Database

Link providing : https://www.w3schools.com/sql/

Insert Statement

INSERT statement is is used to populated a table with rows. used to insert data in to the row of the table.

```
insert into table_name values (.., ... , ..., ..);
```

Example:

insert single record:

```
INSERT INTO users(name, gender, one_signal_id, is_student, student_card_id) values ('seangleng', 'M', 'kjhdifuhsdh3498weeifh83
9', TRUE, '3234853');
```

Insert multiple insert:

```
INSERT INTO users(name, gender, one_signal_id, is_student, student_card_id) values , ('seangleng', 'M', 'kjhdifuhsdh3498weeifh8
39', TRUE, '3234853'), ('jenzy', 'Male', 'wysdfjns84i3y5', TRUE, '2145212');
```

SELECT statement

used to select to get something from the table.

```
SELECT * FROM table_name;
```

Example:

```
-- select every column of the table.
SELECT * FROM users;

-- use select to select columns that you want to know.
select name, gender, is_student from users;
-- or you can write the query like this with rename to column name, but in database it is not rename the table name in database.:
select name as studentName, gender, is_student from users
-- studentName is column-name of column name.
```

SELECT distinct

Select distinct used to select the elements of the table, without duplicate the elements.

```
SELECT DISTINCT name, gender, is_student FROM users;
```

CLAUSES

Clauses in sql included, **WHERE, GROUP BY, ORDER BY, TOP, WITH, LIKE, FROM, LIMIT, AND, OR**.

▼ WHERE

```
SELECT * FROM users WHERE id BETWEEN 1 AND 15;
```

▼ IN

```
SELECT * FROM users WHERE id IN(1, 3, 5, 10)
```

▼ specific condition with where

used to find the specific data.

```
SELECT * FROM users WHERE name = 'jenzy';
```

▼ LIKE

```
SELECT * FROM users, WHERE name like 'jen%';

SELECT * FROM users, WHERE name ILIKE '%jen%';

SELECT * FROM users, WHERE name ILIKE '%jen';
```

▼ GROUP BY

```
SELECT sender_account_id, SUM(withdrawal_amount) FROM transactions
WHERE sender_account_id = 15 GROUP BY sender_account_id;
-- the result will have only sender_account_id that equal to 15.

-- if you want to find all the amount of withdrawal_account of sender_account_id
SELECT sender_account_id, SUM(withdrawal_amount) FROM transactions GROUP BY sender_account_id;
```

▼ HAVING

```
SELECT sender_account_id, SUM(withdrawal_amount)
FROM transactions
GROUP BY sender_account_id
HAVING SUM(withdrawal_amount) < 10056009034::money
```

▼ ORDER BY

```
-- Z to A
SELECT sender_account_id, SUM(withdrawal_amount) as total
FROM transactions
GROUP BY sender_account_id
HAVING SUM(withdrawal_amount) > 1056009034::money
ORDER BY total DESC

-- A to Z (default)
SELECT sender_account_id, SUM(withdrawal_amount) as total
FROM transactions
GROUP BY sender_account_id
```

```
HAVING SUM(withdrawal_amount) > 1056009034::money
ORDER BY total ASC
```

UPDATE statement

UPDATE statement used to update the data in database :

Syntax:

```
UPDATE table_name SET column_name where condition;
```

Example :

```
UPDATE users SET name = 'jenzy' where name = 'seangleng'
```

DELETE statement

Delete statement used to delelte data in database table by properties.

syntax :

```
DELETE FROM table_name where conditon;
```

SORTING ROW

syntax:

```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1, column2, ... ASC|DESC;
```

example:

```
SELECT * FROM Customers
ORDER BY Country ASC, CustomerName DESC;
```

JOIN CLAUSE

Join clause used to combine records from other tables.

▼ INNER JOIN or JOIN

find something that have the same values.

syntax:

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

Inner join more than 2 tables: (`hint` : used `JOIN` multiple times)

```
SELECT a.id, d.name, a.account_name, a.phone_number, a.transfer_limit,
  b.name AS account_Types
  FROM accounts AS a
  JOIN account_types AS b
  ON a.account_type = b.id
  JOIN user_accounts as c ON a.id = c.account_id
  JOIN users as d
  ON c.user_id = d.id
```

▼ LEFT JOIN

syntax:

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

▼ RIGHT JOIN

syntax:

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

INDEX

indexes : are the special look up tables that the database search engine can use to speed up data retrieval. an a index is a pointer to data in a table.

- it helps us to speed up `SELECT` queries and where clause, however, it shows down data input, with `UPDATE` and `INSERT` statement.

The syntax :

```
CREATE INDEX index_name
ON table_name (column1, column2, ...);
```

The syntax of creating unique index (Dubplicate values are not allowed) :

```
CREATE UNIQUE INDEX index_name
ON table_name (column1, column2, ...);
```

Using built-In Function

Built-In function is a kind of function which bult for user use normally.

Example:

- **AVG():** It returns the average value of the column.
- **COUNT():** It returns the number of rows in the table.
- **FIRST():** It returns the first value of the column.

- **LAST():** It returns the last value
- **MAX():** It returns the largest value of the column.
- **MIN():** It returns the smallest value of the column.
- **SUM():** It returns the sum of rows of the table.
- And other :

MySQL Functions

W3Schools offers free online tutorials, references and exercises in all the major languages of the web. Covering popular subjects like HTML, CSS, JavaScript, Python, SQL, Java, and many, many more.

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

Creating function

Syntax :

```
CREATE FUNCTION "public"."function_name"()
 AS $BODY$
  BEGIN

  -- Routine body goes here...

  RETURN;
  END
  $BODY$
  LANGUAGE plpgsql

// Call to use
SELECT function_name();
```

Example:

Coding the function to update the data in the table.

```
CREATE OR REPLACE FUNCTION "public"."my_query"()
  RETURNS "pg_catalog"."void" AS $BODY$ BEGIN
    UPDATE users SET "name" = 'sokKha' WHERE "id" = 3;
  RETURN;
  END
  $BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100

// call function to use in another queries :
SELECT my_query(parameter); // if we have parameter
```

More detail:

Functions In SQL Server: A Beginner's Guide

This article explains how to create functions in SQL Server using the SQL function operator.

https://www.c-sharpcorner.com/UploadFile/b926a6/function-operation-in-sql-database/

Stored Procedures

Syntax :

```
CREATE OR REPLACE PROCEDURE pro_student (p_id INTEGER, p_name VARCHAR)
LANGUAGE plpgsql
AS $BODY$
  BEGIN
   -- sql code
  END;
$BODY$;

 -- When we need to call the procedure function
CALL pro_student();  -- just use CALL with function name to use the function.
```

Note : Procedure is not used for returning tables, but we almost use it with `insert`, `update`, and `delete`.

Store procedures is type of function that use to perform actions on the database.

Function can not create costum aggregate function or to define custom operation.

SQL Stored Procedures (With Examples)

In SQL, a stored procedure is a set of statement(s) that perform some defined actions. In this tutorial, you will learn about stored procedures in SQL with the help of examples.

https://www.programiz.com/sql/stored-procedures

Function that Return a Table

syntax :

```
CREATE OR REPLACE FUNCTION myreacord()
RETURN TABLE(name VARCHAR)
AS $$
BEGIN
  -- your code here
END
$$
LANGUAGE 'plpgsql';
```

example :

```
-- Call the function after created the function.
SELECT * from myreacordejenzy();

CREATE OR REPLACE FUNCTION myreacordejenzy()
RETURNS TABLE(stu_name_h VARCHAR, gender_student VARCHAR, is_student_orNot BOOLEAN)
AS $$
BEGIN

  RETURN QUERY SELECT name, gender, is_student
  FROM users;
END
$$
LANGUAGE 'plpgsql';
```

More detail :

How to Develop a PL/pgSQL Function That Returns a Table

In this tutorial, we will show you how to develop PostgreSQL functions that return a table.

https://www.postgresqltutorial.com/postgresql-plpgsql/plpgsql-function-returns-a-table/