DATABASE SYSTEMS IMPLEMENTATION
HOMEWORK 1 - DUE TO 17/11/2022
VERSION 1.0

1. Data Layer. **SQLite**

For this group of exercises you are going to use an SQLite database for a digital media store. Database diagram 1 illustrates the tables and their relationships.
There are 11 tables/entities:

- **employees table** stores employees data such as employee id, last name, first name, etc. It also has a field named ReportsTo to specify who reports to whom.
- **customers table** stores customers data.
- **invoices & invoice_items tables** these two tables store invoice data. The invoices table stores invoice header data and the invoice_items table stores the invoice line items data.
- **artists table** stores artists data. It is a simple table that contains only the artist id and name.
- **albums table** stores data about a list of tracks. Each album belongs to one artist. However, one artist may have multiple albums.
- **media_types table** stores media types such as MPEG audio and AAC audio files. genres table stores music types such as rock, jazz, metal, etc.
- **tracks table** stores the data of songs. Each track belongs to one album.
- **playlists & playlist_track tables** playlists table store data about playlists. Each playlist contains a list of tracks. Each track may belong to multiple playlists. The relationship between the playlists table and tracks table is many-to-many. The playlist_track table is used to reflect this relationship.

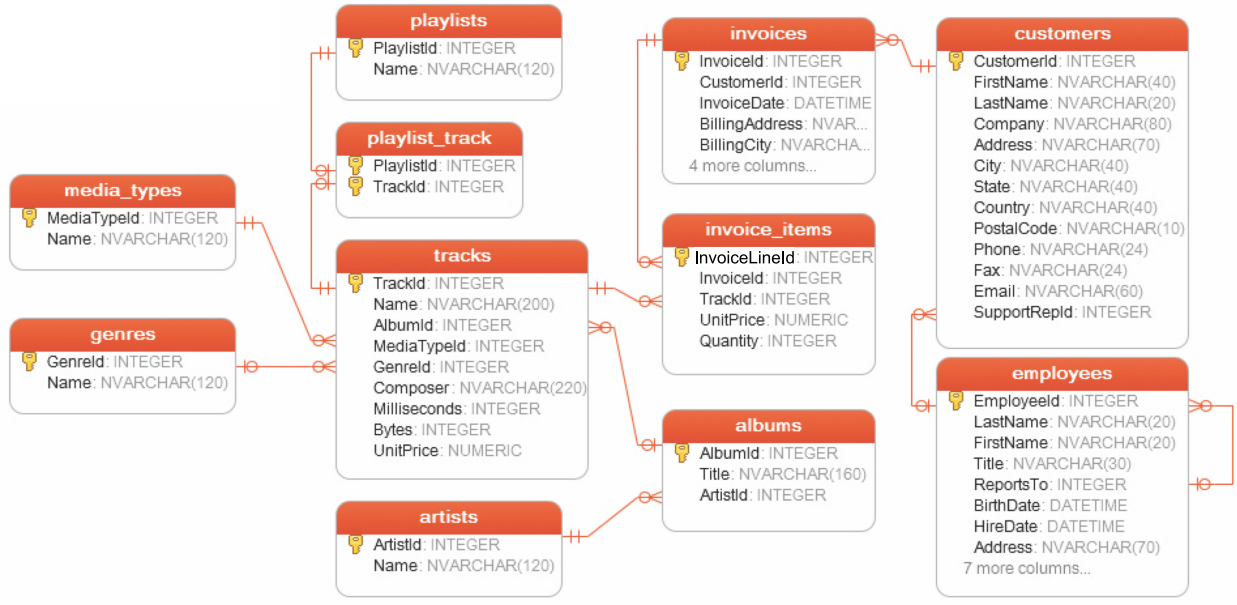

Figure 1. Diagram for the Digital Media Store Database

1.1. **Search queries exercises.** Use diagram 1 as reference and write the SQL queries to extract the following information requests.

(A) How many customers are there in the system?

(B) Who is the artist with the highest number of albums?

(C) How much money did the store make in 2010? For this exercise you have to get the income from the **invoice items**. *Recommendation: use STRFTIME*.

(D) What is the first and last name of the customer that bought more classical tracks (genres table)?

(E) Which artist is the most productive? In other words, get the artist's and album's name whose album has the highest number of tracks.

(F) Who is the first and last name of the employee higher in the hierarchy?

(G) How many costumers bought tracks in the format of the most expensive media type?

Recommendation: If you need to check the columns in a table run PRAGMA table_info(tablename)

2. Application Layer. **SQLITE Python Interface**

For this part of the exercises, the goal is for you to familiarize yourself with the sqlite3 api for python.

2.1. **Abstract search queries.** Turn all the search queries in the exercises 1.1 into search functions in python.

The goal is to provide a python interface for any programmer to get information from the data. Although it is a very simple interaction between the data and the programmer, it creates the first step towards an abstraction between both entities.

Use the method names that abstract each search query in table 1.

Exercise	Function Name	Output Type
1.A	customers_count()	int
1.B	most_productive_artist()	str
1.C	get_income_for_2010()	float
1.D	customer_that_like_classical()	str
1.E	most_successful_artist()	dict("Name": str, "Album": str)
1.F	who_is_the_boss()	str
1.G	costumers_that_likes_quality()	int

Table 1. Functions to be used in the abstraction of each search query.

2.2. **Abstract get income date.** Create a new function *get_income_for_2010()* based on *get_income_for(year)* where year is the function's only argument.

3. Optional Homework (**Not Graded**)

3.1. **Other SQL queries exercises.** Write the SQL queries based on the CRUD operations.

(A) Create a new db called *customers_expenditures* with columns **ExpenditureId**, which is the primary key, the **CustomerId** and the **MoneySpent**.

(B) Do a search query to obtain the expenditure from each customer. Insert into *customers_expenditures* table.

(C) Update the total money spent for Luis Rojas (CustomerId is 57). Add 5.99 to the total amount of money he has spent.

(D) Delete all rows from *customers_expenditures* where the total money spent on records is less than 40.

3.2. **Abstract CRUD queries.** Create a new function: *money_spent_by_customer(id, add_more)*, where the parameters are the Customer Id and the extra money the customer spent (Exercise 1.2.C). The function needs to apply all of the CRUD operations in exercise 3.1. The parameters are optional. *Recommendation: Use default value None in both parameters*.

3.3. **Presentation Layer. Executive Summary.** Create a function that prints to the standard output a quick summary based on the application level functions.

It should be something along the lines of ”There were a total of 59 customers in the digital media store. (...) ”

Add an argument to the function where it receives a year. The function should print to the stdout a summary of the media store profitability for that particular year.