CREATE DATABASE database_Name;

SHOW databases;

USE database_Name;

CREATE TABLE table_Name(Value1 data_type,Value2 data_type, . . .);

SHOW TABLES;

INSERT INTO *table_name* (*column1, column2, column3, ...*)
VALUES (*value1(column1), value2(column2), value3(column3), ...*);

DESCRIBE table_name;

ALTER TABLE table_name **Change** old_column_name (Space) new_column_name (Space) Data _type;

SELECT *column1,column2 ...*
FROM *table_name*
WHERE *column_name* **IS NULL;**

SELECT *column1,column2 ...*
FROM *table_name*
WHERE *column_name* **IS NOT NULL;**

```
SELECT LastName, FirstName, Address FROM Persons
WHERE Address IS NOT NULL;
```

The result-set will look like this:

LastName	FirstName	Address
Doe	John	542 W. 27th Street
Smith	John	110 Bishopsgate

UPDATE *table_name*
SET *column1 = value1, column2 = value2, ...*
WHERE *condition*;

```
UPDATE Customers
SET ContactName='Juan'
WHERE Country='Mexico';
```

Try it Yourself »

e selection from the "Customers" table will now look like this:

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Alfred Schmidt	Obere Str. 57	Frankfurt	12209	Germany
2	Ana Trujillo Emparedados y hel[illegible]s	Juan	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Juan	Mataderos 2312	México D.F.	05023	Mexico

DELETE FROM *table_name*
WHERE *condition*;

DELETE * **FROM** *table_name*;

Example

```
DELETE FROM Customers
WHERE CustomerName='Alfreds Futterkiste';
```

SELECT column1, column2, ...

FROM table_Name;

SELECT DISTINCT column1, column2, ...

FROM table_Name;

SELECT COUNT(DISTINCT column) **FROM** table_Name;

SELECT column1, column2, ...

FROM table_Name

WHERE condition;

Operators in The WHERE Clause

The following operators can be used in the WHERE clause:

Operator	Description
=	Equal
<>	Not equal. **Note:** In some versions of SQL this operator may be written as !=
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
BETWEEN	Between an inclusive range
LIKE	Search for a pattern
IN	To specify multiple possible values for a column

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 AND condition2 AND condition3 ...;
```

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 OR condition2 OR condition3 ...;
```

```
SELECT column1, column2, ...
FROM table_name
WHERE NOT condition;
```

Example

```
SELECT * FROM Customers
WHERE Country='Germany' AND (City='Berlin' OR City='München');
```

Example

```
SELECT * FROM Customers
WHERE NOT Country='Germany' AND NOT Country='USA';
```

```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1, column2, ... ASC(Sorted)|DESC(Dis Sorted);
```

Example

```
SELECT * FROM Customers
ORDER BY Country ASC, CustomerName DESC;
```

```
SELECT TOP number|percent column_name(s)
FROM table_name
WHERE condition;
```

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

```
SELECT column_name(s)
FROM table_name
WHERE ROWNUM <= number;
```

Example

```
SELECT * FROM Customers
WHERE Country='Germany'
LIMIT 3;
```

Try it Yourself »

he following SQL statement shows the equivalent exan

Example

```
SELECT * FROM Customers
WHERE Country='Germany' AND ROWNUM <= 3;
```

```
SELECT MAX(Price) AS Big_Value(Use AS column_name Show)
FROM Products;
```

```
SELECT MIN(Price) AS Small_Value(Use AS than column_name Show)
FROM Products;
```

MIN() Syntax

```
SELECT MIN(column_name)
FROM table_name
WHERE condition;
```

MAX() Syntax

```
SELECT MAX(column_name)
FROM table_name
WHERE condition;
```

SELECT COUNT(ProductID)
FROM Products;(How many value have are this column show)

SELECT AVG(Price)
FROM Products;(All value average number are this column show)

SELECT SUM(Quantity)
FROM OrderDetails;(Add all value are this column show)

SELECT *column1, column2, ...(another column name just show and just firstone are apply pattern)*
FROM *table_name*
WHERE *columnN* **LIKE** *pattern*;

LIKE Operator	Description
WHERE CustomerName LIKE 'a%'	Finds any values that starts with "a"
WHERE CustomerName LIKE '%a'	Finds any values that ends 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 starts with "a" and are at least 3 characters in length
WHERE ContactName LIKE 'a%o'	Finds any values that starts with "a" and ends with "o"

Using the [charlist] Wildcard

The following SQL statement selects all customers with a City starting with "b", "s", or "p":

Example

```
SELECT * FROM Customers
WHERE City LIKE '[bsp]%';
```

Try it Yourself »

The following SQL statement selects all customers with a City starting with "a", "b", or "c":

Example

```
SELECT * FROM Customers
WHERE City LIKE '[a-c]%';
```

Example

```
SELECT * FROM Customers
WHERE CustomerName LIKE 'a%';
```

SELECT *column_name(s)/*from*
FROM *table_name*
WHERE *column_name* **IN** (*'value1', 'value2', ...*);

SELECT *column_name(s)/*from*
FROM *table_name*
WHERE *column_name* **BETWEEN** *value1* **AND** *value2;*

Example

```
SELECT * FROM Products
WHERE (Price BETWEEN 10 AND 20)
AND NOT CategoryID IN (1,2,3);
```

Example

```
SELECT * FROM Products
WHERE ProductName BETWEEN 'Carnarvon Tigers' AND 'Mozzarella di Giovanni'
ORDER BY ProductName;
```

SELECT *column_name* **AS** *alias_name(alias_name mean show new column name)*
FROM *table_name;*

SELECT *column_name(s)*
FROM *table_name* **AS** *alias_name;(alias_name mean show new column name)*

Example

```
SELECT CustomerName, Address + ', ' + PostalCode + ' ' + City + ', ' + Country AS Address
FROM Customers;
```

Try it Yourself »

1.SQL aliases are used to give a table, or a column in a table, a temporary name.

2.Aliases are often used to make column names more readable.

3.An alias only exists for the duration of the query.

Example

```
SELECT Orders.OrderID, Orders.OrderDate, Customers.CustomerName
FROM Customers, Orders
WHERE Customers.CustomerName="Around the Horn" AND Customers.CustomerID=Orders.CustomerID;
```

JOIN:-

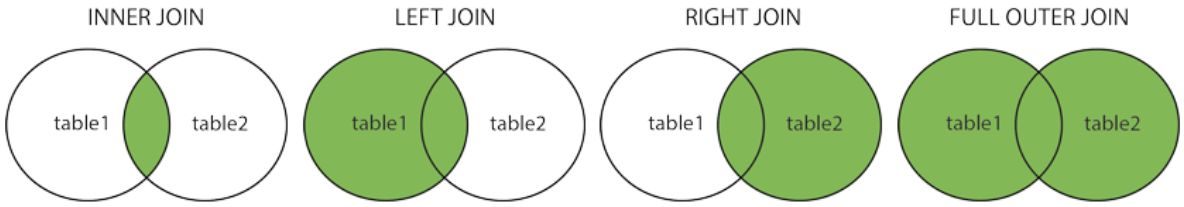
Example

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

Different Types of SQL JOINs

Here are the different types of the JOINs in SQL:

- **(INNER) JOIN**: Returns records that have matching values in both tables
- **LEFT (OUTER) JOIN**: Return all records from the left table, and the matched records from the right table
- **RIGHT (OUTER) JOIN**: Return all records from the right table, and the matched records from the left table
- **FULL (OUTER) JOIN**: Return all records when there is a match in either left or right table



SELECT *column_name(s)/(table1.showable column_name,...)*
FROM *table1*
INNER JOIN *table2* **ON** *table1.column_name = table2.column_name*;

Three tables value add (onetime by table1(orders)) than use this syntax:-

Example

```
SELECT Orders.OrderID, Customers.CustomerName, Shippers.ShipperName
FROM ((Orders
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID)
INNER JOIN Shippers ON Orders.ShipperID = Shippers.ShipperID);
```

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

Example

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

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

Example

```
SELECT Orders.OrderID, Employees.LastName, Employees.FirstName
FROM Orders
RIGHT JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID
ORDER BY Orders.OrderID;
```

SELECT *column_name(s)*
FROM *table1*
FULL OUTER JOIN *table2* **ON** *table1.column_name = table2.column_name*;

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
FULL OUTER JOIN Orders ON Customers.CustomerID=Orders.CustomerID
ORDER BY Customers.CustomerName;
```

SELECT *column_name(s)/(Use AS change to column name)*
FROM *table1 T1, table1 T2*
WHERE *condition*;

Example

```
SELECT A.CustomerName AS CustomerName1, B.CustomerName AS CustomerName2, A.City
FROM Customers A, Customers B
WHERE A.CustomerID <> B.CustomerID
AND A.City = B.City
ORDER BY A.City;
```

The SQL 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 column_name(s) FROM table1
UNION
SELECT column_name(s) FROM table2; /(Not allow duplicate values)

SELECT column_name(s) FROM table1
UNION ALL
SELECT column_name(s) FROM table2; /( To allow duplicate values)
```

Example

```
SELECT City, Country FROM Customers
WHERE Country='Germany'
UNION ALL
SELECT City, Country FROM Suppliers
WHERE Country='Germany'
ORDER BY City;
```

Example

```
SELECT 'Customer' As Type, ContactName, City, Country
FROM Customers
UNION
SELECT 'Supplier', ContactName, City, Country
FROM Suppliers;
```

*('**Customer**' Are Shown by Type_Column & Similarly '**Supplier**' Shown Because Separable data shown there own column & rows)...*

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);
```

Example

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

(DESC USE Value Show Not Shorted)...

Example

```
SELECT Shippers.ShipperName, COUNT(Orders.OrderID) AS NumberOfOrders FROM Orders
LEFT JOIN Shippers ON Orders.ShipperID = Shippers.ShipperID
GROUP BY ShipperName;
```

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);
```

Example

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

Example

```
SELECT Employees.LastName, COUNT(Orders.OrderID) AS NumberOfOrders
FROM Orders
INNER JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID
WHERE LastName = 'Davolio' OR LastName = 'Fuller'
GROUP BY LastName
HAVING COUNT(Orders.OrderID) > 25;
```

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

Example

```
SELECT SupplierName
FROM Suppliers
WHERE EXISTS (SELECT ProductName FROM Products WHERE SupplierId = Suppliers.supplierId AND Price
< 20);
```

The SQL ANY and ALL Operators

The ANY and ALL operators are used with a WHERE or HAVING clause.

The ANY operator returns true if any of the subquery values meet the condition.

The ALL operator returns true if all of the subquery values meet the condition.

SELECT *column_name(s)*
FROM *table_name*
WHERE *column_name operator* **ANY**
(**SELECT** *column_name* **FROM** *table_name* **WHERE** *condition*);

SELECT *column_name(s)*
FROM *table_name*
WHERE *column_name operator* **ALL**
(**SELECT** *column_name* **FROM** *table_name* **WHERE** *condition*);

Example

```
SELECT ProductName
FROM Products
WHERE ProductID = ANY (SELECT ProductID FROM OrderDetails WHERE Quantity > 99);
```

Example

```
SELECT ProductName
FROM Products
WHERE ProductID = ALL (SELECT ProductID FROM OrderDetails WHERE Quantity = 10);
```

The SQL INSERT INTO SELECT Statement

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

- INSERT INTO SELECT requires that data types in source and target tables match
- The existing records in the target table are unaffected

INSERT INTO *table2*
SELECT * **FROM** *table1*
WHERE *condition*;

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

Example

```
INSERT INTO Customers (CustomerName, City, Country)
SELECT SupplierName, City, Country FROM Suppliers
WHERE Country='Germany';
```

DATA Export:- mysqldump –u root –p(ex.admin/12345/ip) database_name>new_database_name.sql

DATA Import:- mysql –u root –p(ex.admin/12345/ip) save_database_name<new_database_name.sql

Create By Md. Abdul Mannan Hridoy

CSE 38th D