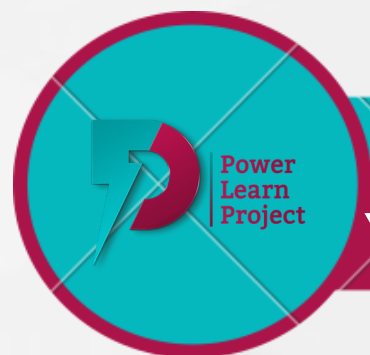


# Introduction to Grouping

## Data

### week 12

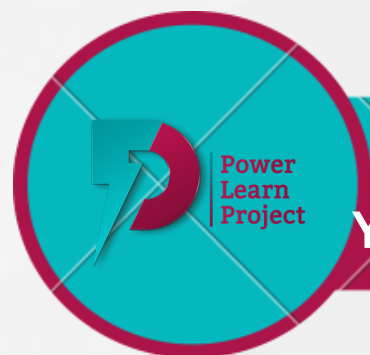


Database Design

You are live with Stanley Munga

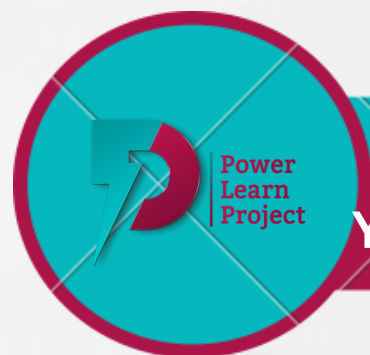
# Learning Objectives

- Understand the concept of grouping data in a database.
- Learn the basic syntax of the GROUP BY clause.
- Explore the role of aggregate functions with grouped data (COUNT, SUM, AVG, MIN, MAX).
- Use the HAVING clause for filtering grouped data.
- Comprehend the difference between WHERE and HAVING in the context of grouped data.
- Apply grouping to single and multiple columns.
- Learn about the importance of column aliases when using grouping.
- Explore the concept of rollup and cube for hierarchical grouping.



# I. Introduction to Grouping Data

In SQL, grouping data is a powerful way to aggregate and analyze information based on specific criteria. The GROUP BY clause, along with HAVING, GROUPING SETS, ROLLUP, and CUBE, provides flexibility in organizing and summarizing data.



Database Design

You are live with Stanley Munga

## II. GROUP BY Clause

### 1. Basic Syntax:

```
SELECT column1, aggregate_function(column2), ...
```

```
FROM table_name
```

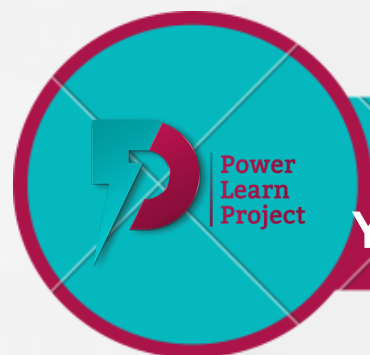
```
GROUP BY column1;
```

### 2. Example:

```
SELECT department_id, AVG(salary) AS avg_salary
```

```
FROM employees
```

```
GROUP BY department_id;
```



# III. HAVING Clause

## III. HAVING Clause

### 1. Purpose:

Filters grouped results based on aggregate conditions.

Used with the GROUP BY clause.

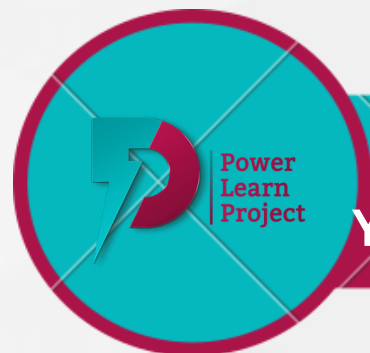
### 2. Syntax:

```
SELECT column1, aggregate_function(column2), ...
```

```
FROM table_name
```

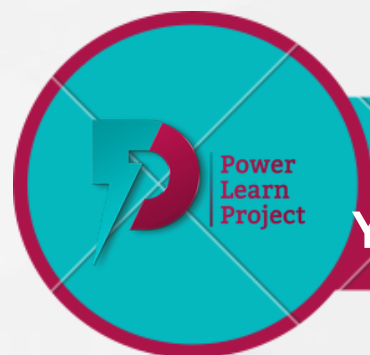
```
GROUP BY column1
```

```
HAVING condition;
```



3. Example:

```
SELECT department_id, AVG(salary) AS avg_salary  
FROM employees  
GROUP BY department_id  
HAVING AVG(salary) > 50000;
```



Database Design

You are live with Stanley Munga

# V. Conclusion

## 1. Purpose:

Groups data based on different combinations of columns.

## 2. Syntax:

```
SELECT column1, column2, aggregate_function(column3), ...
```

```
FROM table_name
```

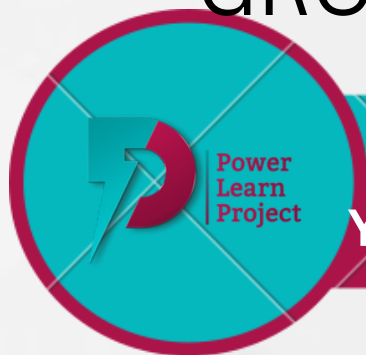
```
GROUP BY GROUPING SETS ((column1), (column2), ());
```

## 3. Example:

```
SELECT department_id, job_id, AVG(salary) AS avg_salary
```

```
FROM employees
```

```
GROUP BY GROUPING SETS ((department_id, job_id), (department_id), ());
```





# V. ROLLUP

## 1. Purpose:

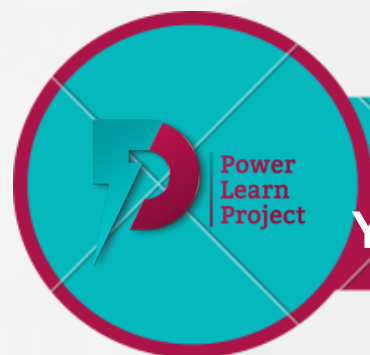
Generates subtotals and grand totals for a set of columns.

## 2. Syntax:

```
SELECT column1, column2, aggregate_function(column3), ...
```

```
FROM table_name
```

```
GROUP BY ROLLUP (column1, column2);
```





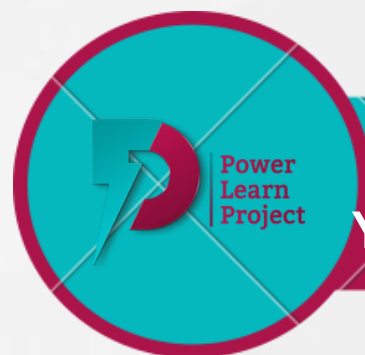
3. Example:

```
SELECT department_id, job_id, AVG(salary) AS avg_salary  
FROM employees  
GROUP BY ROLLUP (department_id, job_id);
```

## VI. CUBE

1. Purpose:

Generates subtotals and grand totals for all possible combinations of columns.

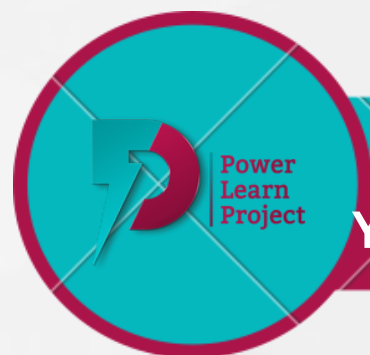


## 2. Syntax:

```
SELECT column1, column2, aggregate_function(column3), ...  
FROM table_name  
GROUP BY CUBE (column1, column2);
```

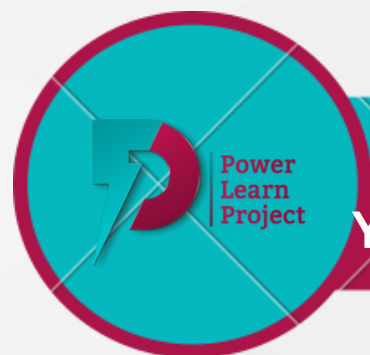
## 3. Example:

```
SELECT department_id, job_id, AVG(salary) AS avg_salary  
FROM employees  
GROUP BY CUBE (department_id, job_id);
```



# VII. Conclusion

Understanding how to group data using GROUP BY, HAVING, GROUPING SETS, ROLLUP, and CUBE provides a comprehensive toolkit for performing complex analyses in SQL. These clauses empower you to organize and summarize data in various ways, facilitating the extraction of valuable insights from your databases.



Database Design

You are live with Stanley Munga