INFOMAX
COMPUTER ACADEMY

MySQL®

What is MySQL?

- MySQL is an open-source relational database management system (RDBMS) developed by Oracle Corporation.
- It's widely used for managing large databases in web applications, powering many popular websites and platforms.
- MySQL follows the relational model and uses SQL (Structured Query Language) for querying and managing data.

Key Features:

- **Scalability:** It's scalable and can handle large amounts of data efficiently.
- **Performance:** Known for its high performance, MySQL is optimized for speed and reliability.
- **Security:** Offers various security features such as encryption, authentication, and access controls.
- **Replication:** Supports replication for distributing data across multiple servers for redundancy and performance.
- **Transactions:** Provides ACID (Atomicity, Consistency, Isolation, Durability) compliance for ensuring data integrity.
- **Stored Procedures and Triggers:** Allows developers to create stored procedures and triggers for automating tasks and enforcing business rules.

Basic Concepts of MySQL

- **Database:** A container for tables and other database objects.
- **Table:** A collection of related data organized in rows and columns.
- **Column:** Represents a specific attribute or field in a table.
- **Row:** A single record in a table containing data for each column.
- **Primary Key:** A unique identifier for each row in a table.
- **Foreign Key:** A column that establishes a link between two tables, enforcing referential integrity.
- **Index:** Data structure used for improving the speed of data retrieval operations on a table.
- **Query:** A request for data or information from a database using SQL statements.

Categories of MySQL Commands

- MySQL commands can be categorized based on their functionality and the operations they perform. Here's a breakdown of the categories of MySQL commands:

  - Data Definition Language (DDL) Commands:
  - Data Manipulation Language (DML) Commands :
  - Data Query Language (DQL) Commands:
  - Data Control Language (DCL) Commands:

Data Definition Language (DDL) Commands:

- **CREATE:** Used to create databases, tables, indexes, views, procedures, and other database objects.
- **ALTER:** Allows modifications to the structure of existing database objects, such as tables, indexes, and columns.
- **DROP:** Deletes databases, tables, indexes, views, procedures, and other database objects.
- **TRUNCATE:** Removes all records from a table, but retains the table structure.
- **RENAME:** Renames a table.

Data Manipulation Language (DML) Commands :

- **INSERT:** Adds new records (rows) into a table.
- **SELECT**: Retrieves data from one or more tables based on specified criteria.
- **UPDATE**: Modifies existing records in a table based on specified criteria.
- **DELETE**: Removes records from a table based on specified criteria.

Data Query Language (DQL) Commands:

- **SELECT**: Primarily used for querying data from one or more tables in a database.

Data Control Language (DCL) Commands:

- **GRANT**: Provides specific privileges to database users or roles.
- **REVOKE**: Removes previously granted privileges from database users or roles.

Data base Operations

- Show the list of Exiting database:
  - *Show databases;*
- Create new Database:
  - Syntax: *Create database <database name>*
  - Example: *create database abc_store;*
- Delete an exiting database:
  - Syntax: *drop database <database name>*
  - Example: *drop database abs_store;*
- Select or Activate the database:
  - Syntax: *use <database name>*
  - example: *use abc_store*

Working with Tables

- Show the list of table in selected databse:
  - *Show tables*
- Create a new table:
  - Syntax: *create table <table_name>(<field_name data_type(size)>, <field_name data_type(size)>,……);*
  - Example: *create table product(pro_id int,pro_name varchar(50),pro_price float,pro_dom date);*
- *Delete Exiting Table:*
  - *Syntax: drop table <table_name>*
  - *Example: drop table product;*

Data type in MySQL

- Numeric Data Types:
  - ***INT***: A whole number that can be signed or unsigned. It's commonly used for storing integers.
  - ***FLOAT***: A floating-point number with single precision.
  - ***DOUBLE***: A floating-point number with double precision.
- String Data Types:
  - ***CHAR(n):*** Fixed-length string with a maximum length of n characters.
  - ***VARCHAR(n):*** Variable-length string with a maximum length of n characters.
  - ***TEXT:*** Variable-length string with a maximum length of 65,535 characters.

Data type in MySQL (cont..)

- Date and Time Data Types:
  - ***DATE:*** Date value in the format 'YYYY-MM-DD'.
  - ***TIME:*** Time value in the format 'HH:MM:SS'.
  - ***DATETIME:*** Combination of date and time values in the format 'YYYY-MM-DD HH:MM:SS'.
  - ***TIMESTAMP:*** A timestamp representing a specific moment in time. It's generally used to record when a row was inserted or updated and is stored in the format 'YYYY-MM-DD HH:MM:SS'.
  - ***YEAR:*** A four-digit year value in the format 'YYYY'.

Data type in MySQL (cont..)

- Miscellaneous Data Types
  - ***BOOLEAN / BOOL:*** Synonyms for TINYINT(1), used to represent boolean values (0 or 1).
  - ***ENUM(value1, value2, ...):*** A list of enumerated values, where each column can have one of the specified values.
  - ***SET(value1, value2, ...):*** A set of enumerated values, where each column can have zero or more of the specified values.
  - ***BLOB:*** Binary large object for storing large binary data.

DESCRIBE Statement

- The DESCRIBE statement is used to display the structure of a table, including column names, data types, and constraints.

- Syntax: *DESCRIBE <table_name>;*
  or
  desc <table_name>;
- Example: *desc product or describe product*

Create table with constrain

Syntax:

```
CREATE TABLE table_name (
    column1 data_type constraint,
    column2 data_type constraint,
    ...
    constraint_name CONSTRAINT constraint_type (column_name(s)),
    ...
);
```

- Example:

Create table employee (emp_id int primary key, emp_name varchar(50) not null, emp_mobile varchar(10) unique not null, emp_salary float default 50000);

Constraints in MySQL

Constraints are rules defined on columns in a table that enforce data integrity and ensure that data follows certain rules or conditions.

- ***Primary Key Constraint***:
  - Ensures that each row in a table is uniquely identified by a specific column or combination of columns.
  - Only one primary key constraint is allowed per table.
- ***Foreign Key Constraint:***
  - Establishes a relationship between two tables by enforcing referential integrity. Ensures that values in a column (or columns) of one table match values in another table's primary key or unique key column(s).
  - Helps maintain data consistency by preventing actions that would violate the integrity of the relationship.

Constraints in MySQL (Conti..)

- ***Unique Constraint***:
  - Ensures that all values in a column (or columns) are unique (no duplicate values).
  - Unlike the primary key constraint, unique constraints allow NULL values, but only one NULL value is allowed.
- ***Check Constraint:***
  - Specifies a condition that must be satisfied by the values in a column.
  - Prevents insertion or updating of data that does not meet the specified condition.
- ***Not Null Constraint:***
  - Ensures that a column does not contain NULL values.
  - Requires every row to have a value in the specified column.

Example : Table with constrains

```
CREATE TABLE employees (
    employee_id INT PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    email VARCHAR(100) UNIQUE,
    department_id INT,
    CONSTRAINT fk_department FOREIGN KEY (department_id)
REFERENCES departments(department_id)
);
```

AUTO_INCEREMENT Keyword

- Auto-increment allows a unique number to be generated automatically when a new record is inserted into a table.
- Example:

```
CREATE TABLE product (
    pro_id int PRIMARY KEY AUTO_INCREMENT,
    pro_name varchar(255) NOT NULL,
    pro_price float,
    pro_dom date);
```

Insert data in table

- Syntax:

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

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

- Example:

Insert into product values (1,'printer',18900,'2020-12-23')

Insert into product (pro_id,pro_name) values (2,'Keyboard');

Display data from table

- Syntax:

  *Select * from <table_name>*

  select <field_name>,<field_name>, .. From <table name>

- Example:

  *Select * from product*

  select pro_name,pro_price,pro_dom from product;

Column Aliasing

- Aliases are used to give a table, or a column in a table, a temporary name. Aliases are often used to make column names more readable.
  - Syntax:

    SELECT column_name AS alias_name FROM table_name;
  - Example:

    SELECT CustomerID AS ID, CustomerName AS Customer FROM Customers;

Sort data from table (order by clause)

- Syntax:

```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1, column2, ... ASC|DESC;
```

- Example:

```
Select * from product order by pro_price;
Select * from product order by pro_price desc;
Select * from product order by pro_dom,pro_price;
```

Display distinct (unique) data from table

- Syntax:

  Select distinct <field_name> from <table_name>;

- Example:

  select distinct city from student;

MySQL LIMIT Clause

- Syntax:

  SELECT column_name(s) FROM table_name
  WHERE condition LIMIT number;

- Example:

  *Select * from product limit 10;*
  *Select * from product order by pro_dom desc limit 10;*

Display data based on conditions

- Syntax:
  - Select * from <table_name> where <condition>
- Example:
  - Select * from product where pro_price<4000;
  - Select * from product where pro_name='printer';
  - Select * from product where pro_dom<'2000-12-23''

Operators in Where clause

- = (Equal to):
  - SELECT * FROM employees WHERE department_id = 101;
- != or <> (Not equal to)
  - SELECT * FROM employees WHERE department_id != 101;
- < (Less than):
  - SELECT * FROM employees WHERE salary < 50000;
- > (Greater than)
  - SELECT * FROM employees WHERE hire_date > '2020-01-01';
- <= (Less than or equal to):
  - SELECT * FROM employees WHERE age <= 30;

Operators in Where clause…..

- >= (Greater than or equal to):
  - SELECT * FROM employees WHERE experience >= 5;
- Logical Operators:
  - AND:
    - SELECT * FROM employees WHERE department_id = 101 AND salary > 50000;
  - OR:
    - SELECT * FROM employees WHERE department_id = 101 OR department_id = 102;
  - NOT:
    - SELECT * FROM employees WHERE NOT department_id = 101;

Operators in Where clause……..

- IN Operator:
  - SELECT * FROM employees WHERE department_id IN (101, 102);
  - SELECT * FROM employees WHERE department_id NOT IN (101, 102);
- BETWEEN Operator:
  - SELECT * FROM employees WHERE salary BETWEEN 50000 AND 80000;
  - SELECT * FROM employees WHERE salary NOT BETWEEN 50000 AND 80000;

Operators in Where clause……..

- LIKE Operator:
  - Matches rows where a column value matches a specified pattern.
  - % represents zero or more characters, and _ represents a single character.
- SELECT * FROM employees WHERE last_name LIKE 'S%’;
- SELECT * FROM employees WHERE last_name LIKE ‘%a';
- SELECT * FROM employees WHERE last_name LIKE ‘%a%';
- SELECT * FROM employees WHERE last_name LIKE ‘_u%’;
- SELECT * FROM employees WHERE last_name not LIKE ‘_u%';

Operators in Where clause……..

- IS NULL / IS NOT NULL Operators:
  - Matches rows where a column value is (or is not) NULL.
- SELECT * FROM employees WHERE email IS NULL;
- SELECT * FROM employees WHERE email IS not NULL;

Delete data from table

- Syntax: delete from <table_name> where <condition>
- Example: Delete from product where pro_dom<‘2020-12-30’;
- Delete all data from table:
  - Delete from <table name>
  - Delete from product
- Truncate Data (Delete table and re create same table)
  - Syntax: truncate <table name>
  - Ex: truncate product

Update Table data

- Syntax:
  - Update <table name> set <field_name>=<value>,<field_name>=<value>,…… where <condition>;
- Example:
  - Update product set pro_price=5000 where pro_id=3;
  - Update product set pro_name='laser Printer' where pro_name='printer';
  - Update product pro_price=pro_price+200 where pro_price<300;

ALTER TABLE

- ADD Column
  - Syntax: Alter table <table name> add column <field_name data_type(size)> [first | after <field name>];
  - Example:
    - Alter table product add column pro_brand varchar(100) default ‘itc’;
    - Alter table product add column pro_id int first;
    - Alter table product add column pro_gst float after pro_price;

- DROP Column
  - Syntax : ALTER TABLE <table_name> DROP COLUMN <column_name>;
  - Example: ALTER TABLE product DROP COLUMN discount;

ALTER TABLE Conti..

- MODIFY COLUMN
  - Syntax: ALTER TABLE table_name MODIFY COLUMN column_name datatype;
  - Example: alter table product modify price float;
- Change Column Name
  - Syntax: alter table <table_name> change <old_field_name> <new_field_name data_type(size)>;
  - Example:
    - alter table product change dis discount float

ALTER TABLE

- Add PRIMARY KEY Constraint in Exiting Table
  - Syntax:
  - ALTER TABLE <table_name> ADD PRIMARY KEY (<field_name>);
  - ALTER TABLE <table_name> ADD CONSTRAINT <constrain name> PRIMARY KEY (<field_name1>, <field_name2>,..);
  - Example:
  - ALTER TABLE Persons ADD PRIMARY KEY (ID);
  - ALTER TABLE Persons ADD CONSTRAINT PK_Person PRIMARY KEY (ID,LastName);
- DROP a PRIMARY KEY Constraint
  - Syntax : alter table <table_name> drop PRIMARY KEY
  - Example: ALTER TABLE Persons DROP PRIMARY KEY;

ALTER Table

- Add UNIQUE Constraint in Exiting table
  - Syntax:
  - ALTER TABLE <table_name> UNIQUE (<field_name>);
  - ALTER TABLE <table_name> ADD CONSTRAINT <constrain name> UNIQUE (<field_name1>,<field_name 2>,..);
  - Example:
  - ALTER TABLE Persons DD UNIQUE (ID);
  - ALTER TABLE Persons ADD CONSTRAINT UC_Person UNIQUE (ID,LastName);:

ALTER TABLE

- Add FOREIGN KEY in existing table
  - Syntax: ALTER TABLE <table_name> ADD FOREIGN KEY (field_name from table 1) REFERENCES Persons(field_name from table 2);
  - Example: ALTER TABLE Orders ADD FOREIGN KEY (PersonID) REFERENCES Persons(PersonID);
- DROP a FOREIGN KEY Constraint
  - Syntax : ALTER TABLE <table_name> DROP FOREIGN KEY <constrain name>;
  - Example: ALTER TABLE Orders DROP FOREIGN KEY FK_PersonOrder;

MySQL Joining Tables

- A **JOIN** is used to combine rows from two or more tables based on a related column.
- MySQL Support following type of Join Operation:
  - INNER JOIN: Returns only matching records from both tables.
  - LEFT JOIN (LEFT OUTER JOIN): Returns all records from the left table and matching records from the right table.
  - RIGHT JOIN (RIGHT OUTER JOIN): Returns all records from the right table and matching records from the left table..
  - CROSS JOIN: Returns the Cartesian product (every row from the first table combined with every row from the second table).Natural Join :
  - SELF JOIN: A table joined with itself.
  - NATURAL JOIN: A **NATURAL JOIN** automatically joins two tables based on all columns that have the **same name and compatible data types** in both tables.

INNER JOIN

- Returns only matching rows from both tables.
- Syntax:
  - SELECT <columns>,.. FROM <table1> INNER JOIN <table2> ON <table1.column >= <table2.column>;
- Example:
  - SELECT student_id, name, course FROM students INNER JOIN courses ON student_id = stu_course_id;

LEFT JOIN (LEFT OUTER JOIN)

- Returns all rows from the left table and matching rows from the right table.
- Syntax:
  - SELECT <columns>,.. FROM <table1> LEFT JOIN <table2> ON <table1.column> = <table2.column>;
- Example:
  - SELECT student_id, name, course FROM students
    LEFT JOIN courses ON student_id = stu_course_id;

RIGHT JOIN (RIGHT OUTER JOIN)

- Returns all rows from the right table and matching rows from the left table.
- Syntax:
  - SELECT <columns> ,..FROM <table1> RIGHT JOIN <table2> ON <table1.column> = <table2.column>;
- Example:
  - SELECT student_id, name, course FROM students RIGHT JOIN courses ON student_id = stu_course_id;

CROSS JOIN

- Returns the Cartesian Product of both tables.
- Syntax:
  - SELECT <columns>,..FROM <table1>,<table2>,…;
- Example:
  - Select name,city,cname,cfees from student,course.

SELF JOIN

- A table joined with itself.
- Syntax:
  - SELECT <a.column>, <b.column> FROM <table_name> a JOIN <table_name> b ON <a.column> = <b.column>;
- Example:
  - SELECT e.emp_name AS Employee, m.emp_name AS Manager FROM employee e LEFT JOIN employee m ON e.manager_id = m.emp_id;

NATURAL JOIN

- Automatically joins tables using columns having the same name.
- Syntax
  - SELECT * FROM table1 NATURAL JOIN table2;
  - SELECT * FROM table1 NATURAL LEFT JOIN table2;
  - SELECT * FROM table1 NATURAL RIGHT JOIN table2;

- Example:
  - SELECT * FROM students NATURAL JOIN courses;
  - SELECT * FROM students NATURAL LEFT JOIN courses;
  - SELECT * FROM students NATURAL RIGHT JOIN courses;

Aggregate Functions in SQL

- MAX(column) : Returns the largest value from the specified column.
- MIN(column) : Returns the smallest value from the specified column.
- AVG(column) : Returns the average of the values in the specified column.
- SUM(column) : Returns the sum of the values for the specified column.
- COUNT(column) : Returns the number of values in the specified column ignoring the NULL values.
- COUNT(*) : Returns the number of records in a table.
- Example:
  - *Select sum(fees),max(fees),count(*) from student;*

GROUP BY in SQL

- At times we need to fetch a group of rows on the basis of common values in a column. This can be done using a GROUP BY clause.
- Syntax:
  - *Select aggregate_function(field_name),……. from <table_name> group by <field_name>,…..*
- Example:
  - *Select city,count(*) as num_of_stu from student group by city;*

Having Clause

- The **HAVING** clause is used to filter records **after grouping** with GROUP BY.
- Syntax:
  - SELECT column_name, aggregate_function(column_name)
    FROM table_name GROUP BY column_name HAVING condition;
- Example:
  - SELECT course, COUNT(*) AS total_students
    FROM students GROUP BY course HAVING COUNT(*) > 2;

Text Function in MySQL

- LENGTH(string) : Returns the length of a string in characters.
  - Ex: SELECT LENGTH('Hello World’);
- CONCAT(string1,string 2,..) : Concatenates two or more strings.
  - Ex: SELECT CONCAT('Hello', ' ', 'World’);
- LOWER(string) or lcase(string) : Converts a string to lowercase.
  - Ex : SELECT LOWER('HELLO WORLD’);
- UPPER(string) or ucase(string): Converts a string to uppercase.
  - Ex : SELECT UPPER('hello world’);
- LEFT(string ,N) : Returns the leftmost number of characters from a string.
  - Ex : SELECT LEFT('Hello World', 5);

Text Function

- RIGHT(string, N): Returns N number of characters from the right side of the string.
- MID(String,pos,n) or Substring(Strinng,pos,n) or substr(String,pos,n): Returns a substring of size n starting from the specified position (pos) of the string. If n is not specified, it returns the substring from the position pos till end of the string.
- LTRIM(string) : Returns the given string after removing leading white space characters.
- RTRIM(string) : Returns the given string after removing trailing white space characters.
- TRIM(string) : Returns the given string after removing both leading and trailing white space characters.

Text Function

- REPLACE(str, from_string, to_string) : used to replace all occurrences of a substring within a string.
- REVERSE(STR): return the reverse of inputted string.

Number Function

- POWER(X,Y) or Pow() : Calculates X to the power Y.
- ROUND(N,D) : Rounds off number N to D number of decimal places.
- MOD(A, B) : Returns the remainder after dividing number, A by number B.
- TRUNCATE(X,D): truncate the number to specified number of decimal places.
- SQRT(N): return the square root of number.
- ABS(N) : return the absolute value of number

Date and Time Function

- NOW() : It returns the current system date and time.
- DATE() : It returns the date part from the given date/time expression.
- MONTH(date) : It returns the month in numeric form from the date.
- MONTHNAME(date) : It returns the month name from the specified date.
- YEAR(date) : It returns the year from the date.
- DAY(date) : It returns the day part from the date.
- DAYNAME(date) : It returns the name of the day from the date.

Set Operations in MySQL

- Set operations are used to combine the results of two or more SELECT statements.
- UNION
- UNION ALL

Union

- Combines results from two queries and removes duplicate rows.
- Syntax:
  - SELECT column1, column2 FROM table1
    UNION
    SELECT column1, column2 FROM table2;
- Example:
  - SELECT name FROM students_2024
    UNION
    SELECT name FROM students_2025;

UNION ALL

- Combines results and keeps duplicates.
- Syntax:
  - SELECT column1 FROM table1
    UNION ALL
    SELECT column1 FROM table2;
- Example:
  - SELECT name FROM students_2024
    UNION ALL
    SELECT name FROM students_2025;

INTERSECT (MySQL 8.0.31+)

- Returns only common rows from both queries.
- Syntax:
  - SELECT column1 FROM table1
    INTERSECT
    SELECT column1 FROM table2;
- Example:
  - SELECT name FROM students_2024
    INTERSECT
    SELECT name FROM students_2025;

EXCEPT (MySQL 8.0.31+)

- Returns rows from the first query that are not in the second query.
- Syntax:
  - SELECT column1 FROM table1
    EXCEPT
    SELECT column1 FROM table2;
- Example:
  - SELECT name FROM students_2024
    EXCEPT
    SELECT name FROM students_2025;

Thank You

Address: G. R. Complex Preetam Nagar, Dhoomanganj
Prayagraj Uttar Pradesh 211011

Mobile No.: 9598948810,8874588766,9532878816
Website : www.infomax.org.in



