# Database: SQL Query II

Hari, Tanggal

Data Analytics

Program Zenius Studi Independen Bersertifikat Bersama Kampus Merdeka

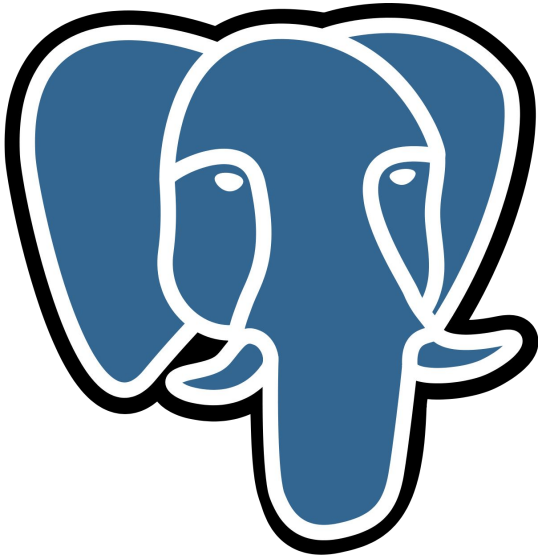
# Introduction to PostgreSQL

# Select Condition (Case When)

# Aggregation (Group By)

# Introduction to PostgreSQL

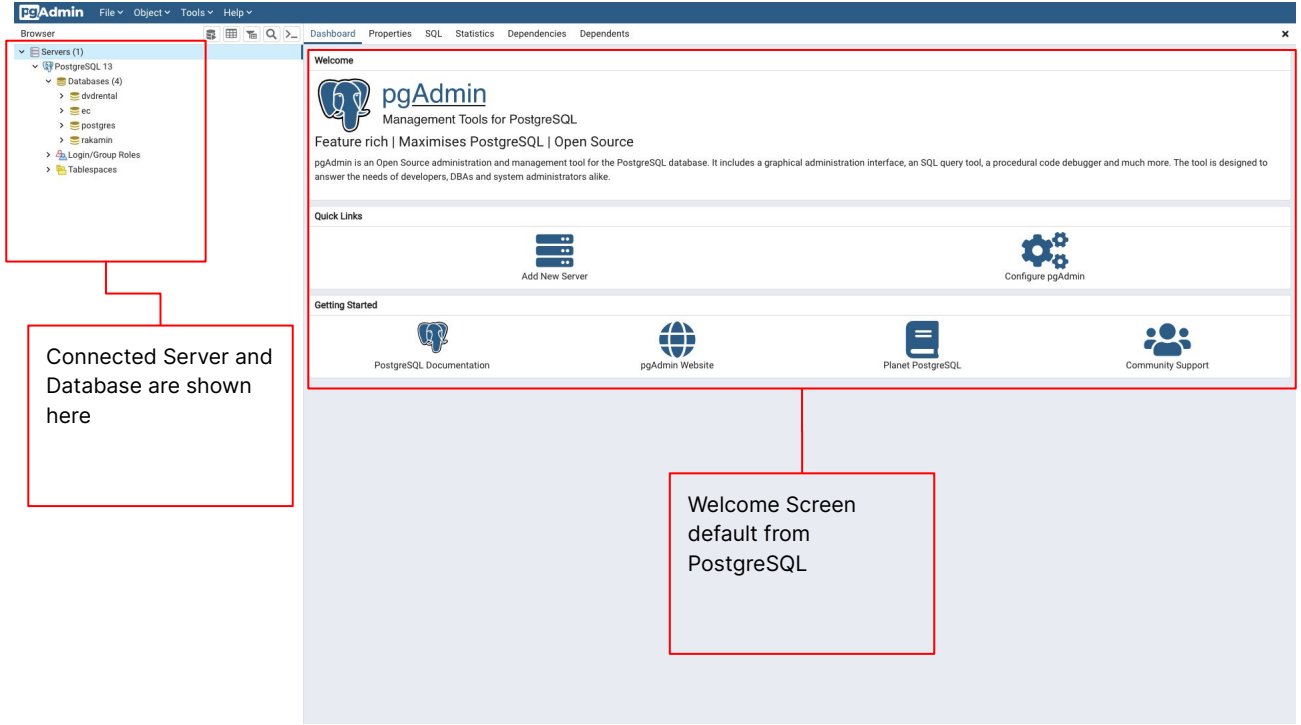
# Introduction to PostgreSQL



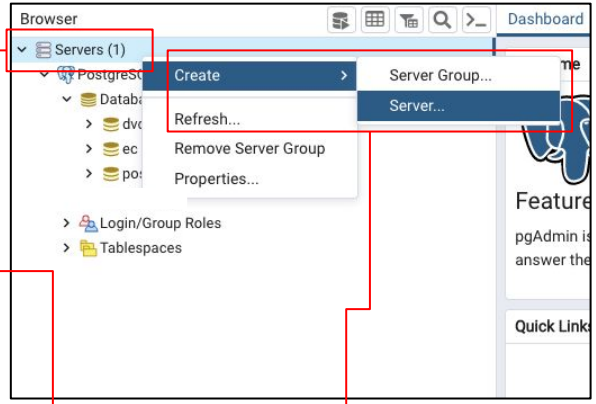
**PostgreSQL** is one form of **Relational Database Management System (RDBMS)** that is popular and used by enterprises and businesses

**PostgreSQL** is *open source.*

# pgAdmin interface

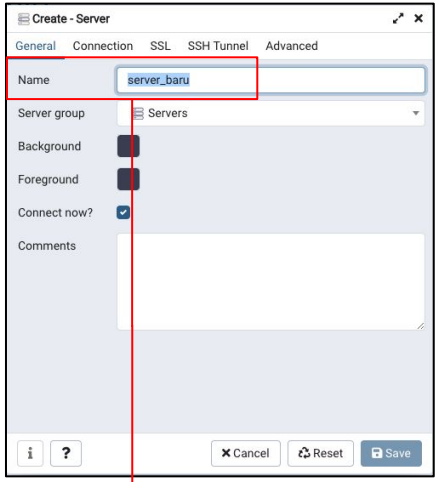


# pgAdmin interface

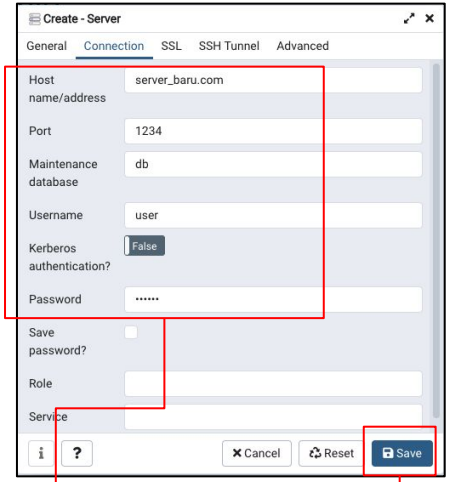



**1.** Right Click on Servers

**2.** Choose Create > Server




**3.** Input desired server name (custom) on General tab




**4.** In Connection tab, insert the credentials

**5.** Click Save

# DVD Rental Database

https://www.postgresqltutorial.com/postgresql-getting-started/postgresql-sample-database/

# Select Condition (Case When)

# Case Statement

The CASE statement goes through conditions and returns a value when the first condition is met (like an if-then-else statement).

So, once a condition is true, it will stop reading and return the result. If no conditions are true, it returns the value in the ELSE clause.

If there is no ELSE part and no conditions are true, it returns NULL.

# Case Statement

```
CASE
    WHEN condition1 THEN result1
    WHEN condition2 THEN result2
    WHEN conditionN THEN resultN
    ELSE result
END;
```

SQL 1

```
SELECT
    CASE
        WHEN Total > 5 THEN "High"
        WHEN Total > 3 AND Total <= 5 THEN "Middle"
        ELSE "Low"
    END
FROM invoices
```

| | CASE WHEN Total > 5 THEN "High" WHEN Total > 3 AND Total <= 5 THEN "Middle" ELSE "Low" END |
|---|---|
| 1 | Low |
| 2 | Middle |
| 3 | High |

# Case Statement

## Using Alias

SQL 1

```
SELECT
    CASE
        WHEN Total > 5 THEN "High"
        WHEN Total > 3 AND Total <= 5 THEN "Middle"
        ELSE "Low"
    END AS categorization
FROM invoices
```

| | categorization |
|---|---|
| 1 | Low |
| 2 | Middle |
| 3 | High |
| 4 | High |

# Case Statement

## Case Statement on Order by Clause

SQL 1

```
SELECT CustomerId, FirstName, LastName
FROM customers
ORDER BY
    (CASE
        WHEN LastName IS NOT NULL THEN LastName
        ELSE FirstName
        END
    )
```

| | CustomerId | FirstName | LastName |
|---|---|---|---|
| 1 | 12 | Roberto | Almeida |
| 2 | 28 | Julia | Barnett |
| 3 | 39 | Camille | Bernard |
| 4 | 18 | Michelle | Brooks |

# Aggregation (Group By)

# Aggregation

SQL aggregation is the task of collecting a set of values to return a single value. It is done with the help of aggregate functions, such as SUM, COUNT, and AVG. For example, in a database of products, you might want to calculate the average price of the whole inventory.

# Aggregation

| title | genre | qty |
|---|---|---|
| book 1 | adventure | 4 |
| book 2 | fantasy | 5 |
| book 3 | romance | 2 |
| book 4 | adventure | 3 |
| book 5 | fantasy | 3 |
| book 6 | romance | 1 |

| genre | total |
|---|---|
| adventure | 7 |
| fantasy | 8 |
| romance | 3 |

# Aggregation

An aggregate function in SQL returns one value after calculating multiple values of a column. We often use aggregate functions with the GROUP BY and HAVING clauses of the SELECT statement.

Various types of SQL aggregate functions are:

- Count()
- Sum()
- Avg()
- Min()
- Max()

# Syntax for Aggregation and Grouping

*The syntax for GROUP BY clause is:*

```
GROUP BY ColumnName1, ColumnName2;
```

Here, ColumnName is the name(s) of the column(s) you want to apply the Group By Clause to.

*The syntax for aggregation in SQL is:*

```
AggregateFunctionName(DISTINCT or ALL GroupName)
```

**Distinct:**

**Eliminating Duplicated Rows**

**Eg:**

| City |
|---|
| New York |
| New York |
| Los Angeles |

**Count: 3**

**Count Distinct: 2**

# COUNT

- **COUNT(*)**

  Counting all rows in the table

- **COUNT(DISTINCT|ALL Groupname)**

  Counting distinct rows/all rows in specific column/expression that we want to aggregate

# COUNT

SQL 1

```
1  SELECT COUNT(*)
2  FROM customers
```

| | COUNT(*) |
|---|---|
| 1 | 59 |

SQL 1

```
1  SELECT COUNT(DISTINCT FirstName)
2  FROM customers
```

| | COUNT(DISTINCT FirstName) |
|---|---|
| 1 | 57 |

# COUNT with GROUP BY

**Group by is used to aggregate values based on certain categories/for each category**

SQL 1

```
SELECT City, COUNT(CustomerId)
FROM customers
GROUP BY City
```

| | City | COUNT(CustomerId) |
|---|---|---|
| 1 | Amsterdam | 1 |
| 2 | Bangalore | 1 |
| 3 | Berlin | 2 |

# COUNT with GROUP BY & HAVING

**Having is for conditional clause in grouping statement**

SQL 1

```
SELECT City, COUNT(CustomerId)
FROM customers
GROUP BY City
HAVING COUNT(CustomerId) >=2
```

| | City | COUNT(CustomerId) |
|---|---|---|
| 1 | Berlin | 2 |
| 2 | London | 2 |
| 3 | Mountain View | 2 |
| 4 | Paris | 2 |
| 5 | Prague | 2 |

# SUM

**The SUM() function returns the total sum of a numeric column.**

SQL 1

```
1  SELECT SUM(Total)
2  FROM invoices
```

| | SUM(Total) |
|---|---|
| 1 | 2328.6 |

SQL 1

```
1  SELECT SUM(DISTINCT Total)
2  FROM invoices
```

| | SUM(DISTINCT Total) |
|---|---|
| 1 | 257.17 |

# SUM with GROUP BY

SQL 1

```
SELECT BillingCity, SUM(DISTINCT Total)
FROM invoices
GROUP BY BillingCity
```

| | BillingCity | SUM(DISTINCT Total) |
|---|---|---|
| 1 | Amsterdam | 38.64 |
| 2 | Bangalore | 36.64 |
| 3 | Berlin | 35.64 |
| 4 | Bordeaux | 39.62 |
| 5 | Boston | 35.64 |

# SUM in HAVING Clause

SQL 1

```
SELECT BillingCity, SUM(Total)
FROM invoices
GROUP BY BillingCity
HAVING SUM(Total) > 50
```

| | BillingCity | SUM(Total) |
|---|---|---|
| 1 | Berlin | 75.24 |
| 2 | London | 75.24 |
| 3 | Mountain View | 77.24 |
| 4 | Paris | 77.24 |
| 5 | Prague | 90.24 |
| 6 | São Paulo | 75.24 |

SQL 1

```
SELECT BillingCity, COUNT(DISTINCT CustomerId)
FROM invoices
GROUP BY BillingCity
HAVING SUM(Total) > 50
```

| | BillingCity | COUNT(DISTINCT CustomerId) |
|---|---|---|
| 1 | Berlin | 2 |
| 2 | London | 2 |
| 3 | Mountain View | 2 |
| 4 | Paris | 2 |
| 5 | Prague | 2 |
| 6 | São Paulo | 2 |

# AVG

**The AVG() function calculates the average of a set of numeric values**

SQL 1

```
1   SELECT AVG(Total)
2   FROM invoices
3
```

| | AVG(Total) |
|---|---|
| 1 | 5.65194174757283 |

# AVG with Group By & Having

SQL 1

```sql
SELECT BillingCity, AVG(Total)
FROM invoices
GROUP BY BillingCity
HAVING SUM(Total) < 40
```

| | BillingCity | AVG(Total) |
|---|---|---|
| 1 | Bangalore | 6.10666666666667 |
| 2 | Bordeaux | 5.66 |
| 3 | Boston | 5.37428571428571 |
| 4 | Brasília | 5.37428571428571 |
| 5 | Brussels | 5.37428571428571 |

# MIN and MAX

**The MIN() & MAX() aggregate function returns the lowest value (minimum) and the highest value (maximum) of numerical data**

SQL 1

```
SELECT MIN(Total)
FROM invoices
```

| | MIN(Total) |
|---|---|
| 1 | 0.99 |

SQL 1

```
SELECT MAX(Total)
FROM invoices
```

| | MAX(Total) |
|---|---|
| 1 | 25.86 |

# MIN and MAX

**For non numeric columns, it will work by alphabetically.**

SQL 1

```
1   SELECT MIN(City)
2   FROM customers
3
```

| | MIN(City) |
|---|---|
| 1 | Amsterdam |

SQL 1

```
1   SELECT MAX(City)
2   FROM customers
3
```

| | MAX(City) |
|---|---|
| 1 | Yellowknife |

# MIN and MAX with Group By & Having

SQL 1

```sql
SELECT BillingCity, MAX(Total)
FROM invoices
GROUP BY BillingCity
HAVING MIN(Total) > 1
```

| | BillingCity | MAX(Total) |
|---|---|---|
| 1 | Bangalore | 13.86 |
| 2 | Cupertino | 13.86 |
| 3 | Delhi | 13.86 |

# HAVING vs WHERE

**HAVING is like a where clause for the group by (applies to the groupings)**

**We can still have a WHERE clause before the GROUP BY clause, but that only applies to the individual rows before the grouping.**

SQL 1

```
SELECT BillingCity, MAX(Total)
FROM invoices
WHERE BillingCity = "Delhi"
GROUP BY BillingCity
HAVING MIN(Total) > 1
```

| | BillingCity | MAX(Total) |
|---|---|---|
| 1 | Delhi | 13.86 |

# Mini Practice

**Questions :**

1. Display the total amount paid by all customers in the payment table.
2. Display the total amount paid by each customer in the payment table.
3. What is the highest total_payment done.
4. Show the number of movies each actor acted in.
5. How many actors have 8 letters only in their first_names.
6. For each store, display the number of customers that are members of that store.
7. Display the movie title for the most rented movie in the store with store_id 1
8. Count the number of actors who's first_names don't start with an 'A'.

# Thank you!

## Any Questions?