SQL

*__SQL__ is a standard language for accessing and manipulating databases.

*__RDBMS__ stands for Relational Database Management System

SELECT & WHERE

```
SELECT (TOP N) column1, ...
FROM table_name
WHERE condition;
```

SELECT DISTINCT

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

ORDER BY

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

INSERT INTO

```
INSERT INTO table_name (column1,  ...)
VALUES (value1,  ...);
```

IS NULL/IS NOT NULL

```
SELECT column_names
FROM table_name
WHERE column_name IS (NOT)
```

UPDATE

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

DELETE

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

* **aggregate function** is a function that performs a calculation on a set of values, and returns a single value.

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

LIKE

% zero, one, or multiple characters

_ one, single character

EXP1":all customers that starts with the letter "a"

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

EXP2":city that starts with 'L' followed by one character, then 'nd' and then two characters

```
SELECT * FROM Customers
WHERE city LIKE 'L_nd__';
```

IN

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

BETWEEN

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

JOIN

```
ELECT column_name(s)
FROM table1 (INNER/RIGHT/LEFT/FULL OUTER) JOIN table2
ON table1.column_name = table2.column_name;
```

UNION combine two or more SELECT statements,must have the same number of columns, (To allow duplicate values, use **UNION ALL**)

```
SELECT column_name(s) FROM table1
UNION
SELECT column_name(s) FROM table2;
```

GROUP BY statement groups rows that have the same values , like "find the number of

```
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
ORDER BY COUNT(CustomerID) DESC;
```

HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate functions

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

EXISTS checks if the subquery's condition is met. If the subquery returns at least one row that satisfies the condition, EXISTS evaluates to true, and the row from the main query is included in the result set

```
SELECT SupplierName
FROM Suppliers
WHERE EXISTS (SELECT * FROM Products WHERE Products.SupplierID = Suppliers.supplierID AND Price < 20);
```

All use like EXISTS but it returns TRUE if ALL of the subquery values meet the condition

SQL

CASE expression goes through conditions and returns a value when the first condition is met (like an if-then-else statement)

```
CASE
   WHEN condition1 THEN result1
   WHEN conditionN THEN resultN
   ELSE result
END;
```

stored procedure is a prepared SQL code that you can save, so the code can be reused over and over again.

```
Create: CREATE PROCEDURE procedure_name
AS
sql_statement
GO;
Execute: EXEC procedure_name;
```

SP With Parameter

```
CREATE PROCEDURE SelectAllCustomers @City nvarchar(30)
AS
SELECT * FROM Customers
WHERE City = @City
GO;
EXEC SelectAllCustomers @City = 'London';
DELETE FROM table_name
WHERE condition;
```

Common Table Expression (CTE) is a temporary result set that you can reference within a SELECT, INSERT, UPDATE, or DELETE statement

```
WITH cte_name AS (
   -- CTE Query)
```

create a new SQL

```
CREATE DATABASE databasename;
```

drop an existing SQL database.

```
DROP DATABASE databasename;
```

create a back up

```
BACKUP DATABASE databasename
TO DISK = 'filepath'
(WITH DIFFERENTIA);
```

create a new table

```
CREATE TABLE table_name (
   column1 datatype
   ....
);
```

drop an existing table

```
DROP TABLE table_name;
```

delete the data inside a table

```
TRUNCATE TABLE table_name;
```

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

```
ALTER TABLE table_name
ADD column_name datatype
DROP COLUMN column_name
RENAME COLUMN old_name to new_name;
ALTER COLUMN column_name datatype;
```

constraints

1.NOT NULL enforces to NOT accept NULL values

```
CREATE TABLE Persons (
   ID int NOT NULL
```

2.UNIQUE: all values in a column are different.

3. PRIMARY KEY constraint uniquely identifies each record in a table

```
CREATE TABLE Persons (
   ID int NOT NULL PRIMARY KEY,)
```

4.FOREIGN KEY is a field (or collection of fields) in one table, that refers to the PRIMARY KEY in another table.

```
CREATE TABLE Orders (
   OrderID int NOT NULL PRIMARY KEY,
   PersonID int FOREIGN KEY REFERENCES Persons(PersonID)
);
```

5.CHECK constraint is used to limit the value range that can be placed in a column

```
CREATE TABLE Persons (
   ID int NOT NULL,
   Age int CHECK (Age>=18)
```

6.Indexes are used to retrieve data from the database more quickly than otherwise.they are just used to speed up searches/queries.Updating a table with indexes takes more time than updating a table

```
CREATE (UNIQUE) INDEX index_name
ON table_name (column1, column2, ...);
```

View is a virtual table based on the result-set of an SQL statement.

The database engine recreates the view, every time a user queries it.

```
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```