CMR INSTITUTE OF TECHNOLOGY

USN | | | | | | | | | | |

Internal Assesment Test –I, August 2024

Sub:	**Database Management System**	**Code:**	**22MCA21**

	Answer Key	Marks	OBE	
			CO	RBT
1)	List and explain the advantages of using the DBMS approach.	10		

A Database Management System (DBMS) is software that allows users to define, create, and manage databases. It provides a systematic way to organize, store, retrieve, and manage data efficiently. Here are some key advantages of using the DBMS approach:

1**. Data Independence**

Logical Data Independence: The DBMS separates the logical data structure (schema) from the physical data storage, enabling changes in the logical schema without affecting the physical storage and vice versa.

Physical Data Independence: Users can change the physical storage structure or devices without needing to modify the logical data structures.

2. **Efficient Data Access**

DBMSs use sophisticated algorithms and indexing to retrieve and manipulate large volumes of data quickly. This results in faster query processing and data retrieval compared to traditional file-based systems.

3. **Data Integrity and Security**

Data Integrity: DBMSs enforce data integrity constraints like primary keys, foreign keys, and unique constraints, ensuring that the data entered into the database is accurate and consistent.

Data Security: DBMSs provide robust security mechanisms, such as user authentication and authorization, ensuring that only authorized users can access or modify data.

4. **Data Redundancy Control**

DBMSs minimize data redundancy by normalizing the data, which reduces the chances of data inconsistency and ensures that data is stored in a single location. This saves storage space and simplifies data management.

5**. Data Consistency**

With a DBMS, all users have access to the same, consistent view of the data, preventing conflicts and inconsistencies that might arise from multiple users updating data simultaneously.

6. **Concurrent Access and Transaction Management**

DBMSs support concurrent access to data by multiple users, managing transactions in a way that ensures data consistency and integrity. They use techniques like locking and transaction logging to manage concurrency effectively.

7. **Backup and Recovery**

DBMSs provide automated backup and recovery procedures, ensuring that data can be restored in case of system failures or data corruption. This feature is critical for maintaining data availability and business continuity.

CO1 L1

	8**. Data Abstraction** DBMSs offer different levels of data abstraction (physical, logical, and view levels), allowing users to interact with the data at a level appropriate for their needs without needing to know the underlying complexities. 9**. Data Sharing** DBMSs allow multiple users or applications to access the database simultaneously, facilitating data sharing across departments or within an organization while maintaining control over data access. 10. **Improved Data Management** With a DBMS, data management becomes more systematic, enabling easier data maintenance, updates, and reporting. It also simplifies complex data relationships, making data management more efficient and less error-prone. 11. **Enhanced Reporting and Query Capabilities** DBMSs provide powerful query languages like SQL that enable complex querying, reporting, and data analysis, making it easier to generate insights from data. 12**. Scalability** Modern DBMSs are designed to scale horizontally and vertically, accommodating growing amounts of data and increasing numbers of users without a significant drop in performance.			
	OR			
2)	Explain about actors on the scene and workers behind the scene.	10	CO1	L1

Actors on the Scene

1. Database Administrators
2. Database Designers
3. End Users
4. System Analysts and Application Programmers (Software Engineers)

Database Administrators

In any organization where many persons use the same resources, there is a need for a chief administrator to oversee and manage these resources. In a database environment, the primary resource is the database itself and the secondary resource is the DBMS and related software. Administering these resources is the responsibility of the **database administrator (DBA).** The DBA is responsible for authorizing access to the database, for coordinating and monitoring its use, and for acquiring software and hardware resources as needed.

Database Designers

Database designers are responsible for identifying the data to be stored in the database and for choosing appropriate structures to represent and store this data. It is the responsibility of database designers to communicate with all prospective database users, in order to understand their requirements, and to come up with a design that meets these requirements.

End Users

End users are the people whose jobs require access to the database for querying, updating, and generating reports; the database primarily exists for their use. There are several categories of end users:

- **Casual end users** occasionally access the database, but they may need different information each time. They use a sophisticated database query language to specify their requests and are typically middle- or high-level managers or other occasional browsers.
- **Naive** or **parametric end users** make up a sizable portion of database end users. Their main job function revolves around constantly querying and updating the database, using standard types of queries and updates—called **canned transactions**—that have been carefully programmed and tested.

  Bank tellers check account balances and post withdrawals and deposits.

- **Sophisticated end users** include engineers, scientists, business analysts, and others who thoroughly familiarize themselves with the facilities of the DBMS so as to implement their applications to meet their complex requirements.
- **Stand-alone users** maintain personal databases by using ready-made program packages that provide easy-to-use menu- or graphics-based interfaces. An example is the user of a tax package that stores a variety of personal financial data for tax purposes.

System Analysts and Application Programmers (Software Engineers)

System analysts determine the requirements of end users, especially naive and parametric end users, and develop specifications for canned transactions that meet these requirements. **Application programmers** implement these specifications as programs; then they test, debug, document, and maintain these canned transactions. Such analysts and programmers (nowadays called **software engineers**) should be familiar with the full range of capabilities provided by the DBMS to accomplish their tasks.

Workers behind the Scene

In addition to those who design, use, and administer a database, others are associated with the design, development, and operation of the DBMS *software and system environment*. These persons are typically not interested in the database itself. We call them the "workers behind the scene," and they include the following categories.

- **DBMS system designers and implementers** are persons who design and implement the DBMS modules and interfaces as a software package. A DBMS is a complex software system that consists of many components or **modules,** including modules for implementing the catalog, query language, interface processors, data access, concurrency control, recovery, and security.
- **Tool developers** include persons who design and implement **tools**—the software packages that facilitate database system design and use, and help improve performance. Tools are optional packages that are often purchased separately. They include packages for database design, performance monitoring, natural language or graphical interfaces, prototyping, simulation, and test data generation .
- **Operators and maintenance personnel** are the system administration personnel who are responsible for the actual running and maintenance of the hardware and software environment for the database system.

3)	**Explain SQL data definition and data types in brief and explain different CREATE and ALTER command.** SQL Data Definition Language (DDL) is a subset of SQL commands used to define	10	CO3	L2

and manage the structure of a database. It includes commands for creating, altering, and deleting database objects like tables, indexes, and views.

SQL Data Types
Data types define the type of data that can be stored in a table column. Here are some common SQL data types:

Numeric Types:

INT: Stores integer values.
FLOAT/REAL: Stores floating-point numbers with varying precision.
DECIMAL/NUMERIC: Stores fixed-point numbers with exact precision, often used for monetary values.
Character/String Types:

CHAR(size): Fixed-length string, padded with spaces if necessary.
VARCHAR(size): Variable-length string, storing only the characters used.
TEXT: Stores large text data, typically used for long paragraphs of text.
Date and Time Types:

DATE: Stores a date value (YYYY-MM-DD).
TIME: Stores a time value (HH:MI
).
DATETIME: Stores both date and time (YYYY-MM-DD HH:MI
).
TIMESTAMP: Stores a timestamp, often used to record the exact time a record was inserted or updated.
Binary Types:

BINARY(size): Fixed-length binary data.
VARBINARY(size): Variable-length binary data.
BLOB: Stores large binary data like images, audio, or video files.
Boolean Type:

BOOLEAN: Stores true/false values.
Miscellaneous Types:

ENUM: Stores one value from a defined list of values.
SET: Stores multiple values from a defined list.

SQL CREATE and ALTER Commands
1. CREATE Command
The CREATE command is used to create new database objects like tables, indexes, views, and schemas.

```
CREATE TABLE table_name (
    column1_name data_type constraints,
    column2_name data_type constraints,
    ...
);

CREATE TABLE employees (
    id INT PRIMARY KEY,
    name VARCHAR(100),
    hire_date DATE,
    salary DECIMAL(10, 2)
```

	); **2. ALTER Command** The ALTER command is used to modify the structure of an existing database object, like adding, dropping, or modifying columns in a table. ALTER TABLE table_name ADD column_name data_type; ALTER TABLE employees ADD department VARCHAR(50); ALTER TABLE table_name MODIFY column_name new_data_type; ALTER TABLE table_name DROP COLUMN column_name; ALTER TABLE old_table_name RENAME TO new_table_name; **Data Definition**: Involves defining the database structure using DDL commands like CREATE and ALTER. **Data Types**: Specify the type of data that can be stored in table columns, ensuring data integrity and efficient storage. **CREATE Command**: Used to create database objects such as tables, views, and indexes. **ALTER Command**: Used to modify the structure of existing database objects, such as adding or dropping columns or changing data types.			
	OR			
4)	**Create the table named 'employee' with the attributes: empno, empname, dept, designation, salary, doj, place. Write SQL statements for the following:** **i. Display all the fields of employee table. ii. Display details of employee number and their salary.iii. Display average salary of all employees. iv. Display distinct name of employees in descending order. v. Count number of employees.** **vi. Display the employees whose name contains second and third letters as 'um'.**	10	CO3	L3

vii. Display salary of employee which is greater than 120000.
viii. Display details of employee whose name is 'Amit' and salary greater than 50000

```
CREATE TABLE employee (
   empno INT PRIMARY KEY,
   empname VARCHAR(100),
   dept VARCHAR(50),
   designation VARCHAR(50),
   salary DECIMAL(10, 2),
   doj DATE,
   place VARCHAR(50)
);
```

1. SELECT * FROM employee;
2. SELECT empno, salary FROM employee;
3. SELECT AVG(salary) AS average_salary FROM employee;
4. SELECT DISTINCT empname FROM employeeORDER BY empname DESC;
5. SELECT COUNT(*) AS total_employees FROM employee;
6. SELECT * FROM employee WHERE empname LIKE '_um%';
7. SELECT * FROM employee WHERE salary > 120000;
8. SELECT * FROM employeeWHERE empname = 'Amit' AND salary > 50000;

5)	**List and explain the data types that are allowed for SQL attributes with example. How char data type differs from varchar?** **In SQL, attributes (columns in a table) can have various data types that define the kind of data they can hold. Here's a list of commonly used SQL data types:**	10	CO3	L2

1. Numeric Data Types
INT / INTEGER: Used for whole numbers without decimals.
Example: age INT can store values like 25, 0, -12.
FLOAT: Stores approximate numeric values with floating decimal points.
Example: price FLOAT can store values like 9.99, 3.14159.
DECIMAL / NUMERIC: Stores exact numeric values with fixed decimal points, commonly used for monetary data.
Example: salary DECIMAL(10, 2) can store values like 50000.75, 1234.56.
SMALLINT: Similar to INT but uses less storage and has a smaller range.
Example: year SMALLINT can store values like 2024, 1980.

2. Character String Data Types
CHAR(n): Fixed-length string where n defines the length.
Example: code CHAR(5) will always store 5 characters. If you store 'A1', it will be stored as 'A1 ' (padded with spaces).
VARCHAR(n): Variable-length string where n defines the maximum length.
Example: name VARCHAR(50) can store any string up to 50 characters long, like 'John' or 'Maria Garcia'.
TEXT: Stores large strings of text. It's similar to VARCHAR but without a specific length limit in many SQL implementations.
Example: description TEXT can store entire paragraphs or documents.

3. Date and Time Data Types
DATE: Stores date values (year, month, day).
Example: birthdate DATE can store values like 2024-08-12.
TIME: Stores time values (hour, minute, second).

Example: start_time TIME can store values like 14:30:00.
DATETIME: Stores both date and time values.
Example: event DATETIME can store values like 2024-08-12 14:30:00.
TIMESTAMP: Similar to DATETIME, often used to store a combination of date and time, including time zone information.
Example: created_at TIMESTAMP can store values like 2024-08-12 14:30:00.

4. Boolean Data Type
BOOLEAN: Stores TRUE or FALSE values. Some SQL implementations use integers where 0 represents FALSE and 1 represents TRUE.
Example: is_active BOOLEAN can store values TRUE or FALSE.

5. Binary Data Types
BINARY / VARBINARY: Stores binary data, like images or files. BINARY is fixed-length, while VARBINARY is variable-length.
Example: file VARBINARY(255) can store binary data up to 255 bytes.

6. Other Data Types
ENUM: Stores a predefined list of values.
Example: status ENUM('active', 'inactive', 'pending') allows only one of these three values to be stored.
BLOB: Stores binary large objects, typically used for storing large binary files such as images or audio files.
Example: image BLOB can store binary data for an image file.
Difference between CHAR and VARCHAR
CHAR:

Fixed-Length: The storage size is always equal to the declared length, regardless of the actual data length.
Padding with Spaces: If the data is shorter than the declared length, it is padded with spaces.
Use Case: Ideal for storing data of a fixed size, such as country codes ('USA', 'IND').
VARCHAR:

Variable-Length: The storage size is equal to the actual data length, up to the declared maximum length.
No Padding: No extra space is allocated if the data is shorter than the maximum length.
Use Case: Best for storing data where the length can vary, such as names, addresses, or descriptions.

```
CREATE TABLE Users (
   id INT PRIMARY KEY,
   username CHAR(10),  -- Will store up to 10 characters, always 10 characters long due to padding
   email VARCHAR(255)  -- Will store up to 255 characters, but only as much space as the actual data needs
);
```

OR

6)	**You are managing the database of a small retail company, which consists of several tables that store essential information. The Customers table contains details such as CustomerID, FirstName, LastName, Email, and Phone. The Products table includes ProductID, ProductName, Category, and Price. The Orders table tracks customer purchases with fields like OrderID, CustomerID, and OrderDate. Finally, the OrderDetails table links specific products to each order, with fields like OrderDetailID, OrderID, ProductID, Quantity, and UnitPrice.** ***Based on this scenario, answer the following SQL questions:*** **1. Write an SQL query to list the full names (FirstName and LastName) of all customers.** **2. Write an SQL query to find the total number of products in the Products table.** 3. **Write an SQL query to retrieve the details (ProductName, Price) of all products in the 'Electronics' category.** CREATE TABLE Customers (CustomerID INT PRIMARY KEY, FirstName VARCHAR(50), LastName VARCHAR(50), Email VARCHAR(100), Phone VARCHAR(15)); CREATE TABLE Products (ProductID INT PRIMARY KEY, ProductName VARCHAR(100), Category VARCHAR(50), Price DECIMAL(10, 2)); CREATE TABLE Orders (OrderID INT PRIMARY KEY, CustomerID INT, OrderDate DATE, FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)); CREATE TABLE OrderDetails (OrderDetailID INT PRIMARY KEY, OrderID INT, ProductID INT,	10	CO3	L3

	Quantity INT, UnitPrice DECIMAL(10, 2), FOREIGN KEY (OrderID) REFERENCES Orders(OrderID), FOREIGN KEY (ProductID) REFERENCES Products(ProductID)); • SELECT FirstName, LastName FROM Customers; • SELECT COUNT(*) AS TotalProducts FROM Products; • SELECT ProductName, Price FROM Products WHERE Category = 'Electronics';			
7)	**Explain in detailed about any 5 aggregate functions with an example of each** **1. COUNT()** Purpose: Counts the number of rows that match a specified condition, or the total number of non-null values in a column. Example: SELECT COUNT(*) AS TotalOrders FROM Orders; **2. SUM()** Purpose: Calculates the total sum of a numeric column. Example: SELECT SUM(Quantity) AS TotalItemsSold FROM OrderDetails; **3. AVG()** Purpose: Computes the average value of a numeric column. Example SELECT AVG(Price) AS AveragePrice FROM Products; **4. MIN()** Purpose: Finds the minimum value in a column. Example SELECT MIN(Price) AS LowestPrice FROM Products; **5. MAX()** Purpose: Finds the maximum value in a column. Example SELECT MAX(OrderDate) AS MostRecentOrder FROM Orders;	10	CO3	L3
	OR			
8)	**Explain with proper example Data Definition Language (DDL)** Data Definition Language (DDL) is a subset of SQL used to define and manage database structures, such as creating, altering, and deleting database objects like tables, indexes, and schemas. DDL statements don't manipulate data directly; instead, they define the structure and organization of the data. Key DDL Commands:	10	CO3	L2

	CREATE ALTER DROP TRUNCATE RENAME Let's explore each of these with proper examples: **1. CREATE** Purpose: The CREATE statement is used to create new database objects such as tables, indexes, or views. Example: Creating a new table called Employees CREATE TABLE Employees (EmployeeID INT PRIMARY KEY, FirstName VARCHAR(50), LastName VARCHAR(50), Email VARCHAR(100), HireDate DATE); **2. ALTER** Purpose: The ALTER statement is used to modify an existing database object, such as adding, deleting, or modifying columns in a table. Example: Adding a new column PhoneNumber to the Employees table. ALTER TABLE Employees ADD PhoneNumber VARCHAR(15); ALTER TABLE Employees MODIFY Email VARCHAR(150); **3. DROP** Purpose: The DROP statement is used to delete an existing database object like a table, view, or index. Example: Dropping the Employees table DROP TABLE Employees; **4. TRUNCATE** Purpose: The TRUNCATE statement is used to delete all rows from a table without removing the table structure itself. Example: Truncating the Employees table TRUNCATE TABLE Employees; **5. RENAME** Purpose: The RENAME statement is used to change the name of an existing database object. Example: Renaming the Employees table to Staff RENAME TABLE Employees TO Staff;			
9)	Explain any 5 features of SQL that helps to make the complex queries □ **Subqueries**: Enable dynamic queries where results from one query depend on another. □ **Aggregate Functions**: Summarize data, often used with GROUP BY. □ **CASE Statements**: Implement conditional logic within queries. □ **Window Functions**: Perform calculations across related rows without reducing the result set.	10	CO3	L2

	• **Window Functions:** Pattern matching is a powerful tool in SQL that helps in filtering and retrieving data based on specific string patterns, making it useful for searching and data validation tasks. With Examples			
	OR			
10)	**Explain in detailed about nested query with any 5 examples of your own** • SELECT ProductName, Price FROM Products WHERE Price > (SELECT AVG(Price) FROM Products); • SELECT FirstName, LastName FROM Customers WHERE CustomerID IN (SELECT CustomerID FROM Orders); • SELECT EmployeeID, FirstName, LastName, Salary FROM Employees e1 WHERE Salary > (SELECT AVG(Salary) FROM Employees e2 WHERE e1.DepartmentID = e2.DepartmentID); • SELECT Category, AvgPrice FROM (SELECT Category, AVG(Price) AS AvgPrice FROM Products GROUP BY Category) AS AvgPricePerCategory WHERE AvgPrice > 50; • SELECT CustomerID, COUNT(OrderID) AS OrderCount FROM Orders GROUP BY CustomerID HAVING COUNT(OrderID) > 3;	10	CO3	L3