Assignment 1: Data Extraction

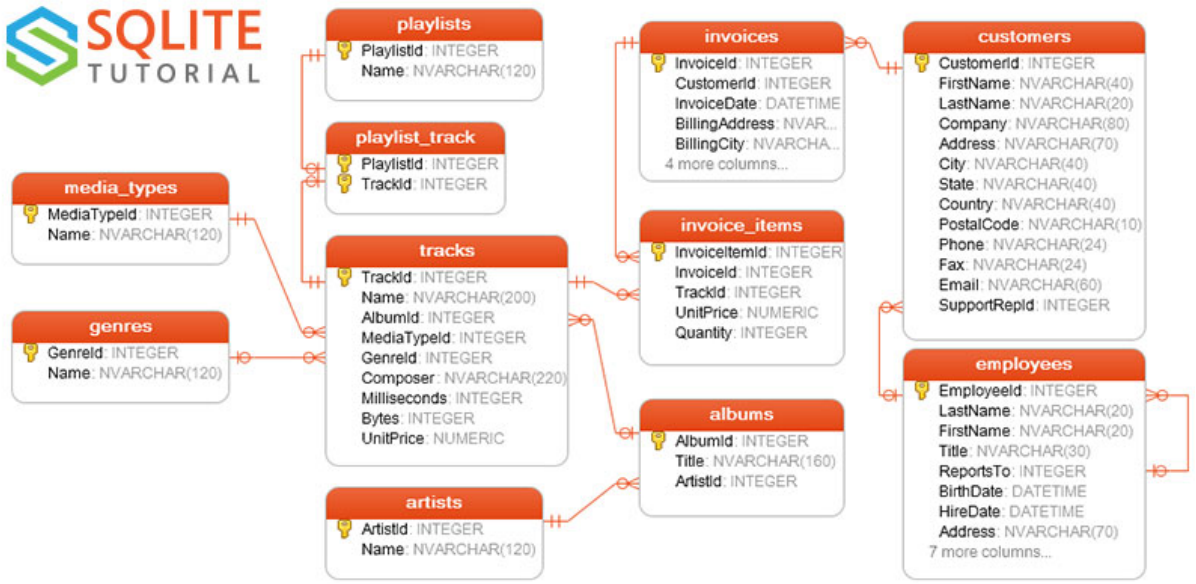
Group 11:

Robert Locher (5747465)**, Jose Enrique Leal Castillo** (9066381)**, Niek Lieon** (6520448)

Task 1

Description of the database

We used the chinook database which contains information on a digital music store, the products they sell and information on the customers. The domain is "Online Music Store" and it shows relations between, for example, music albums or playlists and customers that bought them. There are 11 tables in the database, listed below:



- albums
  - This table contains information on the title and artist of the album. The primary key is *AlbumId*, a unique numerical value to distinguish each album. The table *albums* has a relation with the tables *tracks* and *artists*, with common attributes respectively *AlbumId* and *ArtistId*.
- artists
  - Contains artist names. The primary key is *ArtistId*, a unique numerical value to distinguish each artist. The table has a relation with table *albums*, with common attribute ArtistId.
- customers
  - Personal information on customers such as contact information, name and place of residence. The primary key is *CustomerId.* The table has a relation with the table *invoices*, with common attribute *CustomerId.*
- employees

- Personal information on the employees such as name, title and birthdate. The primary key is *EmployeeId*. The table has a relation with the table *customers*, with common attribute *CustomerId.*

- genres
  - The genre of a particular song/album/playlist. The primary key is *GenreId*. The table has a relation with *tracks*, with common attribute *GenreId.*
- invoices
  - Contains billing information for purchases made by customers. The primary key is *InvoiceId*. This key is shared with the table *invoice_items.* The table has a relation with the tables *customers* and *invoice_items*, with common attribute *InvoiceId.*
- invoice_items
  - Information on product sold: which product, price and quantity. More information can be found by checking the related table *invoices*, which has the common attribute and primary key *InvoiceId.* The table *invoice_items* is also related with the table *tracks,* with common attribute *TrackId*.
- media_types
  - Contains MediaId for product. Contains two attributes, both are shared with the table *tracks*. The primary key for this relation is *MediaTypeId*.
- playlists
  - This relation contains the names of the playlists. It is related to the table *playlist_track*. Its primary key is *PlaylistId,* which is shared with the table *playlist_track*.
- playlist_track
  - TrackId of track in particular playlist. Has a relation with the tables *tracks* (*TrackId*) and *playlists* (*PlaylistId*). Its primary key is *PlaylistId*, which is shared with the table *playlists*.
- tracks
  - Information on track, e.g., composer, genre, price and *TrackId*. This table has a lot of relations with other tables. Commonly used attributes include *TrackId*; *AlbumId*; *Name*; *MediaTypeId*. Its primary key is *TrackId*.

Source of the database:

The database did not need modification to be used for this assignment, the source is listed below.

SQLite Sample Database. (n.d.). SQLite Tutorial.
https://www.sqlitetutorial.net/wp-content/uploads/2018/03/chinook.zip

Task 2

Question 1)

Write a query that contains an outer join (left is sufficient).

Answer:

NL	Give a relation of each album title that exists in the table Albums and its creator artist name
RA	$\pi_{title,name}\ (albums \bowtie artists)$
SQL	"SELECT Title,Name FROM albums LEFT JOIN artists on (artists.ArtistID = albums.ArtistID) "

Question 2)

Write a query that returns the number of missing values in a given attribute (look for an attribute that contains missing values).

Answer:

NL	Provide the number of missing values in track attribute composer
RA	$\pi_{COUNT(trackID)}\left(\gamma_{count(trackID)}\right)\left(\sigma_{composer=NULL}{}^{tracks}\right)$
SQL	SELECT COUNT(*) FROM (SELECT composer FROM tracks Where composer ISNULL)

Question 3)

Write a query that contains (has no) constraint (e.g. return the names of the students who has no courses this period).

Answer:

NL	Provide a table of tracks that have not been purchased
RA	$\pi_{name}\left(\left(\sigma_{trackID}{}^{tracks}\right) - \left(\sigma_{trackID}{}^{invoice\_teams} \bowtie tracks\right)\right)$
SQL	Select distinct tracks.Name FROM tracks EXCEPT Select all tracks.Name FROM tracks join invoice_items on (tracks.trackID = invoice_items.trackID)

Question 4)

Write a query that contains (has only) constraint (e.g. return the names of the students who has only one course this period).

Answer:

NL	Provide the Artists names who have only 1 album published
RA	$\pi_{name}\ \left(\sigma_{conta=1}{}^{\left(artistID\ \rho_{conta}(\gamma_{count(artistID)})\right)}{}^{artists \bowtie\ albums}\right)$
SQL	Select Name From (SELECT Name,Count(*) as conta FROM artists join albums on (artists.ArtistID = albums.ArtistID) Group by (artists.ArtistID) Having (conta =1))

Question 5)

Write a query that categorizes the records in one of the tables using a specific attribute that has a maximum of 5 distinct values and displays the number of records in each category.

Answer:

NL	Select all support representative ID's assigned to customers, and display in a table their ID's and the number of customers assigned to each one.
RA	$\pi_{SupportRepId,COUNT(SupportRepId)}\ (\gamma_{COUNT(SupportRepId)}{}^{customers})$
SQL	SELECT SupportRepId, COUNT(*) FROM customers GROUP BY SupportRepId

Question 6)

Using a table that contains at least one numerical attribute, write a query that displays the records that contain the min and max values of a numerical attribute.

Answer:

NL	Display the records that contain the max and min values of tracks Milliseconds.
RA	$\pi_{MIN(Milliseconds),MAX(Milliseconds)}(\gamma_{MIN(Milliseconds)\ ,\gamma_{MAX(Milliseconds)}}{}^{tracks})$
SQL	SELECT min(tracks.Milliseconds), max(tracks.Milliseconds) FROM tracks

Question 7)

Write a query that requires self-join.

Answer:

NL	Pair the songs that have the same duration in Milliseconds and display its duration in the same relation
RA	$\pi_{T.Name,S.Name,T.Milliseconds}(\sigma_{T.Milliseconds=S.Milliseconds \wedge T.trackID< S.trackID}{}^{(\rho_T(tracks),\ \rho_S(tracks))})$

SQL	SELECT T.Name ,S.Name, T.Milliseconds FROM tracks T, tracks S WHERE T.Milliseconds = S.Milliseconds AND T.trackID< S.trackID

Task 4

Question 1.a)
With the division operator (R ÷ S) you can find the tuples of a certain relation associated with all of the tuples in another relation. This way you can find for example students in relation R who have enrolled for *all* courses specified in relation S.

Question 1.b)

$$R \div S = \pi_{A-B}(R) - \pi_{A-B}((\pi_{A-B}(R) \times S) - R)$$

With R and S being two relations and A and B being their respective attributes. First you project attributes of both relations by joining both relations using the cartesian product of S and R (without the attributes that are also in S). After that you take the difference with the tuples who were also in the original R. This gives you a relation with all tuples that do not satisfy the query. Finally we eliminate from this relation the attributes that are not specified in the query and remove the value of the intended attribute from the original attribute in R. This gives us the final result of R ÷ S. This final relation shows all tuples from R that correspond with all the values specified in S.

Question 2)
The operation (=) means that both sides of the expression have to be equal. This means if X and Y are on both sides, they either both have to be true or both have to be false. With boolean operators this can be written as:

$$(X \wedge Y) \vee (\sim X \wedge \sim Y)$$

The *or* operator combined with on the right side the complement of the left side, will result in a *TRUE* even if both *X* and *Y* are set to *FALSE.* This way the outcome of the expression will be *TRUE* in all cases where *X* and *Y* are equal to each other.

Question 3)
The expression stated in the assignment is an example of the Redundancy Theorem in boolean algebra. The three conditions for applying this theorem are:

1. There must be three variables present.
    a. In this example there are three variables (A, B and C)
2. Every variable must be repeated twice in the expression.
    a. Which is the case for A, B and C
3. One variable in complement form
    a. Which is the ~A in the expression

According to the Redundancy Theorem the term which does not contain the complemented variable, in this case BC, can be removed from the expression, resulting in the final (RHS) expression:

$$(A \vee B) \wedge (\sim A \vee C)$$

This can also be proven algebraically:

$$(A \vee B) \wedge (\sim A \vee C) \wedge (B \vee C)$$

Add F

$$(A \vee B) \wedge (\sim A \vee C) \wedge (B \vee C \vee \mathrm{F})$$

$$F = A\sim A$$

$$(A \vee B) \wedge (\sim A \vee C) \wedge (B \vee C \vee \mathrm{A}\sim\mathrm{A})$$

Commutative law

$$(A \vee B) \wedge (\sim A \vee C) \wedge (B \vee C \vee \mathrm{A}) \wedge (B \vee C \vee \sim\mathrm{A})$$

Commutative law

$$(A \vee B) \wedge (A \vee B \vee C) \wedge (\sim A \vee C) \wedge (\sim A \vee B \vee C)$$

Absorption law

$$(A \vee B) \wedge (\sim A \vee C) \wedge (\sim A \vee C \vee B)$$

Absorption law

$$(A \vee B) \wedge (\sim A \vee C)$$