Overview on SQL- Basics

Reference Web Material- Click Here

CST 204
Database Management Systems

Jacob P Cherian
Asst.Professor
Dept.of CSE,Saintgits College of Engineering

SQL CAN DO THE FOLLOWING

- SQL can execute queries against a database
- SQL can retrieve data from a database
- SQL can insert records in a database
- SQL can update records in a database
- SQL can delete records from a database
- SQL can create new databases
- SQL can create new tables in a database
- SQL can create stored procedures in a database
- SQL can create views in a database
- SQL can set permissions on tables, procedures, and views

Create Table Command

```
CREATE TABLE table_name (

    column1 datatype,

    column2 datatype,

    column3 datatype,

   ....

);
```

```
CREATE TABLE Persons (

    PersonID int,

    LastName varchar(255),

    FirstName varchar(255),

    Address varchar(255),

    City varchar(255)

);
```

SQL Select Statement

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

The SELECT statement is used to select data from a database.

The data returned is stored in a result table, called the result-set.

```
SELECT * FROM table_name;
```

```
SELECT NAME,ROLL_NO FROM STUDENT;

SELECT CustomerName, City FROM Customers
```

SQL WHERE CLAUSE

```
SELECT column1, column2, ...

FROM table_name

WHERE condition;
```

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

```
SELECT * FROM Customers
WHERE CustomerID=1;
```

OPERATORS IN THE WHERE CLAUSE

Operator	Description
=	EQUAL
>	GREATER THAN
<	LESS THAN
>=	GREATER THAN OR EQUAL
<=	LESS THAN OR EQUAL
<>	NOT EQUAL
BETWEEN	BETWEEN A RANGE
LIKE	SEARCH FOR A PATTERN
IN	TO SPECIFY MULTIPLE POSSIBLE VALUES FROM A COLUMN

CONNECTORS- AND , OR, NOT

```
SELECT column1, column2, ...

FROM table_name

WHERE condition1 AND condition2 AND condition3 ...;
```

```
SELECT * FROM Customers
WHERE Country='India' AND City='Lucknow';
```

CONNECTORS- AND , OR, NOT

```
SELECT column1, column2, ...

FROM table_name

WHERE condition1 OR condition2 OR condition3 ...;
```

```
SELECT * FROM Customers
WHERE City='Delhi' OR City='Cochin';
```

CONNECTORS- AND , OR, NOT

```
SELECT column1, column2, ...

FROM table_name

WHERE NOT condition;
```

```
SELECT * FROM Customers
WHERE NOT Country='India';
```

SQL UPDATE

```
UPDATE table_name

SET column1 = value1, column2 = value2, ...

WHERE condition;
```

```
UPDATE Customers
SET Contact_Name = 'Muhammed', City= 'Cochin'
WHERE CustomerID = 1;
```

```
UPDATE Customers
SET ContactName='John' WHERE Country='USA';
```

SQL DELETE

```
DELETE FROM table_name WHERE condition;
```

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

```
DELETE FROM table_name;
```

```
DELETE FROM Customers
```

SQL MIN( )

```
SELECT MIN(column_name)

FROM table_name

WHERE condition;
```

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

The MIN() function returns the smallest value of the selected column.

SQL MAX( )

```
SELECT MAX(column_name)

FROM table_name

WHERE condition;
```

```
SELECT MAX(Price) FROM Products;
```

The MAX() function returns the largest value of the selected column.

SQL COUNT( )

```
SELECT COUNT(column_name)

FROM table_name

WHERE condition;
```

```
SELECT COUNT(ProductID) FROM Products;
```

The COUNT() function returns the number of rows that matches a specified criterion.

SQL AVG( )

```
SELECT AVG(column_name)

FROM table_name

WHERE condition;
```

```
SELECT AVG(Price) FROM Products;
```

The AVG() function returns the average value of a numeric column.

SQL SUM( )

```
SELECT SUM(column_name)

FROM table_name

WHERE condition;
```

```
SELECT SUM(Price) FROM Products;
```

The SUM() function returns the total sum of a numeric column.

SQL LIKE OPERATOR

```
SELECT column1, column2, ...

FROM table_name

WHERE column LIKE pattern;
```

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

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

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

```
SELECT * FROM Customers
WHERE CustomerName NOT LIKE 'a%';
```

SQL DISTINCT

```
SELECT DISTINCT column1, column2, ...

FROM table_name;
```

```
SELECT DISTINCT Country FROM
Customers;
```

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

SQL ALIAS

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.

An alias only exists for the duration of that query.

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

```
SELECT column_name(s)
FROM table_name AS
alias_name;
```

Example

```
SELECT CustomerID AS ID, CustomerName AS Customer FROM Customers;
```