SQL (Structured Query Language)

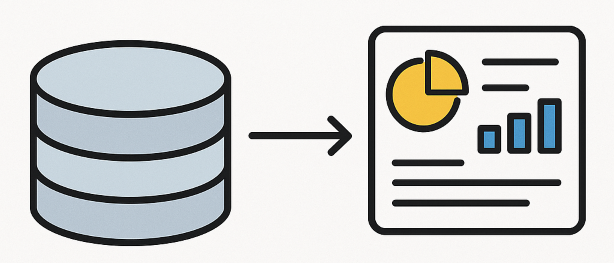
**Data**:- Data is a collection of raw facts, figures, observations, or symbols that represent information but have no meaning until they are processed.

**Examples of Data:**

* Numbers (e.g., 25, 89.5)
* Text (e.g., "Apple", "Student")
* Images (e.g., photos stored digitally)
* Videos
* Audio recordings
* Sensor readings (e.g., temperature = 35°C)
* True/False values

**WHAT IS DATABASE ?**

* This Data will stored in DATABASE.
* A database is an organised collection of data that is stored and accessed electronically from a computer system.
* A database stores data in a structured way so it can be easily accessed, managed, updated, Protected, Modified, Analysed.
* Data includes images, text, videos, Excel files, PDFs, XML files, etc.



* Every website in the world is connected to a production server that serves as a collection of databases.
* Example: Facebook needs multiple databases (servers).

**Servers and Databases:**

* **Servers** require a specific room called a server room when installed on-premises.
* **Cloud:** No physical server room needed; data is stored in data centres.

| **Server (On-premise)** | **Data centre (Cloud)** |
| --- | --- |
| Physical server room | Data stored in data centre |

**Security:** Every company or website must have its database connected and secured.

**Production Server:**

* Multiple databases are connected together in production servers.
* Example: Companies like PhonePe, IRCTC.

**Types of Data:**

* **Structured Database:**
  + Example: Numbers, tables
  + Structured Database==SQL databases ==relational DB ==RDBMS
  + One table relates to another using Primary Key and Foreign Key.
  + Structured DB- MySQL, Oracle, PostgreSQL
* **Unstructured Database:**
  + Includes images, videos, audio, streams, APIs, etc.
  + Structured Database==NoSQL databases ==non-relational DB.
  + Unstructured DB - MongoDB, Apache HBase, Cassandra
* **Vector Database:**
  + Every unstructured data converts to vectors before storing in vector DB.
  + Examples of vector databases: Pinecone, Qdrant, Weaviate, Milvus, Chromadb.
  + Used for RAG (Retrieval Augmented Generation).

**Who Uses Which Database?**

| **Role** | **Database Used** |
| --- | --- |
| Data Engineer | NoSQL DB |
| Data Scientist / Data Analyst | SQL DB |
| ML Developer / Engineer | Vector DB |

**DBMS ( Database Management System ):-**

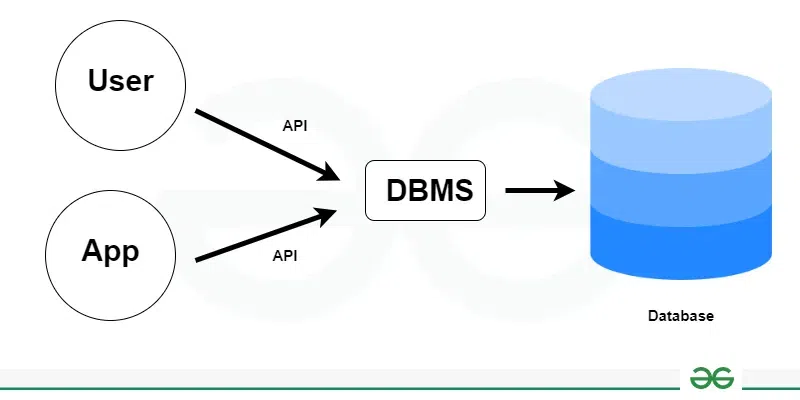
* DBMS ≠ Database
* DBMS (Database Management System) is a software tool used to pull data from the database.

**Popular DBMS Examples:**

* MySQL
* PostgreSQL
* Microsoft SQL Server
* Oracle Database
* SQLite
* MongoDB

**Functions of DBMS:**

* Retrieve data
* Create, modify, or delete databases



**EVOLUTION OF DATABASE :-**

**1. Flat File Database**

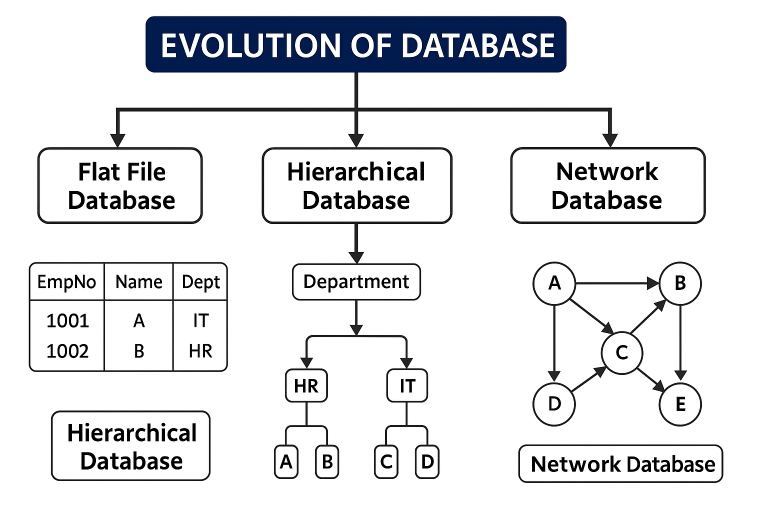
* Stores data in a simple, two-dimensional table (plain text file or spreadsheet).
* No multiple tables or complex relationships between tables.

**2. Hierarchical Database**

* Organises data in a tree-like (parent-child) structure.
* Each record/node has a single parent, except the root.
* Each parent can have multiple children.
* Structure resembles an upside-down tree or organisational chart.

**3. Network Database**

* Represents data using a network model.
* Data organised as collections of records with complex relationships.
* Unlike hierarchical DBs (single parent), each record can have multiple parents.
* Enables more flexible and complex relationships.



**Relational Database = SQL Database = Structured Database :**

* Data stored in tables.
* Each table consists of rows and columns.
* Each column has a name and data type.
* Each row is treated as a record, formed by single or multiple columns.

**Employee Table :- 🡪( Relationship) 🡪 Department Table:-**

| **EMPNO** | **EMPLNAME** | **DEPNO**  **(“**foreign key”**)** |
| --- | --- | --- |
| 1001 | Sahil | 101 |
| 1004 | Kavish | 102 |
| 1006 | Aditya | 103 |
| 1005 | Atul | 104 |

| **DEPNO**  **(“**primary key”**)** | **DNAME** | **Location** |
| --- | --- | --- |
| 101 | HR | Delhi |
| 102 | Sales | Bangalore |
| 103 | Marketing Specialist | Pune |
| 104 | Technical Engineer | Chennai |

🡪

**Relationships:**

* **One-to-One Relationship:**  
  One table relates to only one other table.
* **One-to-Many Relationship:**  
  One table relates to many records in another table.
* **Many-to-One Relationship:**  
  Many tables relate to a single table by ID.

**Non-Relational Database = NoSQL DB = Unstructured DB:**

**Types of NoSQL Databases:**

1. Key-Value Database
2. Document Database
3. Graph Database
4. Wide Column Database (Column-Family)
5. Search Engine Database
6. Time Series Database

**1. Key-Value Database**

* Examples: Redis, Amazon DynamoDB
* Data is stored as key-value pairs.

**Example:**

* + Hostname: Gonville
  + Port number: 1521

**2. Document Database**

* Examples: MongoDB, CouchDB
* Data is stored in JSON-like documents.

**Example :**

ID: iPad

{

"Type": "Tablet",

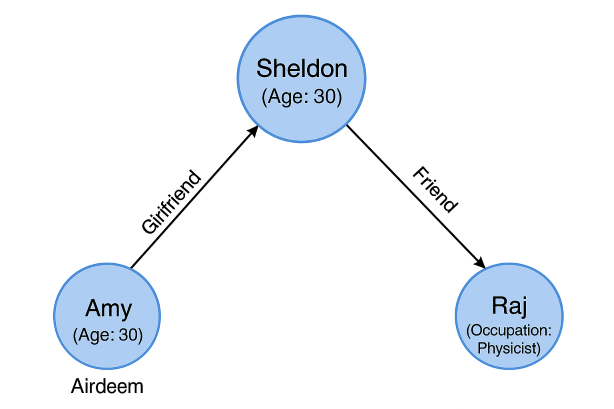
"Apps": ["Safari", "Facetime"]

}

**3. Graph Database**

* Examples: Neo4j, Amazon Neptune
* Data stored in nodes and edges representing parent-child relationships.

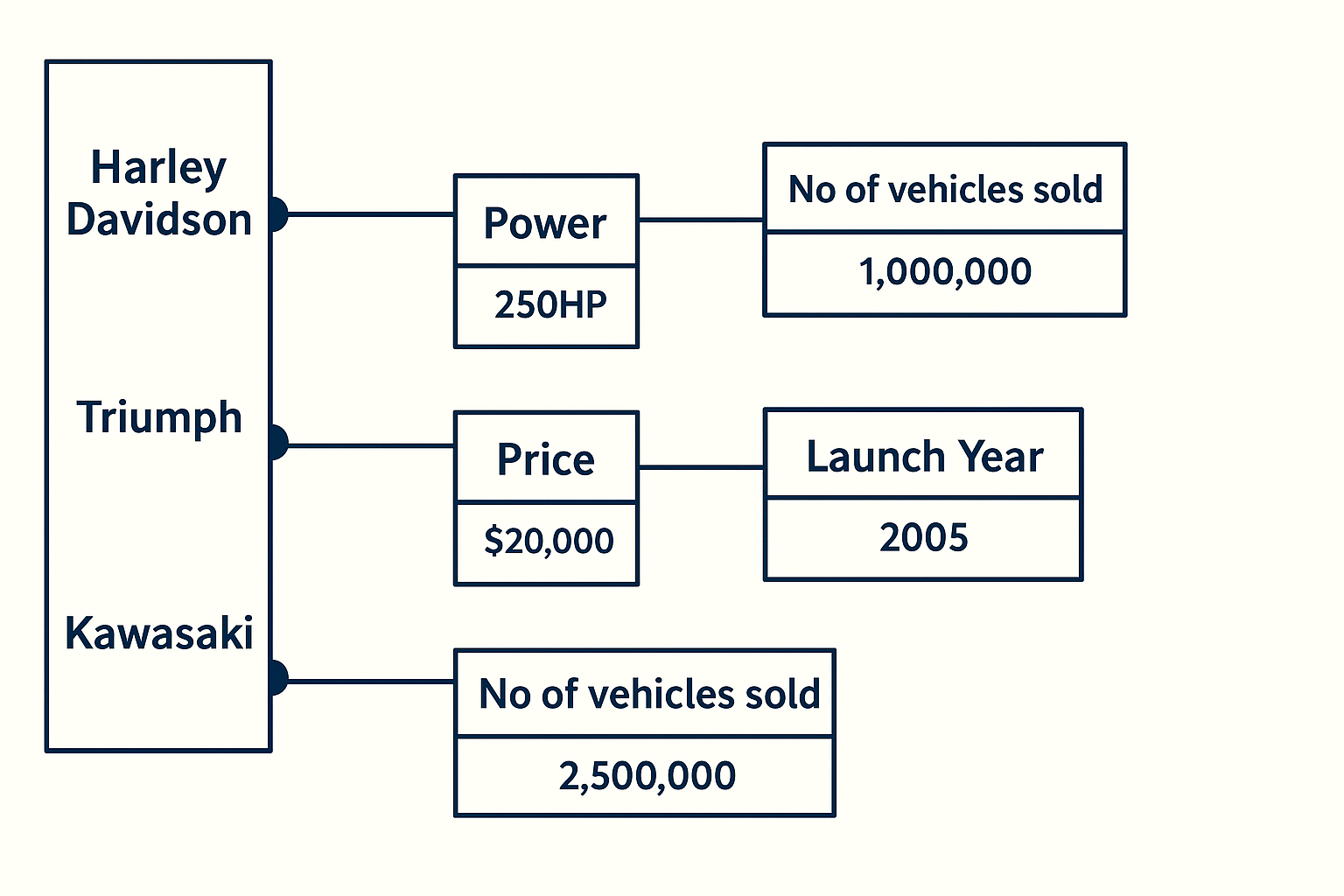
**Example:**



**4. Wide Column Database (Column-Family Store)**

* Examples: Apache Cassandra, Apache HBase
* Data is stored in columns rather than rows, suitable for large amounts of data with varying attributes.

**Example :**



**Applications of DBMS :-**

* + Banking system
  + Telecom
  + Airlines
  + Online shopping
  + Educational institutions
  + Manufacturing

**SQL Language or Types of SQL Commands :-**

1. **DDL — Data Definition Language**

* Used to define/change structure of tables
* Commands:
  + CREATE
  + ALTER
  + DROP
  + TRUNCATE

1. **DCL — Data Control Language**

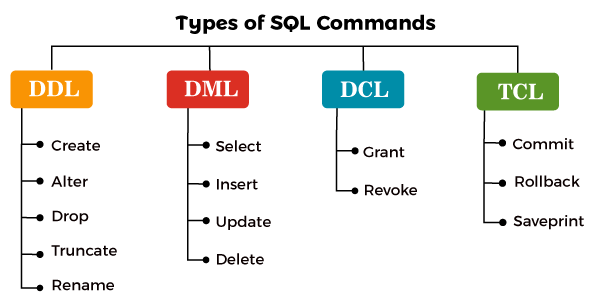
* Used to control access/permissions
* Commands:
  + GRANT
  + REVOKE

1. **DML — Data Manipulation Language**

* Used to manipulate data inside tables
* Commands:
  + SELECT
  + INSERT
  + UPDATE
  + DELETE

1. **TCL — Transaction Control Language**

* Used to manage transactions
* Commands:
  + COMMIT
  + ROLLBACK
  + SAVEPOINT
  + SET TRANSACTION



* **Most popular DB:** MySQL (widely used worldwide)
* Tools: MySQL Workbench, information\_schema, mysql, performance\_schema, sys
* **Schema:** A schema is a collection of tables & database objects

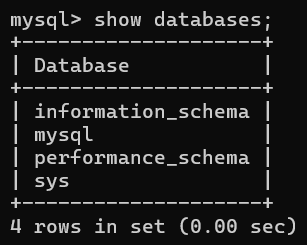
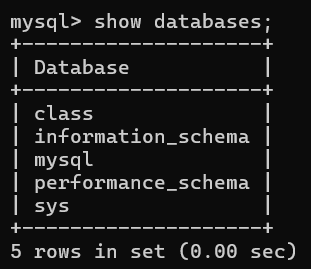
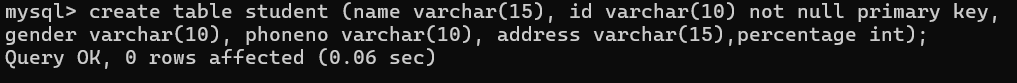
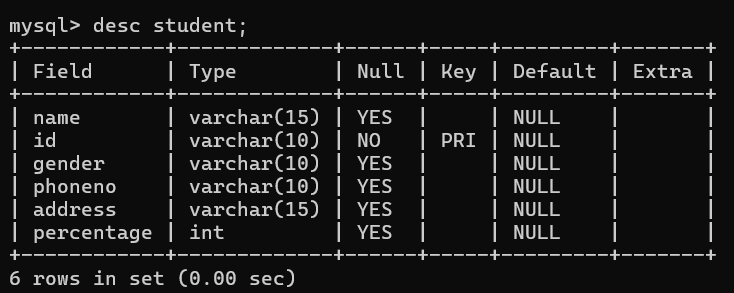
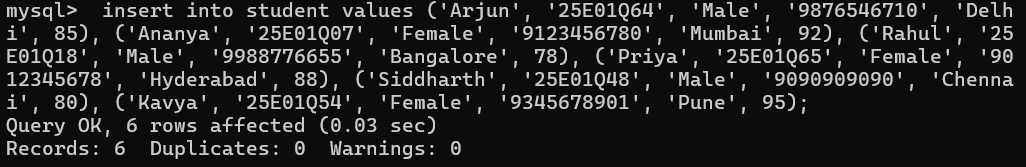
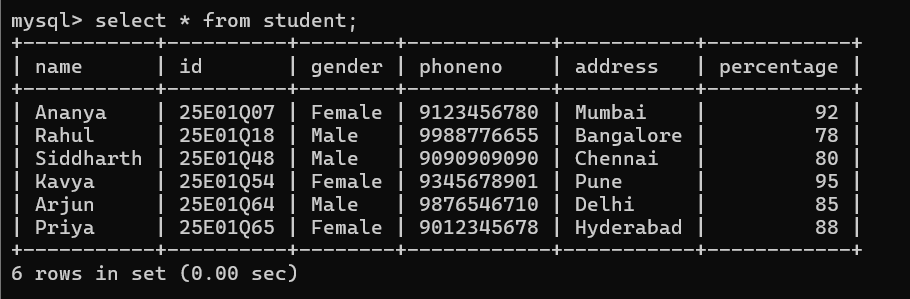
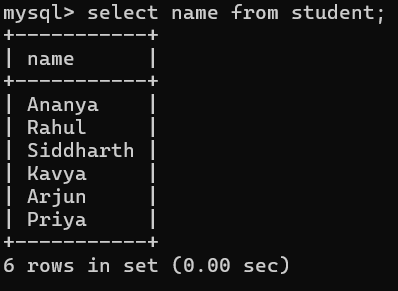
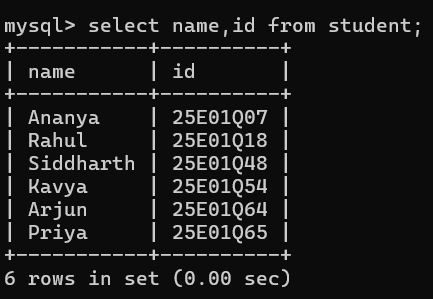
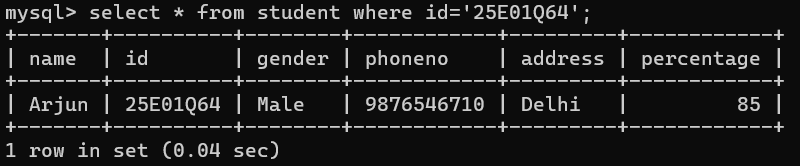
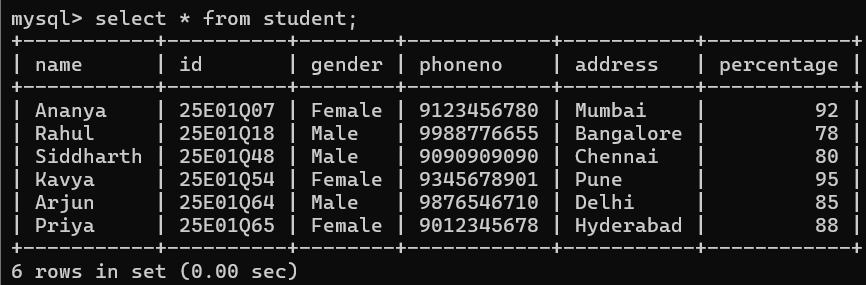
**How to Connect to SQL Server :-**

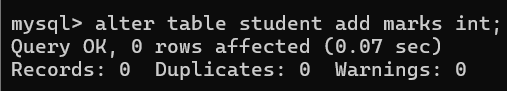
**Three ways:**

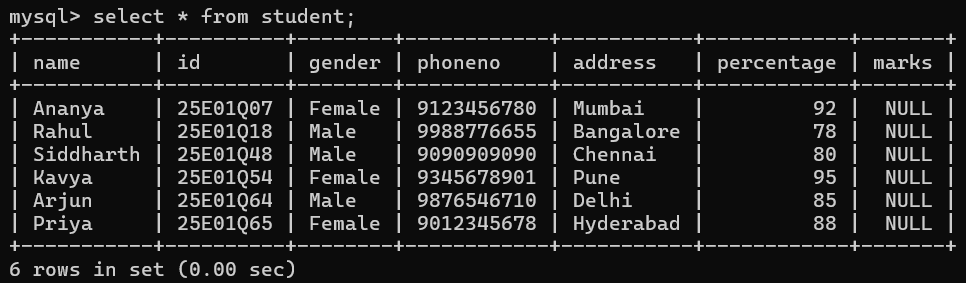
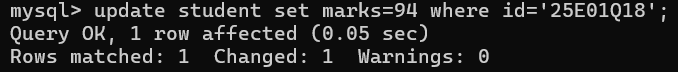
1. **Using MySQL** — enter root password.
2. **Using Workbench** — connect to the table with GUI.
3. **Using Command Prompt (CMD):**

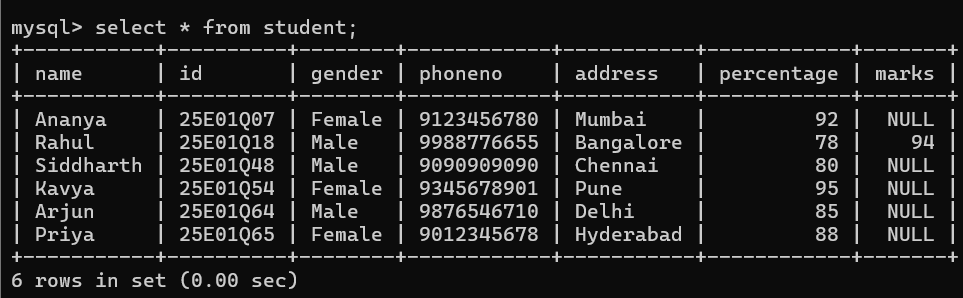
* Command: mysql -u root -p

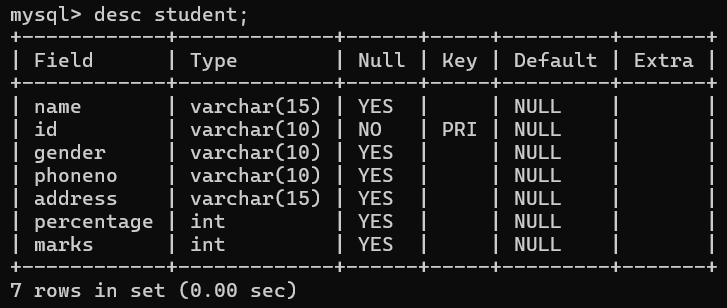
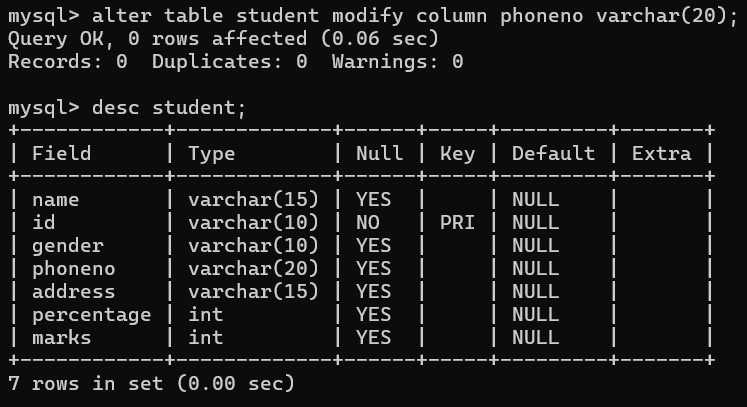
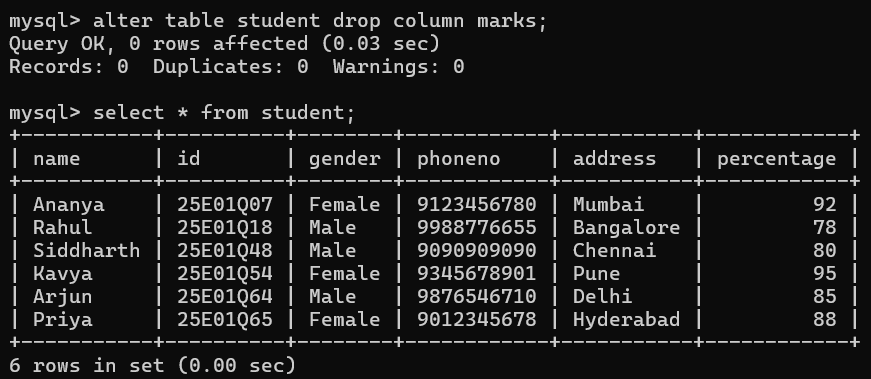
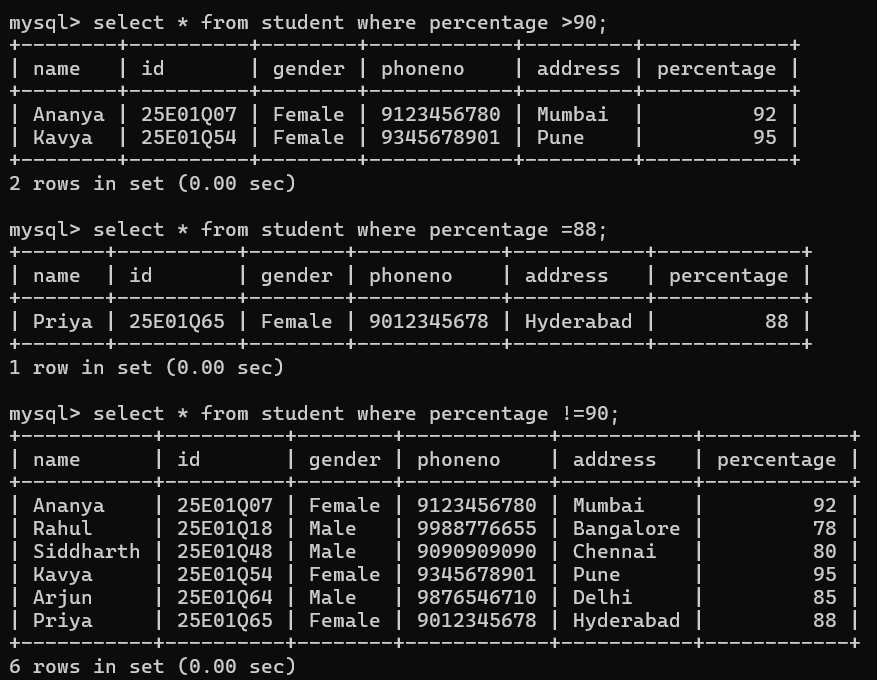
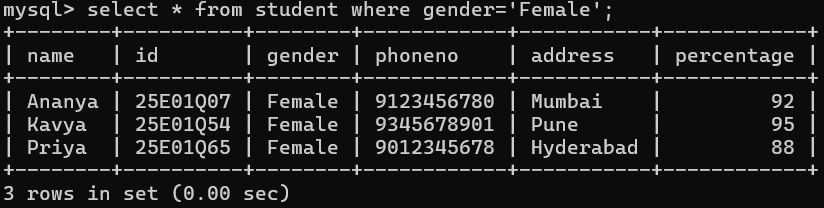
**MYSQL COMMANDS :-**

* To display a list of all databases available in MySQL server.
* Now you can create tables inside class database.
* To display a list of all databases available on server.
* To creates a table named student.
* To display the structure of your student table**.**
* To add data (records) into the table. 
* To display all the records in your student table. 
* To display only the name column from your student table.
* To display only the name and id columns from your student table.
* To display the record of the student whose id is '25E01Q64'
* To display all the records in your student table
* add a new column named marks of type INT to your existing student table.



* To display all the records in your student table.
* update the marks column to 94 for the student whose id is '25E01Q18'.
* To display all the records in your student table



* To display the structure of your student table**.**
* The phoneno column data type was changed from VARCHAR(10) to VARCHAR(20).
* To display the structure of your student table.
* The marks column was deleted from your student table.
* To display the structure of your student table.
* Shows students having percentage greater than 90.
* Shows students having exactly 88%.
* Shows students whose percentage is not equal to 90.
* To display all records where the gender is 'Female'.

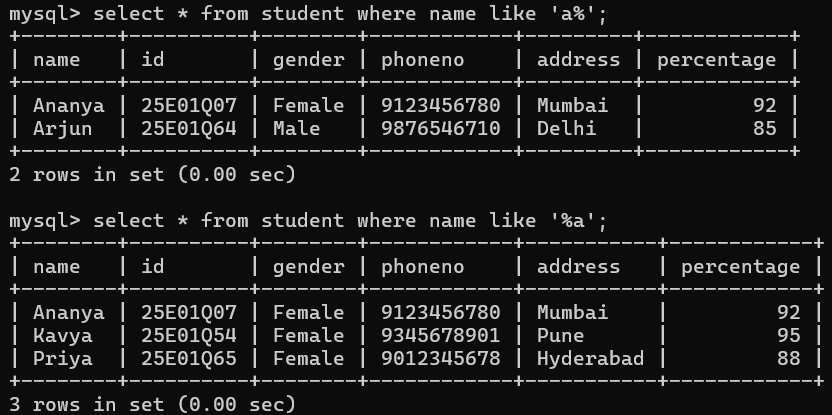
Wild Card Characters :-

* In organization records are plenty. If you want to pull out the data we will use wildcard characters.
* The LIKE operator is used to compare a value to similar values using wildcard operators to filter records based on patterns.

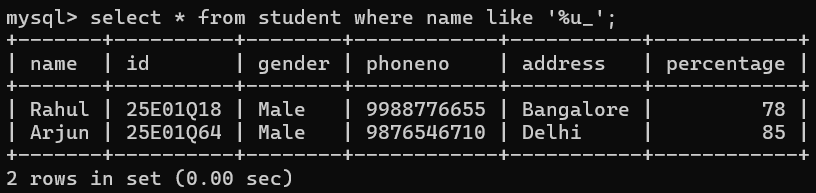
Supported Wildcard Operators :-

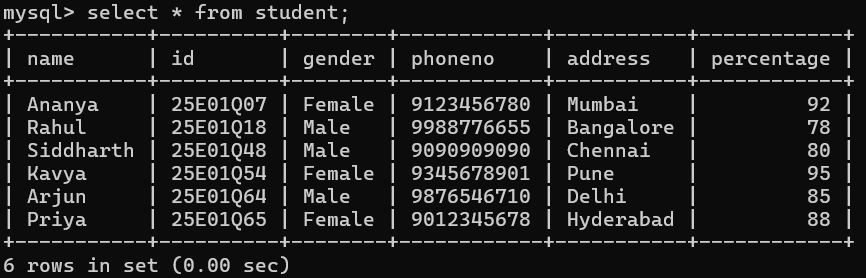
| **Wildcard** | **Description** |
| --- | --- |
| **%** | Matches **zero, one, or multiple characters**. Example: 'a%' finds any value starting with 'a'. MS Access uses \* instead of %. |
| **\_** | Matches **exactly one character**. Example: '\_a%' finds any value with 'a' as the second character. MS Access uses ? instead of \_. |

* This shows all students whose names start with 'a' or 'A' (depending on collation).
* This shows all students whose names end with 'a'.



* To display all records where the second letter of the name is 'a'.
* To display all records where the name ends with any character that has 'u'.



* To display all the records in your student table.

**SQL FUNCTIONS:**

**1. Aggregate Functions**

* SUM() – total sum of a column
* AVG() – average value
* COUNT() – number of rows
* MAX() – highest value
* MIN() – lowest value

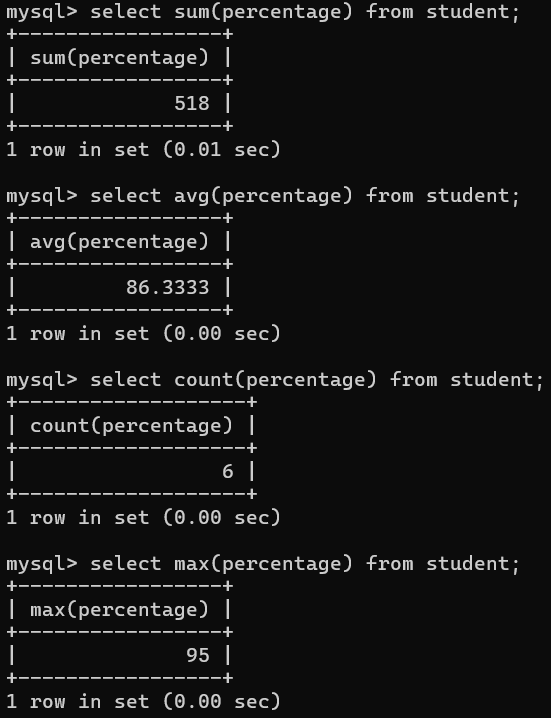
**2. String Functions**

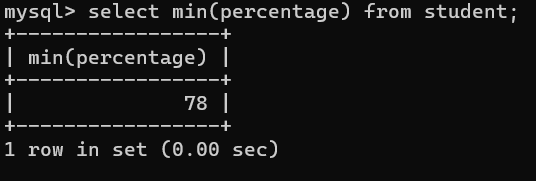
* LENGTH() – length of string
* CHAR\_LENGTH() – number of characters in string
* UPPER() / UCASE() – convert to uppercase
* LOWER() / LCASE() – convert to lowercase
* SUBSTRING() / SUBSTR() – extract part of string
* CONCAT() – combine strings
* TRIM() – remove spaces from both ends
* LTRIM() – remove spaces from left
* RTRIM() – remove spaces from right
* REPLACE() – replace part of string with another
* INSTR() – position of substring
* REVERSE() – reverses string

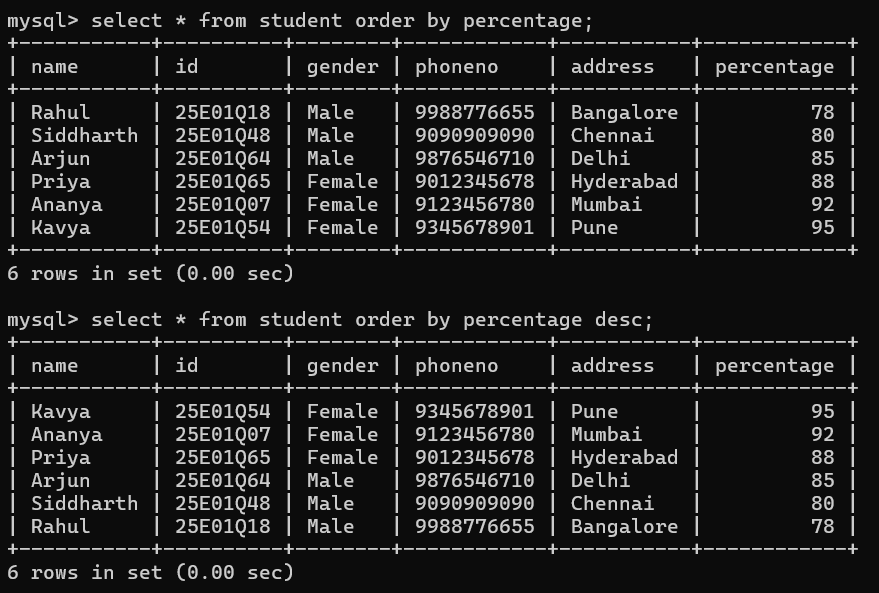
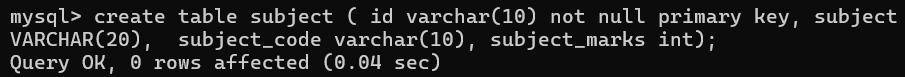
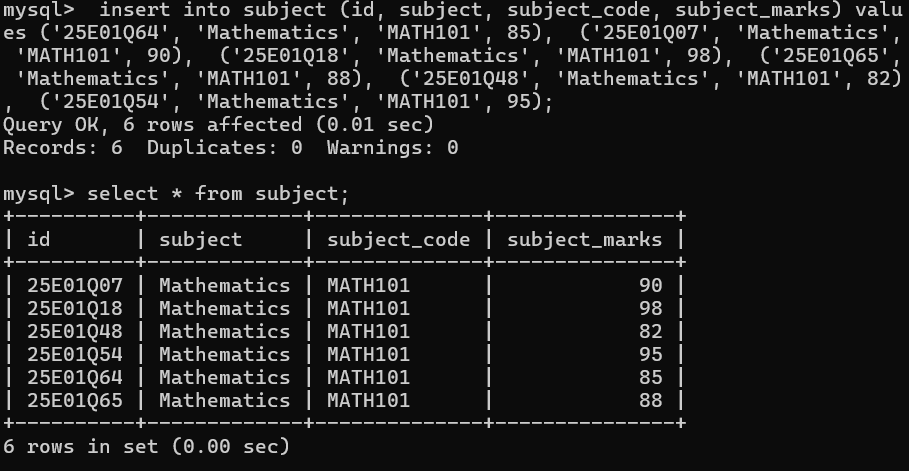
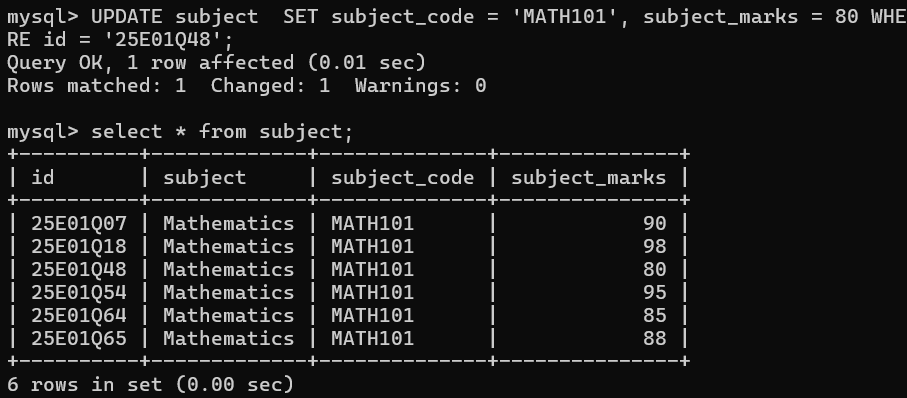
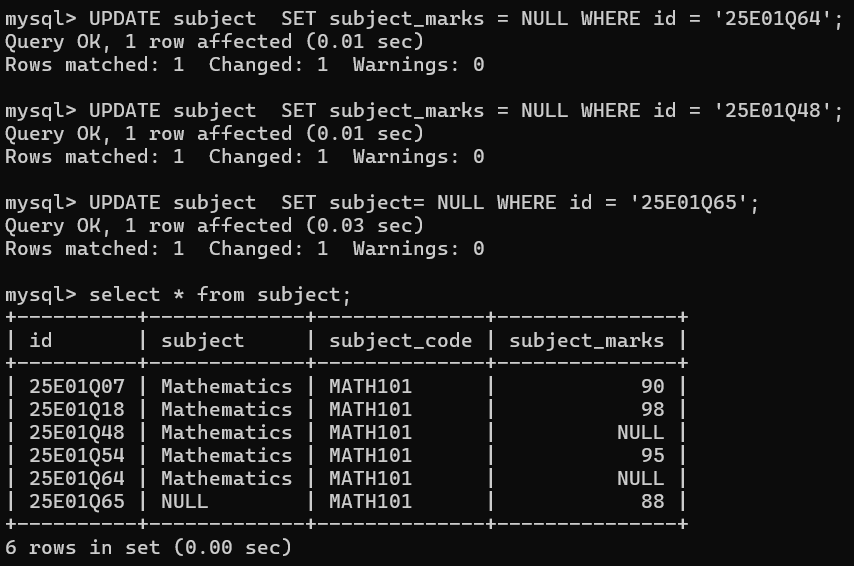
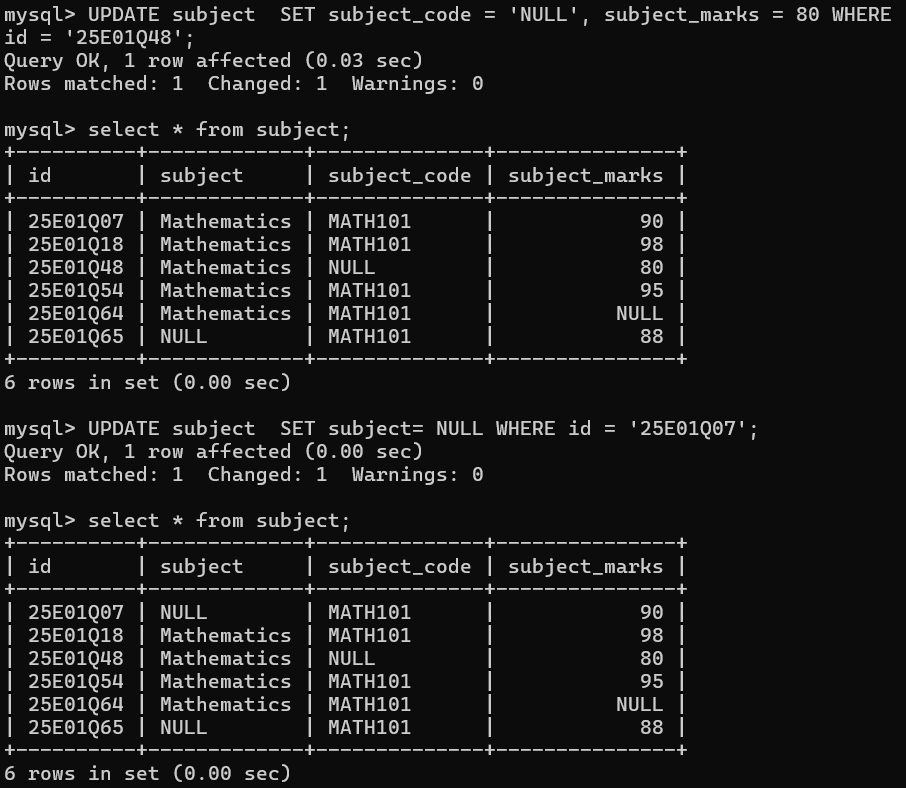
**3. Date and Time Functions**

* NOW() – current date and time
* CURDATE() – current date
* CURTIME() – current time
* DAY() – day from date
* MONTH() – month from date
* YEAR() – year from date
* DAYNAME() – name of weekday
* MONTHNAME() – name of month
* DATEDIFF() – difference between two dates
* DATE\_ADD() – add to date
* DATE\_SUB() – subtract from date
* TIME() – extract time part

**4. Mathematical Functions**

* ROUND() – rounds a number
* CEIL() / CEILING() – smallest integer >= number
* FLOOR() – largest integer <= number
* ABS() – absolute value
* MOD() – remainder
* POWER() – x to the power y
* SQRT() – square root
* EXP() – exponential value of x
* LOG() – natural log
* RAND() – random number
* Adds up all the percentage values.
* Calculates the average of all percentage values.
* Counts the number of entries in the percentage column.
* Finds the highest percentage value.the minimum (smallest) value from the percentage column in the student table



* To displays all records from the student table sorted by percentage in ascending order (lowest to highest).
* To displays all records from the student table sorted by percentage in descending order (highest to lowest).
* creates a subject table with columns for student id, subject name, subject code, and subject marks, setting id as the primary key.
* Inserted 6 rows into the subject table with student IDs, Mathematics subject, and their marks successfully.
* To display all the records in your student table.
* updated the subject\_code to 'MATH101' and subject\_marks to 80 for the student with id '25E01Q48' in the subject table.
* To display all the records in your student table.
* Sets subject\_marks to NULL for student ID 25E01Q64.
* Sets subject\_marks to NULL for student ID 25E01Q48.
* Sets subject to NULL for student ID 25E01Q65.
* To display all the records in your student table.
* set subject\_code to the string 'NULL' (not actual NULL) and subject\_marks to 80 for ID 25E01Q48.
* This sets subject column to NULL (no value) for ID 25E01Q07.
* To display all the records in your student table.

**SQL Joins Descriptions :-**

* 1. **INNER JOIN**
* Returns only matching rows from both tables.
* Excludes non-matching rows.
* Shows common data between tables.

**2. LEFT JOIN (LEFT OUTER JOIN)**

* Returns all rows from the left table.
* Returns matching rows from the right table.
* Shows NULL for right table columns if no match found.

**3. RIGHT JOIN (RIGHT OUTER JOIN)**

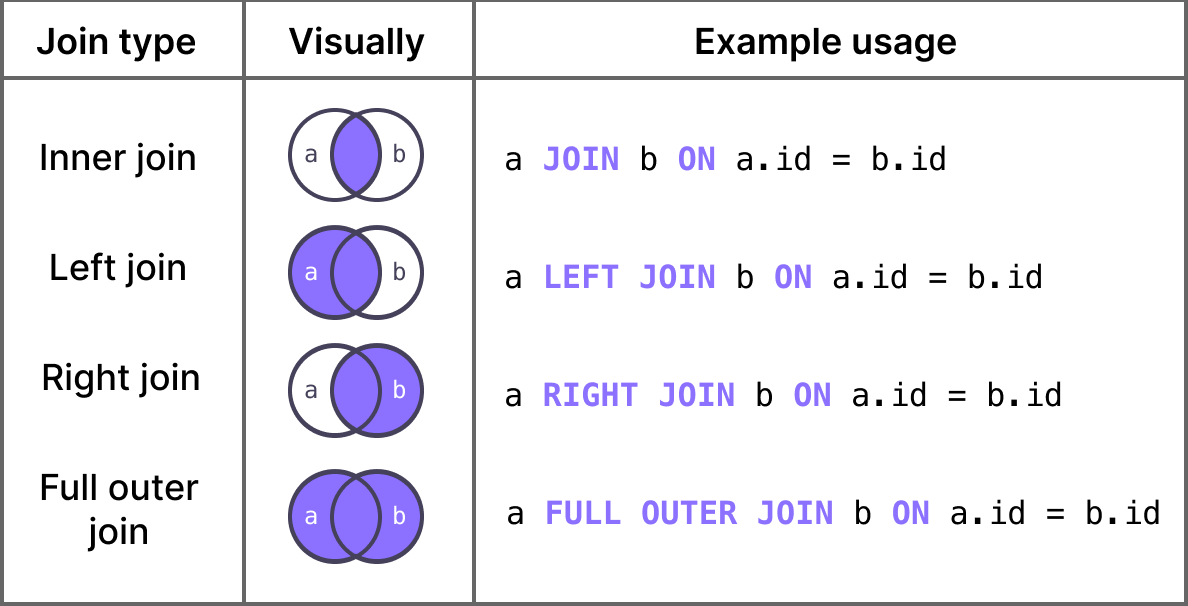
* Returns all rows from the right table.
* Returns matching rows from the left table.
* Shows NULL for left table columns if no match found.

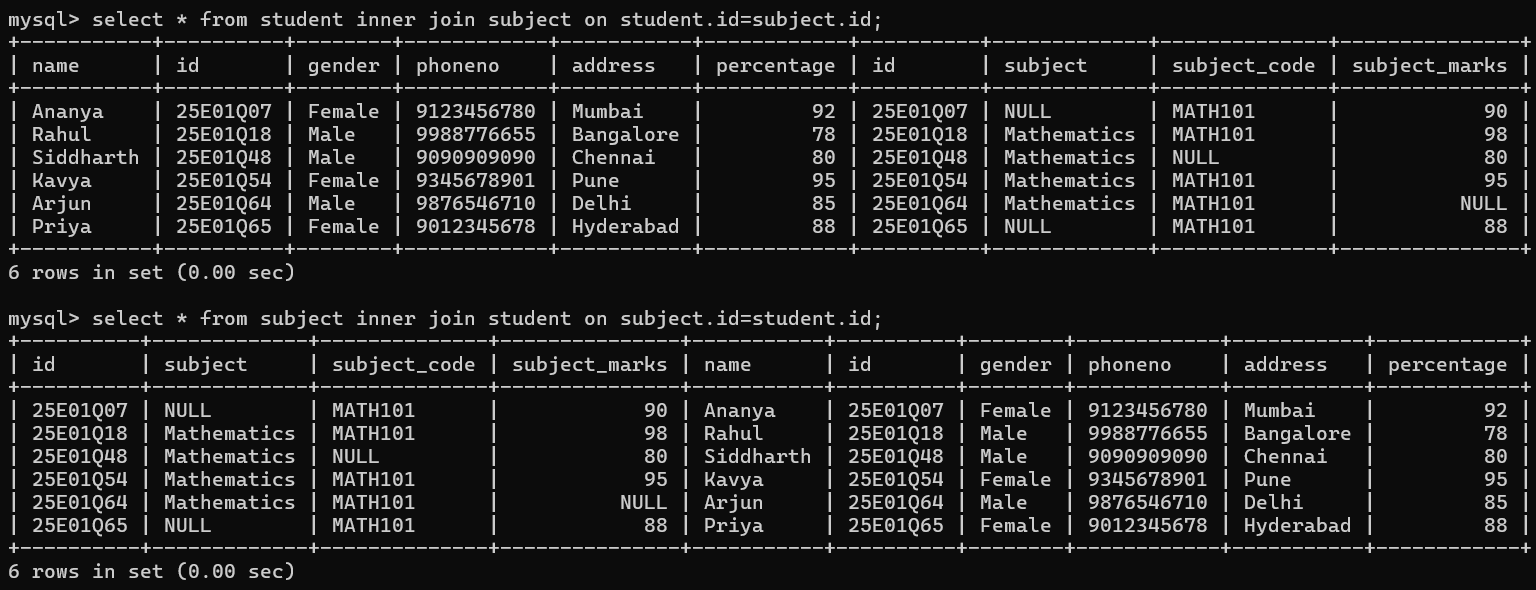
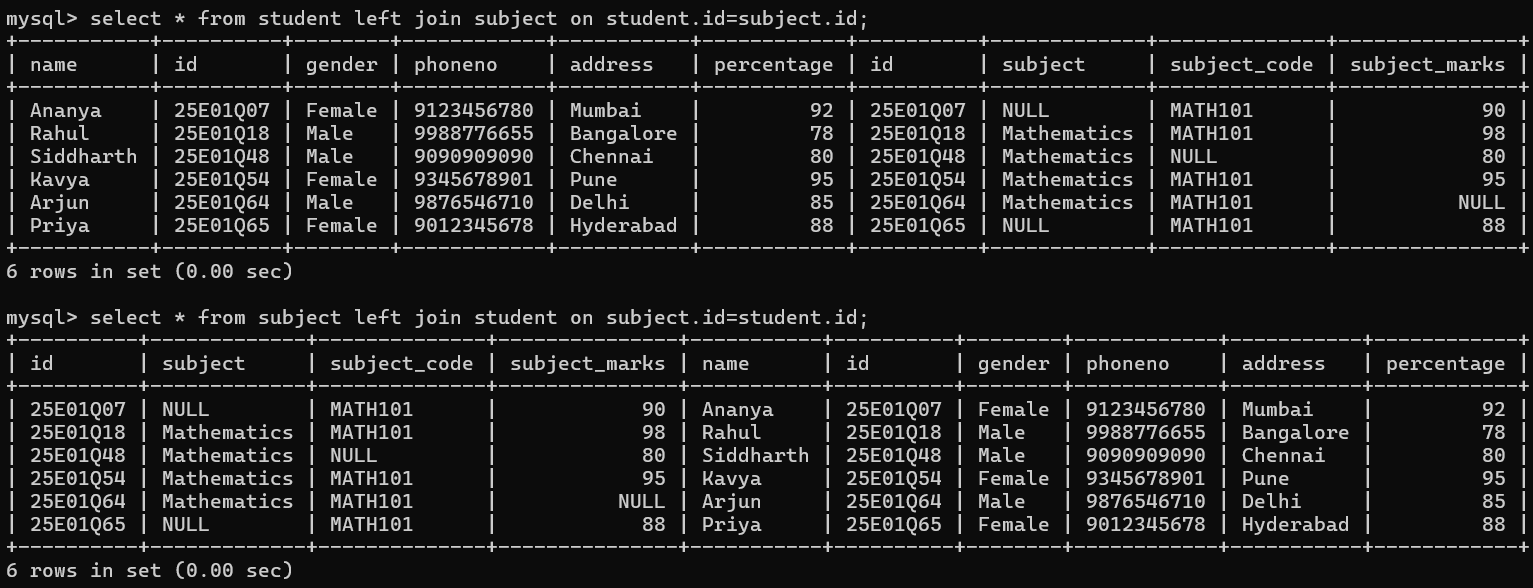
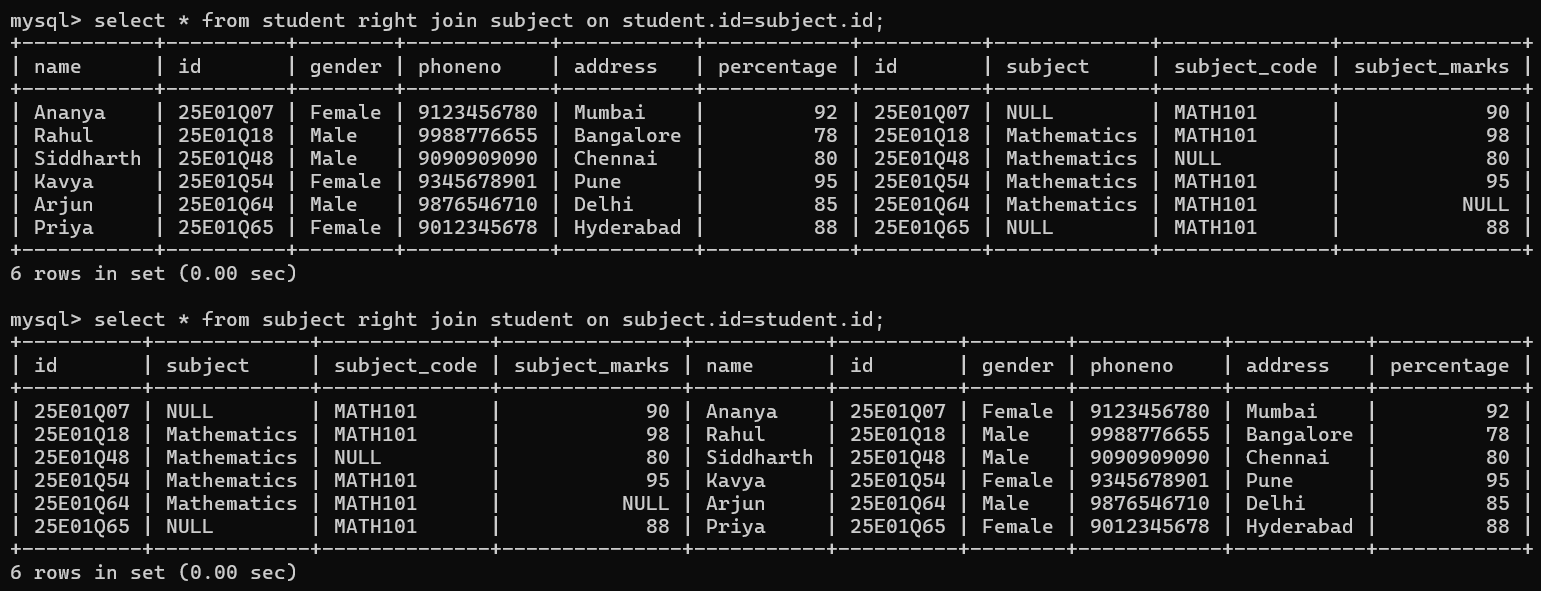
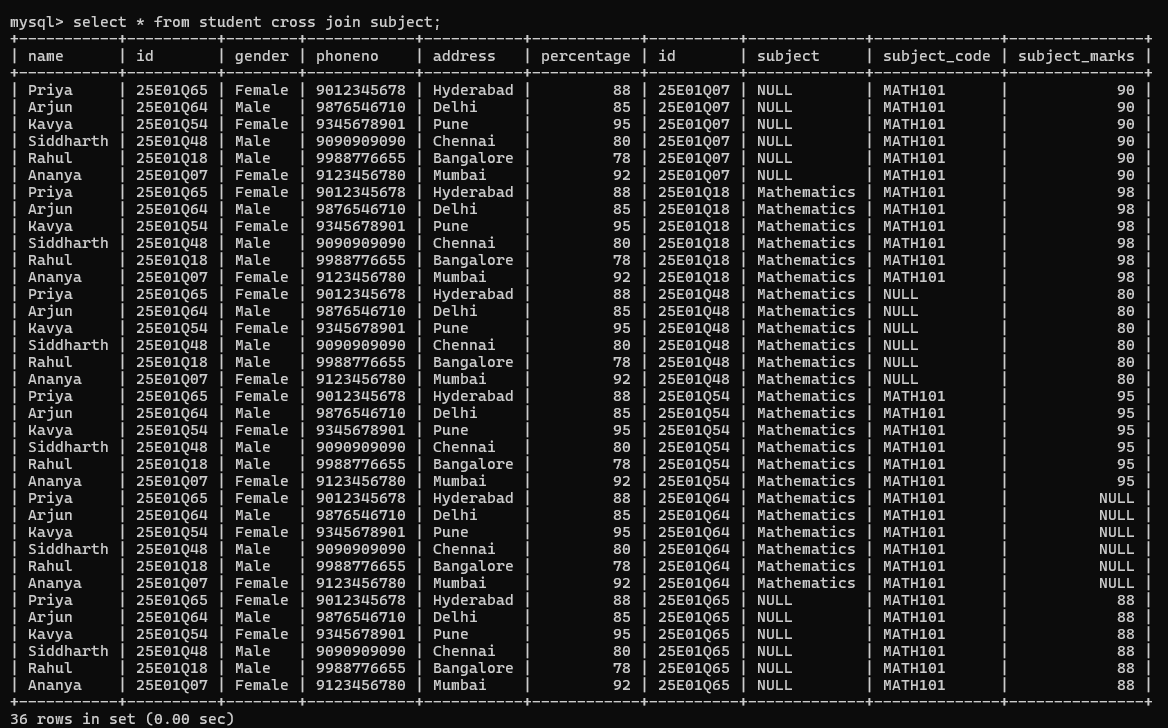
**4. FULL JOIN (FULL OUTER JOIN)**

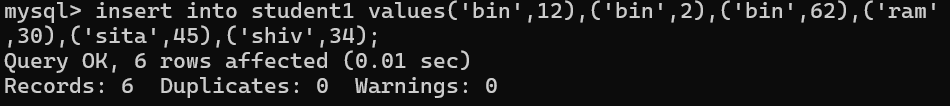
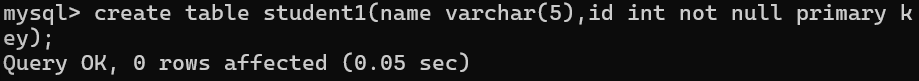
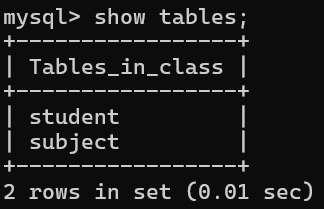
* Returns all rows from both tables.
* Combines matching rows.
* Shows NULL where there is no match in either table.

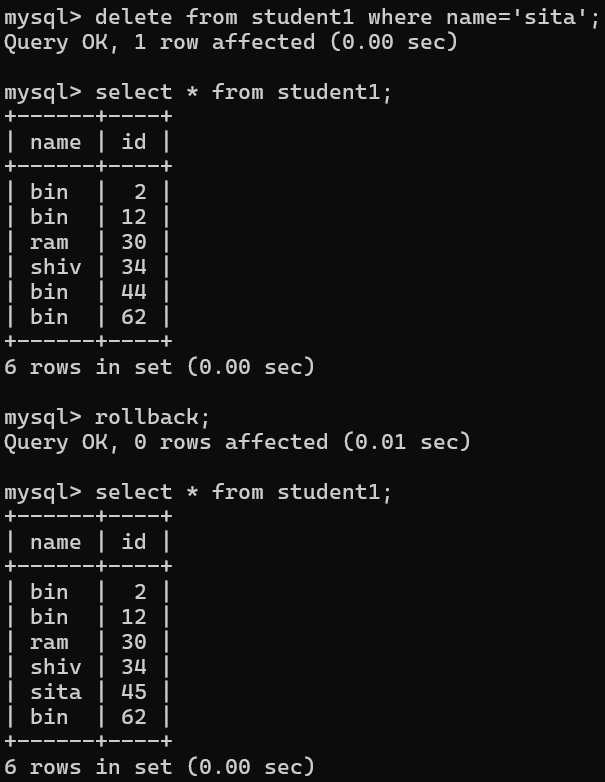
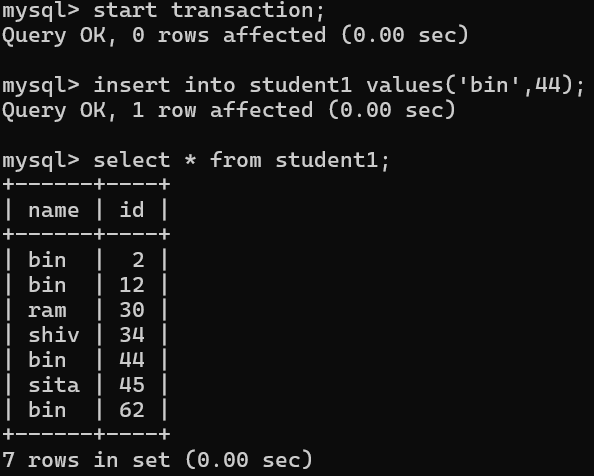
**5. CROSS JOIN**

* Returns the Cartesian product of both tables.
* Every row of first table combines with every row of second table.
* Number of rows in result = rows in table1 \* rows in table2.

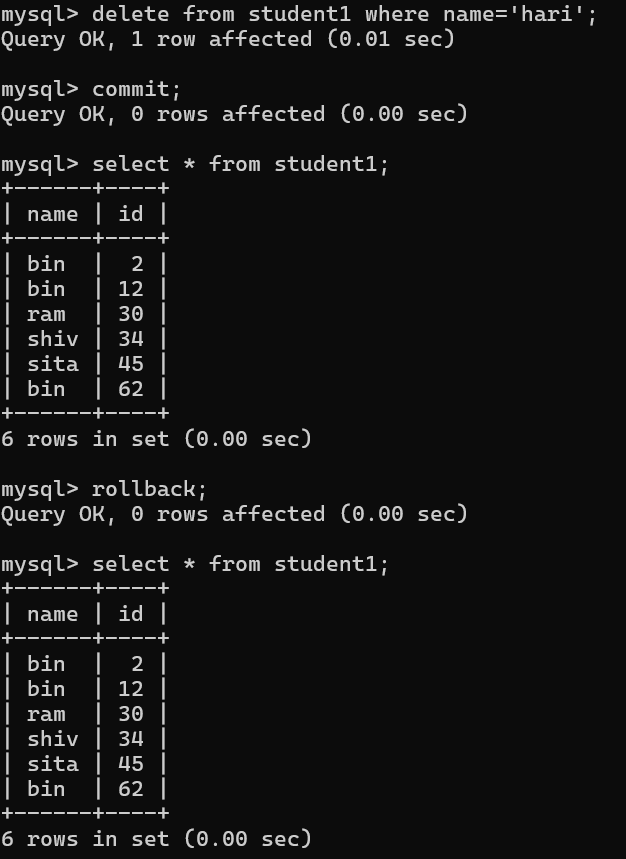
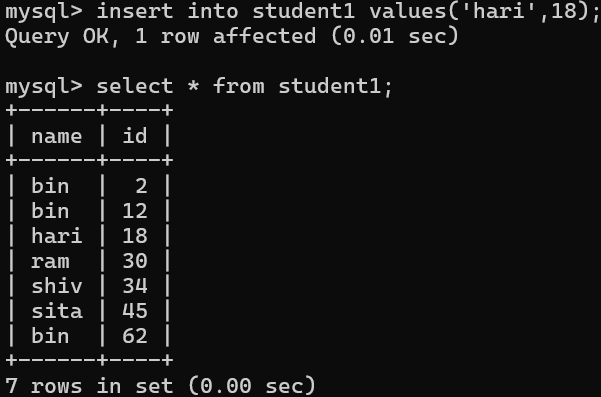


* INNER JOIN combines rows from both tables only where the id matches in both student and subject tables.
* It returns only the records with common id values in both tables.
* INNER JOIN combines rows from subject and student where subject.id = student.id.
* The output columns start with the subject table’s columns followed by student table’s columns (because subject is written first)
* LEFT JOIN returns all rows from the student table and matching rows from the subject table.
* If there is no matching id in subject, the result will show NULL for subject columns.
* LEFT JOIN returns all rows from the subject table and matching rows from the student table.
* If there is no matching id in student, the result will show NULL for student columns.
* RIGHT JOIN returns all rows from the subject table and matching rows from the student table.
* If there is no matching id in student, the result will show NULL for student columns.
* RIGHT JOIN returns all rows from the student table and matching rows from the subject table.
* If there is no matching id in subject, the result will show NULL for subject columns.
* CROSS JOIN returns the Cartesian product of the two tables.
* Every row from student is combined with every row from subject.
* Displays a list of all tables in your current database.
* Creates a new table called student1
* Inserts 6 rows into student1 table



* Transaction started. Changes will not be permanent until commit.
* Added row (bin, 44).
* display all the records in your student table
* Deleted sita.
* All changes since start transaction; are undone
* Your table reverted to its state before starting the transaction
* Transaction started.
* Added row (hari, 18).
* Table now had 7 rows, including hari.
* Deleted the row (hari, 18). Table now back to 6 rows before committing
* This saved both actions (insert and delete) permanently:
* hari was inserted
* Then hari was deleted
* So final table has no hari row.
* No effect because changes were already committed. Rollback only undoes uncommitted change.





* Displayed all tables.
* **Deleted all rows** from student1.**Table structure remains** (empty table).
* student1 still exists (but now empty).
* Result: **Empty set** (no rows).
* Deleted **entire table structure and data** permanently from the database.
* Now only **student** and **subject** remain.

