Analysis with SQL

From the database of a record company which was created, the following analysis was done using Structured Query Language on MySQL Workbench.   
  
At first a database is created of a record company named as “record\_company”.

In that database 3 different tables are added named “bands”, “albums” and “songs”. The bands table has two columns :- id and name where id is the primary key. The albums table comprises of four columns :- id, name, release\_year and band\_id where id is the primary key and band\_id is the foreign key referencing to the bands table primary key column id. The songs table consists of four columns :- id, name, length and album\_id where id is the primary key of that table and album\_id is the foreign key referencing to the albums table primary key column id.

After that the data is collected from a data file and added to these table. The data file consists of bands’ name and their ‘ids’ to be inserted in the ‘bands’ table, the albums which are released by those bands and the releasing year of those albums, the id of the albums and the band’s ids to be inserted in the ‘albums’ table and lastly the songs of those albums and their length, songs’ ids and albums’ ids to be inserted in the ‘songs’ table.

So after creating the database, tables in the database and inserting of the data to those tables, now the analysis part starts.

The analysis is being divided in 13 queries to get the knowledge of most data out of the database and understand how Structured Query Language works.

1. Select only the names of all Bands from the songs table

SELECT DISTINCT

bands.name

FROM

songs

JOIN

albums ON songs.album\_id = albums.id

JOIN

bands ON albums.band\_id = bands.id;

2. Select the oldest album.

SELECT

name

FROM

albums

ORDER BY release\_year

LIMIT 1;

3. Get all the bands that have albums

SELECT DISTINCT

bands.name

FROM

bands

JOIN

albums ON bands.id = albums.band\_id;

4. Get all the bands that have no albums.

SELECT

bands.name

FROM

bands

LEFT JOIN

albums ON bands.id = albums.band\_id

WHERE

albums.id IS NULL;

5. Get the longest album.

SELECT

albums.name, SUM(songs.length) AS total\_length

FROM

albums

JOIN

songs ON albums.id = songs.album\_id

GROUP BY albums.id

ORDER BY total\_length DESC

LIMIT 1;

6. Insert a record for your favorite Band and one of their Albums

INSERT INTO bands (name)

VALUES ('Mafia Mundeer');

INSERT INTO albums (name, release\_year, band\_id)

VALUES ('International Villager', 2012, (SELECT id FROM bands WHERE name = 'Mafia Mundeer'));

7. Delete the Band and Album you added in previous question.

DELETE FROM albums

WHERE

name = 'International Villager';

DELETE FROM bands

WHERE

name = 'Mafia Mundeer';

8. get the Average length of all songs

SELECT

AVG(length) AS average\_length

FROM

songs;

9. Select the longest song of each album

SELECT

a.name AS album\_name, s.name AS song\_name, s.length

FROM

albums a

JOIN

songs s ON a.id = s.album\_id

WHERE

s.length = (SELECT

MAX(s2.length)

FROM

songs s2

WHERE

s2.album\_id = a.id);

10. Get the number of songs for each band

SELECT

b.name AS band\_name, COUNT(s.id) AS song\_count

FROM

bands b

JOIN

albums a ON b.id = a.band\_id

JOIN

songs s ON a.id = s.album\_id

GROUP BY b.name;

11. Create a decade column with dividing the year // 10 \*10.

ALTER TABLE albums

ADD COLUMN decade INT;

UPDATE albums

SET

decade = (release\_year DIV 10) \* 10;

SELECT

\*

FROM

albums;

12. Filter the Albums which start with the word 'The'

SELECT

\*

FROM

albums

WHERE

name LIKE 'The%';

13. Find the album which released in 2008 to 2013.

SELECT

\*

FROM

albums

WHERE

release\_year BETWEEN 2008 AND 2013;