MySQL Cheat Sheet in Simple Terms

Basic Concepts

MySQL Intro: MySQL is a widely-used relational database management system that helps you store and manage data in a structured way. It organizes data into tables with rows and columns, making it easy to retrieve, update, and manage information. MySQL is popular for its reliability, speed, and ease of use, and it's commonly used in web applications and software development.

MySQL RDBMS: MySQL operates as a relational database management system (RDBMS), storing data in tables that can relate to each other.

SQL Commands

- **MySQL SQL:** SQL is the language used to interact with MySQL databases.
- **MySQL SELECT:** Retrieves data from a database.
  Example :
  ```
  SELECT * FROM customers;
  ```
- **MySQL WHERE:** Filters records that fulfill a specified condition.
  Example :
  ```
  SELECT * FROM customers WHERE country='Germany';
  ```
- **MySQL AND, OR, NOT:** Combines conditions when filtering records.
  Example :
  ```
  SELECT * FROM products WHERE price BETWEEN 10 AND 20 AND NOT
  category='Electronics';
  ```
- **MySQL ORDER BY:** Sorts the result set in ascending or descending order.
  Example :
  ```
  SELECT * FROM customers ORDER BY name DESC;
  ```
- **MySQL INSERT INTO:** Inserts new rows into a table.
  Example :
  ```
  INSERT INTO customers (name, email) VALUES ('John Doe', 'john@example.com');
  ```
- **MySQL NULL Values:** Handles operations with rows that contain NULL values.
  Example :
  ```
  SELECT * FROM customers WHERE address IS NULL;
  ```
- **MySQL UPDATE:** Updates existing data within a table.
  Example :
  ```
  UPDATE customers SET status = 'active' WHERE id = 1;
  ```
- **MySQL DELETE:** Deletes data from a table.
  Example :
  ```
  DELETE FROM customers WHERE id = 1;
  ```
- **MySQL LIMIT:** Specifies the maximum number of records to return.
  Example :
  ```
  SELECT * FROM products LIMIT 5;
  ```

Advanced SQL Usage

- **MySQL MIN and MAX:**Retrieves the minimum or maximum value of a chosen column.
  Example :
  ```
  SELECT MIN(price) AS LowestPrice FROM products;
  ```
- **MySQL MIN and MAX:** Performs calculations on numeric columns.
  Example :
  ```
  SELECT COUNT(*) AS TotalCustomers FROM customers;
  ```
- **MySQL LIKE:** Searches for a specified pattern in a column.
  Example :
  ```
  SELECT * FROM customers WHERE name LIKE 'a%';
  ```
- **MySQL Wildcards:** Uses symbols to replace zero or more characters in a pattern.
  Example :
  ```
  SELECT * FROM products WHERE name LIKE 'win%';
  ```
- **MySQL IN:** Allows you to specify multiple values in a WHERE clause.
  Example :
  ```
  SELECT * FROM customers WHERE country IN ('Germany', 'France');
  ```
- **MySQL Aliases:** Provides a temporary name for columns or tables.
  Example :
  ```
  SELECT customer_id AS ID, customer_name AS Customer FROM customers;
  ```

Joins and Set Operations

- **MySQL Joins :** Combines rows from two or more tables based on a related column.
- **MySQL INNER JOIN :** Returns records that have matching values in both tables.
  Example :
  ```
  SELECT orders.order_id, customers.customer_name FROM orders INNER JOIN
  customers ON orders.customer_id = customers.customer_id;
  ```
- **MySQL LEFT JOIN :** Returns all records from the left table, and the matched records from the right table.
  Example :
  ```
  SELECT orders.order_id, customers.customer_name FROM  orders LEFT JOIN
  customers ON orders.customer_id = customers.customer_id;
  ```
- **MySQL RIGHT JOIN :** Returns all records from the right table, and the matched records from the left table.
  Example :
  ```
  SELECT orders.order_id, customers.customer_name FROM orders RIGHT
  JOIN customers ON orders.customer_id = customers.customer_id;
  ```
- **MySQL CROSS JOIN :** Returns the Cartesian product of two or more tables.
  Example :
  ```
  SELECT a.product_name, b.category_name FROM products a  CROSS JOIN categories b;
  ```
- **MySQL Self Join :** Joins a table to itself.
  Example :
  ```
  SELECT A.customer_name AS CustomerName1, B.customer_name AS CustomerName2
  FROM customers A, customers B WHERE A.customer_id <> B.customer_id;
  ```
- **MySQL UNION :** Combines the result sets of two or more SELECT statements.
  Example :
  ```
  SELECT city FROM customers UNION SELECT city FROM suppliers;
  ```
- **MySQL GROUP BY :** Groups rows that have the same values into summary rows.
  Example :
  ```
  SELECT COUNT(customer_id), country FROM customers GROUP BY country;
  ```
- **MySQL HAVING :** Used with the GROUP BY clause to filter groups based on a condition.
  Example :
  ```
  SELECT COUNT(customer_id), country FROM customers
  GROUP BY country HAVING COUNT(customer_id) > 5;
  ```

Advanced Features and Database Management

- **MySQL EXISTS:**Tests whether a subquery returns any records.
  Example :
  ```
  SELECT product_name FROM products WHERE EXISTS (SELECT  product_id FROM
  order_items WHERE products.product_id = order_items.product_id);
  ```
- **MySQL ANY and ALL:** Compares a value to each value in a list or returned by a subquery.
  Example :
  ```
  SELECT * FROM products WHERE price > ANY (SELECT price FROM products
  WHERE price > 100);
  ```
- **MySQL INSERT SELECT:** Inserts data into a table from another table.
  Example :
  ```
  INSERT INTO product_log SELECT * FROM products WHERE discontinued = 1;
  ```
- **MySQL CASE:** Provides if-else logic within SQL statements.
  Example :
  ```
  SELECT order_id, CASE WHEN order_total > 1000 THEN 'High' ELSE 'Low' END AS
  OrderLevel FROM orders;
  ```
- **MySQL Null Functions:** Functions to handle NULL values, like IFNULL or COALESCE.
  Example :
  ```
  SELECT product_name, COALESCE(description, 'No description') FROM products;
  ```
- **MySQL Null Functions:** Comments can be added with -- for single line, or /* */ for multiple lines.
- **MySQL Operators:** Operators are used to specify conditions in an SQL statement, like arithmetic, comparison, and logical operators.

Database Definitions and Table Management

- **MySQL Create DB:** Creates a new database.
  Example :
  ```
  CREATE DATABASE dbname;
  ```
- **MySQL Drop DB:** Deletes an existing database.
  Example :
  ```
  DROP DATABASE dbname;
  ```
- **MySQL Create Table:** Defines a new table and its structure.
  Example :
  ```
  CREATE TABLE customers (customer_id int, name varchar(255), address
  varchar(255));
  ```
- **MySQL Drop Table:** Deletes an existing table.
  Example :
  ```
  DROP TABLE tablename;
  ```
- **MySQL Alter Table:** Changes an existing table structure, such as adding a new column.
  Example :
  ```
  ALTER TABLE customers ADD email varchar(255);
  ```
- **MySQL Constraints:** Defines rules regarding the values allowed in columns.
  Example :
  ```
  CREATE TABLE products (product_id int NOT NULL, product_name varchar(255)
  NOT NULL, PRIMARY KEY (product_id));
  ```
- **MySQL Not Null:** Ensures that a column cannot have a NULL value.
- **MySQL Unique:** Ensures all values in a column are different.
- **MySQL Primary Key:** A field in a table which uniquely identifies each row/record in that table.
- **MySQL Foreign Key:** A field (or collection of fields) in one table that uniquely identifies a row of another table.
- **MySQL Check:** Ensures the value in columns meets a specific condition.
- **MySQL Default:** Sets a default value for a column when no value is specified.
- **MySQL Create Index:** Used to create and retrieve data from the database very quickly.
  Example :
  ```
  CREATE INDEX idx_lastname ON persons (LastName);
  ```
- **MySQL Auto Increment:** AutomaticallyHere's a comprehensive cheat sheet for MySQL, organizing the topics in a manner that provides clear instructions and examples for using MySQL effectively:

Basic Concepts

- **MySql Intro:** Introduction to MySQL, a popular open-source relational database management system.
- **MySql RDBMS:** MySQL operates as a relational database system allowing for effective management and retrieval of large volumes of data.

SQL Commands and Syntax

- **MySql SQL:** The foundational language used for managing and querying a MySQL database.
- **MySql SELECT:** Retrieves data from one or more tables.
  Example :
  ```
  SELECT name, age FROM users;
  ```
- **MySql WHERE:** Specifies conditions to filter records.
  Example :
  ```
  SELECT name, age FROM users WHERE age >= 18;
  ```
- **MySql AND, OR, NOT:** Combines multiple conditions.
  Example :
  ```
  SELECT name, age FROM users WHERE age >= 18 AND age <= 30;
  ```
- **MySql ORDER BY:** Sorts the results.
  Example :
  ```
  SELECT name, age FROM users ORDER BY name;
  ```
- **MySql INSERT INTO:** Adds new rows to a table.
  Example :
  ```
  INSERT INTO users (name, age) VALUES ('John', 28);
  ```
- **MySql NULL Values:** Handles fields with no data.
  Example :
  ```
  SELECT name, age FROM users WHERE address IS NULL;
  ```
- **MySql UPDATE:** Modifies existing records.
  Example :
  ```
  UPDATE users SET age = 29 WHERE name = 'John';
  ```
- **MySql DELETE:** Removes existing records.
  Example :
  ```
  DELETE FROM users WHERE name = 'John';
  ```
- **MySql LIMIT:** Constrains the number of returned records.
  Example :
  ```
  SELECT * FROM users LIMIT 5;
  ```
- **MySql COUNT, AVG, SUM:** Performs calculations on specified columns.
  Example :
  ```
  SELECT COUNT(*), AVG(salary), SUM(salary) FROM employees;
  ```
- **MySql LIKE:** Filters records based on pattern matching.
  Example :
  ```
  SELECT * FROM users WHERE name LIKE 'J%';
  ```
- **MySql Wildcards:** Enhances flexibility in pattern matching.
  Example :
  ```
  SELECT * FROM users WHERE name LIKE 'J_n%';
  ```
- **MySql IN:** Matches any value within a set.
  Example :
  ```
  SELECT * FROM users WHERE age IN (25, 30);
  ```
- **MySql BETWEEN:** Retrieves values within a range.
  Example :
  ```
  SELECT * FROM users WHERE age BETWEEN 25 AND 30;
  ```
- **MySql Aliases:** Renames a column or table for the query duration.
  Example :
  ```
  SELECT name AS username FROM users;
  ```

Joins and Set Operations

- **MySql Joins:** Combines rows from two or more tables.
- **MySql INNER JOIN:** Matches rows in both tables based on a related column.
  Example :
  ```
  SELECT users.name, orders.order_id FROM users INNER JOIN orders ON users.id =
  orders.user_id;
  ```
- **MySql LEFT JOIN:** Returns all rows from the left table, and matched rows from the right.
  Example :
  ```
  SELECT users.name, orders.order_id FROM users LEFT JOIN orders
  ON users.id = orders.user_id;
  ```
- **MySql RIGHT JOIN:** Returns all rows from the right table, and matched rows from the left.
  Example :
  ```
  SELECT users.name, orders.order_id FROM users RIGHT JOIN
  orders ON users.id = orders.user_id;
  ```
- **MySql CROSS JOIN:** Produces a Cartesian product of the rows in the tables.
  Example :
  ```
  SELECT users.name, products.name FROM users CROSS JOIN products;
  ```
- **MySql Self Join:** Joins a table to itself as if the table were two tables.
  Example :
  ```
  SELECT A.name, B.name FROM users A, users B WHERE A.age = B.age
  AND A.id != B.id;
  ```
- **MySql UNION:** Combines the result sets of two or more SELECT statements.
  Example :
  ```
  SELECT name FROM users UNION SELECT name FROM admins;
  ```
- **MySql GROUP BY:** Groups rows sharing a property so aggregate functions can be applied to each group.
  Example :
  ```
  SELECT age, COUNT(*) FROM users GROUP BY age;
  ```
- **MySql HAVING:** Specifies a search condition for a group or an aggregate.
  Example :
  ```
  SELECT age, COUNT(*) FROM users GROUP BY age HAVING COUNT(*) > 1;
  ```

Database and Table Management

- **MySql Create DB:** Creates a new database.
  Example :
  ```
  CREATE DATABASE sampleDB;
  ```
- **MySql Drop DB:** Deletes an existing database.
  Example :
  ```
  DROP DATABASE sampleDB;
  ```
- **MySql Create Table:** Creates a new table in the database.
  Example :
  ```
  CREATE TABLE users (id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(100),
  age INT);
  ```
- **MySql Drop Table:** Deletes an existing table.
  Example :
  ```
  DROP TABLE tablename;
  ```
- **MySql Alter Table:** Modifies an existing table structure, like adding a new column.
  Example :
  ```
  ALTER TABLE users ADD email VARCHAR(255);
  ```
- **MySql Constraints:** Applies rules to table columns.
  Example :
  ```
  CREATE TABLE users (id INT NOT NULL, name VARCHAR(100), PRIMARY KEY(id));
  ```

MySQL Data Types and Functions

MySQL Data Types

MySQL categorizes data types in several broad categories:

Numeric Types

- **INT:** An integer data type.
  Example :
  ```
  age INT
  ```
- **DECIMAL(M, D):** A fixed-point number with M specified as the maximum number of digits (the precision) and D as the number of digits to the right of the decimal.
  Example :
  ```
  price DECIMAL(5,2)
  ```
- **FLOAT:** A single-precision floating point number.
  Example :
  ```
  score FLOAT
  ```
- **DOUBLE:** A double-precision floating point number.
  Example :
  ```
  score DOUBLE
  ```

Date & Time Types

- **DATE:** Stores a date value in YYYY-MM-DD format. Example;
  Example :
  ```
  birth_date DATE
  ```
- **TIME:** Stores time in HH:MM format.
  Example :
  ```
  appointment_time TIME
  ```
- **DATETIME:** Stores both date and time
  Example :
  ```
  created_at DATETIME
  ```
- **TIMESTAMP:** Stores timestamp values, typically used for recording the time of an event.
  Example :
  ```
  last_updated TIMESTAMP
  ```
- **YEAR:** Stores a year in two-digit or four-digit format.
  Example :
  ```
  graduation_year YEAR
  ```