**MODULE: 5 (Database)**

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

* Database is a collection of inter-related data and Management System is a set of programs to store and retrieve those data.
* For Example, university database organizes the data about students, faculty, and admin staff etc. which helps in efficient retrieval, insertion and deletion of data from it.

1. **What is Normalization?**

* Normalization is the process of minimizing redundancy (duplicity) from a relation or set of relations.
* Redundancy in relation may cause insertion, deletion and updation anomalies. So, it helps to minimize the redundancy in relations.
* Most Commonly used normal forms:

**First Normal Form:**

* First normal form(1NF) Second normal form(2NF) Third normal form(3NF) Boyce & Code normal form (BCNF)
* If a relation contain composite or multi-valued attribute, it violates first normal form or a relation is in first normal form if it does not contain any composite or multi-valued attribute.
* A relation is in first normal form if every attribute in that relation is singled valued attribute.

**Second Normal Form:**

* To be in second normal form, a relation must be in first normal form and relation must not contain any partial dependency.
* relation is in 2NF if it has No Partial Dependency, i.e., no non-prime attribute (attributes which are not part of any candidate key) is dependent on any proper subset of any candidate key of the table.
* Partial Dependency – If the proper subset of candidate key determines non-prime attribute, it is called partial dependency.

**Third Normal Form:**

* A relation is in third normal form, if there is no transitive dependency for non-prime attributes as well as it is in second normal form.
* A relation is in 3NF if at least one of the following condition holds in every non-trivial function dependency X –> Y

X is a super key.

Y is a prime attribute (each element of Y is part of some candidate key).

* Transitive dependency – If A->B and B->C are two FDs then A->C is called transitive dependency.

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

**DBMS:**

* DBMS stands for Database Management System. A DBMS stores data in the form of files; it uses the file system to store data.
* In a DBMS, there is no relationship between tables containing data. Thus, DBMS does not support distributed databases. In order to access the stored data, it has to provide some uniform method.
* DBMS are often used in small organizations to deal with a small amount of data handled by a single user. File systems, XML, etc. are the popular examples DBMS.

**RDBMS:**

* RDBMS stands for Relational Database Management System. An RDBMS stores data in the form of a table, and a relationship is also established between tables of the database. One of the major advantages of using an RDBMS is that it defines integrity constraint in the database. Since a relationship can be created among the database tables, it can support distributed database.
* RDBMS are primarily designed to handle large amount of related data that can be handled by multiple users. Some common examples of RDBMS include MySQL, Oracle, MS-Access, etc.

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

* when the same data is stored in different tables, it causes data redundancy.

1. **What is DDL Interpreter?**

* DDL - Data Definition Language
* DDL Interpreter as the name suggests interprets the DDL statements such as schema definition statements like create, delete, etc.

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

* It processes the DML statements into low level instruction (machine language), so that they can be executed.

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

* SQL constraints are used to specify rules for the data in a table.
* Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table. If there is any violation between the constraint and the data action, the action is aborted.
* Constraints can be column level or table level. Column level constraints apply to a column, and table level constraints apply to the whole table.
* For Example,

[NOT NULL](https://www.w3schools.com/sql/sql_notnull.asp) - Ensures that a column cannot have a NULL value

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

* A SAVEPOINT is a point in a transaction in which you can roll the transaction back to a certain point without rolling back the entire transaction.
* Syntax for Savepoint command:

SAVEPOINT SAVEPOINT\_NAME;

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

* A trigger is a stored procedure in a database that automatically invokes whenever a special event in the database occurs.
* Syntax:

create trigger [trigger\_name]

[before | after]

{insert | update | delete}

on [table\_name]

[for each row]

[trigger\_body]

**Task**

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

* **SQL query :**

CREATE TABLE student(Rollno int AUTO\_INCREMENT,

Name varchar(200),

Branch varchar(200),

PRIMARY KEY(Rollno)

);

INSERT INTO student(Name,Branch)

VALUES("Jay","Computer Science"),

("Suhani","Electronic and Com"),

("Kriti","Electronic and Com");

**Dumping data for table student :**

|  |  |  |
| --- | --- | --- |
| **Rollno** | **Name** | **Branch** |
| 1 | Jay | Computer Science |
| 2 | Suhani | Electronic and Com |
| 3 | Kriti | Electronic and Com |

CREATE TABLE Exam(Rollno int AUTO\_INCREMENT ,

S\_code varchar(200),

Marks int,

P\_code varchar(200),

FOREIGN KEY(Rollno) REFERENCES student(Rollno)

);

INSERT INTO exam (Rollno,S\_code,Marks,P\_code)

VALUES(1,"CS11",50,"CS"),

(1,"CS12",60,"CS"),

(2,"EC101",66,"EC"),

(2,"EC102",70,"EC"),

(3,"EC101",45,"EC"),

(3,"EC102",50,"EC");

**Dumping data for table exam :**

|  |  |  |  |
| --- | --- | --- | --- |
| **Rollno** | **S\_code** | **Marks** | **P\_code** |
| 1 | CS11 | 50 | CS |
| 1 | CS12 | 60 | CS |
| 2 | EC101 | 66 | EC |
| 2 | EC102 | 70 | EC |
| 3 | EC101 | 45 | EC |
| 3 | EC102 | 50 | EC |

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

* **SQL query :**

CREATE TABLE Employee(Employee\_id int AUTO\_INCREMENT,

First\_name varchar(200),

Last\_name varchar(200),

Salary int,

Joining\_date varchar(200),

Department varchar(200),

PRIMARY KEY(Employee\_id)

);

INSERT INTO employee(First\_name,Last\_name,Salary,Joining\_date,Department)

VALUES("John","Abraham",1000000,"01-JAN-13 12.00.00 AM","Banking"),

("Michael","Clarke",800000,"01-JAN-13 12.00.00 AM","Insurance"),

("Roy","Thomas",700000,"01-FEB-13 12.00.00 AM","Banking"),

("Tom","Jose",600000,"01-FEB-13 12.00.00 AM","Insurance"),

("Jerry","Pinto",650000,"01-FEB-13 12.00.00 AM","Insurance"),

("Philip","Mathew",750000,"01-JAN-13 12.00.00 AM","Services"),

("TestName1","123",650000,"01-JAN-13 12.00.00 AM","Services"),

("TestName2","Lname%",600000,"01-FEB-13 12.00.00 AM","Insurance");

**Dumping data for table employee :**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Employee\_id** | **First\_name** | **Last\_name** | **Salary** | **Joining\_date** | **Department** |
| 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-FEB-13  12.00.00  AM | Insurance |
| 5 | Jerry | Pinto | 650000 | 01-FEB-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 |

CREATE TABLE Incentive(Employee\_ref\_id int,

Incentive\_date varchar(200),

Incentive\_amount int

);

INSERT INTO incentive(Employee\_ref\_id,Incentive\_date,Incentive\_amount)

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);

**Dumping data for table incentive :**

|  |  |  |
| --- | --- | --- |
| **Employee\_ref\_id** | **Incentive\_date** | **Incentive\_amount** |
| 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 |

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

* SELECT First\_name FROM employee WHERE First\_name="Tom";

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

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

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

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

**d) Get employee details from employee table whose first name contains ‘J’.**

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

**e) Get department wise maximum salary from employee table order by salaryascending?**

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

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

* SELECT employee.First\_name,incentive.Incentive\_amount

FROM employee INNER JOIN incentive

ON employee.Employee\_id=incentive.Employee\_ref\_id AND incentive\_amount>3000;

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

* **SQL query :**

CREATE TABLE salesperson ( `(PK)SNo` int,

SNAME varchar(200),

CITY varchar(200),

COMM float

);

INSERT INTO salesperson(`(PK)SNo` ,SNAME,CITY,COMM)

VALUES(1001,"Peel","Landon",.12),

(1002,"Serres","San Jose",.13),

(1004,"Motika","Landon",.11),

(1007,"Rafkin","Barcelona",.15),

(1003,"Axelrod","New York",.1);

**Dumping data for table salesperson :**

|  |  |  |  |
| --- | --- | --- | --- |
| **(PK)SNo** | **SNAME** | **CITY** | **COMM** |
| 1001 | Peel | Landon | 0.12 |
| 1002 | Serres | San Jose | 0.13 |
| 1004 | Motika | Landon | 0.11 |
| 1007 | Rafkin | Barcelona | 0.15 |
| 1003 | Axelrod | New York | 0.1 |

CREATE TABLE Customer(`(PK)CNM.` int,

CNAME varchar(200),

CITY varchar(200),

RATING int,

`(FK)SNo` int

);

INSERT INTO customer(`(PK)CNM.`,CNAME,CITY,RATING,`(FK)SNo`)

VALUES (201,"Hoffman","London",100,1001),

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

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

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

(206,"Clemens","London",300,1007),

(207,"Pereira","Roe",100,1004);

**Dumping data for table customer :**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **(PK)CNM.** | **CNAME** | **CITY** | **RATING** | **(FK)SNo** |
| 201 | Hoffman | London | 100 | 1001 |
| 202 | Giovanne | Roe | 200 | 1003 |
| 203 | Liu | San Jose | 300 | 1002 |
| 204 | Grass | Barcelona | 100 | 1002 |
| 206 | Clemens | London | 300 | 1007 |
| 207 | Pereira | Roe | 100 | 1004 |

**Retrieve the below data from above table**

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

* SELECT \* FROM salesperson WHERE CITY="Landon" AND COMM>0.12;

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

* SELECT SNAME,CITY FROM salesperson WHERE CITY IN ("Barcelona","Landon");

1. **All salespeople with commission between 0.10 and 0.12. (Boundary values should be excluded).**

* SELECT \* FROM salesperson WHERE COMM BETWEEN 0.10 AND 0.12;

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

* SELECT \* FROM customer WHERE RATING<=100 OR CITY="Roe";