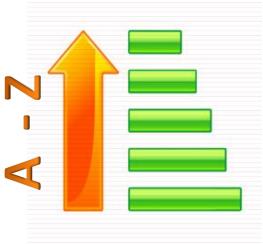
SELECT OPTIONS

Learning Goals

By the end of this lecture students should be able to:

- Understand and use SQL functions
- Use Group, Having, Order clauses to built queries
- Copy data from one table into another, combine the result-set of two or more SELECT statements



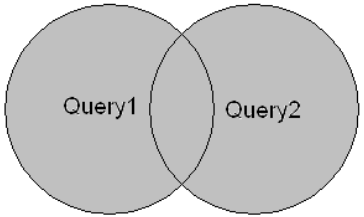


Table of contents

Section1

SQL CLAUSES

Grouping by clause

- Sometimes we want to apply aggregate functions to groups of rows.

Syntax:

```
SELECT column_name, aggregate_function(column_name)
FROM table_name
WHERE column_name operator value
GROUP BY column_name;
```

- Example, find the average mark of each student.

Group

Id	Name	SubjectID	Mark
1	John	DBS	76
2	John	IAI	72
3	Mary	DBS	60
4	Mand	PR1	63
5	Mand	PR2	35
6	Jane	IAI	54

Grades

```
SELECT Name,
AVG(Mark) AS Average
FROM Grades
GROUP BY Name
```

Name	Average
John	74
Mary	60
Mand	49
Jane	54

Having clause

- **HAVING** is like a **WHERE** clause, except that it applies to the results of a **GROUP BY** query.
- It can be used to select groups which satisfy a given condition.
- **Ex**:

```
SELECT Name, AVG(Mark) AS Average
FROM Grades
GROUP BY Name
HAVING AVG(Mark) >= 50
```

Id	Name	SubjectID	Mark
1	John	DBS	76
2	John	IAI	72
3	Mary	DBS	60
4	Mand	PR1	63
5	Mand	PR2	35
6	Jane	IAI	54

Name	Average
John	74
Mary	60
Jane	54

WHERE and HAVING

- **`WHERE`** refers to the rows of tables, and so cannot use aggregate functions
- **`HAVING`** refers to the groups of rows, can use aggregate functions and cannot use columns which are not in the `GROUP BY`

```
SELECT Name,
AVG(Mark) AS Average
FROM Grades
WHERE AVG(Mark) >= 50
GROUP BY Name
```

```
SELECT Name,
AVG(Mark) AS Average
FROM Grades
GROUP BY Name
HAVING AVG(Mark) >= 50
```

Order by clause

- The SQL **ORDER BY clause** is used to sort (ascending or descending) the records in the result set for a SELECT statement.

Syntax:

```
SELECT column_name, column_name
FROM table_name
[WHERE conditions]
ORDER BY column_name, column_name [ASC|DESC]
```

- **Ex:**

Group

Id	Name	SubjectID	Mark
1	John	DBS	76
2	John	IAI	72
3	Mary	DBS	60
4	Mand	PR1	63
5	Mand	PR2	35
6	Jane	IAI	54

```
SELECT Name,
AVG(Mark) AS Average
FROM Grades
GROUP BY Name
ORDER BY Average DESC
```

Grades

Name	Average
John	74
Mary	60
Jane	54
Mand	49

Section2

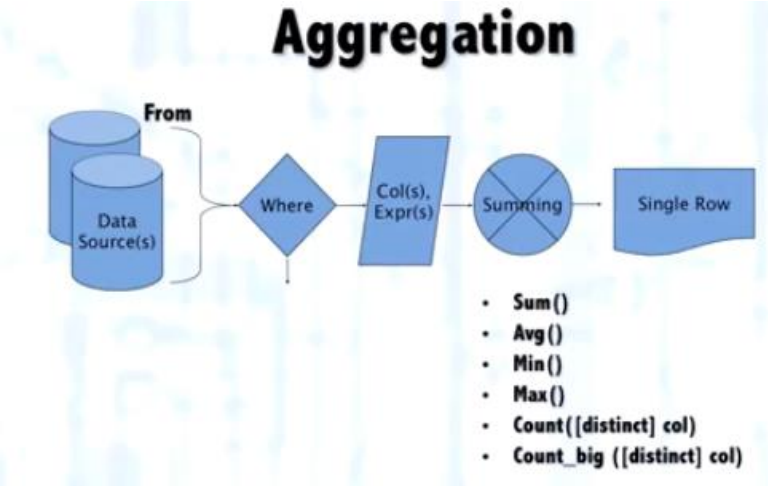
SQL FUNCTIONS

SQL Functions

- SQL has many built-in functions for performing calculations on data:
  - ✓ SQL aggregate functions return a single value, calculated from values in a column.
  - ✓ SQL scalar functions return a single value, based on the input value.

What is an aggregate function

- An aggregate function is function that take a collection of values as input and return a single value.
- Aggregate functions can be used as expressions only in the following:
  - ✓ The select list of a SELECT statement
  - ✓ A HAVING clause.



Aggregate Functions

- Each function eliminates NULL values and operates on Non-NULL values

Function	Description
AVG ()	Return the average value in a column
COUNT()	Return the total number of values in a given column
COUNT(*)	Return the number of rows
MIN ()	Returns the smallest value in a column
MAX ()	Returns the largest value in a column
SUM()	Returns the sum values in a column

Scalar functions

Function	Description
LEN()	Returns the length of a text field
ROUND()	Rounds a numeric field to the number of decimals specified
NOW()	Returns the current system date and time
FORMAT()	Formats how a field is to be displayed

Section3

OTHER OPTIONS

UNION Operator

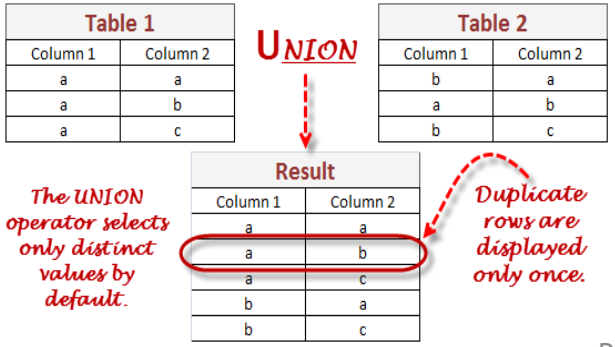
- The SQL UNION operator combines the result of two or more SELECT statements.

Syntax:

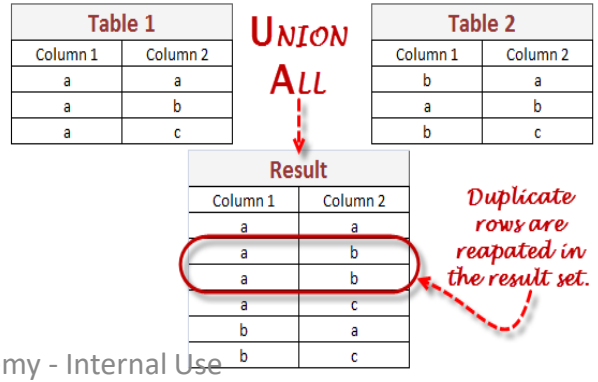
```
SELECT column_name(s) FROM table1
UNION
SELECT column_name(s) FROM table2;
```

Note: The UNION operator selects only distinct values by default. To allow duplicate values, use the ALL keyword with UNION.

```
SELECT Column1, Column2 FROM Table1
UNION
SELECT Column1, Column2 FROM
Table2;
```




```
SELECT Column1, Column2 FROM Table1
UNION ALL
SELECT Column1, Column2 FROM
Table2;
```



SELECT INTO Statement

- With SQL, you can copy information from one table into another.
- The SELECT INTO statement selects data from one table and inserts it into a new table.

Syntax:

(1): copy all columns into the new table:

```
SELECT *
INTO newtable [IN externaldb]
FROM table1;
```

(2): copy only the columns we want into the new table:

```
SELECT column_name(s)
INTO newtable [IN externaldb]
FROM table1;
```

INSERT INTO SELECT Statement

- The **INSERT INTO SELECT** statement selects data from one table and inserts it into an existing table.
- Any existing rows in the target table are unaffected.
- **Syntax:**

✓ *Copy all columns from one table to another, existing table:*

```
INSERT INTO table2
SELECT * FROM table1;
```

✓ *Copy only the columns we want to into another, existing table:*

```
INSERT INTO table2(column_name(s))
SELECT column_name(s)
FROM table1;
```

Summary

- SQL Clauses
  - Group by, Having, Order by
- SQL Functions
  - Aggregate, scalar functions
- Other Options
  - UNION Operator, SQL SELECT INTO, INSERT INTO SELECT
- Demo