

Introduction to Databases and Basic SQL Operations

Objective:

- **Understand the fundamentals of Database, DBMS and SQL:**

- What is Databases?
- What is SQL? (Structure Query Language)?
- What is MySQL?
- Schemas and Data Relationships
- Introduction to RDBMS (Relational Database Management System)

- **Understand the Select Clause:**

- Learn to retrieve data from tables.
- Explore how you can retrieve single or multiple columns from tables.

- **Master the ORDER BY Clause:**

- Discover how to sort query results by specified columns in ascending or descending order. Understand the syntax and key points for using multiple columns in sorting.
- Implement the ORDER BY clause with practical examples.

- **Utilize the LIMIT Clause:**

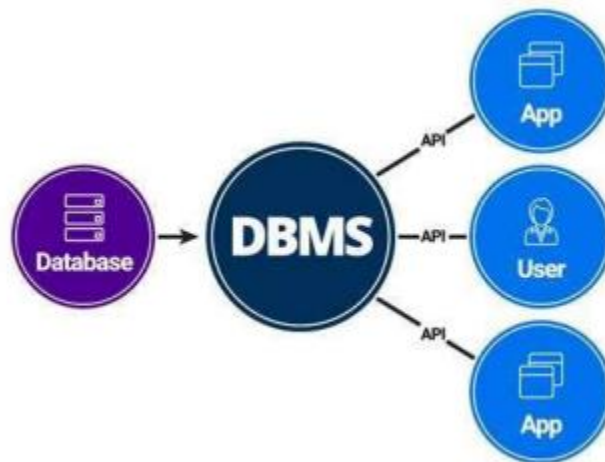
- Learn to restrict the number of rows returned by a query.
- Apply the LIMIT clause in queries to manage large datasets effectively.

1. What is a Relational Database?

A relational database is a type of database that organizes data into one or more tables (or "relations") of rows and columns. Each row represents a unique record, and each column represents a field within the record. Tables in a relational database can be linked to each other through relationships, enabling efficient data organization and retrieval.

2. What are Relational Database Management Systems (RDBMS)?

Relational Database Management Systems (RDBMS) are software tools used to create, manage, and manipulate relational databases. They provide features for storing, querying, and updating data, as well as maintaining data integrity and security. Examples of popular RDBMS include MySQL, PostgreSQL, Oracle Database, and Microsoft SQL Server.



3. What is SQL?

SQL (Structured Query Language) is a standard programming language used to communicate with relational databases. It enables users to perform various operations such as:

- Creating and modifying database structures.
- Inserting, updating, and deleting data.
- Retrieving data through queries.
- Managing access and permissions.

4. What is MySQL?

MySQL is an open-source RDBMS that uses SQL for managing databases. It is widely used for web applications, data warehousing, and logging applications. MySQL is known for its reliability, ease of use, and performance in handling large volumes of data.

5. MySQL vs Excel

Feature	MySQL	Excel
Purpose	Database management	Spreadsheet calculations
Data Size	Handles large datasets	Limited to smaller datasets
Relationships	Supports complex relationships	No native relationship support
Multi-user Access	Yes	Limited
Querying	Advanced SQL queries	Limited formula functions

6. Schemas and Data Relationships

- **Schema:** A schema in a database is the structure that defines how data is organized, including tables, columns, data types, and relationships.
- Data Relationships:
 - **One-to-One:** A single record in one table is related to a single record in another table.
 - **One-to-Many:** A single record in one table is related to multiple records in another table.
 - **Many-to-Many:** Multiple records in one table are related to multiple records in another table (achieved using a junction table).

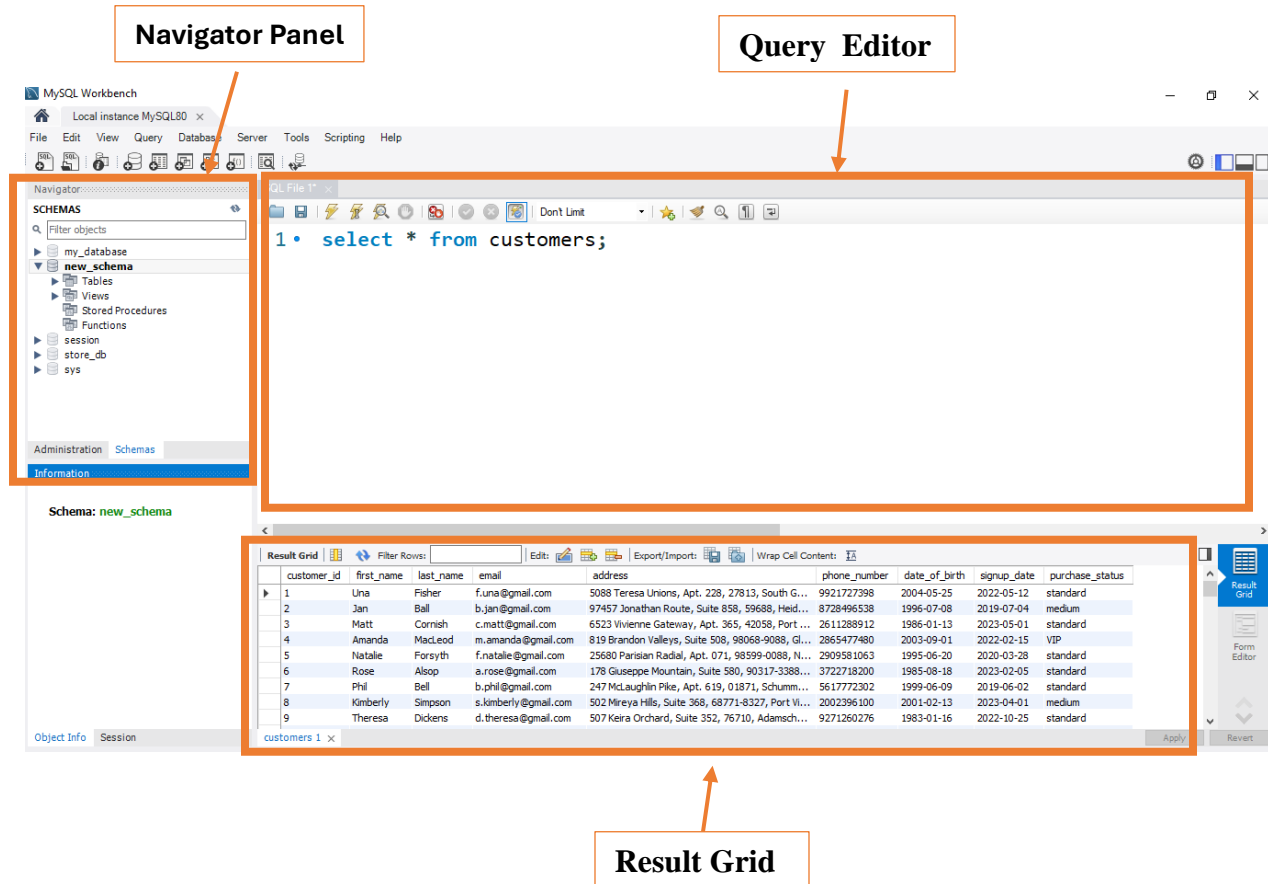
In MySQL, database and schema refers to the same thing.

Starting with MySQL Workbench

Exploring the MySQL Interface

MySQL Workbench is a graphical interface tool used for database management. Key sections of the interface include:

1. **Navigator Panel:** Displays the list of databases and their objects (tables, views, etc.).
2. **Query Editor:** A workspace for writing and executing SQL queries.
3. **Result Grid:** Displays the results of executed queries.

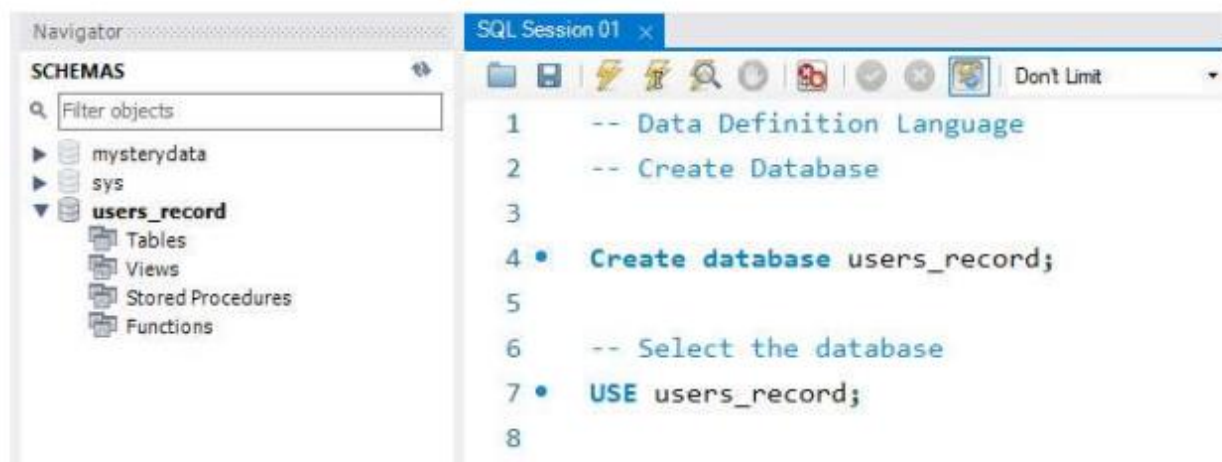


Create a new Database

Creating a new database involves defining the database's name with the keyword `CREATE DATABASE` statement.

Syntax

```
CREATE DATABASE database_name;
```



Create a new Table

Creating a new table involves defining the table's structure with the `CREATE TABLE` statement. You specify the table name, column names, and data types for each column.

Syntax

```
CREATE TABLE table_name (  
    column1 datatype constraint,  
    column2 datatype constraint,  
    column3 datatype constraint,  
    ...  
);
```

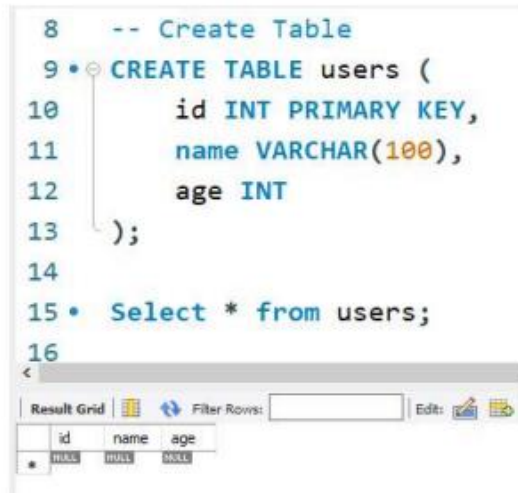
Common Data Types:

- **Numeric Types:** INT, FLOAT, DECIMAL
- **String Types:** VARCHAR, CHAR, TEXT
- **Date and Time Types:** DATE, TIME, DATETIME
- **Binary Types:** BLOB
- **Boolean Types:** BOOLEAN

Example

Let's create a table named `users` with columns for ID, name, and age.

```
-- Create Table  
CREATE TABLE users (  
    id INT PRIMARY KEY,  
    name VARCHAR(100),  
    age INT  
);
```



Inserting Values into New Tables

Once a table is created, you can insert data into it using the `INSERT INTO` statement.

Syntax

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

Example

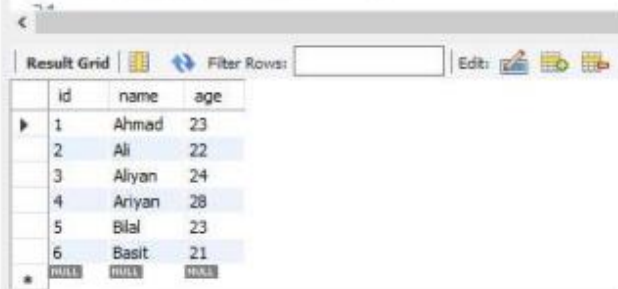
Insert a new user record into the `users` table.

```
INSERT INTO users (id, name, age)  
VALUES (1, 'Ahmad', 23);
```

Insert multiple new users records into the `users` table at once.

```
INSERT INTO users (id, name, age)  
VALUES  
(2, 'Ali', 22),  
(3, 'Aliyan', 24),  
(4, 'Ariyan', 28),  
(5, 'Bilal', 23),  
(6, 'Basit', 21);
```

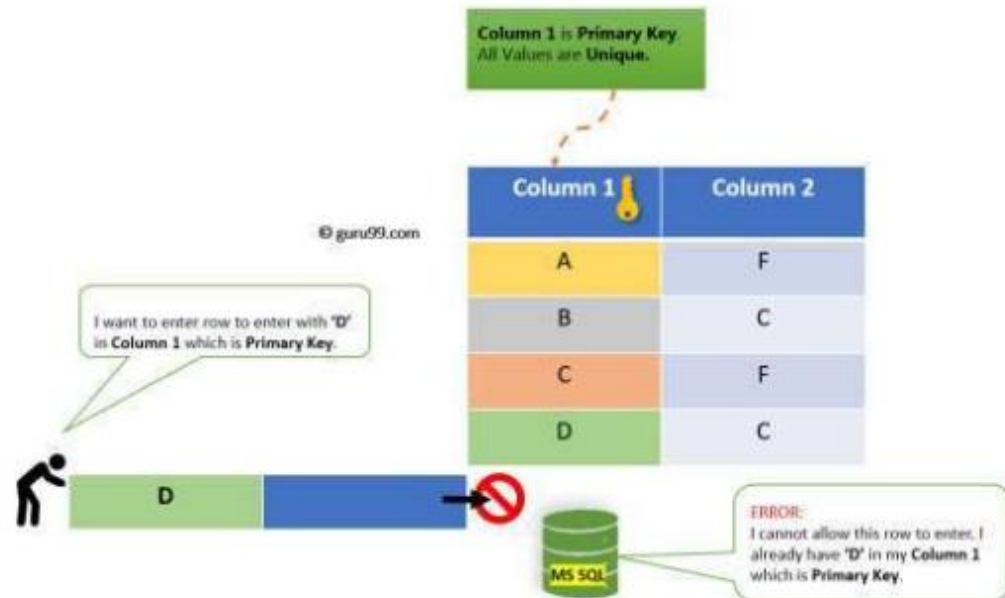
```
17 -- Insert a new record  
18 * INSERT INTO users (id, name, age)  
19 VALUES (1, 'Ahmad', 23);  
20  
21 -- Insert multiple rows at once  
22 * INSERT INTO users (id, name, age)  
23 VALUES  
24 (2, 'Ali', 22),  
25 (3, 'Aliyan', 24),  
26 (4, 'Ariyan', 28),  
27 (5, 'Bilal', 23),  
28 (6, 'Basit', 21);  
29  
30 * select * from users;
```



The screenshot shows a database interface with a SQL query editor and a result grid. The query is: `select * from users;`. The result grid displays the following data:

	id	name	age
1	1	Ahmad	23
2	2	Ali	22
3	3	Aliyan	24
4	4	Ariyan	28
5	5	Bilal	23
6	6	Basit	21
*	NULL	NULL	NULL

Make sure to enter unique ID for users. Because ID is a primary key.

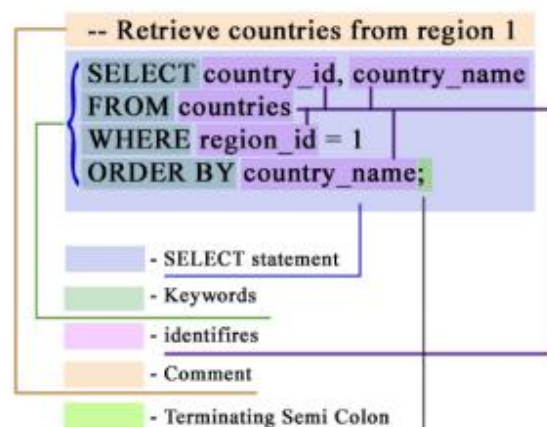


Flow of SQL Commands

SQL commands are executed in a specific order, typically:

1. **FROM:** Identify the table(s).
2. **WHERE:** Filter the rows.
3. **SELECT:** Choose the columns.
4. **ORDER BY:** Sort the results.

SQL Language Elements



1. SELECT:

The SELECT statement is used to select data from a database

Syntax:

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

Using Aliases to temporary Rename Columns:

Utilize column aliases for improved readability and clarity in results.

```
SELECT name AS 'User Name', age AS 'User Age'  
FROM users;
```



```
57 • SELECT name AS 'User Name', age AS 'User Age'  
58 FROM users;  
59
```

	User Name	User Age
▶	Ahmad	23
	Ali	22
	Aliyan	24
	Ariyan	28
	Bilal	23
	Basit	21

2. WHERE Clause:

The WHERE clause filters rows from a table based on a specified condition, allowing you to retrieve only the data that meets certain criteria.

Syntax:

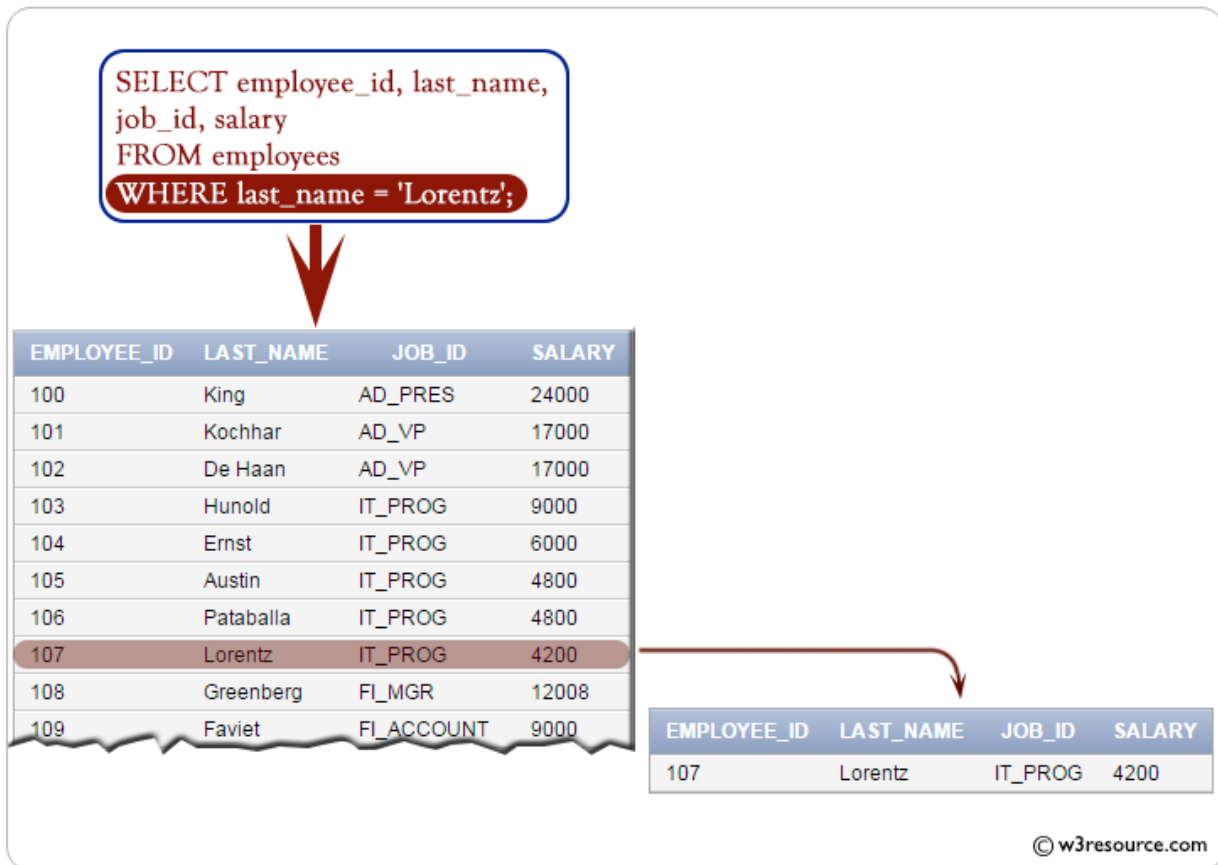
```
SELECT column1, column2, ...  
FROM table_name  
WHERE condition;
```

Example:

```
SELECT * FROM employees WHERE department = 'Sales';
```

Key Points:

- Conditions in the WHERE clause can include comparisons (=, !=, >, <, >=, <=), logical operators (AND, OR, NOT), and pattern matching using LIKE.
- Use single quotes for string values and no quotes for numeric values in conditions.



3. ORDER BY Clause:

The ORDER BY clause sorts the result set of a query based on specified columns, either in ascending or descending order.

Syntax:

SELECT column1, column2, ...

FROM table_name

ORDER BY column1 [ASC|DESC], column2 [ASC|DESC], ...;

ORDER BY Clause in SQL

EmployeeID	EmployeeLastName	EmployeeFirstName	EmailID
003	Jones	Amy	amy@gmail.com
006	Brown	Dan	dan@gmail.com
001	Donald	Jo	jo@gmail.com



```
SELECT *  
FROM Employee  
ORDER BY  
EmployeeLastName;
```



Result

EmployeeID	EmployeeLastName	EmployeeFirstName	EmailID
006	Brown	Dan	dan@gmail.com
001	Donald	Jo	jo@gmail.com
003	Jones	Amy	amy@gmail.com

www.educba.com

Example:

```
SELECT * FROM products ORDER BY price DESC;
```

Key Points:

- You can specify multiple columns for sorting, with the order of precedence determined by the sequence of columns in the ORDER BY clause.
- Use ASC for ascending order (default) and DESC for descending order.

4. LIMIT Clause:

The LIMIT clause is used to restrict the number of rows returned by a query, which is particularly useful when dealing with large datasets.

Syntax:

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

```
FROM table_name
```

```
LIMIT number_of_rows;
```

Example:

SELECT * FROM customers LIMIT 10;

Key Points:

- The LIMIT clause is not supported by all SQL databases, so check the documentation of your specific database management system (DBMS) for compatibility.
- LIMIT is often used in combination with ORDER BY to retrieve the top or bottom N records based on a specified criterion.

Table: Customers

customer_id	first_name	last_name	age	country
1	John	Doe	31	USA
2	Robert	Luna	22	USA
3	David	Robinson	22	UK
4	John	Reinhardt	25	UK
5	Betty	Doe	28	UAE

SELECT first_name, last_name
FROM Customers
LIMIT 2

first_name	last_name
John	Reinhardt
Betty	Doe