Contents

----------SQL Commands DDL----------

Creating Database

This will create database.

```
CREATE DATABASE databasename;
```

Deleting Database

This will delete database

```
DROP DATABASE databasename;
```

Show Database

This will show all database in server

```
SHOW DATABASES;
```

Create Table

This will create new table

```
CREATE TABLE table_name (
column1 datatype,
column2 datatype,
column3 datatype,
....
);
```

Show Tables

This will show all tables in database

```
Show Tables;
```

Create Table Using Another Table

This will create table using existing table

```
CREATE TABLE new_table_name AS
SELECT column1, column2,...
FROM existing_table_name
WHERE ....;
```

Delete Table

This will delete table with structure;

```
DROP TABLE table_name;
```

ALTER TABLE - ADD Column

This will add new column to table structure

```
ALTER TABLE table_name
ADD column_name datatype;
```


ALTER TABLE - DROP COLUMN

This will delete table from structure

```
ALTER TABLE table_name
DROP COLUMN column_name;
```

ALTER TABLE - RENAME COLUMN

This will rename column name ;

```
ALTER TABLE tablename CHANGE `name` `newname` data_type;
```

ALTER TABLE - ALTER/MODIFY DATATYPE

This will modify the datatype

```
ALTER TABLE table_name
MODIFY COLUMN column_name datatype;
```

Truncate table

This will delete table content not structure.

TRUNCATE TABLE table_name;

----------DML (Insert Data into Table)----------

This will insert the data in table

INSERT INTO *table_name*
VALUES (*value1*, *value2*, *value3*, ...);

----------SQL DQL Commands----------

Select data from database

The SELECT statement is used to select data from a database.

SELECT *column1*, *column2*, ...
FROM *table_name*;

Select distinct

The SELECT DISTINCT statement is used to return only distinct (different) values.
SELECT DISTINCT *column1*, *column2*, ...
FROM *table_name*;

where
The WHERE clause is used to filter records.

SELECT *column1*, *column2*, ...
FROM *table_name*
WHERE *condition*;

AND Operator

The WHERE clause can contain one or many AND operators.

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 AND condition2 AND condition3 ...;
```

OR Operator

The WHERE clause can contain one or many OR operators.

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 OR condition2 OR condition3 ...;
```

NOT Operator

The NOT operator is used in combination with other operators to give the opposite result

```
SELECT column1, column2, ...
FROM table_name
WHERE NOT condition;
```

Update

The UPDATE statement is used to modify the existing records in a table.

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

Delete

The DELETE statement is used to delete existing records in a table.

```
DELETE FROM table_name WHERE condition;
```

Limit

The Limit clause is used to specify the number of records to return.

SELECT *column_name(s)*
FROM *table_name*
WHERE *condition*
LIMIT *number*;

NULL Value

A field with a NULL value is a field with no value.

SELECT *column_names*
FROM *table_name*
WHERE *column_name* IS NULL;

SQL Aggregate Functions

An aggregate function is a function that performs a calculation on a set of values, and returns a single value.

- MIN() - returns the smallest value within the selected column
- MAX() - returns the largest value within the selected column
- COUNT() - returns the number of rows in a set
- SUM() - returns the total sum of a numerical column
- AVG() - returns the average value of a numerical column

Min

SELECT MIN(*column_name*)
FROM *table_name*
WHERE *condition*;

Max

SELECT MAX(*column_name*)
FROM *table_name*
WHERE *condition*;

Count

SELECT COUNT(*column_name*)
FROM *table_name*
WHERE *condition*;

Sum

SELECT SUM(*column_name*)
FROM *table_name*
WHERE *condition*;

Avg

SELECT AVG(*column_name*)
FROM *table_name*
WHERE *condition*;

GROUP BY

Group by is used to grouping the columns based on same values.

The GROUP BY statement is often used with aggregate functions
(COUNT(), MAX(), MIN(), SUM(), AVG()) to group the result-set by one or more columns.

SELECT *column_name(s)*
FROM *table_name*
WHERE *condition*
GROUP BY *column_name(s)*
ORDER BY *column_name(s);*

HAVING Clause

The HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate functions.

SELECT *column_name(s)*
FROM *table_name*
WHERE *condition*
GROUP BY *column_name(s)*
HAVING *condition*
ORDER BY *column_name(s);*

LIKE Operator

The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

There are two wildcards often used in conjunction with the LIKE operator:

- The percent sign % represents zero, one, or multiple characters
- The underscore sign _ represents one, single character

SELECT *column1, column2, ...*
FROM *table_name*
WHERE *columnN* LIKE *pattern*;

IN Operator

The IN operator allows you to specify multiple values in a WHERE clause.

SELECT *column_name(s)*
FROM *table_name*
WHERE *column_name* IN (*value1*, *value2*, ...);

NOT IN

SELECT *column_name(s)*
FROM *table_name*
WHERE *column_name* NOT IN (*value1*, *value2*, ...);

BETWEEN Operator

The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates.

SELECT *column_name(s)*
FROM *table_name*
WHERE *column_name* BETWEEN *value1* AND *value2;*

----------Keys----------

Primary key

The PRIMARY KEY constraint uniquely identifies each record in a table.

Primary keys must contain UNIQUE values, and cannot contain NULL values.

A table can have only ONE primary key; and in the table, this primary key can consist of single or multiple columns (fields).

```
CREATE TABLE City (
   ID int NOT NULL,
   City_Name varchar(255) NOT NULL,
   PRIMARY KEY (ID)
);
```

Create Primary key after creating table

```
ALTER TABLE Persons
ADD PRIMARY KEY (ID);
```

Foreign Keys

A FOREIGN KEY is a field (or collection of fields) in one table, that refers to the PRIMARY KEY in another table.

```
CREATE TABLE Employee (
  Employee_id int NOT NULL,
   City_id int NOT NULL,
   PRIMARY KEY (id),
   FOREIGN KEY (City_id) REFERENCES CIty(id)
 ON Delete Cascade
 ON Update Cascade
);
```


Constraints

Define Specific rules for the Table

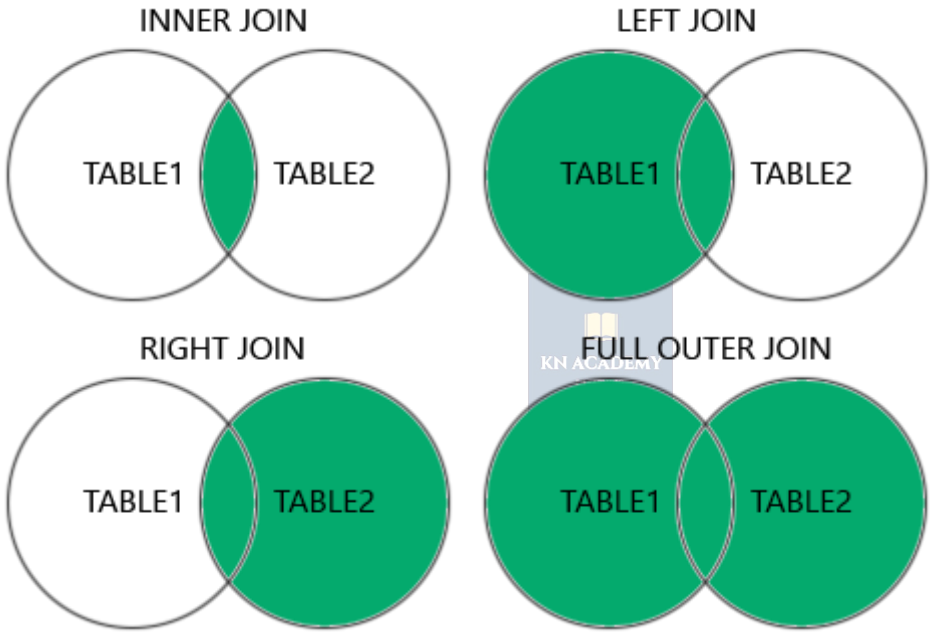
```
CREATE TABLE Persons (
   Student_ID INT NOT NULL,
   Teacher_id INT UNIQUE ,
   Age int check(Age >= 18) ,
   Salary int DEFAULT 10000
   );
```

Joins

A JOIN clause is used to combine rows from two or more tables, based on a related column between them.

Here are the different types of the JOINs in SQL:

- **(INNER) JOIN:** Returns records that have matching values in both tables
- **LEFT (OUTER) JOIN:** Returns all records from the left table, and the matched records from the right table
- **RIGHT (OUTER) JOIN**: Returns all records from the right table, and the matched records from the left table
- **FULL (OUTER) JOIN:** Returns all records when there is a match in either left or right table



SQL Subqueries

A query within SQL Query

1) Select
2) From
3) Where

1) SELECT …. (Subquery) as ColumnName from TableName;

2) SELECT …. From (Subquery) as TableName , TableName;
3) SELECT …. From TableName Where Condition Operator (Subquery)

Views

Virtual tables based on the result set , A view will be automatically updated if table is updated

Create View ViewName as select ColumnNames… from TableName

Date And time

MySQL :: MySQL 8.4 Reference Manual :: 14.7 Date and Time Functions