IV. DDL Commands:

→ DDL (Data Definition Language) used to change the structure of the table like creating the table, altering the table, and deleting the table.

→ DDL Command:

1. CREATE:

. Used to create a new database or table

Syntax:

```
CREATE DATABASE database_name;

CREATE TABLE TableName (
        Attribute 1  Datatype [Constraint],
        Attribute 2  Data type [Constraint],
        Attribute 3  Data type [Constraint],
        [other Constraints]);
```

Example:

```
CREATE TABLE employees (
        employee_number varchar (10) NOT NULL,
        employee_name   varchar (50) NOT NULL,
        department_id   varchar (10),
        salary          decimal (10, 2),
CONSTRAINT employees_pK PRIMARY KEY (employee_number),
CONSTRAINT departments_FK FOREIGN KEY (department_id)
REFERENCES departments (department_id));
```

2. ALTER

. This command allow to add, modify, and delete columns of an existing table

→ syntax:

```
ALTER TABLE table_name
... alter table action ...
```

→ Alter table actions:

. Add column in table

```
ALTER TABLE table_name
ADD column_name column_definition;
```

. Add multiple columns in table

```
ALTER TABLE table_name
ADD (column_1 column_definition,
     column_2 column_definition,
     column_n colum_definition);
```

. Modify column in table

```
ALTER TABLE table_name
MODIFY column_name column_type;
```

. Drop column in table

```
ALTER TABLE table_name
DROP COLUMN column_name;
```

. Rename column in table

```
ALTER TABLE table_name
RENAME COLUMN old_name TO new_name;
```

. Rename table

```
ALTER TABLE table_name
RENAME TO new_table_name ;
```

. Add table constraint

```
ALTER TABLE table_name
Add CONSTRAINT constraint_name constraint_Type
                (Attribute(s)) ;
```

→ Examples:

- ALTER TABLE Vendor
  MODIFY Phone_Number Varchar(40) ;

- ALTER TABLE Vendor
  DROP CONSTRAINT NOT NULL (Phone number) ;

- ALTER TABLE Vendor
  DROP COLUMN Phone_Number ;

- ALTER TABLE Part
  Add (Weight_number Not null,
        Category Varchar(25) Not null) ;

- ALTER TABLE Vendor
  Add Phone_number Varchar(15) Not Null ;

3. DROP

. Used for deleting an entire database or a database table including structure and record stored in table

→ syntax:

```
DROP TABLE table_name;
```

→ Example:

```
DROP TABLE constructor;
```

4. TRUNCATE

Use to remove all the records from a table

→ syntax:

```
TRUNCATE TABLE table_name;
```

→ Example:

```
TRUNCATE TABLE customer_name;
```

5. COMMENT

. Used to write comment within SQL queries

. syntax

```
-- this is a single line comment

/* this is a mulliple
   line comment */
```

V. DML Commands

1. INSERT

. Used to insert a single or multiple row (record) in a table

→ Syntax:

```
INSERT INTO table_name (
            column 1 , column 2 , column 3)
VALUES (value 1 , value 2 , value 3) ;
```

→ Examples:

```
INSERT INTO customers (
 customer_id , sale_date, sale_amount, order_id)
 VALUES ('1005', '2019-12-12', 4200 , '1007');
```

or

```
INSERT INTO customers
VALUES ('1005', '2019-12-12', 4200 , '1007');
```

→ INSERT important notes

. the ordering of attribute and corresponding values is crucial

. The list of attribute and the list of values must have the same nb of elements

. If the list of attribute is missing (like the second example), the list of all attributes is taken, with the ordering taken from the database definition

. If the list of attribute does not contain all attribute of the relation, the default value or the value null (if possible) is inserted for the missing attribute.

2. UPDATE

- Used to modify an existing row (record) in a table

→ Syntax

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

→ Examples:

```
UPDATE customers
SET store_state = 'DL'
WHERE store_state = 'NY' ;
```

```
UPDATE person
SET income = income * 1.1
WHERE age < 30
```

3. DELETE

- Used to remove one or more rows from a table

→ syntax:

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

→ Example:

```
DELETE FROM person
WHERE age < 35 ;
```

VI- Data Query Language

Query statements in SQL start with a SELECT and return result in table form

- **Basics**

Use sql_store ; : حدد من أي DB بدنا كمل كل العمل
DB name
أو بدنا نعمل select

Select customer_id , first name ; : select حدد أي column بدنا نعطلن

or Select * : columns الـ select بكل

From customers حدد الـ table

Where customer_id = 1
condition

ORDER BY first_name حدد الترتيب البدنا يا نعرض العمل

LIMIT 3 ; حدد عدد الـ rows البدنا ياهن

N.B !

. In my SQL, every statement must terminated with ";"

. The order of commands is important :

SELECT
FROM
WHERE
ORDER BY
LIMIT

→ SELECT Clause

-- Using expressions

We can use mathematical expression with the select statement.

ex:

```
SELECT (points +10) * 100 AS discount_factor
FROM customers ;
```

(points +10) * 100 : formula to calculate discount for each customer

AS : we use AS to give the result a proper name

-- Removing duplicates

```
SELECT DISTINCT State
FROM customers ;
```

DISTINCT : remove duplicate value and give unique state

→ WHERE CLAUSE

. where is used to filter data.

. WHERE Condition

↳ Comparison operators: = ; > ; < ; >= ; <= ; != / <> .

↳ Logical operators:

-- AND (both conditions must be true)

```
SELECT *
FROM customers
WHERE birth_date > '1990-01-01' AND points > 1000;
```

-- OR (at least one conditions must be true)

```
SELECT *
FROM customers
WHERE birth_date > '1990_01_01' OR points > 1000;
```

-- NOT (to negate a condition)

```
SELECT *
FROM customers
WHERE NOT (birth_date > '1990-01-01');
```

N.B !!

If we want for example to select customers that lives in "VA", "GA", or "FL" States.

~~WHERE State = "VA" OR "GA" OR "FL";~~

(this query is false)

instead of this we should write:

```
WHERE state = "VA" OR state = "GA" OR state = "FL";
```

And there is a more simple way to write this query: the IN operator

↳ **IN operator**

This statement is used to retrieve records where a column value matches any value in a list

```
SELECT *
FROM customers
WHERE state IN ('VA', 'NY', 'CA');
```

↳ **BETWEEN Operator**

This statement is used to retrieve records where a column value is between two specified values

```
SELECT *
FROM customers
WHERE points BETWEEN 100 AND 200;
```

LIKE Operator

This statement is used to retrieve records where a column value matches a specified pattern

```
-- Returns customers whose first name starts with a
SELECT *
FROM customers
WHERE first_name LIKE 'b%';
```

% : any nb of characters
_ : exactly one character

REGEXP Operator

Same as LIKE but for more complexe pattern

```
-- Returns customers whose first name start with b
SELECT *
FROM customers
WHERE first_name REGEXP '^b'
```

^ : beginning of a string
$: end of a string
| : logical or
[abc] : match any single character
[a-d] : any character from a to d

```
-- Return customers whose first name ends with EY
   or contain SE
WHERE first_name REGEXP '$EY$|SE'
-- Return customers whose first_name contains B followed by R or U
WHERE first_name REGEXP 'b[ru]'
```

IS NULL Operator

```
-- Return customers who doesn't have a phone number
SELECT *
FROM customers
WHERE phone_number IS NULL;
```

ORDER BY Clause

- The order by clause sorts the rows returned by a select statement based on one or more columns in ascending or descending order
- BY default, it set the result in ascending order.
- we can order by an alias or an arithmetic expression

```
-- Sort customers by State (in ascending order), an ther
-- by their first name (in descending order)

SELECT *
FROM customers
ORDER BY state, first_name DESC;
```

LIMIT Clause

Limit the nb of records returned from a query

```
--Return only 3 customers
SELECT *
FROM customers
LIMIT 3;

-- SKip 6 customers and return 3
SELECT *
FROM customers
LIMIT 6, 3
```