[Cheat Sheet] - SQL

- Database
  - Integrating python and SQL
    - https://www.freecodecamp.org/news/connect-python-with-sql/
  - collection of data that is organized in a manner that facilitates ease of access, as well as efficient management and updating
  - made up of **tables** that store relevant information
  - consisting of **columns** and **rows**
- Primary key
  - field in the table that uniquely identifies the table records
  - **unique value** for each row
  - cannot contain **NULL** values
  - Primary Keys

ID	FirstName	LastName	TelephoneNumber
1	John	Smith	715-555-1230
2	David	Williams	569-999-1719
3	Chloe	Anderson	715-777-2010
4	Emily	Adams	566-333-1223
5	James	Roberts	763-777-2956

- SQL
  - **S**tructured **Q**uery **L**anguage
  - **MySQL** is a **program** that understands **SQL**
  - case **INSENSITIVE(**but all commands should be in upper-case**)**
  - **Features**
    - insert, update, or delete records in a database.

        - create new databases, tables, stored procedures and views.
        - retrieve data from a database, etc.
- Commands
    - **SHOW**
        - displays information contained in the database and its tables

```
SHOW DATABASES; --> lists the databases managed by the server
SHOW TABLES; --> display all of the tables in the currently selected MySQL
database
SHOW COLUMNS FROM place; --> displays information about the columns in a
given table(place)
```

        - SHOW COLUMNS FROM ...
            - **Field**: column name
            - **Type**: column data type
            - **Key**: indicates whether the column is indexed
            - **Default**: default value assigned to the column
            - **Extra**: may contain any additional information that is available about a given column
    - **SELECT**
        - select data from a database
        - result is stored in a result table, which is called the **result-set**
        - **query** may retrieve information from selected columns or from all columns in the table

```
SELECT FirstName FROM customers;
SELECT id, lastname, firstname FROM customers; --> creates a table
SELECT * FROM customers --> displays all the table content
```

    - **REGEX**
        - pattern matching

```
SELECT * FROM author
WHERE aut_name REGEXP '^w'; --> '^' starts with...
```

    - **DISTINCT**

        - eliminate all duplicate records and return only unique ones

```
SELECT DISTINCT lastname FROM customers;
```

    - **LIMIT**
        - Creates a list with the limitation given

```
SELECT FirstName FROM customers LIMIT 3; --> gets the first 3 elements of
the row
SELECT FirstName FROM customers OFFSET 2 LIMIT 3; --> gets the first 3
elements of the row starting by the **THIRD** element
```

    - **Fully Qualified Names**
        - table name prior to the column name, by separating them with a **dot**

```
SELECT customers.id FROM customers;
```

    - **ORDER BY**
        - Organize table by element/condition
        - DESC - Orders in a descending order (low -> high)
        - ASC - Orders in a ascending order (high -> low)

```
SELECT * FROM customers ORDER BY LastName;
SELECT * FROM customers ORDER BY LastName, id;
SELECT * FROM customers ORDER BY LastName DESC --> reversed order
alphabetically
```

    - **WHERE**
        - Condition - find only the values that...

```
SELECT column_name FROM database WHERE condition;
SELECT FirstName FROM customer WHERE ID = 7;
SELECT * FROM customer WHERE city = 'Chicago'; --> text values have (')
```

    - **BETWEEN...AND...**
        - Range of results

```
SELECT * FROM customers WHERE ID BETWEEN 3 AND 7;
```

Logical Operators

- combine two Boolean values and return a result of **true**, **false**, or **null**.
-

Operator	Description
AND	TRUE if **both** expressions are TRUE
OR	TRUE if **either** expression is TRUE
IN	TRUE if the operand is equal to one of a list of expressions
NOT	Returns TRUE if expression is not TRUE

-

```
SELECT ID, FirstName, LastName, Age
FROM customers
WHERE Age <= 30 OR Age >= 40;

SELECT * FROM customers
WHERE City = 'New York'
AND (Age=30 OR Age=35); --> AND + OR

SELECT * FROM customers
WHERE City IN ('New York', 'Los Angeles', 'Chicago');
```

Custom Columns

- CONCAT
  - two or more text values and returns the concatenating string

```
SELECT CONCAT(FirstName, ', ' , City) FROM customers;
```

- AS
  - Gives a name to the custom column

```
SELECT CONCAT(FirstName,', ', City) AS new_column --> nao precisa de ''
FROM customers;
```

Functions

- UPPER/LOWER
  - represents a string in UPPER/LOWER case

```
SELECT city, UPPER(LastName) FROM customers;
```

- DROP COLUMN
  - delete the column named

```
ALTER TABLE People
DROP COLUMN DateOfBirth;
```

- **ALTER TABLE**
  - command is used to add, delete, or modify columns in an existing table.

```
ALTER TABLE People ADD ColumnName;
//or
ALTER TABLE People RENAME columnName1 TO ColumnName2
```

- SQRT
  - Square root of a each item individually

```
SELECT Salary, SQRT(Salary)
FROM employees;
```

- AVG
  - Average of all items

```
SELECT AVG(Salary) FROM employees;
```

- SUM
  - Sum of all items

```
SELECT SUM(Salary) FROM employees;
```

- INSERT INTO
  - Inserts an element into the Data Base

```
INSERT INTO customers(
```

```
CustomerName,
Address,
City,
PostalCode,
Country
)

INSERT INTO table (Column1, Column2, ...)
VALUES(a, b, ...)
```

- UPDATE
  - Updates a colum

```
UPDATE  Customers
SET City = 'Oslo'; --> updates all column values to "Oslo"
----------
UPDATE Customers
SET City = "Oslo" WHERE City = "Berlin" --> updates all 'Berlin' values in
column city to 'Oslo'
```

- DELETE
  - Deletes a value in a column

```
DELETE FROM Customers
WHERE Country = 'Norway';
-----------
DELETE Customer
```

- MIN(...)
  - Returns the lowest value of column inside the parenthesis

```
SELECT MIN(Price) FROM Customers
```

- MAX(...)
  - Returns the highets value of column inside the parenthesis

```
SELECT MAX(Price) FROM Customers
```

- LIKE
  - Select a value that begins with...

```
SELECT * FROM Customers
```

```
WHERE City LIKE 'a%'; --> % being rest/starts with a
---------------
SELECT * FROM Customers
WHERE City LIKE '&a'; --> ends with a
-------------
SELECT * FROM Customers
WHERE City LIKE '&a%'; --> has an 'a'
-----------------
SELECT * FROM Customers
WHERE City LIKE ' _a %'; -->second letter is an a
----------------
SELECT * FROM Customers
WHERE City LIKE ' [acs] %'; --> first letter of the City is an "a" or a "c"
or an "s"
-----------------
SELECT * FROM Customers
WHERE City LIKE ' [a-s] %'; --> first letter of the City is an "a" to "s"
```

- ALIAS
  - Gives an nickname to a column
  - SELECT customerName AS Cn

Subqueries

- Subquery is a query within a query

```
Salary > (SELECT AVG(Salary) FROM employees) --> Selects the Salaries that
are above avarega of itself
```

JOINING TABLES

- Creates a temporary table that shows the data
- Must have a common column on both tables
  - EXAMPLE: The list of customers and the list of items bought
  - CUSTOMER TABLE

ID	Name	City	Age

35	Brian	Moscow	35

- PRODUCT TABLE

ID	Name	Price	Customer_ID
200	Book	$12.99	35

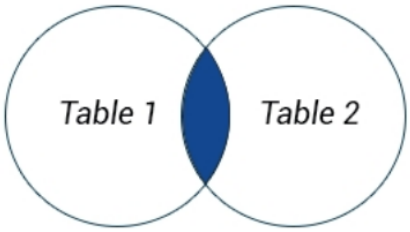
- The Customer_ID and ID in the customer table are common columns that link both of them together("OVERLAP")

```
SELECT customers.ID, customers.Name, orders.Name, orders.Amount
FROM customers, orders
WHERE customers.ID=orders.Customer_ID
ORDER BY customers.ID;
```

Types of join

- Inner join
  - joins when there is a match between the tables.

```
SELECT column_name(s)
FROM table1 INNER JOIN table2
ON table1.column_name=table2.column_name;
```

  - 

- Left Join
  -

- Right Join

UNION

- combine data from multiple tables into one comprehensive dataset
- columns in both tables must be the same

```
SELECT column_name(s) FROM table1
UNION
SELECT column_name(s) FROM table2;
```

UNION ALL

- Combine data from multiple tables including duplicates

```
SELECT column_name(s) FROM table1
UNION ALL
SELECT column_name(s) FROM table2;
```

CREATE A TABLE

```
CREATE TABLE table_name
(
column_name1 data_type(size),
column_name2 data_type(size),
column_name3 data_type(size),
....
columnN data_type(size)
);

CREATE TABLE Sas(
    id INT,
    firstname VARCHAR(10),
    lastname VARCHAR(15),
    age INT,
    prod_id INT
);
```

Data Types

Data types specify the type of data for a particular column.

If a column called "LastName" is going to hold names, then that particular column should have a "varchar" (variable-length character) data type.
The most common data types:
Numeric
INT -A normal-sized integer that can be signed or unsigned.
FLOAT(M,D) - A floating-point number that cannot be unsigned. You can optionally define the display length (M) and the number of decimals (D).
DOUBLE(M,D) - A double precision floating-point number that cannot be unsigned. You can optionally define the display length (M) and the number of decimals (D).

Date and Time
DATE - A date in YYYY-MM-DD format.
DATETIME - A date and time combination in YYYY-MM-DD HH:MM:SS format.
TIMESTAMP - A timestamp, calculated from midnight, January 1, 1970
TIME - Stores the time in HH:MM:SS format.

String Type
CHAR(M) - Fixed-length character string. Size is specified in parenthesis. Max 255 bytes.
VARCHAR(M) - Variable-length character string. Max size is specified in parenthesis.
BLOB - "Binary Large Objects" and are used to store large amounts of binary data, such as images or other types of files.
TEXT - Large amount of text data.

Constrains

SQL **constraints** are used to specify rules for table data.
The following are commonly used SQL constraints:
NOT NULL - Indicates that a column cannot contain any NULL value.
UNIQUE - Does not allow to insert a duplicate value in a column. The UNIQUE constraint maintains the uniqueness of a column in a table. More than one UNIQUE column can be used in a table.
PRIMARY KEY - Enforces the table to accept unique data for a specific column and this constraint create a unique index for accessing the table faster.
CHECK - Determines whether the value is valid or not from a logical expression.
DEFAULT - While inserting data into a table, if no value is supplied to a column, then the column gets the value set as DEFAULT.
For example, the following means that the **name** column disallows NULL values.
AUTO INCREMENT - allows a unique number to be generated when a new record is inserted into a table; By default, the starting value for AUTO_INCREMENT is 1

VIEWS

- is a **virtual table** that is based on a SQL statement
- Allow us to:
    - Structure data in a way that users or classes of users find natural or intuitive.
    - Restrict access to the data in such a way that a user can see and (sometimes) modify exactly what they need and no more.
    - Summarize data from various tables and use it to generate reports