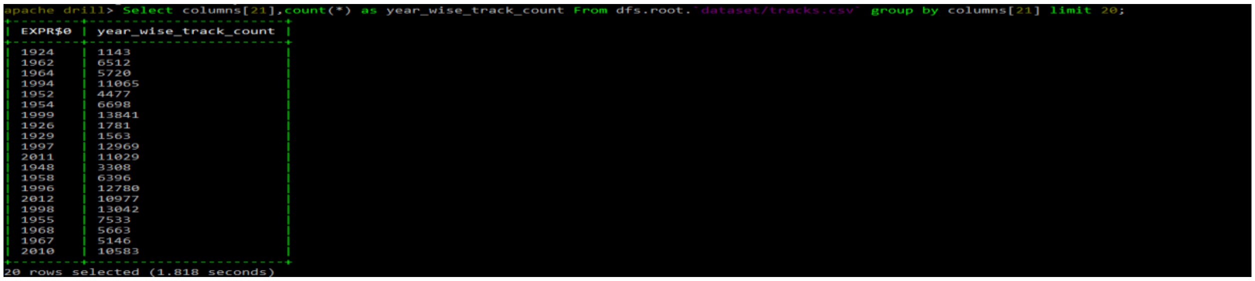
```
20 rows selected (11.4 seconds)
apache drill> Select b.columns[2] as genres,count(*) as c From dfs.root.`dataset/tracks.csv` a inner join dfs.root.`dataset/artists.csv` b on a.columns[6] = b.columns[0
]
. .semicolon>              where CAST(b.columns[2] AS varchar)  like '%classic%' and a.columns[21] = '1999'
. .semicolon>              group by b.columns[2] having count(*) > 10 limit 20;
+----------------------------------------------------------------------+----+
|                               genres                                 | c  |
+----------------------------------------------------------------------+----+
| metal, neo classical metal, progressive metal, rock                  | 12 |
| classic bollywood, desi pop, filmi, modern bollywood, sufi           | 22 |
| classic russian rock, russian rock                                   | 15 |
| classic italian pop, experimental, italian adult pop                 | 11 |
| classic portuguese pop                                               | 18 |
| classic danish pop, dansktop                                         | 20 |
| classic j-rock                                                       | 18 |
| classic italian pop, italian adult pop                               | 45 |
| christian music, christian relaxative, classic praise, messianic praise, world worship, worship | 16 |
| classic swedish pop, swedish alternative rock, swedish pop, swedish singer-songwriter | 12 |
| classic j-pop                                                        | 15 |
| classic finnish pop, classic iskelma, iskelma                        | 12 |
| classic thai pop, thai folk pop                                      | 16 |
| classic russian pop, russian pop                                     | 11 |
| c-pop, classic mandopop, mandopop                                    | 22 |
| c-pop, classic mandopop, vintage chinese pop                         | 13 |
| classic russian rock, horror punk, russian folk rock, russian metal, russian punk, russian punk rock, russian rock, russian ska | 18 |
| classic malaysian pop, malaysian indie, malaysian pop, rock kapak    | 14 |
| classic russian rock, russian rock, tatar pop                        | 14 |
| c-pop, cantopop, classic cantopop, classic mandopop, mandopop        | 18 |
+----------------------------------------------------------------------+----+
20 rows selected (4.245 seconds)
apache drill>
```

8- Show all the artist, their track name, year, hit/flop status on the basis of danceability, followers and genres.



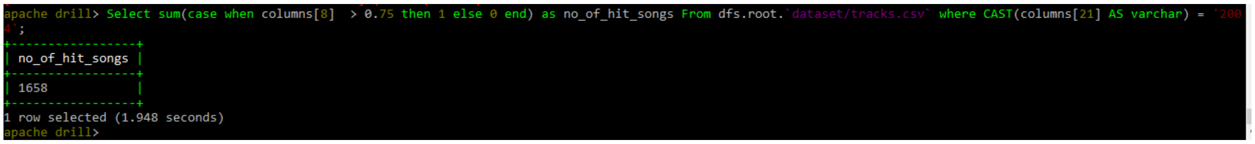
```
apache drill> Select a.columns[5] as artist_name,a.columns[1] as track_name, a.columns[21] as release_year,
. .semicolon> case when a.columns[8] > 0.75 then 'hit' else 'flop' end as status,
. .semicolon> b.columns[1] as followers,b.columns[2] as genre
. .semicolon> From dfs.root.`dataset/tracks.csv` a inner join dfs.root.`dataset/artists.csv` b on a.columns[6] = b.columns[0]
. .semicolon> where CAST(a.columns[1] AS varchar) is not null limit 10;
+-----------------+---------------------------------------+--------------+--------+-----------+------------------------------------------------------+
|  artist_name    |              track_name               | release_year | status | followers |                        genre                         |
+-----------------+---------------------------------------+--------------+--------+-----------+------------------------------------------------------+
| Ignacio Corsini | Vivo para Quererte - Remasterizado    | 1922         | flop   | 3528.0    | tango, vintage tango                                 |
| Ignacio Corsini | El Prisionero - Remasterizado         | 1922         | flop   | 3528.0    | tango, vintage tango                                 |
| Dick Haymes     | Lady of the Evening                   | 1922         | flop   | 11327.0   | adult standards, big band, easy listening, lounge, swing |
| Dick Haymes     | Ave Maria                             | 1922         | flop   | 11327.0   | adult standards, big band, easy listening, lounge, swing |
| Francis Marty   | La Butte Rouge                        | 1922         | flop   | 15.0      |                                                      |
| Mistinguett     | La Java                               | 1922         | flop   | 5078.0    | vintage chanson                                      |
| Greg Fieler     | Old Fashioned Girl                    | 1922         | flop   | 11.0      |                                                      |
| Lucien Boyer    | Tu Verras Montmartre                  | 1922         | flop   | 80.0      |                                                      |
| Félix Mayol     | Elle Prend L'boulevard Magenta        | 1922         | flop   | 294.0     | vintage chanson                                      |
| Victor Boucher  | Ca C'est Une Chose                    | 1922         | flop   | 2.0       |                                                      |
+-----------------+---------------------------------------+--------------+--------+-----------+------------------------------------------------------+
10 rows selected (2.312 seconds)
```

9- Show all the songs recorded in each year.



```
apache drill> Select columns[21],count(*) as year_wise_track_count From dfs.root.`dataset/tracks.csv` group by columns[21] limit 20;
+---------+-------------------------+
| EXPR$0  | year_wise_track_count   |
+---------+-------------------------+
| 1924    | 1143                    |
| 1962    | 6512                    |
| 1964    | 5720                    |
| 1994    | 11065                   |
| 1952    | 4477                    |
| 1954    | 6698                    |
| 1999    | 13841                   |
| 1926    | 1781                    |
| 1929    | 1563                    |
| 1997    | 12969                   |
| 2011    | 11029                   |
| 1948    | 3308                    |
| 1958    | 6396                    |
| 1996    | 12780                   |
| 2012    | 10977                   |
| 1998    | 13042                   |
| 1955    | 7533                    |
| 1968    | 5663                    |
| 1967    | 5146                    |
| 2010    | 10583                   |
+---------+-------------------------+
20 rows selected (1.818 seconds)
```

10- Show the total no of hit songs in 2004.

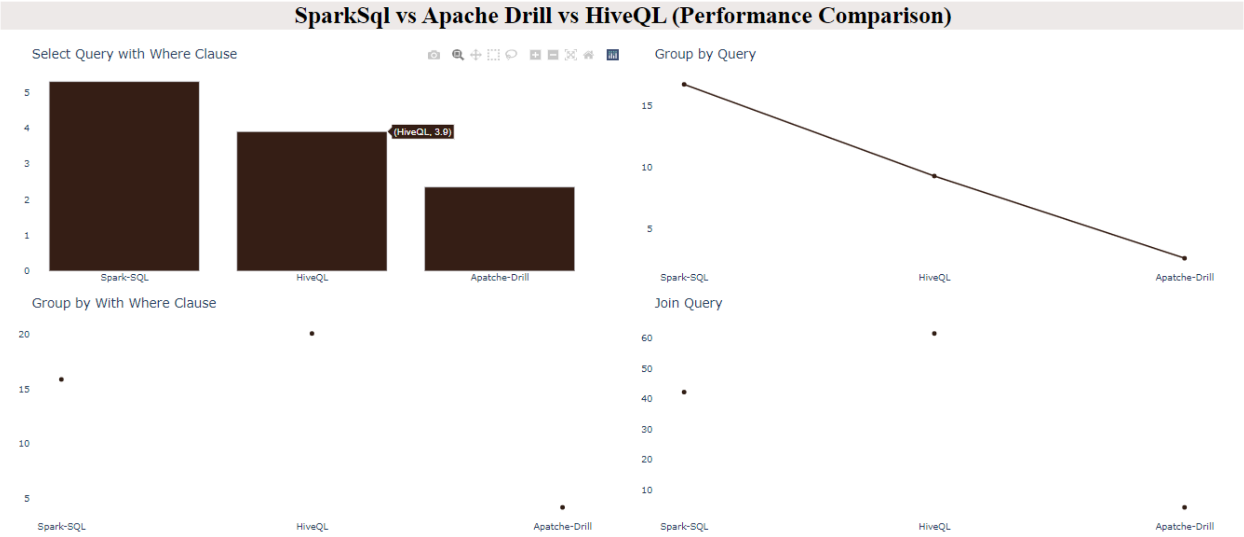


```
apache drill> Select sum(case when columns[8]  > 0.75 then 1 else 0 end) as no_of_hit_songs From dfs.root.`dataset/tracks.csv` where CAST(columns[21] AS varchar) = '200
4';
+-----------------+
| no_of_hit_songs |
+-----------------+
| 1658            |
+-----------------+
1 row selected (1.948 seconds)
apache drill>
```

Comparison:

Throughout this project, we have calculated the time take by each query for SparkSQL, HiveQL and ApacheDrill and that is also represented in the graph.

In the Y-axis, we have seconds and in the X-axis the three SQL based engines, and we have plot their performance on the basis of same query structure.


SparkSql vs Apache Drill vs HiveQL (Performance Comparison)

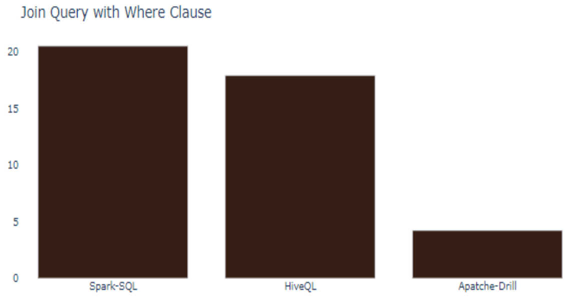
Select Query with Where Clause

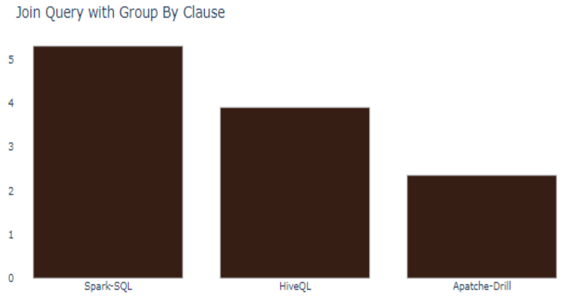
Group by Query

Group by With Where Clause

Join Query

From the above graph, we can observe that ApacheDrill has proved to be the efficient in all query structure. Drill is a low-latency distributed query engine for large scale datasets. Even thou HiveQL and Sprak performance is also good, but Drill has exceptionally proven to be a best Query engine for our dataset.


Join Query with Where Clause


Join Query with Group By Clause