Software Engineering Assignment

**MODULE: 5 SE – DATABASE**

1. **What do you understand By Database**

# Ans:-

**DBMS (DataBase Mangment System)**

Data is a collection of a distinct small unit of information. It can be used in a variety of forms like text, numbers, media, bytes, etc. it can be stored in pieces of paper or electronic memory, etc.

1. **What is Normalization?**

# Ans:-

* **Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate the undesirable characteristics like Insertion, Update and Deletion Anomalies.**
* **Normalization divides the larger table into the smaller table and links them using relationship.**
* **The normal form is used to reduce redundancy from the database table.**
* **Types of Normal Forms**
* **1NF**

A relation is in 1NF if it contains an atomic value.

* **2NF**

A relation will be in 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the primary key.

* **3NF**

A relation will be in 3NF if it is in 2NF and no transition dependency exists..

1. **What is Difference between DBMS and RDBMS?**

# Ans:-

**Difference between DBMS and RDBMS**

**The following are the important differences between DBMS and RDBMS.**

|  |  |  |
| --- | --- | --- |
| **Key** | **DBMS** | **RDBMS** |
| Definition | DBMS stands for Database Management System. | RDBMS stands for Relational Database Management System. |
| Data Storage | Data is stored as file. | Data is stored as tables. |

|  |  |  |
| --- | --- | --- |
| Data Access | In DBMS, each data elements are to be accessed individually. | In RDBMS, multiple data elements can be accessed at same time. |
| Relationship | There is no relationship between data in DBMS. | Data is present in multiple tables which can be related to each other. |
| Normalization | Normalization cannot be achieved. | Normalization can be achieved. |
| Distributed database | DBMS has no support for distributed databases. | RDBMS supports distributed databases. |
| Data Quantity | DBMS deals with small quantity of data. | RDBMS deals with large quantity of data. |
| Data Redundancy | Data Redundancy is common in DBMS. | Data Redundancy can be reduced using key and indexes in RDBMS. |
| User | DBMS supports single user at a time. | RDBMS supports multiple users at a time. |
| Security | DBMS provides low security during data manipulation. | RDBMS has multilayer security during data manipulation. |
| Example | File systems, XML, etc. | Oracle, SQL Server. |

1. **What is EF Codd Rule of RDBMS Systems?**

# Ans:-

**Dr Edgar F. Codd, after his extensive research on the Relational Model of database systems, came up with twelve rules of his own, which according to him, a database must obey in order to be regarded as a true relational database**.

**Rule 1: Information Rule**

**The data stored in a database, may it be user data or metadata, must be a value of some table cell. Everything in a database must be stored in a table format.**

**Rule 2: Guaranteed Access Rule**

**Every single data element (value) is guaranteed to be accessible logically with a combination of table-name, primary-key (row value), and attribute-name (column value). No other means, such as pointers, can be used to access data.**

**Rule 3: Systematic Treatment of NULL Values**

**The NULL values in a database must be given a systematic and uniform treatment. This is a very important rule because a NULL can be interpreted as one the following − data is missing, data is not known, or data is not applicable.**

**Rule 4: Active Online Catalog**

**The structure description of the entire database must be stored in an online catalog, known as data dictionary, which can be accessed by authorized users. Users can use the same query language to access the catalog which they use to access the database itself.**

**Rule 5: Comprehensive Data Sub-Language Rule**

**A database can only be accessed using a language having linear syntax that supports data definition, data manipulation, and transaction management operations. This language can be used directly or by means of some application. If the database allows access to data without any help of this language, then it is considered as a violation.**

**Rule 6: View Updating Rule**

**All the views of a database, which can theoretically be updated, must also be updatable by the system.**

**Rule 7: High-Level Insert, Update, and Delete Rule**

**A database must support high-level insertion, updation, and deletion. This must not be limited to a single row, that is, it must also support union, intersection and minus operations to yield sets of data records.**

**Rule 8: Physical Data Independence**

**The data stored in a database must be independent of the applications that access the database. Any change in the physical structure of a database must not have any impact on how the data is being accessed by external applications.**

**Rule 9: Logical Data Independence**

**The logical data in a database must be independent of its user’s view (application). Any change in logical data must not affect the applications using it. For example, if two tables are merged or one is split into two different tables, there should be no impact or change on the user application. This is one of the most difficult rule to apply.**

**Rule 10: Integrity Independence**

**A database must be independent of the application that uses it. All its integrity constraints can be independently modified without the need of any change in the application. This rule makes a database independent of the front-end application and its interface.**

**Rule 11: Distribution Independence**

**The end-user must not be able to see that the data is distributed over various locations. Users should always get the impression that the data is located at one site only. This rule has been regarded as the foundation of distributed database systems.**

**Rule 12: Non-Subversion Rule**

**If a system has an interface that provides access to low- level records, then the interface must not be able to subvert the system and bypass security and integrity constraints.**

1. **What do you understand By Data Redundancy?**

# Ans:-

**Data redundancy occurs when the same piece of data exists in multiple places, whereas data inconsistency is when the same data exists in different formats in multiple tables.**

1. **What is DDL Interpreter?**

# Ans:-

**(Data Definition Language)**

**A data definition language (DDL) is a computer language used to create and modify the structure of database objects in a database.**

**For Example :-**

**Create Alter Rename**

1. **What is DML Compiler in SQL?**

# Ans:-

**A DML (data manipulation language) refers to a computer programming language that allows you to add (insert), delete (delete), and alter (update) data in a database.**

1. **What is SQL Key Constraints writing an Example of SQL Key Constraints.**

# Ans:-

SQL Constraints

* **terms that needs to be satisfied on data**
* **For ex all students must have unique roll number**
* **Can be defined as primary key, foreign key, unique key etc**
* **Primary key – column of table whose value can be used to uniquely identify records Foreign key – column inside table that is primary key of another table Web**

Application Development in PHP with Laravel 2021 Database

* **Unique key – like primary key can be used to uniquely identify a record • Difference between primary key and unique key is primary key will never allow null where as unique key will allow it for once.**

1. **What is save Point? How to create a save Point write a Query?**

# Ans:-

**A SAVEPOINT is a point in a transaction when you can roll the transaction back to a certain point without rolling back the entire transaction.**

**The Syntax for a SAVEPOINT command is as shown below.**

**SAVEPOINT SAVEPOINT\_NAME;**

**This command serves only in the creation of a SAVEPOINT among all the transactional statements.**

**Query:-**

**CREATE TABLE student(ID INT, Name VARCHAR(20),**

**Percentage INT, Location VARCHAR(20), DateOfBirth DATE);**

**INSERT INTO student(ID, Name, Percentage, Location, DateOfBirth) VALUES(1, "Manthan Koli", 79, "Delhi", "2003-08-20"),**

**(2, "Dev Dixit", 75, "Pune", "1999-06-17"),**

**(3, "Aakash Deshmukh", 87, "Mumbai", "1997-09-12"),**

**(4, "Aaryan Jaiswal", 90, "Chennai", "2005-10-02"),**

**(5, "Rahul Khanna", 92, "Ambala", "1996-03-04"),**

**(6, "Pankaj Deshmukh", 67, "Kanpur", "2000-02-02"),**

**(7, "Gaurav Kumar", 84, "Chandigarh", "1998-07-06"),**

**(8, "Sanket Jain", 61, "Shimla", "1990-09-08"),**

**(9, "Sahil Wagh", 90, "Kolkata", "1968-04-03"); SELECT \*FROM student;**

**START TRANSACTION; SAVEPOINT ini;**

**INSERT INTO student**

**VALUES (10, "Saurabh Singh", 54, "Kashmir", "1989-01-06"); UPDATE student SET Name = "Mahesh Kuwar" WHERE ID =1;**

**SAVEPOINT upd;**

**DELETE FROM student WHERE ID= 6;**

**SELECT \*FROM student; SAVEPOINT del;**

1. **What is trigger and how to create a Trigger in SQL?**

# Ans:-

A trigger is a Stored procedure in database which Automatically invokes whenever a special event in the database occurs.

For example,

A trigger can be invoked when a row is inserted into a specified table.

**How to create a Trigger in SQL / Syntax:-**

**Create Trigger [Trigger\_name] [Befor / After]**

**{INSERT| UPDATE| DELETE}**

**On [Table\_name] [For each rows] [Trigger\_Body]**

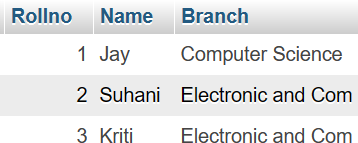
**TASK**

* 1. **Create Table Name : Student and Exam**

1. **Student:- Query:-**

CREATE TABLE student(Rollno int(11) primary key, Name varchar(50),Branch varchar(50));

**Primary key**



**STUDENT:-**

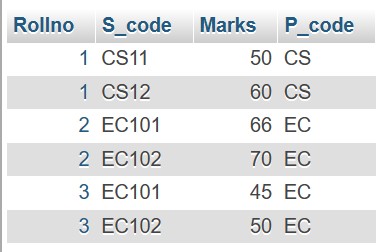
1. **Exam:-**

**Query:-**

CREATE TABLE exam(Rollno int(11), S\_code varchar(255), Marks int(11), P\_code varchar(50),

FOREIGN KEY (Rollno) REFERENCES student(Rollno) ON DELETE CASCADE ON UPDATE CASCADE);

**Foreign key**



**EXAM:-**

* 1. **Create table given below**

**Information:- Query:-**

INSERT Into info (FirstName,Lastname,Address,City,Age) VALUES

('Mickey','Mouse','123 Fantasy Way','Anaheim',73),

('Bat','Man','321 Cavern Ave','Gotham',54),

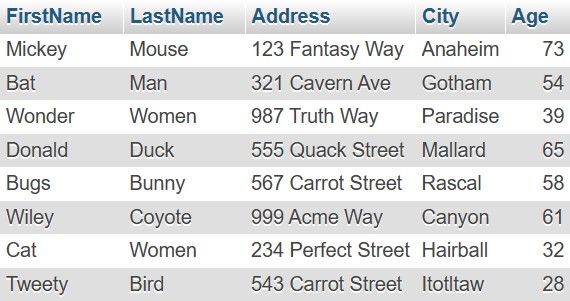
('Wonder','Women','987 Truth Way','Paradise',39),

('Donald','Duck','555 Quack Street','Mallard',65),

('Bugs','Bunny','567 Carrot Street','Rascal',58),

('Wiley','Coyote','999 Acme Way','Canyon',61),

('Cat','Women','234 Perfect Street','Hairball',32), ('Tweety','Bird','543 Carrot Street','Itotltaw',28);



* 1. **Create table given below: Employee and Incentive**

**Table Name: Employee**

**Query:-**

CREATE TABLE employee( Employee\_id int(11) PRIMARY KEY, First\_name varchar(50),

Last\_name varchar(50),

Salary bigint(11), Joining\_date varchar(50), Department varchar(50));

INSERT INTO

employee(Employee\_id,First\_name,Last\_name,Salary,Joinin g\_date,Department) VALUES (1,'John','Abraham',1000000,'01-Jan-13 12.00.00

AM','Banking'),

(2,'Michael','Clarke',800000,'01-Jan-13 12.00.00

AM','Insurance'), (3,'Roy','Thomas',700000,'01-Feb-13 12.00.00

AM','Banking'),

(4,'Tom','Jose',600000,'01-Jan-13 12.00.00

AM','Insurance'), (5,'Jerry','Pinto',650000,'01-Jan-13 12.00.00

AM','Insurance'), (6,'Philip','Mathew',750000,'01-Jan-13 12.00.00

AM','Services'),

(7,'TestName1','123',650000,'01-Jan-13 12.00.00

AM','Services'),

(8,'TestName2','Lname%',600000,'01-Feb-13 12.00.00

AM','Insurance');



**Table Name: Incentive**

**Query:-**

CREATE TABLE incentive (Employee\_ref\_id int(11),Incentive\_date varchar(50),Incentive\_amount int(11),

FORIEGN KEY (Employee\_ref\_id) REFRENCES employee(Employee\_id) ON DELETE CASCADE ON UPDATE CASCADE);

INSERT INTO

incentive(Employee\_ref\_id,Incentive\_date,Incentive\_amoun t) VALUES

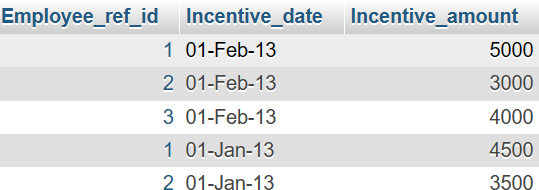
(1,'01-Feb-13',5000),

(2,'01-Feb-13',3000),

(3,'01-Feb-13',4000),

(1,'01-Jan-13',4500),

(2,'01-Jan-13',3500);

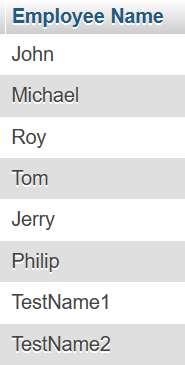


**Query:-**

1. **Get First\_Name from employee table using Tom name “Employee Name”.**

**Ans:-**

SELECT First\_name "Employee Name" FROM employee;



1. **Get FIRST\_NAME, Joining Date, and Salary from employee table.**

**Ans:-**

SELECT First\_name,Joining\_date,Salary FROM employee;



1. **Get all employee details from the employee table order by First\_Name Ascending and Salary descending?**

**Ans:-**

SELECT \* FROM employee ORDER BY First\_name ASC,Salary DESC;



1. **Get employee details from employee table**

**whose first name contains ‘J’.**

**Ans:-**

SELECT \* FROM employee WHERE First\_name LIKE '%j%';



1. **Get department wise maximum salary from employee table order by salary ascending?**

**Ans:-**

SELECT Department, MAX (Salary) MAXSalary FROM employee GROUP BY Department ORDER BY MAXSalary ASC;



1. **Select first\_name, incentive amount from employee and incentives table for those employees who have incentives and incentive amount greater than 3000**

**Ans:-**

SELECT First\_name ,Incentive\_amount FROM employee A INNER JOIN incentive B ON

A.Employee\_id=B.Employee\_ref\_id AND Incentive\_amount >3000;



1. **Create After Insert trigger on Employee table which insert records in view table**

**Ans:-**

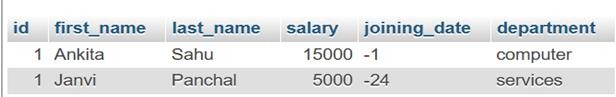
CREATE TRIGGER 'after insert' AFTER INSERT ON

'employee' FOR EACH ROW INSERT INTO Views (id,first\_name,last\_name, salary,joining\_date,department) values (new.Employee\_id,new.First\_name,new.Last\_name,new

.Salary,new.Joining\_date,new.Department);





View Table:-

* 1. **Create table given below: Salesperson and Customer**

TABLE 1:

TABLE NAME SALESPERSON :-

**Query:-**

CREATE TABLE salesperson(Sno int(11) PRIMARY KEY, Sname varchar(50), City varchar(50), Comm int(11));

NSERT INTO salesperson(Sno,Sname,City,Comm) VALUES (1001,'Peel','London',12),

(1002,'Serrs','San Jose',13),

(1004,'Motika','London',11),

(1007,'Rafkin','Barcelone',15),

(1003,'Axelrod','New York',1);



TABLE 2:

TABLE NAME CUSTOMER:-

**Query:-**

CREATE TABLE customer(Cno int(11) PRIMARY KEY,Cname varchar(50),City varchar(50),Rating int(11),Sno int(11),

FOREIGN KEY (Sno) REFERENCES salesperon (Sno) ON DELETE CASCADE ON UPDATE CASCADE);

INSERT INTO customer(Cno,Cname,City,Rating,Sno) VALUES

(201,'Hoffman','London',100,1001),

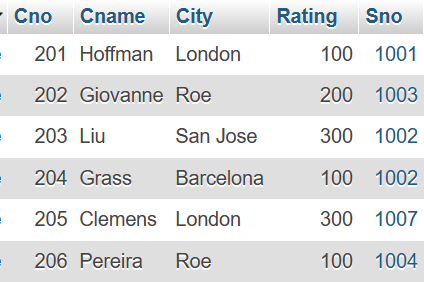
(202,'Giovanne','Roe',200,1003),

(203,'Liu','San Jose',300,1002),

(204,'Grass','Barcelona',100,1002),

(205,'Clemens','London',300,1007),

(206,'Pereira','Roe',100,1004);



**Retrieve the below data from above table**

1. **All orders for more than $1000.**

# Ans:-

**Query:-**

SELECT \* FROM customer where RATING>100;

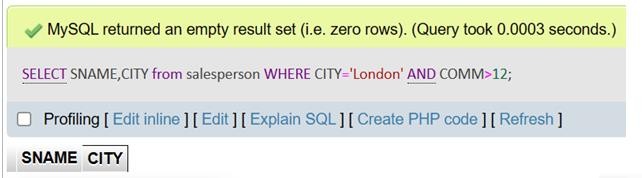


1. **Names and cities of all salespeople in London with commission above 0.12**

# Ans:-

**Query:-**

SELECT SNAME,CITY from salesperson WHERE CITY='London' AND COMM>12;



1. **All salespeople either in Barcelona or in**

**London Ans:- Query:-**

SELECT \* from salesperson where CITY='barcelona' OR CITY='london';



1. **All salespeople with commission between**

**0.10 and 0.12. (Boundary values should be excluded).**

# Ans:-

**Query:-**

SELECT \* from salesperson where COMM>10 and COMM<12;



1. **All customers excluding those with rating <= 100 unless they are located in Rome**

# Ans:-

**Query:-**

SELECT \* FROM CUSTOMER WHERE RATING>100 AND CITY !='ROM';

