Welcome to the Databases - SQL module!

Trainer: Diana Cavalcanti

Scope

- Relations
- Databases, Tables: Creating and Designing
- Data types, indexes, limitations
- SQL
- CRUD
- Complex queries with JOIN (INNER, OUTER, LEFT, RIGHT)
- having, group by, order by, limit
- (Optional)
- triggers, procedures
- Transactions
- ACID

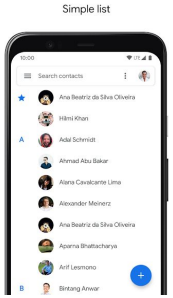
Software:
- MySQL 5.7.x+/8.x.y+
- MySQL Workbench 5.x.y+/8.x.y+

Important

Attendance list
Break time

Fundamentals

- Do you know what a database is?
  - A database is an organized collection of data
  - Would you know how to measure how much this area is present in your life?



Database system

A Database system is basically a computerized information storage system, that is, a computerized system whose main purpose is to maintain, store and make information available. ” (C.J. Date)

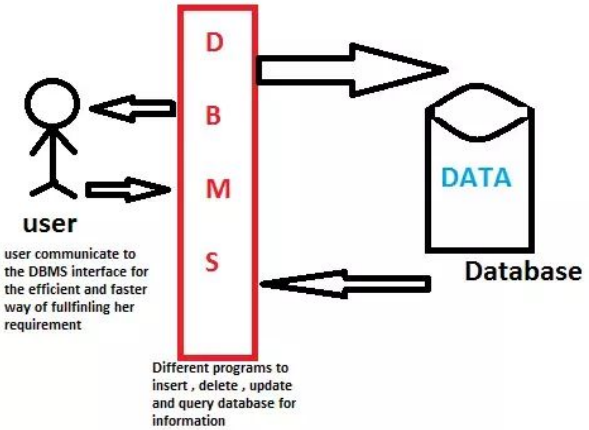
Main purpose:

- Organized storage aimed at:
  - System optimization
  - Facilitate insert, update, processing and consultation

https://en.wikibooks.org/wiki/Introduction_to_Database_Systems

A Database Management System (DBMS)

- DBMS is a system (software) that provides an interface to database for information storage and retrieval
  - capacity for large amount of data
  - an easy to use interface language (SQL-structured query language)
  - efficient retrieval mechanisms
  - multi-user support
  - security management
  - concurrency and transaction control
  - persistent storage with backup and recovery for reliability



https://en.wikibooks.org/wiki/Introduction_to_Database_Systems

A Database Management System (DBMS)

Examples of popular DBMS used these days:

- MySql
- Oracle
- SQL Server
- IBM DB2
- PostgreSQL

Relational databases

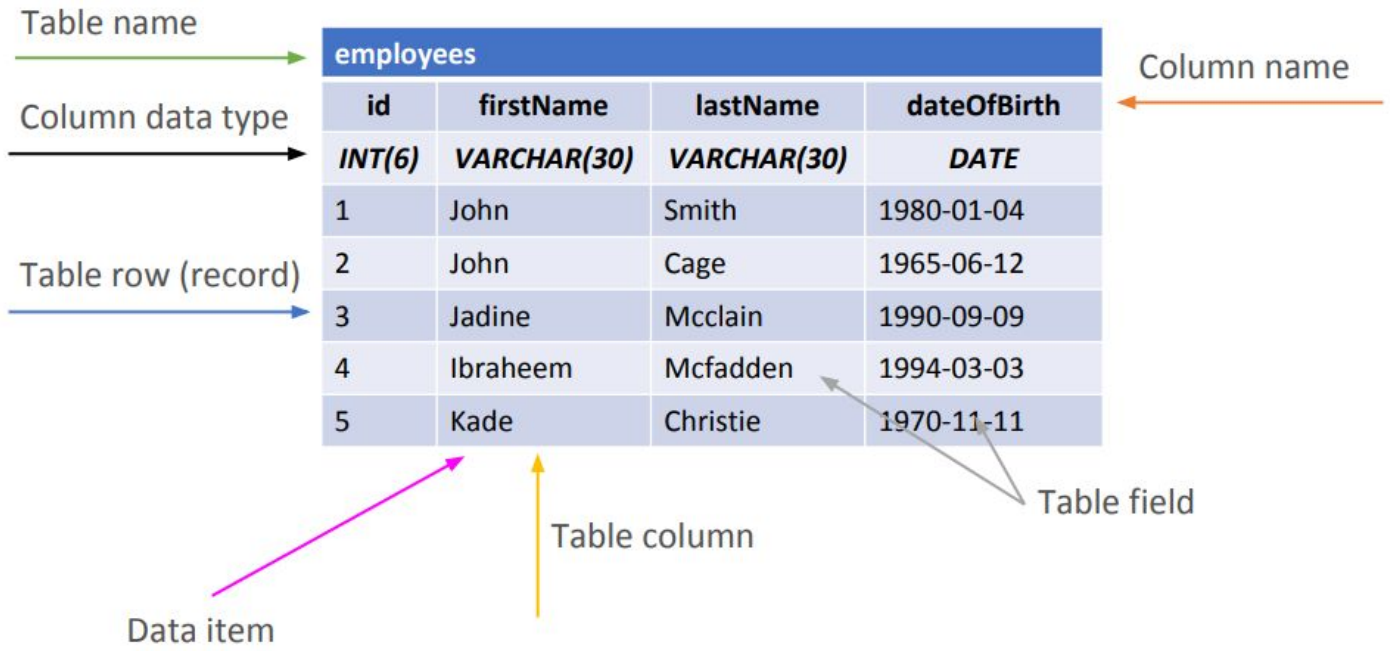
- This model organizes data into one or more tables (or "relations") of columns and rows, with a unique key identifying each row.

- A table is a collection of data held in a two dimensional structure.
- The two dimensions are rows and columns.
- A table is identified by a name.

https://www.oracle.com/database/what-is-database.html

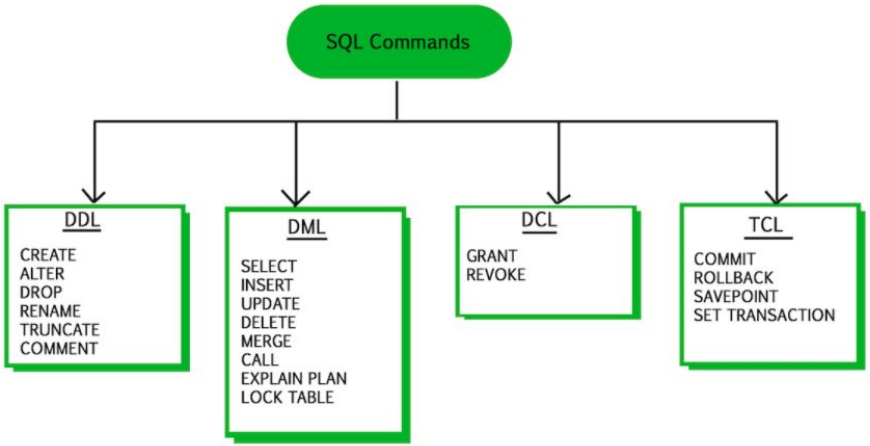
Relational databases

- Table



SQL - Structured Query Language

- **DDL** - **data definition language**. Helps users define what kind of data they are going to store and how they are going to model this data.
- **DML** - **data manipulation language**. Allows users to insert, update and delete data from the database.
- **DQL** - **data query language**. Helps users retrieve information from the database.
- **DCL** - **data control language**. Allows users to restrict and control access to the database.



SQL - Structured Query Language

- **DDL - Data Definition Language**
-
  - Create a database
    - **CREATE DATABASE** sda_course;
  - Select the database
    - use sda_course;
  - Delete a database
    - **DROP DATABASE** sda_course;

SQL - **DDL - Data Definition Language**

- Create a table

```
CREATE TABLE employees (
    id_employees INT,
    first_name VARCHAR(30),
    last_name VARCHAR(30),
    salary INT
);
```

- Column data types: The column data types define the type of information you can store in that particular column:
- **numeric:** int, tinyint, bigint, float, real, etc.,
- **date and time:** Date, Time, Datetime, etc.,
- **character and string:** char, varchar, text, etc.,
- **logical values:** TINYINT type value (0 or 1).

SQL - Structured Query Language

- **DDL - Data Definition Language**

  - describe employees;
  - Delete a table
    - **DROP TABLE** employees;

SQL - Structured Query Language

- **DDL - Data Definition Language**
-
    - Add a column

      ALTER TABLE employees
      ADD dateOfBirth **VARCHAR(10);**

    - Update a column

      ALTER TABLE employees
        MODIFY dateOfBirth **VARCHAR(50);**

SQL - Structured Query Language

- **DDL - Data Definition Language**
  - RENAME a column

    ALTER TABLE employees
    CHANGE COLUMN dateOfBirth date_of_birth **DATE**

  - DELETE a column

    ALTER TABLE employees
    DROP COLUMN date_of_birth **;**

SQL - Structured Query Language

- **DDL - Data Definition Language**

When defining a table the user can set certain properties on the columns:

- data type controls the type of values stored in the column,
- **NOT NULL** defines whether a column must be filled or not,
- **AUTOINCREMENT** states that the column value will be generated automatically (incrementation of the last inserted value) - this only works for numeric columns,
- **UNIQUE** states that there cannot be more than one row with the same value for that particular column.

SQL - Structured Query Language

-
- **NOT NULL**
  - **ALTER TABLE** employees **MODIFY** first_name **VARCHAR(30) NOT NULL;**
-
- **AUTOINCREMENT**
  - **ALTER TABLE** employees **CHANGE** id_employees id_employees **INT NOT NULL AUTO_INCREMENT PRIMARY KEY;**
  -
- **UNIQUE**
  - **ALTER TABLE employees ADD UNIQUE (last_name);**

Exercises

1. Create a new database: humanResources
2. Create a new table employees, with the following columns:
   a. employeeId - INTEGER ,
   b. firstName - VARCHAR,
   c. lastName - VARCHAR,
   d. dateOfBirth - DATE,
   e. postalAddress - VARCHAR.
3. Alter table employees and add the following columns:
   a. phoneNumber - VARCHAR,
   b. email - VARCHAR,
   c. salary - INTEGER.
4. Alter table employees and remove the postalAddress column.
5. Create a new table employeeAddresses,
   a. country_id - INTEGER
   b. country_name - VARCHAR.
6. Remove table employeeAddresses.

DML - Data Manipulation Language

- **Adding data**

INSERT INTO employees (id_employees, first_name, last_name, salary, date_of_birth) **VALUES**
(1, 'Michael', 'Harding', 20, '1937-07-25'),
(2, 'Ariana', 'Fox', 30, '1992-09-30'),
(3, 'Madelyn', 'Flynn', 35, '1953-03-05'),
(4, 'Fynley', 'Dodd', 40, '1973-03-27'),
(5, 'Aliza', 'Wyatt', 55, '1969-02-14'),
(6, 'Michael', 'Doss', 67, '1964-12-11')
(7, 'Michael', 'Watshon', 37, '1983-12-11');

*ALTER TABLE employees add date_of_birth DATE;

DML - Data Manipulation Language

- **Updating data**

```
UPDATE employees SET date_of_birth = '1988-12-11'
WHERE id_employees = 1 ;
```

```
SET SQL_SAFE_UPDATES=0;
```

```
SELECT * FROM employees
```

DML - Data Manipulation Language

- **Deleting data**

```
DELETE FROM employees WHERE id_employees = 7 ;
```

Exercises

Use the database: humanResources

1. Insert a new entry into employees table:
   a. employeeId - 1,
   b. firstName - John,
   c. lastName - Johnson,
   d. dateOfBirth - 1975-01-01,
   e. phoneNumber - 0-800-800-314,
   f. email - john@johnson.com,
   g. salary - 1000.
2. Update dateOfBirth of John Johnson to 1980-01-01.
3. Delete everything from employees table.
4. Add two more entries in employees:
   a. 1, 'John' , 'Johnson', '1975-01-01', '0-800-800-888' , 'john@johnson.com', 1000
   b. 2,'James' , 'Jameson', '1985-02-02', '0-800-800-999' , 'james@jameson.com', 2000

Exercises - Answer

Use the database: humanResources

1. Insert a new entry into employees table:
   a. employeeId - 1,
   b. firstName - John,
   c. lastName - Johnson,
   d. dateOfBirth - 1975-01-01,
   e. phoneNumber - 0-800-800-314,
   f. email - john@johnson.com,
   g. salary - 1000.

INSERT INTO employees (employeeId, firstName, lastName, dateOfBirth , phoneNumber, email, salary)
VALUES (1, 'John', 'Johnson', '1975-01-01', ' 0-800-800-314', 'john@johnson.com', 100)**;**

*If employeeId is auto-increment, remove it.

Exercises - Answer

1. Update dateOfBirth of John Johnson to 1980-01-01.

```
UPDATE employees SET dateOfBirth = '1980-01-01'
WHERE id_employees = 1;
```

also

```
UPDATE employees SET dateOfBirth = '1980-01-01'
WHERE first_name = '1980-01-01' AND  last_name = 'Johnson ';
```

1. Delete everything from employees table.

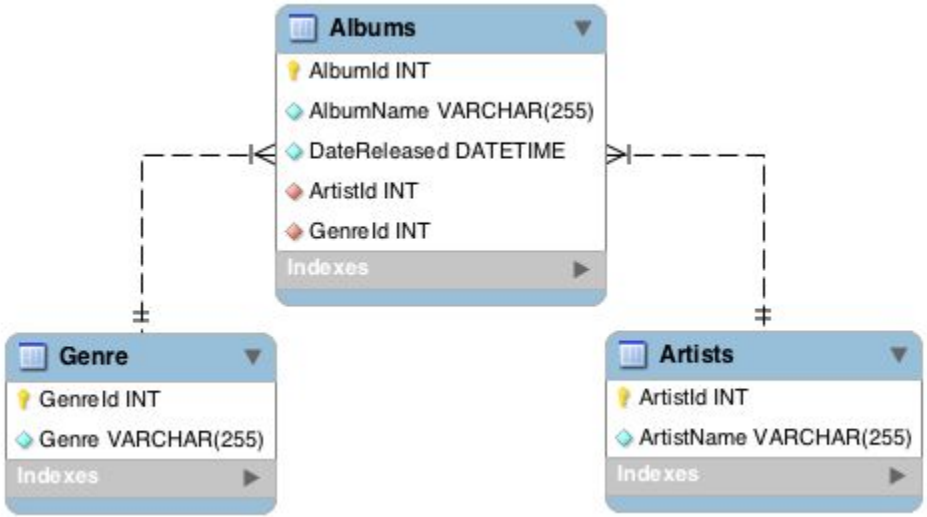
   ```
   DELETE FROM employees;
   ```

2. Add two more entries in employees:
   a. 1, 'John' , 'Johnson', '1975-01-01', '0-800-800-888' , 'john@johnson.com', 1000
   b. 2, 'James' , 'Jameson', '1985-02-02', '0-800-800-999' , 'james@jameson.com', 2000

Exercises

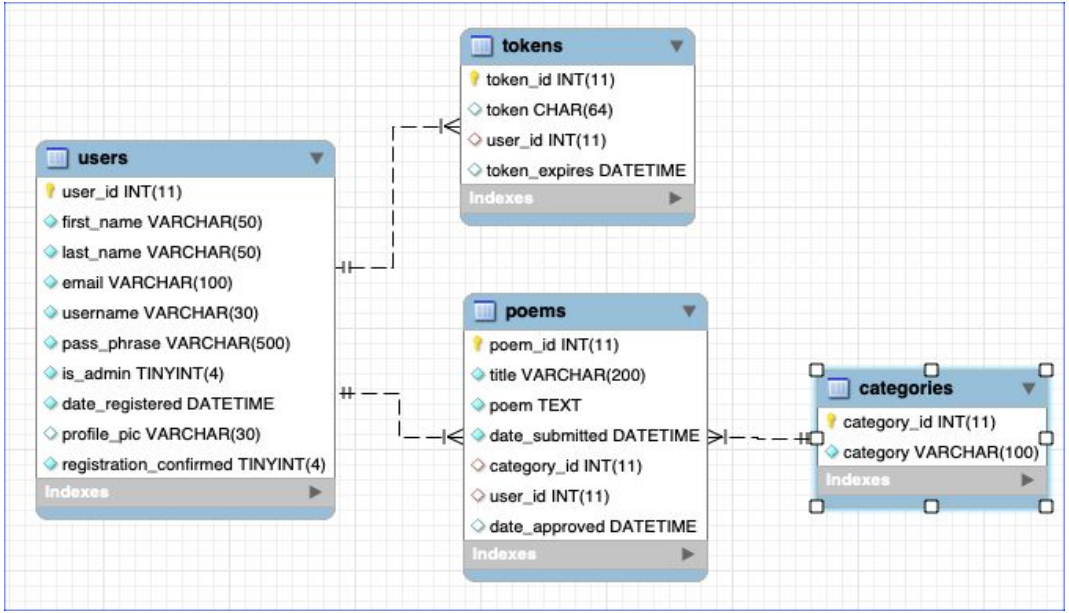
Using DDL

Create a new schema “music” and add the tables following the diagram below



Exercises

- Using DDL create a new schema “db_poems” and add the tables following the diagram below
- Use DML to insert data
- Ids are auto_increment
- Read and search about functions for Date
  - https://www.geeksforgeeks.org/sql-date-functions/
  - https://dataschool.com/learn-sql/dates/
  - https://www.tutorialspoint.com/sql/sql-date-functions.htm
- Insert data using a date function for the attribute ‘date_registered’



Read about string functions

https://www.w3schools.com/sql/sql_ref_sqlserver.asp

Day 2

DQL - Data Query Language

- **SELECT FROM**
  - The SELECT statement allows you to read data from one or more tables.

SELECT select_list FROM table_name [WHERE condition];

SELECT * FROM employees;

DQL - Data Query Language

- **SELECT FROM**
- **WHERE clause**
  - The WHERE clause allows you to specify a search condition for the rows returned by a query.
  - The search condition is a combination of one or more predicates using the logical operator AND, OR and NOT.

DQL - Data Query Language

- **SELECT FROM … WHERE clause**
  - **SELECT** * **FROM** employees **WHERE** last_name = 'Fox';
  - **SELECT DISTINCT** first_name **FROM** employees;
  - **SELECT** * **FROM** employees **WHERE** last_name **= 'Wyatt' AND** first_name **= 'Aliza';**
  - **SELECT** * **FROM** employees **WHERE** salary **>** 40;
  - **SELECT** * **FROM** employees **WHERE** salary **IN (10, 20, 30)**;
  - **SELECT** * **FROM** employees **WHERE** salary **IS NULL**;
  - **SELECT** * **FROM** employees **WHERE** salary **IS NOT NULL** ;
  - **SELECT** * **FROM** employees **WHERE** salary **!= 20**;
  - **SELECT** * **FROM** employees **WHERE** salary **BETWEEN 30 AND 50** ;
  - **SELECT** * **FROM** employees **WHERE** first_name **LIKE 'A%';**
  - **SELECT** * **FROM** employees **WHERE** first_name **LIKE '%n';**
  - **SELECT** * **FROM** employees **WHERE** first_name **LIKE '%e%';**

AGGREGATE functions

An aggregate function performs a calculation on multiple values and returns a single value

- **AVG** - takes multiple numbers and returns the average value of the numbers
  - **SELECT AVG**(salary) **FROM** employees;
- **SUM** - returns the summation of all values
  - **SELECT SUM**(salary) **FROM** employees;
- **MAX** - returns the highest value
  - **SELECT MAX**(salary) **FROM** employees;
- **MIN** - returns the lowest value
  - **SELECT MIN**(salary) **FROM** employees;
- **COUNT** - returns the number of rows
  - **SELECT COUNT**(*) **FROM** employees;

SQL EXTRAS

- **ORDER BY**
  - Used to sort the result-set in ascending or descending order:
    SELECT column1, column2, … FROM table_name ORDER BY column1 [ASC|DESC];

```
SELECT first_name
FROM employees
ORDER BY first_name ASC;
```

```
SELECT first_name
FROM employees
ORDER BY first_name DESC;
```

SQL EXTRA

- **AS**
  - Aliases are used to give a table, or a column in a table, a temporary name:
  - **SELECT column1 as newName, column2, … FROM table_name;**
    - **SELECT** first_name **as FIRST_NAME FROM** employees;

- **LIMIT**
  - Used to restrict the number of results retrieved from the database
  - **SELECT * FROM** employees **LIMIT 3;**

SQL EXTRAS

- **GROUP BY**
  - statement groups rows that have the same values into summary rows, like “find the number of customers in each country”:
  - SELECT column1, column2, … FROM table_name GROUP BY column1;
  - **SELECT COUNT(CustomerID), Country FROM Customers GROUP BY Country;**

```
SELECT first_name, COUNT(*) AS 'occurences count'
FROM employees
GROUP BY first_name;
```

SQL EXTRA

- **HAVING**
  - clause was added to SQL because the WHERE keyword could not be used with aggregate functions:
  - **SELECT column1, column2, … FROM table_name GROUP BY column1 HAVING condition;**

```
SELECT first_name AS 'NAME'
FROM employees
GROUP BY first_name
HAVING COUNT(*) > 1;
```

SubQueries

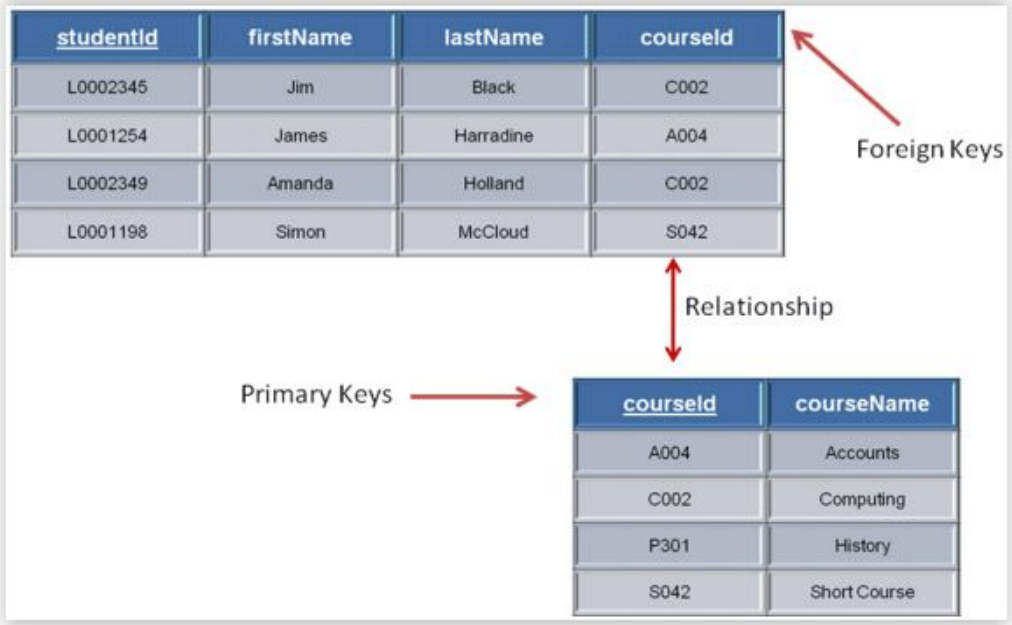
https://www.mysqltutorial.org/mysql-subquery/

https://www.essentialsql.com/get-ready-to-learn-sql-server-20-using-subqueries-in-the-select-statement/

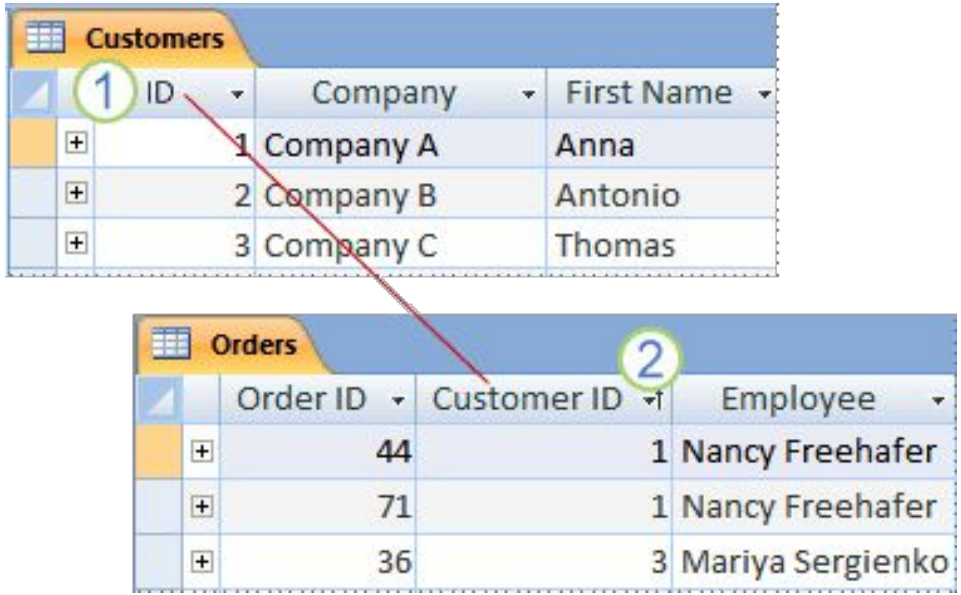
https://levelup.gitconnected.com/how-and-when-to-write-mysql-subqueries-8d5d580b1729

```
SELECT first_name, salary
FROM   employees
WHERE  salary = (SELECT MIN(salary) FROM   employees);
```

PRIMARY and FOREIGN Keys



PRIMARY and FOREIGN Keys



PRIMARY Keys

- A primary key is a column or a set of columns that uniquely identifies each row in the table.
- A primary key must contain unique values. If the primary key consists of multiple columns, the combination of values in these columns must be unique.
- A primary key column cannot have NULL values.
- A table can have one an only one primary key.
- A primary key column often has the AUTO_INCREMENT attribute that automatically generates a sequential integer whenever you insert a new row into the table.

FOREIGN Keys

```
CREATE TABLE employees (
    id_employees INT AUTO_INCREMENT PRIMARY KEY NOT NULL,
    first_name VARCHAR(30),
    last_name VARCHAR(30),
    salary INT,
    date_of_birth DATE
);
```

OR

```
ALTER TABLE employees ADD PRIMARY KEY NOT NULL (id_employees);
```

FOREIGN Keys

- A foreign key is a column or group of columns in a table that links to a column or group of columns in another table.
- The foreign key places constraints in the related tables, so MySQL can maintain referential integrity. The table containing the foreign key is called the child table, and the referenced table is the parent table.
- Typically, the foreign key columns of the child table often refer to the primary key columns of the parent table.
- A table can have more than one foreign key where each foreign key references to a primary key of the different parent tables.
- Once a foreign key constraint is in place, the foreign key columns from the child table must have the corresponding row in the parent key columns of the parent table or values in these foreign key columns must be NULL.

FOREIGN Keys

```
ALTER TABLE employees ADD id_departments INT(6);

CREATE TABLE departments (
    id_departments INT(6) AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(30) NOT NULL
);


ALTER TABLE employees ADD FOREIGN KEY(id_departments)
(REFERENCES departments (id_departments);
```

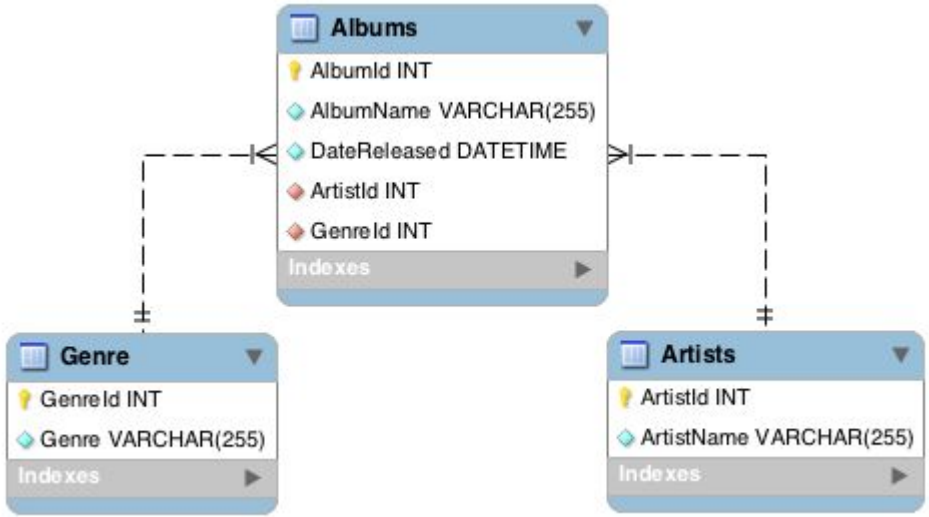
EXERCISES

Use the database: humanResources

1. Select everything from table employees.
2. Select only firstName and lastName from table employees.
3. Select all employees with lastName Johnson.
4. Select all employees whose lastName starts with J.
5. Select all employees whose lastName contains so.
6. Select all employees born after 1980.
7. Select all employees born after 1980 and whose firstName is John.
8. Select all employees born after 1980 or whose firstName is John.
9. Select all employees whose lastName is not Jameson.
10. Select maximum salary.
11. Select minimum salary.
12. Select average salary.

Exercises

- Using SQL add the relationship between the tables described on diagram below, use the reverse engineer and compare your diagram
- Add data, create a query to answer how many albums exist by ‘genre’
- Create a query to answer what is the lasted album released?



Exercises

- Using SQL add the relationship between the tables described on diagram below, use the reverse engineer and compare your diagram
- How many users was registered by date?
- List the ‘token_id’ that has expired