SQL

~SQL is a standard language for storing, manipulating and retrieving data in databases.

~SQL stands for Structured Query Language.

~SQL keywords are NOT case sensitive.

CREATE DATABASE: The CREATE DATABASE statement is used to create a new SQL database.

```
--> CREATE  DATABASE  databasename;
```

DROP DATABASE:The DROP DATABASE statement is used to drop an existing SQL database.

```
--> DROP DATABASE databasename;
```

BACKUP DATABASE:The BACKUP DATABASE statement is used in SQL Server to create a full back up of an existing SQL database.

```
--> BACKUP DATABASE databasename TO DISK = 'filepath';
```

CREATE TABLE:The CREATE TABLE statement is used to create a new table in a database.

```
CREATE TABLE table_name (
   column1 datatype,
   column2 datatype,
   column3 datatype,
  ....
);
```

```
Exa: CREATE TABLE Persons (
   PersonID int,
   LastName varchar(255),
   FirstName varchar(255),
   Address varchar(255),
   City varchar(255) );
```

Create Table Using Another Table: A copy of an existing table can also be created using CREATE TABLE.

[আপনি যদি বিদ্যমান টেবিলটি ব্যবহার করে একটি নতুন টেবিল তৈরি করেন তবে নতুন টেবিলটি পুরানো সারণি থেকে বিদ্যমান মানগুলি দিয়ে পূর্ণ হবে।]

```
--> CREATE TABLE new_table_name AS
   SELECT column1, column2,...
   FROM existing_table_name
   WHERE ....;
```

```
Exa: CREATE TABLE TestTable AS
SELECT customername, contactname
FROM customers;
```

DROP TABLE: The DROP TABLE statement is used to drop an existing table in a database.

```
DROP TABLE table_name;
```

ALTER TABLE:The ALTER TABLE statement is used to add, delete, or modify columns in an existing table.

ADD Column:

```
--> ALTER TABLE table_name ADD column_name datatype;
Exa: ALTER TABLE Customers ADD Email varchar(255);
```

[Now we want to add a column named "DateOfBirth" in the "Persons" table.]

```
Exa: ALTER TABLE Persons ADD DateOfBirth date;
```

DROP COLUMN:

```
--> ALTER TABLE table_name DROP COLUMN column_name;
Exa: ALTER TABLE Customers DROP COLUMN Email;
Exa: ALTER TABLE Persons DROP COLUMN DateOfBirth;
```

MODIFY COLUMN:

--> ALTER TABLE table_name MODIFY COLUMN column_name datatype;

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

--> SELECT column1, column2, ...FROM table_name;

Exa:- SELECT CustomerName, City FROM Customers;

[Here, column1, column2, ... are the field names of the table you want to select data from.]

[If you want to select all the fields available in the table, use the following syntax:-]

--> SELECT * FROM table_name;

Exa:- SELECT * FROM Customers;

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

--> SELECT DISTINCT column1, column2, ...FROM table_name;

Exa:- SELECT DISTINCT Country FROM Customers;

--> SELECT COUNT(DISTINCT column1) FROM table_name;

Exa:- SELECT COUNT(DISTINCT Country) FROM Customers;

--> SELECT Count(*) AS DistinctCountries FROM (SELECT DISTINCT column1 FROM table_name);

Exa:- SELECT Count(*) AS DistinctCountries FROM (SELECT DISTINCT Country FROM Customers);

WHERE CLAUSE: The WHERE clause is used to extract only those records that fulfill a specified condition.

--> SELECT column1, column2, ...FROM table_name WHERE condition;

--> SELECT * FROM table_name WHERE condition;

Exa: SELECT * FROM Customers WHERE Country='Mexico';

SELECT * FROM Customers WHERE CustomerID=1;

SELECT * FROM Products WHERE Price = 18;

SELECT * FROM Products WHERE Price >= 30;

SELECT * FROM Products WHERE Price <> 18; [Note: Not equal.In some versions of SQL this operator may be written as !=]

SELECT * FROM Products WHERE Price BETWEEN 50 AND 60;

SELECT * FROM Customers WHERE City IN ('Paris','London');

[The WHERE clause is not only used in SELECT statement, it is also used in UPDATE, DELETE statement, etc.!]

AND, OR and NOT Operators :The WHERE clause can be combined with AND, OR, and NOT operators.

--> AND Syntax :- SELECT column1, column2, ...FROM table_name WHERE condition1 AND condition2 AND condition3 ...;

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

--> OR Syntax :- SELECT column1, column2, ... FROM table_name WHERE condition1 OR condition2 OR condition3 ...;

Exa: SELECT * FROM Customers WHERE Country='Germany' OR City='Berlin';

--> NOT Syntax :- SELECT column1, column2, ...FROM table_name WHERE NOT condition;

Exa: SELECT * FROM Customers WHERE NOT Country='Germany';

--> Combining AND, OR and NOT:-

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

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

ORDER BY Keyword: The ORDER BY keyword is used to sort the result-set in ascending or descending order.

--> SELECT column1, column2, ...FROM table_name ORDER BY column1, column2, ... ASC|DESC;

Exa: SELECT * FROM Customers ORDER BY Country;

SELECT * FROM Customers ORDER BY Country DESC;

SELECT * FROM Customers ORDER BY Country, CustomerName;

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

[The ORDER BY keyword sorts the records in ascending order by default. To sort the records in descending order, use the DESC keyword.]

INSERT INTO Statement: The INSERT INTO statement is used to insert new records in a table.

It is possible to write the INSERT INTO statement in two ways.

The first way:

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

Exa: INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country)

VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006', 'Norway');

(I am Use)The second way:

--> INSERT INTO table_name VALUES (value1, value2, value3, ...);

Exa: INSERT INTO Customers (CustomerName, City, Country)

VALUES ('Cardinal', 'Stavanger', 'Norway');

NULL Values: Find A field with a NULL value is a field with no value?

--> IS NULL Syntax: SELECT column_names FROM table_name WHERE column_name IS NULL;

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

--> IS NOT NULL Syntax: SELECT column_names FROM table_name WHERE column_name IS NOT NULL;

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

UPDATE Statement: The UPDATE statement is used to modify the existing records in a table.

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

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

[দ্রষ্টব্য: কোনও টেবিলে রেকর্ড আপডেট করার সময় সাবধান! আপডেটের বিবৃতিতে WHERE ধারাটি লক্ষ্য করুন। WHERE ধারাটি নির্দিষ্ট করে যে কোন রেকর্ড (গুলি) আপডেট করা উচিত। আপনি WHERE ধারাটি বাদ দিলে টেবিলের সমস্ত রেকর্ড আপডেট করা হবে!]

DELETE Statement: The DELETE statement is used to delete existing records in a table.

--> DELETE FROM table_name WHERE condition;

Exa: DELETE FROM Customers WHERE CustomerName='Alfreds Futterkiste';

[Be careful!!WHERE ধারাটি নির্দিষ্ট করে যে কোন রেকর্ড (গুলি) মুছে ফেলা উচিত। আপনি WHERE ধারাটি বাদ দিলে টেবিলের সমস্ত রেকর্ড মুছে ফেলা হবে!]

Delete All Records:

--> DELETE FROM table_name;

Exa: DELETE FROM Customers;

SQL SELECT TOP Clause: The SELECT TOP clause is used to specify the number of records(top) to return.

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

Exa: SELECT * FROM Customers LIMIT 3; [Show 1st 3 data]

SELECT * FROM Customers WHERE Country='Germany' LIMIT 3; [Show 1st 3 Germany country data]

MIN() and MAX() Function:The MIN() / MAX() function returns the smallest value/ largest value of the selected column.

--> SELECT MIN/MAX(column_name) FROM table_name WHERE condition;

Exa: SELECT MIN(Price) AS SmallestPrice FROM Products;

SELECT MAX(Price) AS LargestPrice FROM Products;

Exa: SELECT MIN(*) FROM Products WHERE Price = 18;

COUNT(), AVG() and SUM() Functions:

--> SELECT COUNT(column_name) FROM table_name WHERE condition;

Exa: SELECT COUNT(ProductID) FROM Products;

--> SELECT AVG(column_name) FROM table_name WHERE condition;

Exa: SELECT AVG(Price) FROM Products;

--> SELECT SUM(column_name) FROM table_name WHERE condition;

Exa: SELECT SUM(Quantity)FROM OrderDetails;

Exa: SELECT COUNT(*) FROM Products WHERE Price = 18;

SQL LIKE Operato: LIKE অপারেটর একটি কলামে একটি নির্দিষ্ট প্যাটার্ন সন্ধান করতে WHERE ধারাটিতে ব্যবহৃত হয়।

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

[You can also combine any number of conditions using AND or OR operators.]

'a%'-- Finds any values that start with "a"

'%a'-- Finds any values that end with "a"

'%or%'-- Finds any values that have "or" in any position

'_r%' -- Finds any values that have "r" in the second position

'a__%' -- Finds any values that start with "a" and are at least 3 characters in length

'a%o' --Finds any values that start with "a" and ends with "o"

'a_%_%' -- Finds any values that starts with "a" and are at least 3 characters in length

Exa: SELECT * FROM Customers WHERE CustomerName LIKE 'a%'

SQL Wildcards: একটি ওয়াইল্ডকার্ড অক্ষর একটি স্ট্রিংয়ে এক বা একাধিক অক্ষরের বিকল্প ব্যবহার করতে ব্যবহৃত হয়।

Wildcard characters are used with the SQL LIKE operator.The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

Exa: SELECT * FROM Customers WHERE City LIKE 'ber%';

[SQL statement selects all customers with a City starting with "ber"]

Exa: SELECT * FROM Customers WHERE City LIKE '%es%';

[SQL statement selects all customers with a City containing the pattern "es"]

Exa: SELECT * FROM Customers WHERE City LIKE '_ondon';

[SQL statement selects all customers with a City starting with any character, followed by "ondon"]

Exa: SELECT * FROM Customers WHERE City LIKE 'L_n_on';

[SQL statement selects all customers with a City starting with "L", followed by any character, followed by "n", followed by any character, followed by "on"]

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

[SQL statement selects all customers with a City starting with "b", "s", or "p"]

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

[SQL statement selects all customers with a City starting with "a", "b", or "c"]

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

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

[SQL statements select all customers with a City NOT starting with "b", "s", or "p"]

SQL IN Operator:

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

or

--> SELECT column_name(s) FROM table_name WHERE column_name IN (SELECT STATEMENT);

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

[SQL statement selects all customers that are located in "Germany", "France" or "UK"]

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

Exa: SELECT * FROM Customers WHERE Country IN (SELECT Country FROM Suppliers);

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;

Exa: SELECT * FROM Products WHERE Price BETWEEN 10 AND 20;

Exa: SELECT * FROM Products WHERE Price NOT BETWEEN 10 AND 20;

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

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

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

SQL Aliases:SQL aliases are used to give a table, or a column in a table, a temporary name.[এসকিউএল এলিয়াস একটি টেবিল, বা একটি টেবিলের একটি কলাম, একটি অস্থায়ী নাম দিতে ব্যবহৃত হয়।]

Alias Column Syntax:

--> SELECT column_name AS alias_name FROM table_name;

Exa: SELECT CustomerID AS ID, CustomerName AS Customer FROM Customers;

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

Exa: SELECT CustomerName AS Customer, ContactName AS "Contact Person" FROM Customers;

Alias Table Syntax:

--> SELECT column_name(s) FROM table_name AS alias_name;

[creates an alias named "Address" that combine four columns (Address, PostalCode, City and Country)]

[একটি aliases(temporary name) undare একাধিক কলামকে একত্রিত করে all Data রাখতে চাইলে ...]

--> SELECT column_name , alias_name + ',' + column_name +... FROM table_name;

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

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

SQL JOIN: A JOIN clause is used to combine rows from two or more tables, based on a related column between them.

[তাদের মধ্যে সম্পর্কিত কলামের ভিত্তিতে দুটি বা ততোধিক সারণী থেকে সারি একত্রিত করতে একটি JOIN ক্লজ ব্যবহার করা হয়।]

Joining Tables-->

--> SELECT table1.column_name(Condition Base),table2.column_name,table1.column_name,.. FROM table1,table2 WHERE table1.column_name(Condition Base)= table2.column_name(Condition Base)

(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

INNER JOIN:

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

Exa: SELECT Orders.OrderID, Customers.CustomerName FROM Orders

```
                    INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;
```

JOIN Three Tables:

```
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);
```

FULL OUTER JOIN:

```
SELECT column_name(s)

FROM table1

FULL OUTER JOIN table2

ON table1.column_name = table2.column_name

WHERE condition;
```

Exa:

```
SELECT Customers.CustomerName, Orders.OrderID

FROM Customers

FULL OUTER JOIN Orders ON Customers.CustomerID=Orders.CustomerID

ORDER BY Customers.CustomerName;
```

UNION/UNION BY:The UNION operator is used to combine the result-set of two or more SELECT statements. [দুই বা ততোধিক SELECT স্টেটমেন্টের ফলাফল-সংমিশ্রণের জন্য ইউনিয়ন অপারেটর ব্যবহার করা হয়।]

# Each SELECT statement within UNION must have the same number of columns[ইউনিয়নের মধ্যে প্রতিটি নির্বাচনী বিবৃতিতে একই সংখ্যক কলাম থাকতে হবে]

# The columns must also have similar data types[কলামগুলিতে অবশ্যই একই জাতীয় ডেটা থাকতে হবে]

# The columns in each SELECT statement must also be in the same order[প্রতিটি নির্বাচনী বিবৃতিতে কলামগুলিও একই ক্রমে থাকতে হবে]

```
--> SELECT column_name(s) FROM table1

   UNION

   SELECT column_name(s) FROM table2;

   Exa: SELECT City FROM Customers UNION SELECT City FROM Suppliers ORDER BY City;
```

[The UNION operator selects only distinct values by default.(ডিফল্টরূপে পৃথক মানগুলি নির্বাচন করে)

if allow duplicate values, use UNION ALL]

```
--> SELECT column_name(s) FROM table1

   UNION ALL

   SELECT column_name(s) FROM table2;
```