**Module 4 – Introduction to DBMS**

**Introduction to SQL**

**Theory Questions:**

**1: What is SQL, and why is it essential in database management?**

(SQL) is a programming language for storing and processing information in a relational database. A relational database stores information in tabular form, with rows and columns representing different data attributes and the various relationships between the data values**.**

**2. Explain the difference between DBMS and RDBMS.**

**DBMS (Database Management System):**

* General term: DBMS is a broad category of software that manages databases.
* Data storage: Can store data in various formats, including files, hierarchical structures, or network models.
* Data relationships: May or may not enforce relationships between data.
* Data integrity: May not have robust mechanisms for ensuring data consistency and accuracy.

**RDBMS (Relational Database Management System):**

* Specific type: RDBMS is a DBMS that uses the relational model.
* Data storage: Stores data in tables with rows and columns, with relationships defined between tables.
* Data relationships: Emphasizes relationships between tables using primary and foreign keys.
* Data integrity: Provides features like integrity constraints, normalization, and ACID properties Atomicity, Consistency, Isolation, Durability) to ensure data integrity and reliability.

**3. Describe the role of SQL in managing relational databases.**

**:** SQL plays a central role in managing **relational databases**. It is the standard language used to interact with and manipulate data stored in **relational database management systems (RDBMS)** such as MySQL, PostgreSQL, Oracle, SQL Server, and SQLite.

**4. What are the key features of SQL?**

* **Data Definition Language (DDL):**
* **Data Manipulation Language (DML)**
* **Relational Foundation**
* **High-performance**
* **Scalability**
* **Security and authentication**
* **Vendor Independence**
* **Portability across different computer systems**

**LAB EXERCISES:**

**Lab 1: Create a new database named school\_db and a table called students with the following columns: student\_id, student\_name, age, class, and address.**

CREATE DATABASE schooldb;

USE schooldb;

CREATE TABLE students (

student\_id INT PRIMARY KEY,

student\_name VARCHAR(200),

age INT,

class VARCHAR(50),

address VARCHAR(200)

);

**Lab 2: Insert five records into the students table and retrieve all records using the SELECT statement.**

CREATE TABLE students (

student\_id INT PRIMARY KEY,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

age INT,

major VARCHAR(50)

);

**Insert five records into the students table**

INSERT INTO students (student\_id, first\_name, last\_name, age, major) VALUES

(1, 'Alice', 'Smith', 20, 'Computer Science'),

(2, 'Bob', 'Johnson', 22, 'Electrical Engineering'),

(3, 'Charlie', 'Brown', 19, 'Mathematics'),

(4, 'Darsh', 'Gajjar', 21, 'Physics'),

(5, 'ben', 'Davis', 20, 'Biology');

**Retrieve all records from the students table**

SELECT \* FROM students;

**SQL Syntax**

**Theory Questions:**

**1. What are the basic components of SQL syntax?**

**1**. SQL Commands (Categorized by Function)

**2**. Core Clauses (Commonly used with SELECT)

**3**. Language Elements

**4**. General Syntax Rules

**2. Explain the role of clauses in SQL statements.**

In SQL, **clauses** are fundamental building blocks that act as modifiers or instructions within an SQL statement, dictating how the database should process the data. They provide specific conditions, filters, groupings, and ordering for the data being manipulated or retrieved.

**LAB EXERCISES:**

**Lab 1: Write SQL queries to retrieve specific columns (student\_name and age) from the students table.**

SELECT student\_name, age

FROM students;

**Lab 2: Write SQL queries to retrieve all students whose age is greater than 10.**

SELECT \*

FROM students

WHERE age > 10;

**SQL Constraints**

**Theory Questions:**

**1. What are constraints in SQL? List and explain the different types of constraints.**

**NOT NULL**:

Ensures that a column cannot have a NULL value. This means that every row must have a value for that column.

If you try to insert a new row or update an existing row with a NULL value in a NOT NULL column, the database will return an error.

**UNIQUE**:

Ensures that all values in a column (or a set of columns) are different. While multiple rows can have NULL values in a UNIQUE column.

This constraint prevents duplicate entries in the specified column(s).

**PRIMARY KEY**:

Uniquely identifies each record in a table. A primary key column (or set of columns) must contain unique values, and it cannot contain NULL values.

A table can have only one primary key. It's a combination of NOT NULL and UNIQUE constraints. It's crucial for establishing relationships between tables.

**FOREIGN KEY**:

Links two tables together by referencing the PRIMARY KEY of another table. It ensures referential integrity, meaning that relationships between tables are maintained.

**CHECK**:

You define a boolean expression that must be true for every value in the column. If an insert or update operation violates this condition, it will be rejected.

**DEFAULT**:

Provides a default value for a column when no value is specified during an INSERT operation.

If you don't explicitly provide a value for a column with a DEFAULT constraint, the database will automatically assign the predefined default value.

**2. How do PRIMARY KEY and FOREIGN KEY constraints differ?**

**PRIMARY KEY**

Uniquely identifies each record (row) in a database table. It acts as the principal identifier for a table.

Must contain unique values for each row. No two rows can have the same primary key value.

Cannot contain NULL values. Every row must have a value for its primary key.

A table can have only one primary key. This primary key can be composed of one or more columns (a composite primary key).

**FOREIGN KEY**

Establishes a link or relationship between two tables. It references the PRIMARY KEY of another table, the parent table.

A table can have multiple foreign keys, each referencing a different primary key (or even the same primary key) in other tables.

Primarily used to enforce referential integrity between tables, ensuring that relationships between data are valid and consistent. It prevents actions that would destroy links between tables.

**3. What is the role of NOT NULL and UNIQUE constraints?**

**1. NOT NULL Constraint**

* Role The primary role of the NOT NULL constraint is to ensure that a specific column cannot contain NULL values.

**2. UNIQUE Constraint**

* Role The main role of the UNIQUE constraint is to ensure that all values in a specific column are distinct. No two rows can have the same non-NULL value in a UNIQUE column.

**LAB EXERCISES:**

**Create a table teachers with the following columns: teacher\_id (Primary Key), teacher\_name (NOT NULL), subject (NOT NULL), and email (UNIQUE).**

CREATE TABLE teachers (

teacher\_id INT PRIMARY KEY,

teacher\_name VARCHAR(255) NOT NULL,

subject VARCHAR(255) NOT NULL,

email VARCHAR(255) UNIQUE

**);**

**: Main SQL Commands and Sub-commands(DDL)**

**Theory Questions:**

**1. Define the SQL Data Definition Language (DDL).**

The SQL Data Definition Language is a subset of SQL used for defining, managing, and modifying the structure of database objects. DDL commands deal with how data is stored and organized.

**2. What is the purpose of specifying data types and constraints during table creation?**

The purpose of specifying data types and constraints during table creation is fundamental to building a reliable, and efficient database. They work hand in hand to ensure data integrity, data quality, and optimized performance.

**LAB EXERCISES:**

**Lab 1: Create a table courses with columns: course\_id, course\_name, and course\_credits. Set the course\_id as the primary key.**

CREATE TABLE courses (

course\_id INT PRIMARY KEY,

course\_name VARCHAR(255) NOT NULL,

course\_credits INT NOT NULL

);

**Lab 2: Use the CREATE command to create a database university\_db.**

**: ALTER Command**

**Theory Questions:**

**1. What is the use of the ALTER command in SQL?**

The ALTER command in SQL is a Data Definition Language command used to modify the structure of an existing database object, most commonly a table. It allows for various structural changes without recreating the entire object.

**2. How can you add, modify, and drop columns from a table using ALTER?**

**1. Adding a Column:**

ALTER TABLE Employees

ADD COLUMN HireDate DATE;

**2. Modifying a Column:**

ALTER TABLE Products

MODIFY COLUMN Price DECIMAL(10, 2);

**3. Dropping a Column:**

ALTER TABLE Customers

DROP COLUMN OldAddress;

**LAB EXERCISES:**

**Lab 1: Modify the courses table by adding a column course\_duration using the ALTER command.**

ALTER TABLE course

ADD COLUMN course\_duration INT;

**Lab 2: Drop the course\_credits column from the courses table.**

ALTER TABLE course\_table\_name DROP COLUMN credits;

**: DROP Command**

**Theory Questions:**

**1. What is the function of the DROP command in SQL?**

The DROP command in SQL is a Data Definition Language (DDL) command used to permanently remove objects from a database. Unlike DELETE or TRUNCATE, which operate on data within a table, DROP removes the entire structure of the object along with its data.

**2. What are the implications of dropping a table from a database?**

Dropping a table from a database has resulting in the permanent deletion of the tables data and structure, constraints, triggers, and views. This action is irreversible without a backup, and dependent objects like views and stored procedures referencing and the dropped table become invalid.

**Lab 1: Drop the teachers table from the school\_db database**

**DROP TABLE teachers;**

**Lab 2: Drop the students table from the school\_db database and verify that the table has been removed.**