**SQL**

**Create database**

CREATE DATABASE testDB;

**Create table**

CREATE TABLE Employee(EM\_ID INT PRIMARY KEY AUTO\_INCREMENT,FIRST\_NAME VARCHAR(255),LAST\_NAME VARCHAR(255),SALARY INT(10),JOINING\_DATE DATE,DEPARTMENT VARCHAR(255));

**Insert data**

INSERT INTO employee(FIRST\_NAME,LAST\_NAME,SALARY,JOINING\_DATE,DEPARTMENT)VALUES("JOHN","ABRAHAM","1000000","2013-01-01","BANKING");

**Foreign key**

CREATE TABLE incentives(insentive\_id INT PRIMARY KEY AUTO\_INCREMENT,EMPLOYEE\_REF\_ID INT,INCENTIVE\_DATE DATE,INCENTIVE\_AMT INT,FOREIGN KEY(EMPLOYEE\_REF\_ID)REFERENCES employee(EM\_ID));

SELECT CustomerName, City FROM Customers;

SELECT Country FROM Customers;

Country

Maxico

Canada

Maxico

Canada

Singapore

(double data hase to pan dekhase)

SELECT DISTINCT Country FROM Customers;

Country

Maxico

Canada

Singapore

(distinct different data)

Count with distinct

SELECT COUNT(DISTINCT Country) FROM Customers;

**AS output vakhate column nu name change karva**

SELECT Count(\*) AS DistinctCountries  
FROM (SELECT DISTINCT Country FROM Customers);

where

SELECT \* FROM Customers  
WHERE Country='Mexico';

=

SELECT \* FROM Products

WHERE Price = 18;

SELECT \* FROM Products

WHERE Price > 30;

SELECT \* FROM Products

WHERE Price < 30;

SELECT \* FROM Products

WHERE Price >= 30;

SELECT \* FROM Products

WHERE Price <= 30;

|  |  |
| --- | --- |
| <> | Not equal. **Note:** In some versions of SQL this operator may be written as != |

< > same as !=

SELECT \* FROM Products

WHERE Price <> 18;

SELECT \* FROM Products

WHERE Price != 18;

BETWEEN

SELECT \* FROM Products

WHERE Price BETWEEN 50 AND 60;

**Like**

|  |  |
| --- | --- |
| **LIKE Operator** | **Description** |
| 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 2 characters in length |
| 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" |

SELECT \* FROM Customers

WHERE City LIKE 's%';

SELECT \* FROM Customers

WHERE City IN ('Paris','London');

SELECT \* FROM Customers  
WHERE Country='Germany' AND City='Berlin';

SELECT \* FROM Customers

WHERE City='Berlin' OR City='München';

SELECT \* FROM Customers

WHERE NOT Country='Germany';

(not equal jevu j che)

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

SELECT \* FROM Customers

WHERE NOT Country='Germany' AND NOT Country='USA';

Bydefault orderby assending ma j out put ape

SELECT \* FROM Customers  
ORDER BY Country;

SELECT \* FROM Customers  
ORDER BY Country DESC;

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

(ASS DESC MA 2 condition jode hoi , ave)

SELECT CustomerName, ContactName, Address  
FROM Customers  
WHERE Address IS NULL;

SELECT CustomerName, ContactName, Address  
FROM Customers  
WHERE Address IS NOT NULL;

Update ni query

UPDATE Customers  
SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'  
WHERE CustomerID = 1;

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

DELETE FROM Customers; (DROP pan use thai)

DELETE FROM Customers WHERE CustomerName='Alfreds Futterkiste';

**Note:** Not all database systems support the SELECT TOP clause. MySQL supports the LIMIT clause to select a limited number of records, while Oracle uses FETCH FIRST n ROWS ONLY and ROWNUM.

My sql ma limit use thai

SELECT TOP 3 \* FROM Customers;

SELECT \* FROM Customers  
LIMIT 3;

SELECT TOP 3 \* FROM Customers

WHERE Country='Germany';

SELECT MIN(Price) AS SmallestPrice  
FROM Products;

SELECT MAX(Price) AS LargestPrice  
FROM Products;

SELECT COUNT(ProductID)  
FROM Products;

SELECT AVG(Price)  
FROM Products;

SELECT SUM(Quantity)  
FROM OrderDetails;

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

(b ,s,or p badha thi start thai eva data ave)

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

(a to c sudhi na data ave)

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

(not equal)

Or

SELECT \* FROM Customers  
WHERE City NOT LIKE '[bsp]%';

SELECT \* FROM Customers  
WHERE Country IN ('Germany', 'France', 'UK');

SELECT \* FROM Customers  
WHERE Country NOT IN ('Germany', 'France', 'UK');

SELECT \* FROM Customers  
WHERE Country IN (SELECT Country FROM Suppliers);

SELECT \* FROM Products  
WHERE Price BETWEEN 10 AND 20;

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

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

SELECT \* FROM Products  
WHERE ProductName NOT BETWEEN 'Carnarvon Tigers' AND 'Mozzarella di Giovanni'  
ORDER BY ProductName;

SELECT \* FROM Orders  
WHERE OrderDate BETWEEN #07/01/1996# AND #07/31/1996#;

Or

SELECT \* FROM Orders  
WHERE OrderDate BETWEEN '1996-07-01' AND '1996-07-31';

**Alies**

SELECT CustomerID AS ID, CustomerName AS Customer  
FROM Customers;

SELECT CustomerName AS Customer, ContactName AS [Contact Person]  
FROM Customers;

Sql use thai

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

Mysql ma use thai

SELECT CustomerName, CONCAT(Address,', ',PostalCode,', ',City,', ',Country) AS Address  
FROM Customers;

Join ni query ma alias use

SELECT o.OrderID, o.OrderDate, c.CustomerName  
FROM Customers AS c, Orders AS o  
WHERE c.CustomerName='Around the Horn' AND c.CustomerID=o.CustomerID;

**Join (inner join ek j)**

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: Returns all records from the left table, and the matched records from the right table
* RIGHT (OUTER) JOIN: Returns all records from the right table, and the matched records from the left table
* FULL (OUTER) JOIN: Returns all records when there is a match in either left or right table

  

  

Three table join

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);

Left join

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

Right join

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

Full join

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