Intro To Database

(Database Fundamental using MySQL)

Mohamed ELshafei

SELECT with Condition

```
Select  dept_id, dept_name
from    department
where   location = ‘Cairo’;
```

Comparison Conditions

- = Equal.
- > greater than.
- >= greater than or equal.
- < less than.
- <= less than or equal.
- <>not equal.

```
Select   last_name, salary
from     employee
where    salary >1000
```

Logical Conditions

- AND.

```
Select   last_name, salary
from     employee
where  city = ‘Assiut’ and salary > 1000;
```

- OR.

```
Select   last_name, salary
from     employee
where  city = ‘Assiut’ OR salary > 1000;
```

- NOT.

```
Select   emp_id, last_name, salary, manager_id
From     employee
where   manager_id NOT IN (100, 101, 200);
```

Other Comparison Conditions

- Between …… AND ….. (between two values - Inclusive).

```
Select   last_name, salary
from     employee
where  salary between 1000 and 3000;
```

- IN (set) (Match any of a list of values)

```
Select    emp_id, last_name, salary, manager_id
From      employee
where     manager_id IN (100, 101, 200);
```

- Like (Match a character Pattern)

```
Select   first_name
from     employee
where    first_name Like ‘s%’;
```

Arithmetic Expressions

```
Select   last_name, salary, salary + 300
from     employee;
```

- Order of precedence: * , / , +, -
- You can enforce priority by adding parentheses.

```
Select    last_name, salary, 10 * (salary + 300)
from      employee;
```

Order by Clause

- It is used to sort results either in ascending or descending order.

✓
```
Select      fname, dept_id, hire_date
From        employee
Order by    hire_date  [ ASC ];
```

✓
```
Select      fname, dept_id, hire_date
From        employee
Order by    hire_date  DESC;
```

✓
```
Select      fname, dept_id, salary
From        employee
Order by    dept_id, Salary  DESC;
```

IN Operator

- The IN operator allows you to specify multiple values in a WHERE clause.
- The IN operator is a shorthand for multiple OR conditions.

```
SELECT column_name(s)
FROM table_name
WHERE column_name IN (value1, value2, ...);
```

- SELECT * FROM Customers

  WHERE Country NOT IN ('Germany', 'France', 'UK');

BETWEEN Operator

- The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates.

```
SELECT column_name(s)
FROM table_name
WHERE column_name BETWEEN value1 AND value2;
```

- SELECT * FROM Products

  WHERE Price BETWEEN 10 AND 20;

Aliases

- SQL aliases are used to give a table, or a column in a table, a temporary name.
- Aliases are often used to make column names more readable.

```
SELECT column_name AS alias_name
FROM table_name;
```

- SELECT CustomerID AS ID, CustomerName AS Customer

  FROM Customers;

DISTINCT

- The SELECT DISTINCT statement is used to return only distinct (different) values.

```
SELECT DISTINCT column1, column2, ...
FROM table_name;
```

SELECT DISTINCT Country FROM Customers;

NULL Value

- A field with a NULL value is a field with no value.
- A NULL value is different from a zero value or a field that contains spaces. A field with a NULL value is one that has been left blank during record creation!

- SELECT columnNames FROM tableName WHERE columnName IS NULL;
- SELECT columnNames FROM tableName WHERE columnName IS NOT NULL;

NULL Functions

- The MySQL IFNULL() function lets you return an alternative value if an expression is NULL:

```
SELECT ID , IFNULL (First_Name , ‘  ‘) from student ;
```

- SELECT ProductName, UnitPrice * (UnitsInStock + IFNULL(UnitsOnOrder, 0))
  FROM Products

LIKE

- The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

```
SELECT column1, column2, ... FROM table_name
WHERE columnN LIKE pattern;
```

WHERE CustomerName LIKE 'a%'	Finds any values that start with "a"
WHERE CustomerName LIKE '%a'	Finds any values that end with "a"
WHERE CustomerName LIKE '%or%'	Finds any values that have "or" in any position
WHERE CustomerName LIKE '_r%'	Finds any values that have "r" in the second position
WHERE CustomerName LIKE 'a_%_%'	Finds any values that start with "a" and are at least 3 characters in length
WHERE ContactName LIKE 'a%o'	Finds any values that start with "a" and ends with "o"

INSERT INTO SELECT

- The INSERT INTO SELECT statement copies data from one table and inserts it into another table.

```
INSERT INTO table2 (column1, column2, column3, ...)
SELECT column1, column2, column3, ...
FROM table1
WHERE condition;
```

INSERT INTO Customers (CustomerName, City, Country)
SELECT SupplierName, City, Country FROM Suppliers;

Comments

- Comments are used to explain sections of SQL statements, or to prevent execution of SQL statements.
- **Single Line Comments**

  Single line comments start with --.
- **Multi-line Comments**

  Multi-line comments start with /* and end with */.

TOP(LIMIT)

- The SELECT TOP clause is used to specify the number of records to return.

```
SELECT column_name(s)
FROM table_name
WHERE condition
LIMIT number;
```

- SELECT * FROM Customers LIMIT 3;

Aggregate Function

- MIN(),MAX(),COUNT(), AVG() and SUM()

```
SELECT MIN(Price) AS SmallestPrice FROM Products;
```

- The COUNT() function returns the number of rows that matches a specified criteria.
- The AVG() function returns the average value of a numeric column.
- The SUM() function returns the total sum of a numeric column.

GROUP BY

- The GROUP BY statement is often used with aggregate functions (COUNT, MAX, MIN, SUM, AVG) to group the result-set by one or more columns.

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
ORDER BY column_name(s)
```

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country;
```

HAVING

- The HAVING clause was added to SQL because the WHERE keyword could not be used with aggregate functions.

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition
ORDER BY column_name(s);
```

```
SELECT COUNT(CustomerID), Country  FROM Customers
GROUP BY Country
HAVING COUNT(CustomerID) > 5;
```

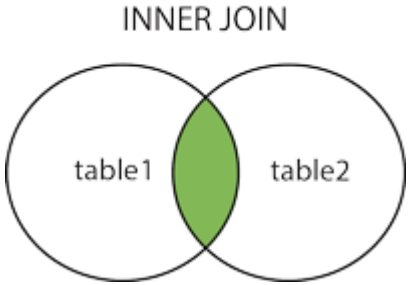
JOIN

- A JOIN clause is used to combine rows from two or more tables, based on a related column between them.
- **Different Types of SQL JOINs:**
  - CROSS JOIN
  - INNER JOIN
  - OUTER JOIN
  - SELF JOIN

INNER JOIN

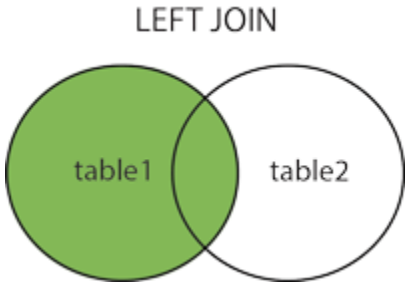
- The INNER JOIN keyword selects records that have matching values in both tables.

```
SELECT column_name(s)
FROM table1
INNER JOIN table2 ON table1.column_name = table2.column_name;
```

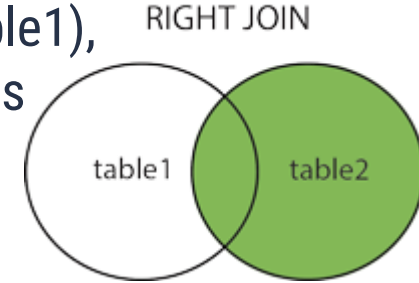


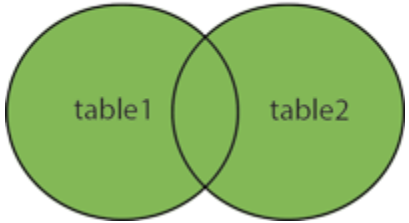

```
SELECT Orders.OrderID, Customers.CustomerName
FROM Orders
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;
```

OUTER JOIN



- The LEFT JOIN keyword returns all records from the left table (table1), and the matched records from the right table (table2). The result is NULL from the right side, if there is no match.






```
SELECT column_name(s)
FROM table1
LEFT JOIN table2 ON table1.column_name = table2.column_name;
```

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID
```

Self JOIN

- A self JOIN is a regular join, but the table is joined with itself.

```
SELECT column_name(s)
FROM table1 T1, table1 T2
WHERE condition;
```

```
SELECT A.Name AS CustomerName1, B.Name AS CustomerName2, A.City
FROM Customers A, Customers B
WHERE A.CustomerID <> B.CustomerID
AND A.City = B.City
```

UNION Operator

- The UNION operator is used to combine the result-set of two or more SELECT statements.
  - Each SELECT statement within UNION must have the same number of columns
  - The columns must also have similar data types
  - The columns in each SELECT statement must also be in the same order

```
SELECT City FROM Customers
UNION
SELECT City FROM Suppliers
```

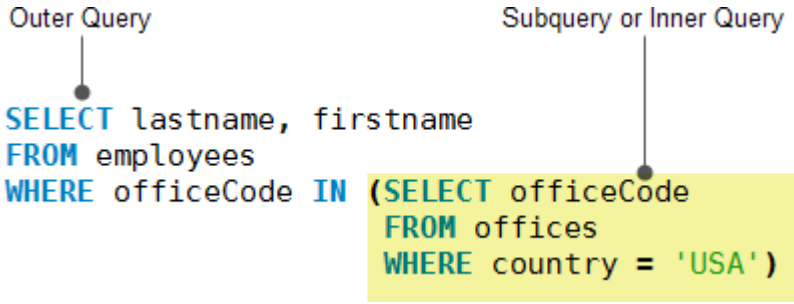
UNION ALL Operator

- The UNION operator selects only distinct values by default. To allow duplicate values, use UNION ALL

```
SELECT City FROM Customers
UNION ALL
SELECT City FROM Suppliers
```

Subquery

- A MySQL subquery is a query nested within another query such as SELECT, INSERT, UPDATE or DELETE. In addition, a MySQL subquery can be nested inside another subquery.



EXISTS Operator

- The EXISTS operator is used to test for the existence of any record in a subquery.
- The EXISTS operator returns true if the subquery returns one or more records.

```
SELECT column_name(s)
FROM table_name
WHERE EXISTS
(SELECT column_name FROM table_name WHERE condition);
```

```
SELECT SupplierName  FROM Suppliers
WHERE EXISTS (SELECT ProductName FROM Products WHERE Price < 20);
```

Order Of Execution

1. FROM clause
2. WHERE clause
3. GROUP BY clause
4. HAVING clause
5. SELECT clause
6. DISTINCT clause
7. ORDER BY clause
8. TOP clause

THANKS!

Any questions?