DML

DML - Data Manipulation Language are languages and commands that deal with manipulating data in the database
It's data languages used to retrieve, add, update, and delete records in a table in a database. In sql, these commands are SELECT, UPDATE, INSERT, and DELETE.

Select - Retrieve a result set of records from one or more tables. Through the use of clauses, this data can be filtered or customized as desired.

Logical Order of Operations

- FROM / JOIN - Determines the total working set of data being queried
- WHERE - Applies constraint to individual rows and discards any that don't mean them
    - Use AND and OR to apply multiple constraints
- GROUP BY - Groups together rows based on common values
- HAVING - Discards grouped row (if applicable) if they do not meet the constraints
    - Must have GROUP BY in query in order to use HAVING
- SELECT - Get all rows that are produced by above clauses
- DISTINCT - Discards any rows with duplicate values
- ORDER BY - Sort the data in either ascending / descending order by the specified data
- LIMIT / OFFSET - Discards any rows that fall outside of the specified range

Select Query Format with Every Clause

- SELECT DISTINCT columns
  FROM table
  JOIN table2
  WHERE [condition]
  GROUP BY column
  HAVING [condition2]
  ORDER BY column [ASC/DESC]
  LIMIT count
    - Order of these clauses is important
    - Not every clause in the SELECT query needs to be included.

Example

- SELECT id, name FROM customer, WHERE id < 10

To select the entire table, use SELECT * FROM table

Update- Is used to modify existing records in the table

Update Query Format

- UPDATE table_name
  SET column1 = value1, column2 = value2, columnN = valueN
  WHERE [condition];

Example

- UPDATE CUSTOMERS
  SET ADDRESS = 'Pune', SALARY = 1000.00;

- Update needs a Where clause, otherwise every column would be updated.

Insert- Adds new rows to the table.

Insert Query Format

- INSERT INTO table_name (column_1, column_2, column_3, … columnN)
  VALUES (value1, value2, value3, … valueN);

If adding all the values, you do not need to specify the columns, but the values need to be in the same order as the columns of the table.

INSERT INTO TABLE_NAME VALUES (value1,value2,value3,...valueN);

Example
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (1, 'Ramesh', 32, 'Ahmedabad', 2000.00);

Delete-Removes existing records from the table in the database

Delete Query Format

- DELETE FROM table_name
  WHERE [condition];

Example

- DELETE FROM CUSTOMERS
  WHERE ID = 6;

To delete all records from table, the command is
DELETE FROM table_name;
So be careful to put conditions

DML Query Examples

```
SELECT *
    FROM Company_Example.dbo.Employees;

SELECT FirstName, LastName AS Surname
    FROM Company_Example.dbo.Employees;
```

100 %

Results | Messages

	EmployeeID	FirstName	LastName	JobTitle	Salary	Age	BuildingID
1	2	Jane	Smith	Engineer	80000.00	23	1
2	3	Tom	Riddle	CEO	1000000.00	45	2
3	4	Mark	Antony	NULL	90000.00	50	2
4	5	Clark	Kent	Journalist	50000.00	24	2
5	6	Peter	Parker	NULL	50000.00	24	2
6	8	Delete	Me	NULL	10.00	0	1
7	9	John	Smith	Engineer	10.00	23	1

	FirstName	Surname
1	Jane	Smith
2	Tom	Riddle
3	Mark	Antony
4	Clark	Kent
5	Peter	Parker
6	Delete	Me
7	John	Smith

```sql
/* Unlimited AND/OR
    Operators
    =, !=, <, <=, >, >=             Standard operators
    BETWEEN ... AND                 Number is within range of two values (inclusive)
    NOT BETWEEN ... AND...          Number is not within range of two values (inclusive)
    IN (list of numbers)            Number is in a list
    NOT IN (list of numbers)        Number is not in list
*/

SELECT *
    FROM Company_Example.dbo.Employees
    WHERE Salary > 90000;

SELECT *
    FROM Company_Example.dbo.Employees
    WHERE Salary > 90000 OR AGE > 46;

SELECT *
    FROM Company_Example.dbo.Employees
    WHERE AGE NOT BETWEEN 10 AND 100;

SELECT *
    FROM Company_Example.dbo.Employees
    WHERE AGE IN (44, 45, 46);
```

	EmployeeID	FirstName	LastName	JobTitle	Salary	Age	BuildingID
1	3	Tom	Riddle	CEO	1000000.00	45	2

	EmployeeID	FirstName	LastName	JobTitle	Salary	Age	BuildingID
1	3	Tom	Riddle	CEO	1000000.00	45	2
2	4	Mark	Antony	NULL	90000.00	50	2

	EmployeeID	FirstName	LastName	JobTitle	Salary	Age	BuildingID
1	8	Delete	Me	NULL	10.00	0	1

	EmployeeID	FirstName	LastName	JobTitle	Salary	Age	BuildingID
1	3	Tom	Riddle	CEO	1000000.00	45	2

```sql
/* Operators for text
    =                       Case sensitive exact string comparison
    != or <>                Case sensitive exact string inequality comparison
    LIKE                    Case insensitive exact string comparison
    NOT LIKE                Case insensitive exact string inequality comparison
    %                       Used anywhere in string to match 0 or more characters '%hot%'
    _                       Used to match a single character 'AN_'
    IN (String List)        String in a list
    NOT IN (Sting List)     String not in a list
*/
SELECT *
    FROM Company_Example.dbo.Employees
    WHERE FirstName = 'John';

SELECT *
    FROM Company_Example.dbo.Employees
    WHERE LastName LIKE 'SMIT_';

SELECT *
    FROM Company_Example.dbo.Employees
    WHERE FirstName IN ('John', 'Tom');
```

	EmployeeID	FirstName	LastName	JobTitle	Salary	Age	BuildingID
1	9	John	Smith	Engineer	10.00	23	1

	EmployeeID	FirstName	LastName	JobTitle	Salary	Age	BuildingID
1	2	Jane	Smith	Engineer	80000.00	23	1
2	9	John	Smith	Engineer	10.00	23	1

	EmployeeID	FirstName	LastName	JobTitle	Salary	Age	BuildingID
1	3	Tom	Riddle	CEO	1000000.00	45	2
2	9	John	Smith	Engineer	10.00	23	1

```sql
/* Filtering and Sorting
    DISTINCT discards duplicate records
    ORDER BY column ASC/DESC
*/

SELECT LastName
    FROM Company_Example.dbo.Employees
    WHERE LastName = 'Smith';

SELECT DISTINCT LastName
    FROM Company_Example.dbo.Employees
    WHERE LastName = 'Smith';

SELECT *
    FROM Company_Example.dbo.Employees
    WHERE LastName = 'Smith'
    ORDER BY Salary ASC;
```

100 %

Results | Messages

	LastName
1	Smith
2	Smith

	LastName
1	Smith

	EmployeeID	FirstName	LastName	JobTitle	Salary	Age	BuildingID
1	9	John	Smith	Engineer	10.00	23	1
2	2	Jane	Smith	Engineer	80000.00	23	1

```
/*
SELECT columns
FROM mytable
    JOIN another_table
      ON mytable.column = another_table.column
*/

SELECT *
    FROM Company_Example.dbo.Employees as Employees
    INNER JOIN Company_Example.dbo.Buildings as Buildings
        ON Employees.BuildingID = Buildings.BuildingID;

SELECT *
    FROM Company_Example.dbo.Employees as Employees
    INNER JOIN Company_Example.dbo.Buildings as Buildings
        ON Employees.BuildingID = Buildings.BuildingID
    WHERE BuildingName = 'White House';
```

100 %

Results | Messages

	EmployeeID	FirstName	LastName	JobTitle	Salary	Age	BuildingID	BuildingID	BuildingName
1	2	Jane	Smith	Engineer	80000.00	23	1	1	White House
2	3	Tom	Riddle	CEO	1000000.00	45	2	2	Green House
3	4	Mark	Antony	NULL	90000.00	50	2	2	Green House
4	5	Clark	Kent	Journalist	50000.00	24	2	2	Green House
5	6	Peter	Parker	NULL	50000.00	24	2	2	Green House
6	8	Delete	Me	NULL	10.00	0	1	1	White House
7	9	John	Smith	Engineer	10.00	23	1	1	White House

	EmployeeID	FirstName	LastName	JobTitle	Salary	Age	BuildingID	BuildingID	BuildingName
1	2	Jane	Smith	Engineer	80000.00	23	1	1	White House
2	8	Delete	Me	NULL	10.00	0	1	1	White House
3	9	John	Smith	Engineer	10.00	23	1	1	White House

```sql
/*
SELECT DISTINCT column, AGG_FUNC(column_or_expression), …
FROM mytable
    JOIN another_table
      ON mytable.column = another_table.column
    WHERE constraint_expression
    GROUP BY column
    HAVING constraint_expression
    ORDER BY column ASC/DESC
    LIMIT(TOP) count OFFSET COUNT;

    Select the sum of salaries greater than 5000 by building ordered ascending.
*/
SELECT BuildingName, SUM(Salary) AS Combined_Salary
    FROM Company_Example.dbo.Employees as Employees
    INNER JOIN Company_Example.dbo.Buildings as Buildings
        ON Employees.BuildingID = Buildings.BuildingID
    WHERE Salary > 5000
    GROUP BY BuildingName
    ORDER BY Combined_Salary ASC;
```

100 %

Results | Messages

	BuildingName	Combined_Salary
1	White House	80000.00
2	Green House	1190000.00