DATA QUERY LANGUAGE

DQL

- DQL is a short name for Data Query Language.
- **DQL** statements are used for **performing queries** on the data within schema objects.
- The purpose of the DQL Command is to get some schema relation based on the query passed to it.
- It includes the **SELECT** statement. This command allows getting the data out of the database to perform operations with it.
- When we use a SELECT statement against a table or tables the result is compiled into a further **temporary table**, which is displayed as a result set.
- DQL is also considered part of **DML** by some.

DQL COMMAND

DQL Command in SQL :

S.No	Command	Description
1	**SELECT**	It is used to **retrieve data** from the table/tables.

SELECT COMMAND

- Select is the most commonly used statement in SQL.
- The SELECT Statement in SQL is used to retrieve or fetch data from a database. We **can fetch either the entire table** or **according to some specified rules**. The data returned is stored in a **result table**.

SELECT COMMAND…

- *Note:* We will be using "**CustomerDB**" Database for DQL Examples.
- The below command will fetch records from the table.

Syntax :	Example :
SELECT column_name1, column_name2… **FROM** table_name ;	`SELECT CustomerID, CustomerName` `FROM Customer;`

- **Output(Result set)**:

CustomerID	CustomerName
1	Alfreds Futterkiste
2	Ana Trujillo Emparedados y helados
3	Antonio Moreno Taquería

SELECT COMMAND...

- We can fetch **whole table** with the help of following command.

Syntax :	Example :
SELECT * **FROM** table_name;	`SELECT *` `FROM Customer;`

- **Output**:

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	5021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	5023	Mexico

SELECT DISTINCT

- When querying data from a table, you may get **duplicate rows**. To remove these duplicate rows, we use the **DISTINCT clause** in the SELECT statement.

Syntax :	Example :
SELECT DISTINCT column_name1 **FROM** table_name;	`SELECT DISTINCT Country` `FROM Customer ;`

- **Output**:

Country
Germany
Mexico
UK
Sweden
France

SELECT DISTINCT…

- You can provide multiple columns in a SELECT DISTINCT statement, it will fetch a unique combination of these columns.

Syntax :	Example :
SELECT DISTINCT column_name1 **FROM** table_name;	`SELECT DISTINCT City, Country` `FROM Customer ;`

- **Output**:

City	Country
Berlin	Germany
México D.F.	Mexico
London	UK
Luleå	Sweden
Mannheim	Germany
Strasbourg	France

WHERE CLAUSE

- The WHERE clause allows you to **specify a search condition** for the rows returned by a query.
- It is mostly with SELECT, INSERT, UPDATE and DELETE queries.

Syntax :	Example :

SELECT column_name1, column_name2, column_name3
FROM table_name
WHERE condition;
	`SELECT CustomerID, CustomerName, Country`
	`FROM Customer`
	`WHERE Country = 'Germany';`

- **Output**:

CustomerID	CustomerName	Country
1	Alfreds Futterkiste	Germany
6	Blauer See Delikatessen	Germany
17	Drachenblut Delikatessend	Germany

SQL OPERATORS

Operator	Description
=	Equal to
!=	Not equal to
<>	Not equal to
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
	Arithmetic Operators

EXAMPLES

```
#Example 1:
SELECT CustomerID, CustomerName, COUNTRY
FROM Customer
WHERE Country <> 'Brazil';

#Example 2:
SELECT ProductName, Price
FROM Products
WHERE Price >= 60;
```

SQL OPERATORS

Operator	Description
AND	TRUE if all the conditions separated by AND is TRUE
OR	TRUE if any of the conditions separated by OR is TRUE
NOT	Displays a record if the condition(s) is NOT TRUE
BETWEEN	Between a specified range of values
IN	To check for a value in a given set of items (list)
LIKE	This is used to search for a pattern in value.

EXAMPLES

```
#Example 3:
SELECT ProductName, SupplierID,  Price
FROM Products
WHERE SupplierID > 4 AND Price >=40;

#Example 4:
SELECT CustomerName, City
FROM Customer
WHERE City = 'London' OR City = 'Berlin';
```

EXAMPLES

```sql
#Example 5:
SELECT CustomerName, City
FROM Customer
WHERE NOT City = 'London';

#Example 6:
SELECT FirstName, LastName, BirthDate
FROM Employees
WHERE BirthDate BETWEEN '1952-01-01' AND '1960-01-01';
```

EXAMPLES

```sql
#Example 7:
SELECT FirstName, LastName, BirthDate
FROM Employees
WHERE BirthDate NOT BETWEEN '1952-01-01' AND '1960-01-01';

#Example 8:
SELECT ProductName, SupplierID, Price
FROM Products
WHERE  (PRICE BETWEEN 20 AND 40)
AND SupplierID = 5;
```

LIKE OPERATORS

- The LIKE operator in SQL is used with the WHERE clause to get a result set that matches the given string pattern.
- A wildcard character in SQL is used with the **LIKE** clause **to replace a single or set of characters** in any string.
- For Example(Percent and Underscore wildcards):
  - The % wildcard in SQL is used to **represent zero or more characters**.
  - The _ wildcard in SQL is used to **represent exactly one character** in a string.
  - One can use these wildcards in **combinations** too.

EXAMPLES

```sql
#Example 9:
SELECT CustomerName, City
FROM Customer
WHERE City LIKE '_erlin';

#Example 10:
SELECT CustomerName, City
FROM Customer
WHERE City LIKE 'B%';
```

EXAMPLES

```sql
#Example 11:
SELECT CustomerName, City
FROM Customer
WHERE City LIKE 'B_r%';

#Example 12:
SELECT CustomerID, CustomerName, COUNTRY
FROM Customer
WHERE Country IN ( 'Germany', 'France');
```

ORDER BY CLAUSE

When we use the **SELECT** statement to query data from a table, the order of rows in the result set is **unspecified**. **To sort the rows** in the result set, we add the ORDER BY clause.

Syntax :	Example :

SELECT column_name1, column_name2
FROM table_name
ORDER BY column_name1 **ASC/DESC**;
	`SELECT CustomerID, CustomerName, City, Country`
	`FROM Customer`
	`ORDER BY City DESC;`

```
#Example:
SELECT CustomerID, CustomerName, City, Country
FROM Customer
WHERE Country = 'France'
ORDER BY City;
```

ORDER BY CLAUSE

Que:1 Write a query to fetch information on Customers(ID, Name, City, Country) from Germany, France and Mexico. And get a result set where cities are arranged in descending order and countries in ascending.

EXAMPLES

```sql
#Solution: 1
SELECT CustomerID, CustomerName, City, Country
FROM Customer
WHERE Country IN ( 'Germany', 'France', 'Mexico')
ORDER BY City DESC, Country ASC;
```

THANK YOU