Lab 1

1️⃣ DDL – Data Definition Language

Full Form: Data Definition Language
Main Use: Defines and changes the **structure of database objects**

Commands:

- `CREATE` – Creates database objects (table, view, index)
- `ALTER` – Modifies table structure
- `DROP` – Deletes table or database permanently
- `TRUNCATE` – Removes all records from a table
- `RENAME` – Renames a table

Example:

```
CREATE TABLE Student (
    id INT,
    name VARCHAR(50)
);
```

Key Points:

- Affects **database structure**
- Changes are **permanent**
- Auto-commit is applied

Below are the **DDL commands with SYNTAX + EXAMPLE** (easy & exam-oriented):

1️⃣ CREATE

Use: Creates database objects (table, view, index)

Syntax:

```
CREATE TABLE table_name (
    column_name datatype,
    column_name datatype
);
```

Example:

```
CREATE TABLE Student (
    student_id INT,
    student_name VARCHAR(50),
    course VARCHAR(20)
);
```

2️⃣ ALTER

Use: Modifies table structure (add / modify / drop column)

Syntax (Add column):

```
ALTER TABLE table_name
ADD column_name datatype;
```

Example:

```
ALTER TABLE Student
ADD mobile_no VARCHAR(10);
```

3️⃣ DROP

Use: Deletes table or database permanently

Syntax:

```
DROP TABLE table_name;
```

Example:

```
DROP TABLE Student;
```

4️⃣ TRUNCATE

Use: Removes all records from a table (structure remains)

Syntax:

```
TRUNCATE TABLE table_name;
```

Example:

```
TRUNCATE TABLE Student;
```

5️⃣ RENAME

Use: Renames a table

Syntax:

```
RENAME TABLE old_table_name TO new_table_name;
```

Example:

```
RENAME TABLE Student TO Student_Info;
```

2 DML – Data Manipulation Language

Full Form: Data Manipulation Language
Main Use: Used to **insert, update, and delete data**

Commands:

- `INSERT` – Adds new records
- `UPDATE` – Modifies existing records
- `DELETE` – Removes records

Example:

```
INSERT INTO Student VALUES (1, 'Amit');
```

Key Points:

- Works on **data**
- Changes can be **rolled back**
- Used frequently in applications

1 INSERT

Use: Adds new records into a table

Syntax:

```
INSERT INTO table_name VALUES (value1, value2, ...);
```

Example:

```
INSERT INTO Student VALUES (1, 'Amit', 'BCA');
```

Syntax (specific columns):

```
INSERT INTO table_name (column1, column2)
VALUES (value1, value2);
```

Example:

```
INSERT INTO Student (student_id, student_name)
VALUES (2, 'Neha');
```

2 UPDATE

Use: Modifies existing records

Syntax:

```
UPDATE table_name
SET column_name = value
WHERE condition;
```

Example:

```
UPDATE Student
SET course = 'BCA'
WHERE student_id = 1;
```

⚠ **Without WHERE**, all records will be updated.

3 DELETE

Use: Removes records from a table

Syntax:

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

Example:

```
DELETE FROM Student
WHERE student_id = 2;
```

Delete all records:

```
DELETE FROM Student;
```

3 DQL – Data Query Language

Full Form: Data Query Language
Main Use: Used to **retrieve data** from database

Command:

- `SELECT`

Example:

```
SELECT * FROM Student;
```

Key Points:

- Does **not modify data**
- Used with `WHERE`, `GROUP BY`, `ORDER BY`
- Read-only operation

4 DCL – Data Control Language

Full Form: Data Control Language
Main Use: Controls **user permissions and access**

Commands:

- `GRANT` – Gives access rights
- `REVOKE` – Removes access rights

Example:

```
GRANT SELECT ON Student TO user1;
```

Key Points:

- Used by **database administrator**
- Controls security
- No effect on data values

5 TCL – Transaction Control Language

Full Form: Transaction Control Language
Main Use: Manages **database transactions**

Commands:

- `COMMIT` – Saves transaction permanently
- `ROLLBACK` – Cancels transaction
- `SAVEPOINT` – Sets a rollback point

Example:

```
ROLLBACK;
```

Key Points:

- Used with DML commands
- Ensures **data consistency**
- Supports ACID properties

Comparison Table (Full)

Language	Full Form	Main Use	Commands	Affects
DDL	Data Definition	Structure	CREATE, ALTER	Table/Schema
DML	Data Manipulation	Data change	INSERT, UPDATE	Records
DQL	Data Query	Data fetch	SELECT	Output only
DCL	Data Control	Access control	GRANT, REVOKE	Permissions
TCL	Transaction Control	Transactions	COMMIT, ROLLBACK	Data state

The SQL CREATE TABLE Statement

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

```
CREATE TABLE table_name (
```

```
    column1 datatype,
    column2 datatype,
    column3 datatype,
   ....
);
```

SQL data types can be broadly categorized into several categories:

1. Numeric data types: It stores numeric values, such as integers and floating-point numbers. Examples include `INT`, `BIGINT`, `FLOAT`, `NUMERIC`, and `DECIMAL`.
2. Character and string data types: It stores character strings, such as names and addresses. Examples include `CHAR`, `VARCHAR`, and `TEXT`.
3. Date and time data types: It stores date and time values. Examples include `DATE`, `TIME`, `DATETIME`, and `TIMESTAMP`.
4. Binary data types: It stores binary data, such as images and files. Examples include `BLOB`, `BINARY`, and `VARBINARY`.
5. Boolean data types: It stores `true`/`false` values. Examples include `BOOL` and `BOOLEAN`.
6. Enumerated data types: It stores a predefined set of values. Examples include `ENUM`.
7. Array data types: It stores multiple values of the same data type. Examples include `ARRAY`.
8. JSON data types: It stores JSON data. Examples include `JSON` and `JSONB`.

Example

SQL CREATE TABLE Example

The following example creates a table called "Persons" that contains five columns: PersonID, LastName, FirstName, Address, and City:

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

```
CREATE TABLE Customers (
  CustomerName VARCHAR(50),
  ContactName VARCHAR(50),
  Address VARCHAR(100),
  City VARCHAR(50),
```

```
    PostalCode VARCHAR(20),
    Country VARCHAR(50)
);
```

SQL INSERT INTO Statement

The SQL INSERT INTO Statement

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

INSERT INTO Syntax

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

1. Specify both the column names and the values to be inserted:

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

2. If you are adding values for all the columns of the table, you do not need to specify the column names in the SQL query. However, make sure the order of the values is in the same order as the columns in the table. Here, the `INSERT INTO` syntax would be as follows:

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

```
CREATE TABLE STUDENT (
    student_id INT,
    student_name VARCHAR(50),
    course VARCHAR(30),
    semester INT,
    gender CHAR(1),
    dob DATE,
    mobile_no VARCHAR(10)
);
```

INSERT INTO STUDENT VALUES

(1, 'Amit Kumar', 'BCA', 1, 'M', '2006-02-10', '9876543210'),

(2, 'Neha Shah', 'BCA', 1, 'F', '2006-05-18', '9123456789');

SQL SELECT Statement

The SQL SELECT Statement

The SELECT statement is used to select data from a database

exampleGet your own SQL Server

Return data from the Customers table:

```
SELECT CustomerName, City FROM Customers;
```

Syntax

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

SELECT * FROM STUDENT;

SQL SELECT DISTINCT Statement

The SQL SELECT DISTINCT Statement

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

Syntax

```
SELECT DISTINCT column1, column2, ...
```

```
FROM table_name;
```

```
SELECT DISTINCT Country FROM Customers;
```

SELECT Example Without DISTINCT

If you omit the `DISTINCT` keyword, the SQL statement returns the "Country" value from all the records of the "Customers" table:

SQL WHERE Clause

The SQL WHERE Clause

The `WHERE` clause is used to filter records.

It is used to extract only those records that fulfill a specified condition.

Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Example[Get your own SQL Server]

Select all customers from Mexico:

```
SELECT * FROM Customers
WHERE Country='Mexico';
```

Select all customers from Mexico:

```
SELECT * FROM Customers
WHERE Country='Mexico';
```

Text Fields vs. Numeric Fields

SQL requires single quotes around text values (most database systems will also allow double quotes).

However, numeric fields should not be enclosed in quotes:

Example

```
SELECT * FROM Customers
WHERE CustomerID=1;
```

Operators in The WHERE Clause

You can use other operators than the = operator to filter the search.

Example

Select all customers with a CustomerID greater than 80:

```
SELECT * FROM Customers
WHERE CustomerID > 80;
```

Operator Description	
=	Equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
<>	Not equal. **Note:** In some versions of SQL this operator may be written as !=
BETWEEN	Between a certain range
LIKE	Search for a pattern

IN	To specify multiple possible values for a column

1) Equal

```
SELECT * FROM Products
WHERE Price = 18;
```

2) Greater than

```
SELECT * FROM Products
WHERE Price > 30;
```

3) Less than

```
SELECT * FROM Products
WHERE Price < 30;
```

4) Greater than or equal

```
SELECT * FROM Products WHERE Price >= 30;
```

5) Less than or equal

```
SELECT * FROM Products WHERE Price <= 30;
```

6)Not equal. **Note:** In some versions of SQL this operator may be written as !=

```
SELECT * FROM Products WHERE Price <> 18;
```

7) Between a certain range

```
SELECT * FROM Products WHERE Price BETWEEN 50 AND 60;
```

8) Like (Search for a pattern)

```
SELECT * FROM Customers WHERE City LIKE 's%';
```

9) IN To specify multiple possible values for a column

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