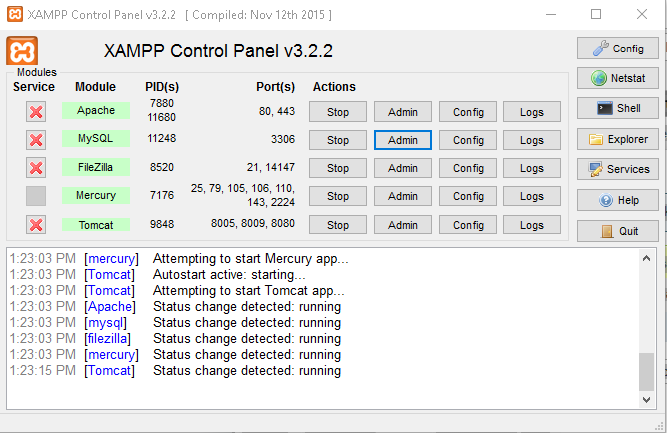
# Databases Lab 1 (MySQL)

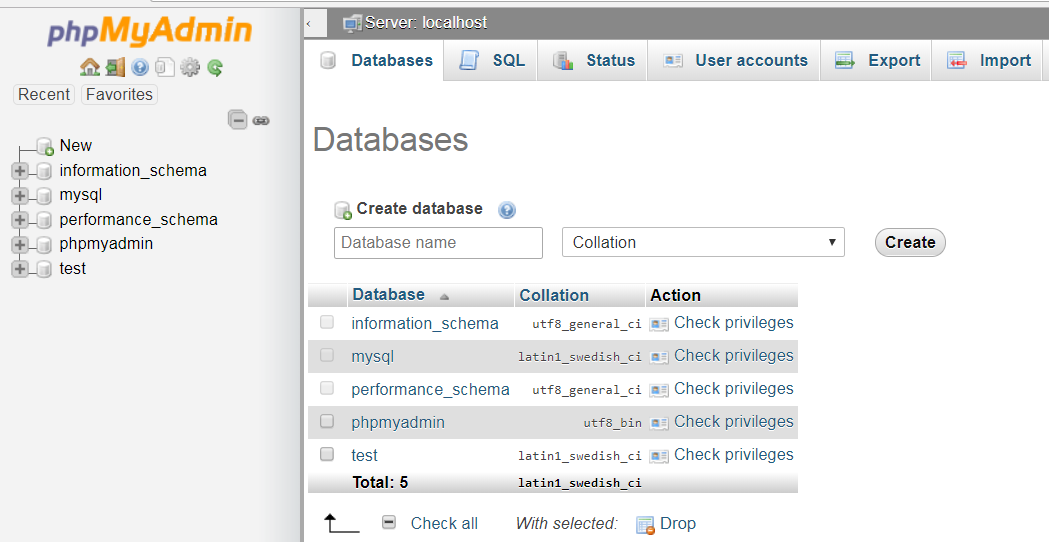
There is a choice of database technologies which can be used to complete the labs: you can register for an online account to use Oracle Application Express 5; or, use the XAMPP stack which incorporates a MySQL implementation. While the two technologies have different IDE support and slight variations in SQL, in essence, the lab work is the same; this document will provide instructions for using MySQL (MariaDB).

## 1. Setting up the database

We are using the XAMPP stack and a control panel application in the Specialised Applications folder which allows you to start both the Apache HTTP server and the MySQL database server (they may be started automatically).



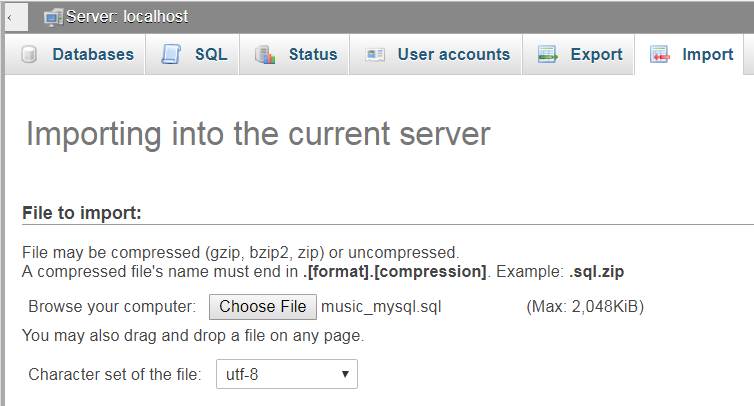
The MySQL admin button in the XAMPP control panel gives you access to a web interface called **phpmyadmin** which allows you to administer a MySQL installation.



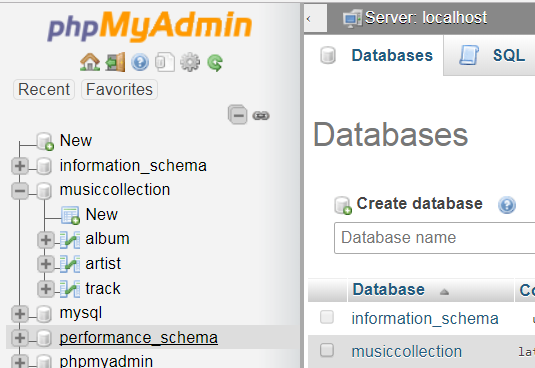
From this page, you can perform MySQL operations such as creating databases, creating tables, running queries, inserting & editing data etc.

#### Exercise 1

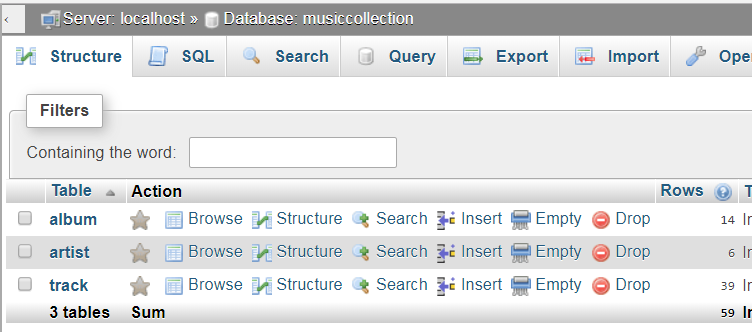
The database for the lab has already been created and exported as a set of SQL statements. You can import the database by running the script file, *musiccollection.sql*, which has the relevant SQL statements:



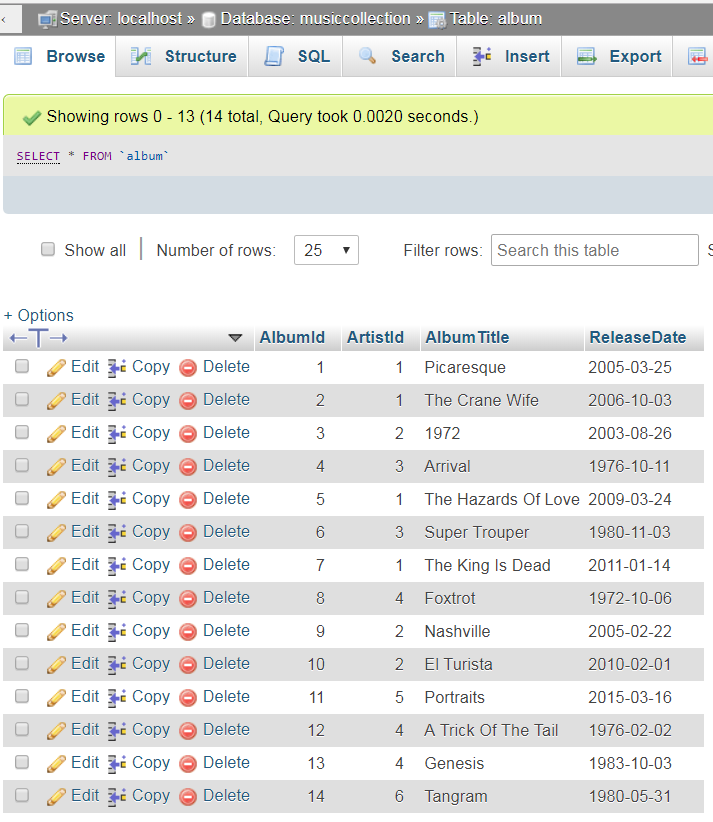
Once you have successfully achieved importing the database, use **phpMyAdmin** to explore the structure of the database:



Selecting the name of the database shows us the tables:



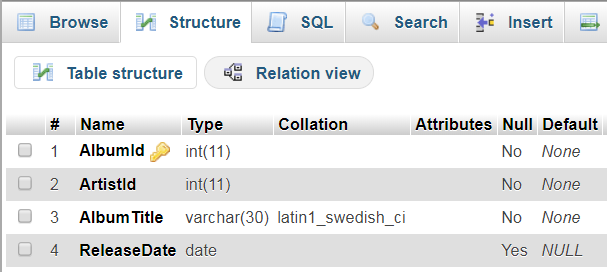
Selecting a table navigates to a page which shows table data:



Note the SQL query is: **select \* from album**

MySQL likes the **`** character to surround table names but it is unnecessary in most instances.

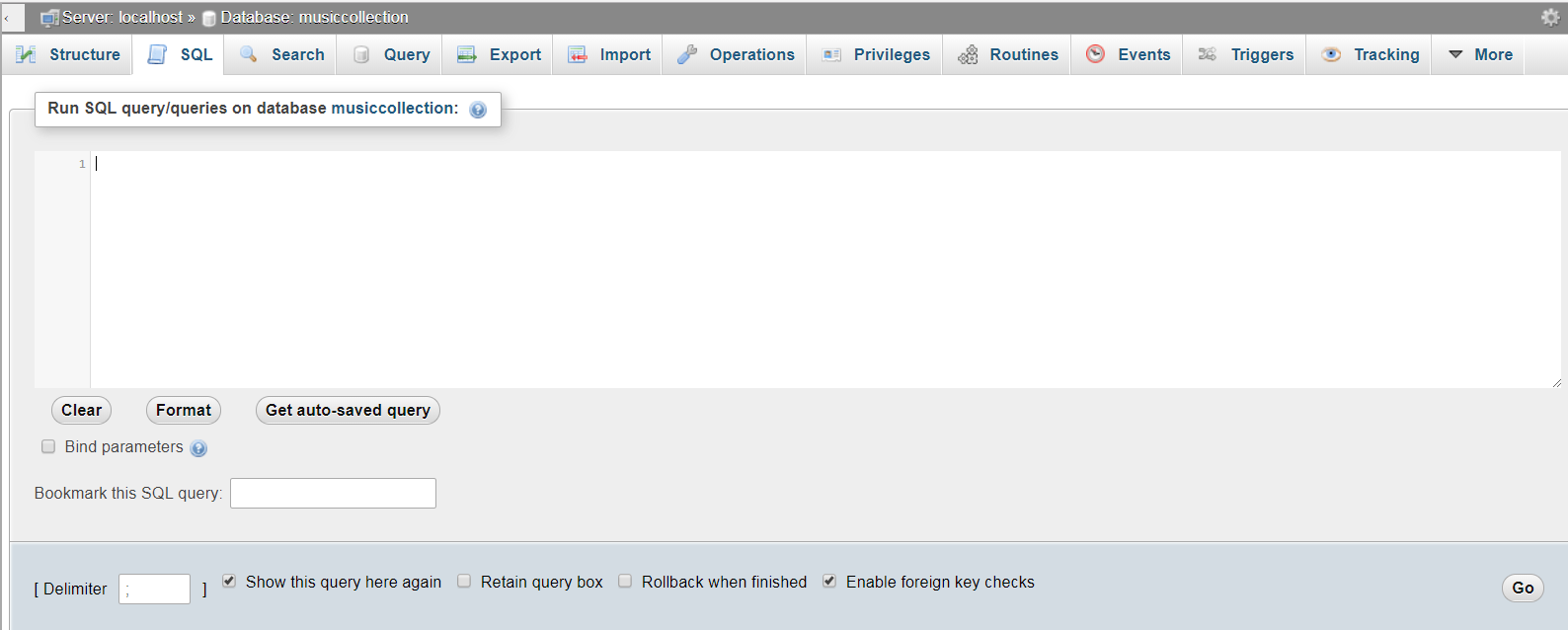
Select the **Structure** tab to view the table structure:



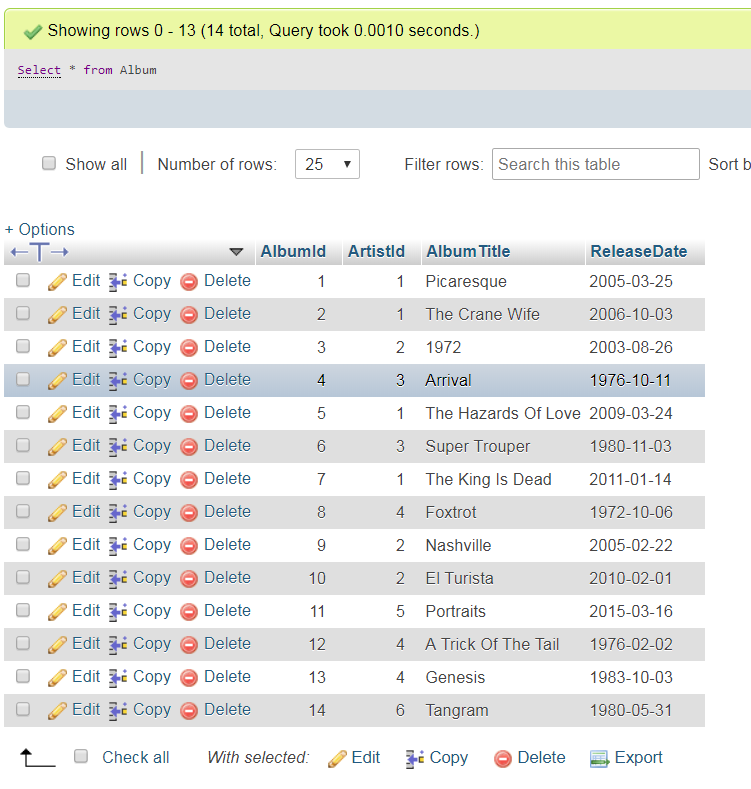
Explore the other two tables in a similar fashion.

## 2. Running SQL Queries

Select the **SQL** tab on either the **Database** page or on an individual table’s page:



We can use this to enter individual SQL commands e.g. **Select \* from Album**

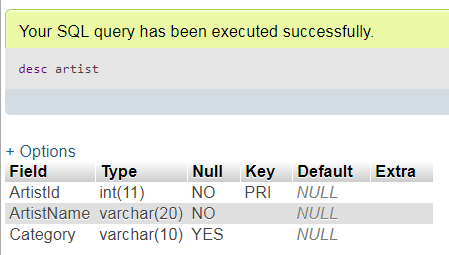


A database table consists of a number of columns (fields) – some of which must have unique values and some may have NULL values. Data within a table is organised in rows where each row describes a different entity occurrence.

SQL provides a way of querying a table definition using the **Describe** command:

### Exercise 2

1. Enter the describe command with the **Artist** table to see its structure:



Note there are 3 columns defined in the **Artist** table of various data types and lengths; note that all columns are Not Null i.e. a value is required for these columns when a new row is inserted into the table i.e. an entity occurrence is stored in the database.

Also note that **ArtistId** is defined as a Primary Key of the **Artist** table. The values of this column are used to distinguish between rows and must be unique. This is one of the main principles of Relational Databases.

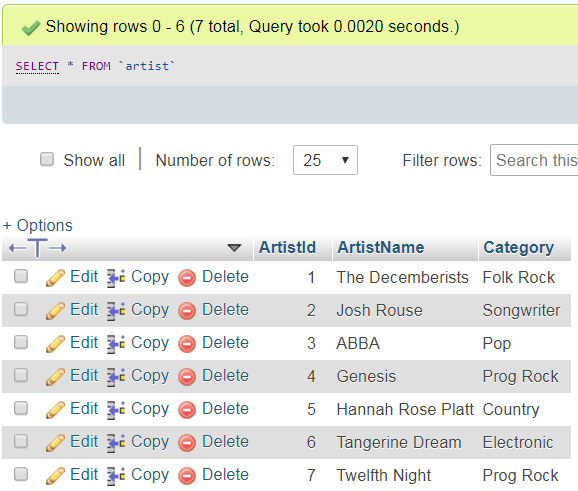
1. Run the **Describe** command on each of the other tables
2. Can you see how the tables are linked together?

### Exercise 3

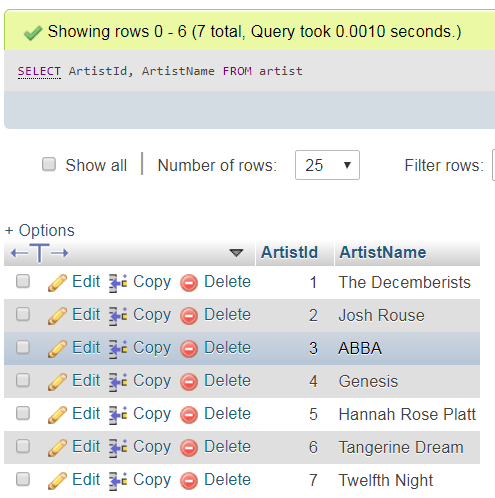
Now we are going to practice some SQL queries:

Enter the following SQL commands as above and check you see the results depicted; ensure you understand the queries:

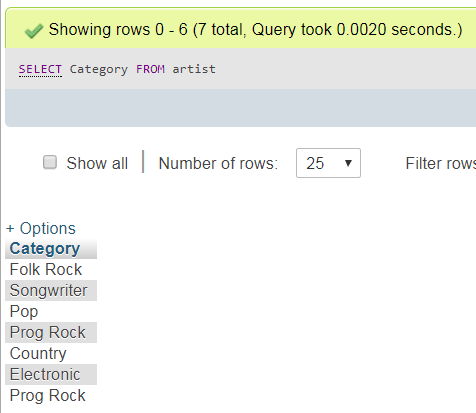
1. Display all details of all artists:



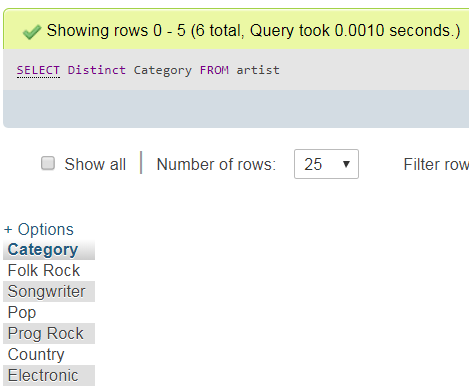
1. Display artist id and artist name for all artists:



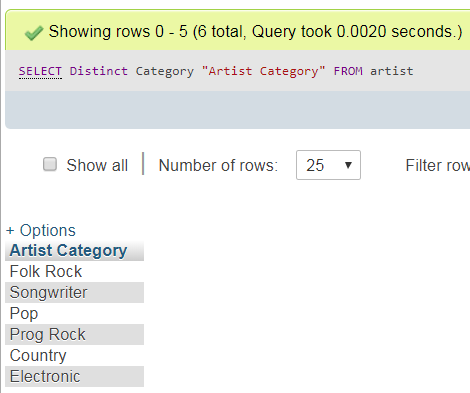
1. Display all artist categories:



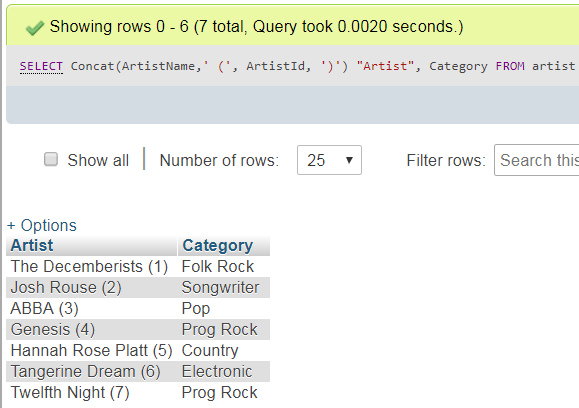
1. Display unique artist categories:



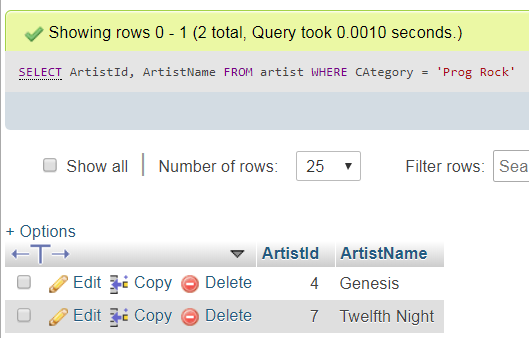
1. Display unique categories with column heading “Artist Category”:



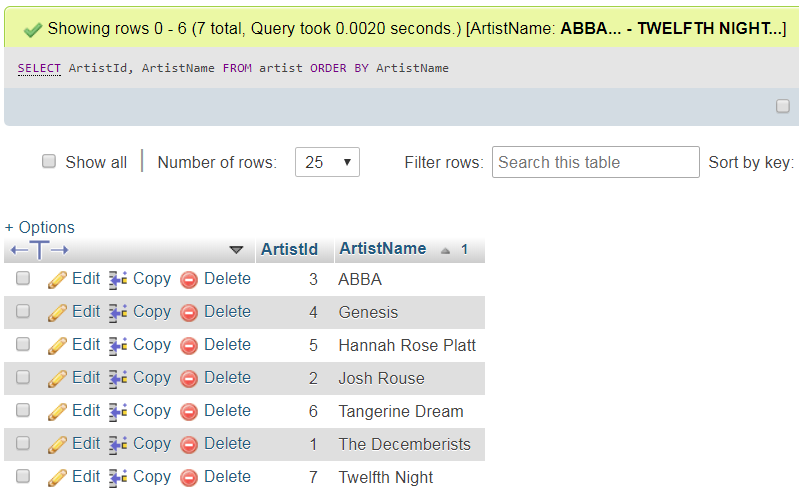
1. Display artist id and artist name with the format *artist name (artist id)* and heading “Artist” and category:



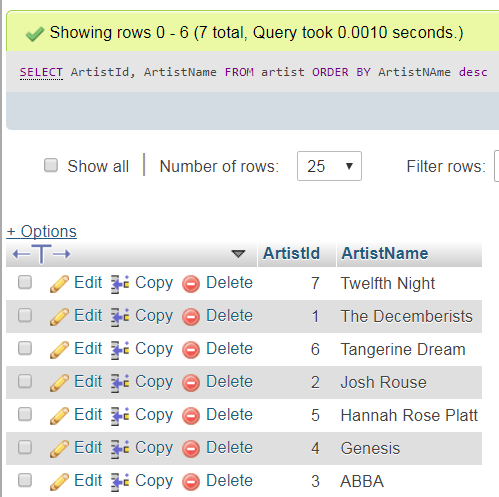
1. Display artist id and artist name for the Prog Rock category:



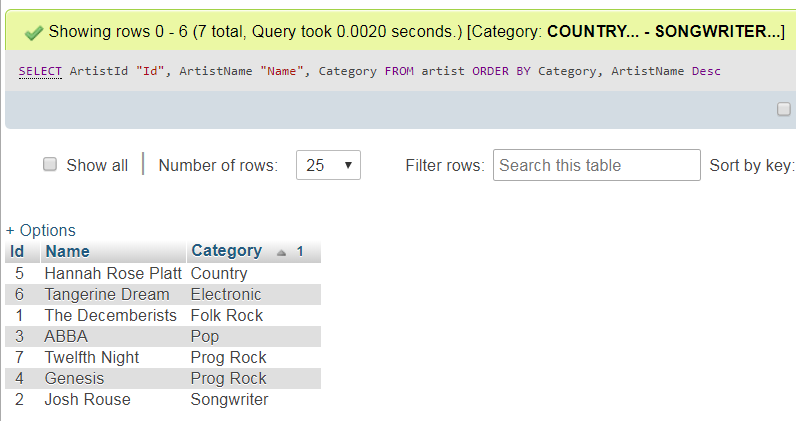
1. Display artist id and artist name for all artists sorted by artist name:



1. Display artist id and artist name for all artists sorted by artist name in reverse order:



1. Display artist id, artist name and category of all artists sorted by reverse order of artist name within category:



### Exercise 4

Write and test SQL queries to answer each of the requests below:

1. Display all details of all tracks (39 rows)
2. Display album id, track number and track title for all tracks (39 rows)
3. Display tracks whose album has id 8 (6 rows)
4. Display all tracks in sorted order of track title (39 rows)
5. Display all albums in descending order of release date (14 rows)
6. Display all albums in descending order of release date within ascending order of artist id (14 rows)
7. Display album id and album title with column headings “Id” and “Album” for artist id = 1 (4 rows)
8. Display album details of albums released this century (8 rows)
9. Display track details for track no = 1 in descending order of length (14 rows)
10. Display for album id = 4, the track no, track title and length in the format:

*track no: track title (track length)*

with heading “Track” and in ascending order of track no. (4 rows)