DDL statements

CREATE

Creating a Table

Syntax :

```
CREATE TABLE table_name (
column1 datatype [constraints],
column2 datatype [constraints],
...
);
```

Example :

```
CREATE TABLE employees (
employee_id SERIAL PRIMARY KEY,
first_name VARCHAR(50),
last_name VARCHAR(50),
department VARCHAR(50),
hire_date DATE
);
```

Example from Lab Experiment :

```
-- Create the buses table
CREATE TABLE bus (
    bus_id INTEGER PRIMARY KEY,
    bus_number VARCHAR(20) UNIQUE NOT NULL,
    capacity INTEGER NOT NULL CHECK (capacity > 0)
);


 -- Create the routes table
CREATE TABLE routes (
    route_id INTEGER PRIMARY KEY,
    origin VARCHAR(100) NOT NULL,
    destination VARCHAR(100) NOT NULL,
    distance INTEGER NOT NULL CHECK (distance > 0)
);


 -- Create the schedules table
CREATE TABLE schedules (
    schedule_id INTEGER PRIMARY KEY,
    route_id INTEGER NOT NULL REFERENCES routes (route_id),
    bus_id INTEGER NOT NULL REFERENCES buses (bus_id),
    departure_time TIMESTAMP NOT NULL,
    arrival_time TIMESTAMP NOT NULL,
    CHECK (departure_time < arrival_time)
);
```

```
-- Create the customers table
CREATE TABLE customers (
   customer_id INTEGER PRIMARY KEY,
   name VARCHAR(100) NOT NULL,
   email VARCHAR(100) UNIQUE NOT NULL,
   phone_number VARCHAR(15) UNIQUE NOT NULL
);

-- Create the bookings table
CREATE TABLE bookings (
   booking_id INTEGER PRIMARY KEY,
   customer_id INTEGER NOT NULL REFERENCES customers (customer_id),
   schedule_id INTEGER NOT NULL REFERENCES schedules (schedule_id),
   booking_date TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
   UNIQUE (customer_id, schedule_id, booking_date)
);

-- Create the tickets table
CREATE TABLE tickets (
ticket_id INTEGER PRIMARY KEY,
booking_id INTEGER NOT NULL REFERENCES bookings (booking_id),
seat_number VARCHAR(10) NOT NULL,
UNIQUE (booking_id, seat_number)
);
```

ALTER

Adding a Column

Syntax :

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

Example :

```
ALTER TABLE employees ADD COLUMN email VARCHAR(100);
```

Modifying a Column

Syntax :

```
ALTER TABLE table_name ALTER COLUMN column_name TYPE new_datatype;
```

Example :

```
ALTER TABLE employees ALTER COLUMN hire_date TYPE TIMESTAMP;
```

Dropping a Column

Syntax :

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

Example :

```
ALTER TABLE employees DROP COLUMN email;
```

Example from Lab Experiment :

```
ALTER TABLE buses ADD COLUMN description TEXT;

ALTER TABLE buses DROP COLUMN description;

ALTER TABLE buses ADD CONSTRAINT unique_bus_number UNIQUE (bus_number);

ALTER TABLE schedules ADD CONSTRAINT fk_bus FOREIGN KEY (bus_id) REFERENCES
buses (bus_id);

ALTER TABLE buses DROP CONSTRAINT unique_bus_number;


ALTER TABLE buses ALTER COLUMN bus_number SET NOT NULL;

ALTER TABLE buses ALTER COLUMN bus_number DROP NOT NULL;
```

DROP

Used to delete objects from the database

Dropping a Table

Example :

```
DROP TABLE table_name;
```

Syntax :

```
DROP TABLE employees;
```

Dropping a Database

Example :

```
DROP DATABASE database_name;
```

Syntax :

```
DROP DATABASE company_db;
```

TRUNCATE

Used to remove all rows from a table without removing the table itself

Syntax :

```
TRUNCATE TABLE table_name;
```

Example :

```
TRUNCATE TABLE employees;
```

DML Operations

INSERT

Syntax :

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

Example :

```
INSERT INTO employees (first_name, last_name, department, hire_date)
VALUES ('John', 'Doe', 'Sales', '2023-01-10');
```

Inserting Multiple Rows :

```
INSERT INTO employees (first_name, last_name, department, hire_date)
VALUES
('Jane', 'Doe', 'Marketing', '2023-02-15'),
('Alice', 'Smith', 'IT', '2023-03-20');
```

Inserting Data without Column List :

```
INSERT INTO employees
VALUES (1, 'Michael', 'Scott', 'Management', '2024-05-25');
```

Example from Lab Assignment :

```
-- Insert rows into the schedules table
INSERT INTO schedules (route_id, bus_id, departure_time, arrival_time) VALUES
(1, 1, '2024-09-08 08:00:00', '2024-09-08 12:00:00'),
(2, 2, '2024-09-08 09:00:00', '2024-09-08 13:00:00'),
(3, 3, '2024-09-08 10:00:00', '2024-09-08 14:00:00'),
(4, 4, '2024-09-08 11:00:00', '2024-09-08 15:00:00'),
(5, 5, '2024-09-08 12:00:00', '2024-09-08 16:00:00');

-- View schedules table after insert
SELECT * FROM schedules;

-- Insert rows into the customers table
INSERT INTO customers (name, email, phone_number) VALUES
('Ritesh Jha', 'ritesh@example.com', '1234567890'),
('Anita Sharma', 'anita@example.com', '0987654321'),
('Nita Patil', 'nita@example.com', '1122334455');

-- View customers table after insert
SELECT * FROM customers;

-- Insert rows into the bookings table
INSERT INTO bookings (customer_id, schedule_id) VALUES
(1, 1),
(2, 2),
(3, 3);

-- View bookings table after insert
SELECT * FROM bookings;

-- Insert rows into the tickets table
INSERT INTO tickets (booking_id, seat_number) VALUES
(1, 'A1'),
(2, 'B1'),

(3, 'C1');

-- View tickets table after insert
SELECT * FROM tickets;
```

UPDATE

Syntax :

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

Example :

```
UPDATE employees
SET department = 'Finance'
WHERE employee_id = 1;
```

Updating Multiple Columns :

```
UPDATE employees
SET first_name = 'Jim', last_name = 'Halpert', department = 'Sales'
WHERE employee_id = 2
```

;

Example from Lab Assignment :

```
-- Update the email of a customer
UPDATE customers SET email = 'ritesh_updated@example.com' WHERE customer_id = 1;

-- View customers table after update
SELECT * FROM customers;

-- Increase the capacity of a bus
UPDATE buses SET capacity = capacity + 10 WHERE bus_id = 2;

-- View buses table after update
SELECT * FROM buses;

-- Change the destination of a route
UPDATE routes SET destination = 'City Z' WHERE route_id = 5;

-- View routes table after update
SELECT * FROM routes;

-- Update the phone number of a customer
UPDATE customers SET phone_number = '9988776655' WHERE customer_id = 3;

-- View customers table after update
SELECT * FROM customers;
```

DELETE

Syntax :

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

Example :

```
DELETE FROM employees
WHERE employee_id = 1;
```

Deleting Multiple Rows :

```
DELETE FROM employees
WHERE department = 'Sales';
```

Example from Lab Assignment :

```
-- Delete a booking
DELETE FROM bookings WHERE booking_id = 2;

-- View bookings table after delete
SELECT * FROM bookings;

-- Remove a customer who has not made any bookings
DELETE FROM customers WHERE customer_id = 2;

-- View customers table after delete
SELECT * FROM customers;

-- Delete a bus with a specific bus number
DELETE FROM buses WHERE bus_number = 'BUS103';

-- View buses table after delete
SELECT * FROM buses;

-- Delete all routes with distance greater than 200
DELETE FROM routes WHERE distance > 200;

-- View routes table after delete
SELECT * FROM routes;
```

‘from’ and ‘where’ clause

SELECT with FROM and WHERE

```
SELECT first_name, last_name
FROM employees
WHERE department = 'IT';
```

DELETE with FROM and WHERE

```
DELETE FROM employees
WHERE hire_date < '2023-01-01';
```

UPDATE with FROM and WHERE

```
UPDATE employees
SET department = 'Marketing'
WHERE department = 'Sales' AND hire_date > '2022-01-01';
```

Example from Lab Assignment :

```
-- Select all customers
SELECT * FROM customers;


-- Select all buses with a capacity greater than 50
SELECT * FROM buses WHERE capacity > 50;

-- Select the name and email of customers who have booked a specific schedule
SELECT name, email FROM customers WHERE customer_id IN (SELECT
customer_id FROM bookings WHERE schedule_id = 1);


-- Select distinct destinations from the routes table
SELECT DISTINCT destination FROM routes;

-- Select schedules where the departure time is before a specific time
SELECT * FROM schedules WHERE departure_time < '2024-09-06 10:00:00';
```

Aggregate Functions

Simple Aggregates

COUNT

Syntax :

```
SELECT COUNT(*)
FROM table_name;
```

Example :

```
SELECT COUNT(*) AS total_employees
FROM employees;
```

SUM

Syntax :

```
SELECT SUM(column_name)
FROM table_name;
```

Example :

```
SELECT SUM(salary) AS total_salary
FROM employees;
```

AVG

Syntax :

```
SELECT AVG(column_name)
FROM table_name;
```

Example :

```
SELECT AVG(age) AS average_age
FROM employees;
```

MIN

Syntax :

```
SELECT MIN(column_name)
FROM table_name;
```

Example :

```
SELECT MIN(hire_date) AS earliest_hire_date
FROM employees;
```

MAX

Syntax :

```
SELECT MAX(column_name)
FROM table_name;
```

Example :

```
SELECT MAX(salary) AS highest_salary
FROM employees;
```

Examples from Lab Assignment :

```
-- Minimum Distance

SELECT MIN(distance) AS min_distance FROM routes;

-- Maximum Distance

SELECT MAX(distance) AS max_distance FROM routes;
```

```
-- Count of Routes

SELECT COUNT(*) AS total_routes FROM routes;
```

```
-- Sum of Distances

SELECT SUM(distance) AS total_distance FROM routes;
```

```
-- Average Distance

SELECT AVG(distance) AS avg_distance FROM routes;
```

Aggregate with Clauses (ORDER BY, GROUP BY, HAVING)

GROUP BY

Syntax :

```
SELECT column1, AGG_FUNC(column2)
FROM table_name
GROUP BY column1;
```

Example :

```
SELECT department, COUNT(*) AS num_employees
FROM employees
GROUP BY department;
```

ORDER BY

Syntax :

```
SELECT column1, AGG_FUNC(column2)
FROM table_name
GROUP BY column1
ORDER BY AGG_FUNC(column2) DESC;
```

Example :

```
SELECT department, AVG(salary) AS avg_salary
FROM employees
GROUP BY department
ORDER BY avg_salary DESC;
```

HAVING

Syntax :

```
SELECT column1, AGG_FUNC(column2)
FROM table_name
GROUP BY column1
HAVING AGG_FUNC(column2) > value;
```

Example :

```
SELECT department, COUNT( ) AS num_employees
FROM employees
GROUP BY department
HAVING COUNT(
) > 10;
```

Examples from Lab Assignment :

```
-- Sum of Distances, Ordered by Distance

SELECT SUM(distance) AS total_distance FROM routes

ORDER BY total_distance DESC;
```

```
-- Count of Routes, Grouped by Origin City

SELECT origin, COUNT(*) AS route_count FROM routes

GROUP BY origin

ORDER BY route_count DESC;
```

```
-- Having Clause - Cities with More than 1 Route

SELECT origin, COUNT(*) AS route_count FROM routes

GROUP BY origin

HAVING COUNT(*) > 1;
```

LIKE Operator

% - Represents zero or more characters

```
WHERE last_name LIKE '%son';
```

Finds all last names that end with 'son' (e.g., Johnson, Jackson).

_ - Represents a single character

```
WHERE first_name LIKE 'A__a';
```

Finds first names that start with 'A', followed by any two characters, and end with 'a' (e.g., 'Aida', 'Alba').

Examples :

```
WHERE email LIKE 'admin%@company.com';
```

start with 'admin' and end with '@company.com'.

```
WHERE department LIKE '%Engineering%';
```

contain the word 'Engineering'.

```
WHERE phone_number LIKE '555-%';
```

start with '555-'.

```
WHERE employee_id LIKE '12_4';
```

Finds employee IDs like '1234', '1244'

Examples from Lab Assignment :

```
-- Find Routes Originating from Cities Starting with 'D'
SELECT * FROM routes WHERE origin LIKE 'D%';
```

```
-- Find Routes Ending in Cities Starting with 'M'
SELECT * FROM routes WHERE destination LIKE 'M%';
```

```
-- Find Routes Where Origin Contains 'h'
SELECT * FROM routes WHERE origin LIKE '%h%';
```

Food Ordering App

DDL

-- Create All Tables

```
CREATE TABLE customers(
Customer_id SERIAL primary key,
Name varchar(50) NOT NULL,
Address varchar(255) NOT NULL,
Contactinfo varchar(50) NOT NULL
);
```

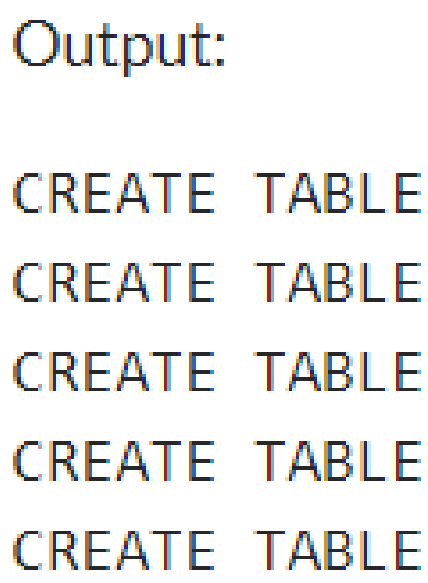
```
CREATE TABLE restaurants (
Restaurant_id SERIAL PRIMARY KEY,
Name varchar(50) NOT NULL,
Location varchar(255) NOT NULL
);
```

```
CREATE TABLE menu_items(
Menuitem_id SERIAL PRIMARY KEY,
Restaurant_id integer references restaurants(Restaurant_id),
Name varchar(50) NOT NULL,
Cost integer NOT NULL check(Cost>0)
);
```

```
CREATE TABLE orders(
Order_id SERIAL PRIMARY KEY,
Customer_id integer references customers(Customer_id),
Date timestamp default current_timestamp,
Amount integer NOT NULL check(Amount>0)
);
```

```
CREATE TABLE orderdetails(
Orderdetail_id SERIAL PRIMARY KEY,
Order_id integer references orders(Order_id),
Menuitem_id integer references menu_items(Menuitem_id),
Quantity integer NOT NULL check(Quantity>0)
);
```

Output :



DML

--INSERT

```
INSERT INTO customers(Name,Address,Contactinfo)
VALUES
('Ritesh','Milkyway Galaxy','ritesh@email.com'),
('Dev','Some Galaxy, somewhere','dev@company.com');
```

```
SELECT * From customers;
```

```
INSERT INTO restaurants(Name, Location)
VALUES('LaPinos Pizza','Jharkhand'),
('Restaurant 2', 'Mumbai');
```

```
SELECT * From restaurants;
```

```
INSERT INTO menu_items(Restaurant_id, Name, Cost)
VALUES
(1,'Cheese Pizza',15),
(1,'Dal Makhani',5),
(1,'Paneer chilly', 20),
(1,'BBQ Chicken', 40),
(1,'Farmhouse Pizza', 20),
(2,'Burger',5),
(2,'Chai', 2),
(2,'Burrito', 19);
```

```
SELECT * From menu_items;
```

Output :

```
INSERT 0 2
 customer_id |  name  |        address          |   contactinfo
-------------+--------+-------------------------+------------------
           1 | Ritesh | Milkyway Galaxy         | ritesh@email.com
           2 | Dev    | Some Galaxy, somewhere | dev@company.com
(2 rows)

INSERT 0 2
 restaurant_id |     name      | location
---------------+---------------+-----------
             1 | LaPinos Pizza | Jharkhand
             2 | Restaurant 2  | Mumbai
(2 rows)

INSERT 0 8
 menuitem_id | restaurant_id |      name       | cost
-------------+---------------+-----------------+------
           1 |             1 | Cheese Pizza    |   15
           2 |             1 | Dal Makhani     |    5
           3 |             1 | Paneer chilly   |   20
           4 |             1 | BBQ Chicken     |   40
           5 |             1 | Farmhouse Pizza |   20
           6 |             2 | Burger          |    5
           7 |             2 | Chai            |    2
           8 |             2 | Burrito         |   19
(8 rows)
```

--UPDATE

```
UPDATE customers
SET Address = '456 Elm St, Springfield'
WHERE Customer_id=1;
```

```
SELECT * From customers;
```

```
UPDATE menu_items
SET Cost = 25
WHERE Menuitem_id = 2;
```

```
SELECT * From menu_items;
```

Output :

```
UPDATE 1
 customer_id |  name  |         address          |   contactinfo
-------------+--------+--------------------------+------------------
           2 | Dev    | Some Galaxy, somewhere   | dev@company.com
           1 | Ritesh | 456 Elm St, Springfield  | ritesh@email.com
(2 rows)

UPDATE 1
 menuitem_id | restaurant_id |      name       | cost
-------------+---------------+-----------------+------
           1 |             1 | Cheese Pizza    |   15
           3 |             1 | Paneer chilly   |   20
           4 |             1 | BBQ Chicken     |   40
           5 |             1 | Farmhouse Pizza |   20
           6 |             2 | Burger          |    5
           7 |             2 | Chai            |    2
           8 |             2 | Burrito         |   19
           2 |             1 | Dal Makhani     |   25
(8 rows)
```

--DELETE

```
DELETE from customers
WHERE Customer_id = 1;
```

```
SELECT * From customers;
```

```
DELETE from menu_items
WHERE Menuitem_id = 2;
```

```
SELECT * From menu_items;
```

Output :

```
DELETE 1
 customer_id | name |        address         |   contactinfo
-------------+------+-------------------------+-----------------
           2 | Dev  | Some Galaxy, somewhere | dev@company.com
(1 row)

DELETE 1
 menuitem_id | restaurant_id |      name       | cost
-------------+---------------+-----------------+------
           1 |             1 | Cheese Pizza    |   15
           3 |             1 | Paneer chilly   |   20
           4 |             1 | BBQ Chicken     |   40
           5 |             1 | Farmhouse Pizza |   20
           6 |             2 | Burger          |    5
           7 |             2 | Chai            |    2
           8 |             2 | Burrito         |   19
(7 rows)
```

Aggregate Functions

-- Simple Aggregates

```
SELECT sum(Cost) as total_cost
from menu_items;
```

```
SELECT avg(Cost) as avg_cost
from menu_items;
```

```
select count(*) as total_dishes
from menu_items;
```

```
select min(Cost) as min_cost
from menu_items;
```

```
select max(Cost) as max_cost
from menu_items;
```

Output :

```
 total_cost
------------
        121
(1 row)

      avg_cost
----------------------
 17.2857142857142857
(1 row)

 total_dishes
--------------
            7
(1 row)

 min_cost
----------
        2
(1 row)

 max_cost
----------
       40
(1 row)
```

-- Aggregate with clauses

-- ORDER BY

```
SELECT Restaurant_id, SUM(Cost) AS total_cost
FROM menu_items
GROUP BY Restaurant_id;
```

-- GROUP BY

```
SELECT Restaurant_id, SUM(Cost) AS total_cost
FROM menu_items
GROUP BY Restaurant_id
ORDER BY total_cost DESC;
```

-- HAVING

```
SELECT Restaurant_id, SUM(Cost) AS total_cost
FROM menu_items
GROUP BY Restaurant_id
HAVING SUM(Cost) > 50;
```

Output :

```
 restaurant_id | total_cost
---------------+------------
             2 |         26
             1 |         95
(2 rows)

 restaurant_id | total_cost
---------------+------------
             1 |         95
             2 |         26
(2 rows)

 restaurant_id | total_cost
---------------+------------
             1 |         95
(1 row)
```

-- Like Operator

```
Select * FROM menu_items
WHERE Name LIKE '%Pizza';
```

```
Select * From menu_items
Where Name like 'Ch_i';
```

Output :

```
 menuitem_id | restaurant_id |      name       | cost
-------------+---------------+-----------------+------
           1 |             1 | Cheese Pizza    |   15
           5 |             1 | Farmhouse Pizza |   20
(2 rows)

 menuitem_id | restaurant_id | name | cost
-------------+---------------+------+------
           7 |             2 | Chai |    2
(1 row)
```