# Relational database-III

Manas Jyoti Das, PhD
Computer Science
SIUE

## Creating a table

```
Create database web;



use web;
CREATE TABLE list_of_names (
  id INT NOT NULL AUTO_INCREMENT,
  name VARCHAR(255) NOT NULL,
  PRIMARY KEY (id)
);
```

```
INSERT INTO my_table (name)
VALUES
  ('John Doe'),
  ('Jane Doe'),
  ('Peter Smith'),
  ('Mary Johnson'),
  ('Michael Brown'),
  ('David Williams'),
  ('Elizabeth Jones'),
  ('Robert Miller'),
  ('Patricia Garcia'),
  ('James Anderson');
```

## Count of monte cristo

```
use web;
SELECT COUNT(*) AS row_count FROM list_of_names;
```

# Avg function

```
use web;
drop table list_of_names;

CREATE TABLE num_table (
  id INT NOT NULL AUTO_INCREMENT,
  name VARCHAR(255) NOT NULL,
  value INT NOT NULL,
  PRIMARY KEY (id)
);
```

```
INSERT INTO num_table (name,
value) VALUES
  ('John Doe', 10),
  ('Jane Doe', 20),
  ('Peter Smith', 30),
  ('Mary Johnson', 40),
  ('Michael Brown', 50),
  ('David Williams', 60),
  ('Elizabeth Jones', 70),
  ('Robert Miller', 80),
  ('Patricia Garcia', 90),
  ('James Anderson', 100);
```

```
use web;
SELECT AVG(value) AS average_value FROM num_table;
```

# Max, min and sum

```
use web;
SELECT MAX(value) AS max_value FROM num_table;
```

```
use web;
SELECT MIN(value) AS min_value FROM num_table;
```

```
use web;
SELECT SUM(value) AS sum_value FROM num_table;
```

## Group by

```
use web;
CREATE TABLE info (
        id INT AUTO_INCREMENT PRIMARY KEY,
        name VARCHAR(255),
        department VARCHAR(255),
        salary DECIMAL(10, 2)
);
```

```
INSERT INTO info (name, department, salary)
VALUES
        ('John', 'HR', 50000.00),
        ('Alice', 'IT', 60000.00),
        ('Bob', 'IT', 55000.00),
        ('Eve', 'HR', 52000.00),
        ('Charlie', 'Sales', 48000.00),
        ('Grace', 'Sales', 49000.00),
        ('Daniel', 'IT', 58000.00),
        ('Sophia', 'HR', 51000.00),
        ('Liam', 'IT', 62000.00),
        ('Olivia', 'Sales', 47000.00);
```

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

## Having clause

```
use web;
SELECT department, AVG(salary) AS avg_salary
FROM info
GROUP BY department
HAVING AVG(salary) > 50000
ORDER BY avg_salary DESC;
```

## Inner join

```
use web;
CREATE TABLE customers (
	id INT AUTO_INCREMENT PRIMARY KEY,
	first_name VARCHAR(255),
	last_name VARCHAR(255)
);
INSERT INTO customers (first_name, last_name) VALUES
	('John', 'Doe'),
	('Alice', 'Smith'),
	('Bob', 'Johnson');
```

## Inner join

```
use web;
CREATE TABLE orders (
        order_id INT AUTO_INCREMENT PRIMARY KEY,
        customer_id INT,
        order_date DATE,
        total_amount DECIMAL(10, 2)
);
INSERT INTO orders (customer_id, order_date, total_amount)
VALUES
        (1, '2023-01-15', 100.00),
        (2, '2023-02-20', 150.00),
        (1, '2023-03-10', 75.50),
        (3, '2023-04-05', 200.00);
```

## Inner join

```
use web;
SELECT customers.first_name, customers.last_name, orders.order_date, orders.total_amount
FROM customers
INNER JOIN orders ON customers.id = orders.customer_id;
```

## Left join

```
use web
CREATE TABLE departments (
	department_id INT AUTO_INCREMENT
PRIMARY KEY,
	department_name VARCHAR(255)
);
INSERT INTO departments (department_name)
VALUES
	('HR'),
	('IT'),
	('Sales');
```

```
use web;
CREATE TABLE employees (
	employee_id INT AUTO_INCREMENT
PRIMARY KEY,
	first_name VARCHAR(255),
	last_name VARCHAR(255),
	department_id INT
);
INSERT INTO employees (first_name, last_name,
department_id) VALUES
	('John', 'Doe', 1),
	('Alice', 'Smith', 2),
	('Bob', 'Johnson', 2),
	('Eve', 'Anderson', 3),
	('Charlie', 'Brown', NULL);
```

## Left join

```
use web;
SELECT employees.first_name, employees.last_name, departments.department_name
FROM employees
LEFT JOIN departments ON employees.department_id = departments.department_id;

use web;
SELECT departments.department_name, employees.first_name, employees.last_name
FROM departments
RIGHT JOIN employees ON departments.department_id = employees.department_id;
```

Common use cases include situations where you want to retain all records from the left table, even if there are no matches in the right table.

RIGHT JOIN is less commonly used than LEFT JOIN and can often be achieved by simply reversing the order of the tables and using LEFT JOIN.

# Joins

- LEFT OUTER JOIN clause - Joins all rows from the left table along with rows from the right table where the join condition is met.
- RIGHT OUTER JOIN clause - Joins all rows from the right table along with rows from the left table where the join condition is met.
- FULL OUTER JOIN clause - Joins all rows from the left and right tables regardless if the join condition is met.