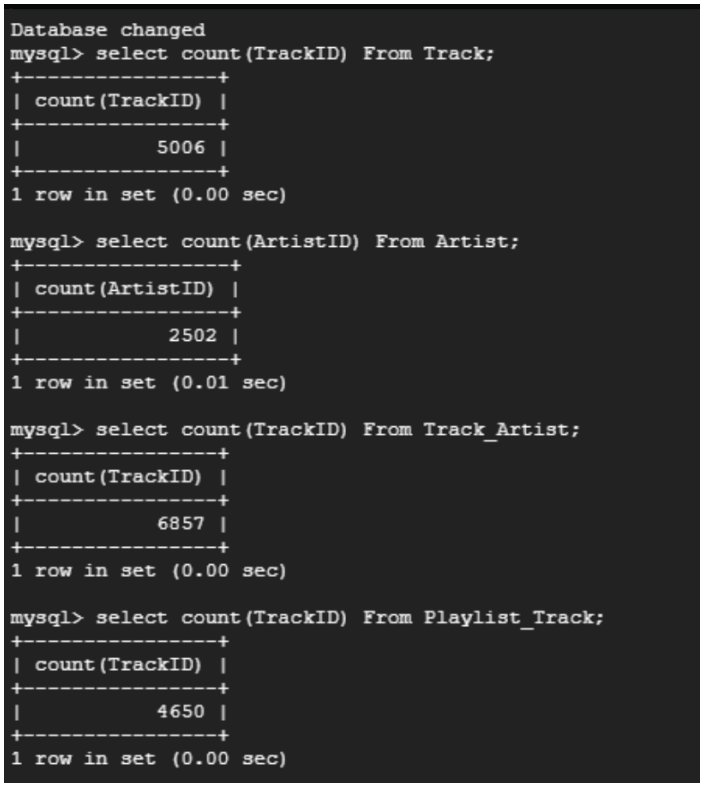
Part 1:

DDL Commands

```
CREATE TABLE User (
   UserID VARCHAR(255) PRIMARY KEY,
   Name VARCHAR(255),
   Email VARCHAR(255) UNIQUE,
   HashedPassword VARCHAR(255)
);


CREATE TABLE SpotifyProfile (
   UserID VARCHAR(255) PRIMARY KEY,
   DisplayName VARCHAR(255),
   ProfileUrl VARCHAR(255),
   ImageUrl VARCHAR(255),
   APIKey VARCHAR(255)
);


CREATE TABLE LinkedProfile (
   UserID VARCHAR(255),
   SpotifyProfileID VARCHAR(255),
   FOREIGN KEY (UserID) REFERENCES User(UserID),
   FOREIGN KEY (SpotifyProfileID) REFERENCES SpotifyProfile(UserID),
   PRIMARY KEY (UserID, SpotifyProfileID)
);


CREATE TABLE Playlist (
   PlaylistID VARCHAR(255) PRIMARY KEY,
   DisplayName VARCHAR(255),
   PlaylistName VARCHAR(255)
);

CREATE TABLE Playlist_Track (
   TrackID VARCHAR(255),
   PlaylistID VARCHAR(255),
```

```
    FOREIGN KEY (PlaylistID) REFERENCES Playlist(PlaylistID),
    FOREIGN KEY (TrackID) REFERENCES Track(TrackID),
    PRIMARY KEY (TrackID, PlaylistID)
);


CREATE TABLE Track (
    TrackID VARCHAR(255) PRIMARY KEY,
    Tempo DECIMAL,
    Valence DECIMAL,
    Liveness DECIMAL,
    Instrumentalness DECIMAL,
    Acousticness DECIMAL,
    Speechiness DECIMAL,
    Mode INT,
    MusicKey INT,
    Energy DECIMAL,
    Danceability DECIMAL,
    Duration_ms INT,
    Popularity INT,
    Track_name VARCHAR(255),
    Album_name VARCHAR(255)
);

CREATE TABLE Track_Artist (
        TrackID VARCHAR(255),
        ArtistID VARCHAR(255),
        FOREIGN KEY (ArtistID) REFERENCES Artist(ArtistID),
FOREIGN KEY (TrackID) REFERENCES Track(TrackID),
        PRIMARY KEY (TrackID, ArtistID)
);
CREATE TABLE Top_Saved_Track (
    TrackID VARCHAR(255),
    UserID VARCHAR(255),
    DateSaved DATE,
    ShortRating INT,
    MediumRating INT,
    LargeRating INT,
    FOREIGN KEY (TrackID) REFERENCES Track(TrackID),
    FOREIGN KEY (UserID) REFERENCES SpotifyProfile(UserID)
```

```
);


CREATE TABLE Artist (
    ArtistID VARCHAR(255) PRIMARY KEY,
    ArtistName VARCHAR(255),
    ImageUrl VARCHAR(255),
    ProfileUrl VARCHAR(255)
);


CREATE TABLE Top_Saved_Artist (
    ArtistID VARCHAR(255),
    UserID VARCHAR(255),
    DateSaved DATE,
    ShortRating INT,
    MediumRating INT,
    LargeRating INT,
    FOREIGN KEY (ArtistID) REFERENCES Artist(ArtistID),
    FOREIGN KEY (UserID) REFERENCES SpotifyProfile(UserID)
);
```



```
Database changed
mysql> select count(TrackID) From Track;
+----------------+
| count(TrackID) |
+----------------+
|           5006 |
+----------------+
1 row in set (0.00 sec)

mysql> select count(ArtistID) From Artist;
+-----------------+
| count(ArtistID) |
+-----------------+
|            2502 |
+-----------------+
1 row in set (0.01 sec)

mysql> select count(TrackID) From Track_Artist;
+----------------+
| count(TrackID) |
+----------------+
|           6857 |
+----------------+
1 row in set (0.00 sec)

mysql> select count(TrackID) From Playlist_Track;
+----------------+
| count(TrackID) |
+----------------+
|           4650 |
+----------------+
1 row in set (0.00 sec)
```

Advanced SQL queries

Return the top 15 artists with the highest average energy

```
SELECT a.ArtistName, AVG(t.popularity) as avg_popularity
FROM Track t
JOIN Track_Artist ta ON ta.TrackID = t.TrackID
JOIN Artist a on a.ArtistID = ta.ArtistID
GROUP BY a.ArtistName
ORDER BY avg_popularity DESC
LIMIT 15;
```

```
mysql> SELECT a.ArtistName, AVG(t.popularity) as avg_popularity
    -> FROM Track t
    -> JOIN Track_Artist ta ON ta.TrackID = t.TrackID
    -> JOIN Artist a on a.ArtistID = ta.ArtistID
    -> GROUP BY a.ArtistName
    -> ORDER BY avg_popularity DESC
    -> LIMIT 15
    -> ;
+------------------------------+----------------+
| ArtistName                   | avg_popularity |
+------------------------------+----------------+
| Kacey Musgraves              |        93.0000 |
| Kali Uchis                   |        92.0000 |
| girl in red                  |        91.0000 |
| Hotel Ugly                   |        90.0000 |
| a-ha                         |        89.0000 |
| Dwele                        |        89.0000 |
| John Legend                  |        87.0000 |
| Oliver Anthony Music         |        87.0000 |
| Foo Fighters                 |        87.0000 |
| R.E.M.                       |        87.0000 |
| Dua Lipa                     |        87.0000 |
| One Direction                |        86.6667 |
| Richy Mitch & The Coal Miners |        86.0000 |
| The Chainsmokers             |        86.0000 |
| Eurythmics                   |        86.0000 |
+------------------------------+----------------+
15 rows in set (0.03 sec)
```

Returns the Playlist with the number of tracks with popularity greater than 10 (can change value)

```
SELECT p.PlaylistID, p.PlaylistName, COUNT(t.TrackId) AS TrackCount
FROM Playlist p
JOIN Playlist_Track pt ON p.PlaylistID = pt.PlaylistID
JOIN Track t on pt.TrackID = t.TrackID
WHERE t.Popularity > 10
GROUP BY p.PlaylistID, p.PlaylistName
ORDER BY TrackCount DESC
LIMIT 15;
```

```
mysql> SELECT p.PlaylistID, p.PlaylistName, COUNT(t.TrackId) AS TrackCount
    -> FROM Playlist p
    -> JOIN Playlist_Track pt ON p.PlaylistID = pt.PlaylistID
    -> JOIN Track t on pt.TrackID = t.TrackID
    -> WHERE t.Popularity > 10
    -> GROUP BY p.PlaylistID, p.PlaylistName
    -> ORDER BY TrackCount DESC
    -> LIMIT 15;
+------------------------+---------------------------+------------+
| PlaylistID             | PlaylistName              | TrackCount |
+------------------------+---------------------------+------------+
| 2czJrrikJEKKQTnd7tBU2T | Childhood Classics        |       1252 |
| 2s9R059mmdc8kz6lrUqZZd | Coffee Shop Vibes         |        501 |
| 6aytqONARLRPsI57VpQHwU | shitpost                  |        160 |
| 37i9dQZF1DX2sUQwD7tbmL | Feel-Good Indie Rock      |        148 |
| 5ifMsPihlkYEGCakqfqj17 | Classical Piano Essentials |        127 |
| 37i9dQZF1DX2Nc3B70tvx0 | Front Page Indie          |        114 |
| 37i9dQZF1F0sijgNaJdgit | Your Top Songs 2022       |        101 |
| 37i9dQZF1DX4OzrY981I1W | my life is a movie        |        100 |
| 15bfV0CrzO4dK7h4jzg0vY | Gym pump                  |         73 |
| 6ddKIHUn32p3Y620jAzHg8 | aluminum                  |         70 |
| 37i9dQZF1DXcbAIldMQMIs | text me back              |         70 |
| 37i9dQZF1DX6R7QUWePReA | Christmas Classics        |         69 |
| 37i9dQZF1DZ06evO4cWDcc | This Is Jimi Hendrix      |         51 |
| 37i9dQZF1EIgVFcj8TnmaA | Instrumental Metal Mix    |         50 |
| 37i9dQZF1DZ06evO1NyWWI | This Is Led Zeppelin      |         49 |
+------------------------+---------------------------+------------+
15 rows in set (0.02 sec)
```

Part 2: Analysis

<u>Before Adding Index:</u>

EXPLAIN ANALYZE
SELECT a.ArtistName, AVG(t.popularity) as avg_popularity
FROM Track t
JOIN Track_Artist ta ON ta.TrackID = t.TrackID
JOIN Artist a on a.ArtistID = ta.ArtistID
GROUP BY a.ArtistName
ORDER BY avg_popularity DESC
LIMIT 15;

```
| EXPLAIN
| -> Limit: 15 row(s)  (actual time=37.655..37.657 rows=15 loops=1)
    -> Sort: avg_popularity DESC, limit input to 15 row(s) per chunk  (actual time=37.653..37.654 rows=15 loops=1)
        -> Table scan on <temporary>  (actual time=36.455..36.859 rows=2501 loops=1)
            -> Aggregate using temporary table  (actual time=36.453..36.453 rows=2501 loops=1)
                -> Nested loop inner join  (cost=4010.27 rows=6443) (actual time=0.789..29.107 rows=6857 loops=1)
                    -> Nested loop inner join  (cost=1755.06 rows=6443) (actual time=0.745..12.311 rows=6857 loops=1)
                        -> Table scan on a  (cost=250.50 rows=2440) (actual time=0.706..1.851 rows=2502 loops=1)
                        -> Covering index lookup on ta using ArtistID (ArtistID=a.ArtistID)  (cost=0.35 rows=3) (actual time=0.002..0.004 rows=3 loops=2502)
                    -> Single-row index lookup on t using PRIMARY (TrackID=ta.TrackID)  (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=6857)
 |
1 row in set (0.04 sec)
```

EXPLAIN ANALYZE
SELECT p.PlaylistID, p.PlaylistName, COUNT(t.TrackId) AS TrackCount
FROM Playlist p
JOIN Playlist_Track pt ON p.PlaylistID = pt.PlaylistID
JOIN Track t on pt.TrackID = t.TrackID
WHERE t.Popularity > 10
GROUP BY p.PlaylistID, p.PlaylistName
ORDER BY TrackCount DESC
LIMIT 15;

```
mysql> EXPLAIN ANALYZE
    -> SELECT p.PlaylistID, p.PlaylistName, COUNT(t.TrackId) AS TrackCount
    -> FROM Playlist p
    -> JOIN Playlist_Track pt ON p.PlaylistID = pt.PlaylistID
    -> JOIN Track t on pt.TrackID = t.TrackID
    -> WHERE t.Popularity > 10
    -> GROUP BY p.PlaylistID, p.PlaylistName
    -> ORDER BY TrackCount DESC
    -> LIMIT 15;
| EXPLAIN
| -> Limit: 15 row(s)  (actual time=19.874..19.876 rows=15 loops=1)
    -> Sort: TrackCount DESC, limit input to 15 row(s) per chunk  (actual time=19.873..19.874 rows=15 loops=1)
        -> Table scan on <temporary>  (actual time=19.833..19.845 rows=33 loops=1)
            -> Aggregate using temporary table  (actual time=19.830..19.830 rows=33 loops=1)
                -> Nested loop inner join  (cost=745.43 rows=478) (actual time=0.152..15.262 rows=3818 loops=1)
                    -> Nested loop inner join  (cost=243.01 rows=1436) (actual time=0.126..2.810 rows=4650 loops=1)
                        -> Table scan on p  (cost=3.55 rows=33) (actual time=0.075..0.102 rows=33 loops=1)
                        -> Covering index lookup on pt using PlaylistID (PlaylistID=p.PlaylistID)  (cost=3.04 rows=44) (actual time=0.018..0.073 rows=141 loops=33)
                    -> Filter: (t.Popularity > 10)  (cost=0.25 rows=0.3) (actual time=0.002..0.003 rows=1 loops=4650)
                        -> Single-row index lookup on t using PRIMARY (TrackID=pt.TrackID)  (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=4650)
 |
1 row in set (0.02 sec)
```

CREATE INDEX track_artist_track_id_index ON Track_Artist(TrackID);

```
mysql> SHOW INDEX FROM Track;
+-------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+-------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Track |          0 | PRIMARY  |            1 | TrackID     | A         |        4830 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
+-------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
1 row in set (0.01 sec)
```

```
| EXPLAIN
                                                                                  |
+-----------------------------------------------------------------------------------+
| -> Limit: 15 row(s)  (actual time=37.655..37.657 rows=15 loops=1)
    -> Sort: avg_popularity DESC, limit input to 15 row(s) per chunk  (actual time=37.653..37.654 rows=15 loops=1)
        -> Table scan on <temporary>  (actual time=36.455..36.859 rows=2501 loops=1)
            -> Aggregate using temporary table  (actual time=36.453..36.453 rows=2501 loops=1)
                -> Nested loop inner join  (cost=4010.27 rows=6443) (actual time=0.789..29.107 rows=6857 loops=1)
                    -> Nested loop inner join  (cost=1755.06 rows=6443) (actual time=0.745..12.311 rows=6857 loops=1)
                        -> Table scan on a  (cost=250.50 rows=2440) (actual time=0.706..1.851 rows=2502 loops=1)
                        -> Covering index lookup on ta using ArtistID (ArtistID=a.ArtistID)  (cost=0.35 rows=3) (actual time=0.002..0.004 rows=3 loops=2502)
                    -> Single-row index lookup on t using PRIMARY (TrackID=ta.TrackID)  (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=6857)
 |
+-----------------------------------------------------------------------------------+
1 row in set (0.04 sec)
```

CREATE INDEX track_artist_artist_id_index ON Track_Artist(ArtistID);

```
mysql> SHOW INDEX FROM Track_Artist;
+--------------+------------+------------------------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------
+---------+------------+
| Table        | Non_unique | Key_name                     | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment
| Visible | Expression |
+--------------+------------+------------------------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------
+---------+------------+
| Track_Artist |          0 | PRIMARY                      |            1 | TrackID     | A         |        3810 |     NULL |   NULL |      | BTREE      |         |
| YES     | NULL       |
| Track_Artist |          0 | PRIMARY                      |            2 | ArtistID    | A         |        5822 |     NULL |   NULL |      | BTREE      |         |
| YES     | NULL       |
| Track_Artist |          1 | track_artist_track_id_index  |            1 | TrackID     | A         |        5006 |     NULL |   NULL |      | BTREE      |         |
| YES     | NULL       |
| Track_Artist |          1 | track_artist_artist_id_index |            1 | ArtistID    | A         |        2502 |     NULL |   NULL |      | BTREE      |         |
| YES     | NULL       |
+--------------+------------+------------------------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------
+---------+------------+
4 rows in set (0.01 sec)
```


```
| EXPLAIN
                                                                                  |
+-----------------------------------------------------------------------------------+
| -> Limit: 15 row(s)  (actual time=37.655..37.657 rows=15 loops=1)
    -> Sort: avg_popularity DESC, limit input to 15 row(s) per chunk  (actual time=37.653..37.654 rows=15 loops=1)
        -> Table scan on <temporary>  (actual time=36.455..36.859 rows=2501 loops=1)
            -> Aggregate using temporary table  (actual time=36.453..36.453 rows=2501 loops=1)
                -> Nested loop inner join  (cost=4010.27 rows=6443) (actual time=0.789..29.107 rows=6857 loops=1)
                    -> Nested loop inner join  (cost=1755.06 rows=6443) (actual time=0.745..12.311 rows=6857 loops=1)
                        -> Table scan on a  (cost=250.50 rows=2440) (actual time=0.706..1.851 rows=2502 loops=1)
                        -> Covering index lookup on ta using ArtistID (ArtistID=a.ArtistID)  (cost=0.35 rows=3) (actual time=0.002..0.004 rows=3 loops=2502)
                    -> Single-row index lookup on t using PRIMARY (TrackID=ta.TrackID)  (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=6857)
 |
+-----------------------------------------------------------------------------------+
1 row in set (0.04 sec)
```


CREATE INDEX track_artist_track_id_artist_id_index ON Track_Artist(TrackID, ArtistID);

```
mysql> SHOW INDEX FROM Track_Artist;
+--------------+------------+---------------------------------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+------
---------+---------+------------+
| Table        | Non_unique | Key_name                              | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index
_comment | Visible | Expression |
+--------------+------------+---------------------------------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+------
---------+---------+------------+
| Track_Artist |          0 | PRIMARY                               |            1 | TrackID     | A         |        3810 |     NULL |   NULL |      | BTREE      |         |
         | YES     | NULL       |
| Track_Artist |          0 | PRIMARY                               |            2 | ArtistID    | A         |        5822 |     NULL |   NULL |      | BTREE      |         |
         | YES     | NULL       |
| Track_Artist |          1 | track_artist_track_id_index           |            1 | TrackID     | A         |        5006 |     NULL |   NULL |      | BTREE      |         |
         | YES     | NULL       |
| Track_Artist |          1 | track_artist_artist_id_index          |            1 | ArtistID    | A         |        2502 |     NULL |   NULL |      | BTREE      |         |
         | YES     | NULL       |
| Track_Artist |          1 | track_artist_track_id_artist_id_index |            1 | TrackID     | A         |        5006 |     NULL |   NULL |      | BTREE      |         |
         | YES     | NULL       |
| Track_Artist |          1 | track_artist_track_id_artist_id_index |            2 | ArtistID    | A         |        6388 |     NULL |   NULL |      | BTREE      |         |
         | YES     | NULL       |
+--------------+------------+---------------------------------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+------
---------+---------+------------+
6 rows in set (0.01 sec)
```

```
| EXPLAIN

                                                                                                          |
+---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------+
| -> Limit: 15 row(s)  (actual time=37.655..37.657 rows=15 loops=1)
    -> Sort: avg_popularity DESC, limit input to 15 row(s) per chunk  (actual time=37.653..37.654 rows=15 loops=1)
        -> Table scan on <temporary>  (actual time=36.455..36.859 rows=2501 loops=1)
            -> Aggregate using temporary table  (actual time=36.453..36.453 rows=2501 loops=1)
                -> Nested loop inner join  (cost=4010.27 rows=6443) (actual time=0.789..29.107 rows=6857 loops=1)
                    -> Nested loop inner join  (cost=1755.06 rows=6443) (actual time=0.745..12.311 rows=6857 loops=1)
                        -> Table scan on a  (cost=250.50 rows=2440) (actual time=0.706..1.851 rows=2502 loops=1)
                        -> Covering index lookup on ta using ArtistID (ArtistID=a.ArtistID)  (cost=0.35 rows=3) (actual time=0.002..0.004 rows=3 loops=2502)
                    -> Single-row index lookup on t using PRIMARY (TrackID=ta.TrackID)  (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=6857)
 |
+---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------+
1 row in set (0.04 sec)
```

Justification: We decided to go with the composite index (CREATE INDEX track_artist_track_id_artist_id_index ON Track_Artist(TrackID, ArtistID)) because we wanted to consider which index covers the most types of queries. With this index, we can be sure to incorporate a wider range of queries and ensure heightened versatility for the database. Additionally, we preserve maintenance overhead as some indexes may require more frequent reorganization or rebuilding, which we do not need to worry about for this index.

--

Second Query:

CREATE INDEX playlist_track_playlist_id_index ON Playlist_Track(PlaylistID);

```
mysql> SHOW INDEX FROM Playlist_Track;
+----------------+------------+----------------------------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------
------+---------+------------+
| Table          | Non_unique | Key_name                         | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_co
mment | Visible | Expression |
+----------------+------------+----------------------------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------
------+---------+------------+
| Playlist_Track |          0 | PRIMARY                          |            1 | TrackID     | A         |        3919 |     NULL |   NULL |      | BTREE      |         |
      | YES     | NULL       |
| Playlist_Track |          0 | PRIMARY                          |            2 | PlaylistID  | A         |        4232 |     NULL |   NULL |      | BTREE      |         |
      | YES     | NULL       |
| Playlist_Track |          1 | playlist_track_playlist_id_index |            1 | PlaylistID  | A         |          33 |     NULL |   NULL |      | BTREE      |         |
      | YES     | NULL       |
+----------------+------------+----------------------------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------
------+---------+------------+
3 rows in set (0.01 sec)
```

```
mysql> EXPLAIN ANALYZE
    -> SELECT p.PlaylistID, p.PlaylistName, COUNT(t.TrackId) AS TrackCount
    -> FROM Playlist p
    -> JOIN Playlist_Track pt ON p.PlaylistID = pt.PlaylistID
    -> JOIN Track t on pt.TrackID = t.TrackID
    -> WHERE t.Popularity > 10
    -> GROUP BY p.PlaylistID, p.PlaylistName
    -> ORDER BY TrackCount DESC
    -> LIMIT 15;
+--------------------------------------------------------------------------------+
| EXPLAIN
                                                  |
+--------------------------------------------------------------------------------+
| -> Limit: 15 row(s)  (actual time=19.874..19.876 rows=15 loops=1)
    -> Sort: TrackCount DESC, limit input to 15 row(s) per chunk  (actual time=19.873..19.874 rows=15 loops=1)
        -> Table scan on <temporary>  (actual time=19.833..19.845 rows=33 loops=1)
            -> Aggregate using temporary table  (actual time=19.830..19.830 rows=33 loops=1)
                -> Nested loop inner join  (cost=745.43 rows=478) (actual time=0.152..15.262 rows=3818 loops=1)
                    -> Nested loop inner join  (cost=243.01 rows=1436) (actual time=0.126..2.810 rows=4650 loops=1)
                        -> Table scan on p  (cost=3.55 rows=33) (actual time=0.075..0.102 rows=33 loops=1)
                        -> Covering index lookup on pt using PlaylistID (PlaylistID=p.PlaylistID)  (cost=3.04 rows=44) (actual time=0.018..0.073 rows=141 loops=33)
                    -> Filter: (t.Popularity > 10)  (cost=0.25 rows=0.3) (actual time=0.002..0.003 rows=1 loops=4650)
                        -> Single-row index lookup on t using PRIMARY (TrackID=pt.TrackID)  (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=4650)
 |
+--------------------------------------------------------------------------------+
1 row in set (0.02 sec)
```

CREATE INDEX track_popularity_index ON Track(Popularity);

```
mysql> SHOW INDEX FROM Track;
+-------+------------+------------------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+--
----------+
| Table | Non_unique | Key_name               | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | E
xpression |
+-------+------------+------------------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+--
----------+
| Track |          0 | PRIMARY                |            1 | TrackID     | A         |        4830 |     NULL |   NULL |      | BTREE      |         |               | YES     | N
ULL       |
| Track |          1 | track_popularity_index |            1 | Popularity  | A         |          96 |     NULL |   NULL | YES  | BTREE      |         |               | YES     | N
ULL       |
+-------+------------+------------------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+--
----------+
2 rows in set (0.01 sec)
```

```
mysql> EXPLAIN ANALYZE
    -> SELECT p.PlaylistID, p.PlaylistName, COUNT(t.TrackId) AS TrackCount
    -> FROM Playlist p
    -> JOIN Playlist_Track pt ON p.PlaylistID = pt.PlaylistID
    -> JOIN Track t on pt.TrackID = t.TrackID
    -> WHERE t.Popularity > 10
    -> GROUP BY p.PlaylistID, p.PlaylistName
    -> ORDER BY TrackCount DESC
    -> LIMIT 15;
+--------------------------------------------------------------------------------+
| EXPLAIN
                                                  |
+--------------------------------------------------------------------------------+
| -> Limit: 15 row(s)  (actual time=19.874..19.876 rows=15 loops=1)
    -> Sort: TrackCount DESC, limit input to 15 row(s) per chunk  (actual time=19.873..19.874 rows=15 loops=1)
        -> Table scan on <temporary>  (actual time=19.833..19.845 rows=33 loops=1)
            -> Aggregate using temporary table  (actual time=19.830..19.830 rows=33 loops=1)
                -> Nested loop inner join  (cost=745.43 rows=478) (actual time=0.152..15.262 rows=3818 loops=1)
                    -> Nested loop inner join  (cost=243.01 rows=1436) (actual time=0.126..2.810 rows=4650 loops=1)
                        -> Table scan on p  (cost=3.55 rows=33) (actual time=0.075..0.102 rows=33 loops=1)
                        -> Covering index lookup on pt using PlaylistID (PlaylistID=p.PlaylistID)  (cost=3.04 rows=44) (actual time=0.018..0.073 rows=141 loops=33)
                    -> Filter: (t.Popularity > 10)  (cost=0.25 rows=0.3) (actual time=0.002..0.003 rows=1 loops=4650)
                        -> Single-row index lookup on t using PRIMARY (TrackID=pt.TrackID)  (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=4650)
 |
+--------------------------------------------------------------------------------+
1 row in set (0.02 sec)
```

CREATE INDEX playlist_track_playlist_id_track_id_index ON Playlist_Track(PlaylistID, TrackID);

```
mysql> SHOW INDEX FROM Playlist_Track;
+----------------+------------+-------------------------------------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+
---------------+---------+------------+
| Table          | Non_unique | Key_name                                  | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment |
 Index_comment | Visible | Expression |
+----------------+------------+-------------------------------------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+
---------------+---------+------------+
| Playlist_Track |          0 | PRIMARY                                   |            1 | TrackID     | A         |        3919 |     NULL |   NULL |      | BTREE      |         |
               | YES     | NULL       |
| Playlist_Track |          0 | PRIMARY                                   |            2 | PlaylistID  | A         |        4232 |     NULL |   NULL |      | BTREE      |         |
               | YES     | NULL       |
| Playlist_Track |          1 | playlist_track_playlist_id_index          |            1 | PlaylistID  | A         |          33 |     NULL |   NULL |      | BTREE      |         |
               | YES     | NULL       |
| Playlist_Track |          1 | playlist_track_playlist_id_track_id_index |            1 | PlaylistID  | A         |          33 |     NULL |   NULL |      | BTREE      |         |
               | YES     | NULL       |
| Playlist_Track |          1 | playlist_track_playlist_id_track_id_index |            2 | TrackID     | A         |        4650 |     NULL |   NULL |      | BTREE      |         |
               | YES     | NULL       |
+----------------+------------+-------------------------------------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+
---------------+---------+------------+
5 rows in set (0.01 sec)
```

```
mysql> EXPLAIN ANALYZE
    -> SELECT p.PlaylistID, p.PlaylistName, COUNT(t.TrackId) AS TrackCount
    -> FROM Playlist p
    -> JOIN Playlist_Track pt ON p.PlaylistID = pt.PlaylistID
    -> JOIN Track t on pt.TrackID = t.TrackID
    -> WHERE t.Popularity > 10
    -> GROUP BY p.PlaylistID, p.PlaylistName
    -> ORDER BY TrackCount DESC
    -> LIMIT 15;
+--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
------------------------------------------------------------------------------+
| EXPLAIN

                                                  |
+--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
------------------------------------------------------------------------------+
| -> Limit: 15 row(s)  (actual time=19.874..19.876 rows=15 loops=1)
    -> Sort: TrackCount DESC, limit input to 15 row(s) per chunk  (actual time=19.873..19.874 rows=15 loops=1)
        -> Table scan on <temporary>  (actual time=19.833..19.845 rows=33 loops=1)
            -> Aggregate using temporary table  (actual time=19.830..19.830 rows=33 loops=1)
                -> Nested loop inner join  (cost=745.43 rows=478) (actual time=0.152..15.262 rows=3818 loops=1)
                    -> Nested loop inner join  (cost=243.01 rows=1436) (actual time=0.126..2.810 rows=4650 loops=1)
                        -> Table scan on p  (cost=3.55 rows=33) (actual time=0.075..0.102 rows=33 loops=1)
                        -> Covering index lookup on pt using PlaylistID (PlaylistID=p.PlaylistID)  (cost=3.04 rows=44) (actual time=0.018..0.073 rows=141 loops=33)
                    -> Filter: (t.Popularity > 10)  (cost=0.25 rows=0.3) (actual time=0.002..0.003 rows=1 loops=4650)
                        -> Single-row index lookup on t using PRIMARY (TrackID=pt.TrackID)  (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=4650)
 |
+--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
------------------------------------------------------------------------------+
1 row in set (0.02 sec)
```

Justification: We decided to go with the composite index (CREATE INDEX playlist_track_playlist_id_track_id_index ON Playlist_Track(PlaylistID, TrackID)) because we wanted to consider which index covers the most types of queries. Additionally, we placed higher importance on query frequency, in which we wanted to consider which index is likely to optimize the most frequently executed queries. Likewise, since the number of playlists will be increasing, having a composite index that tracks the playlistID as well as the multiple tracks in it, we can use the index to continuously track the changes as it covers a wider range of queries.
