SNS COLLEGE OF TECHNOLOGY

(An Autonomous Institution)
Coimbatore – 641 035.
B.E / B.Tech – Internal Assessment Exam- II
Academic Year 2022-2023 (EVEN)

Fourth Semester (Regulation R2019)

19CST202 – Database Management Systems

Answer Key

ANSWER ALL QUESTIONS

1. Define role in SQL and Write the syntax for creating and dropping a role.

A database role specifies a set of database level permissions that let a user perform specific tasks. A role is a way to distinguish among various users as far as what these users can access/update in the database.

SYNTAX:

create a role - create role <name>

Example:

create role instructor

Once a role is created , can assign "users" to the role using:

grant <role> to <users>

drop role <name>

2. Difference between Embedded and Dynamic SQL?

S. No.	Key	Static SQL / Embedded SQL	Dynamic SQL
1	Database Access	In Static SQL, database access procedure is predetermined in the statement.	In Dynamic SQL, how a database will be accessed, can be determine only at run time.
2	Efficiency	Static SQL statements are more faster and efficient.	Dynamic SQL statements are less efficient.
3	Compilation	Static SQL statements are compiled at compile time.	Dynamic SQL statements are compiled at run time.
4	Application Plan	Application Plan parsing, validation, optimization and generation are compile time activities.	Application Plan parsing, validation, optimization and generation are run time activities.
5	Use Cases	Static SQL is used in case of uniformly distributed data.	Dynamic SQL is used in case of non-uniformly distributed data.
6	Dynamic Statements	Statements like EXECUTE IMMEDIATE, EXECUTE, PREPARE are not used.	Statements like EXECUTE IMMEDIATE, EXECUTE, PREPARE are used
7	Flexibility	Static SQL is less flexible.	Dynamic SQL is highly flexible.

3. **Consider the following Functional dependencies to find the canonical cover of F. FD = {A →BC, B → C, A →B, AB→C}**

Steps to find canonical cover

Step 1: Right hand side of all functional dependencies should be single Attribute

F1 = { A →B, A →C, B → C, A →B, AB→C }

Step 2: Remove the Extraneous Attribute

F2 = { A →B, A →C, B → C, A →B, B→C}

Step 3: Eliminate the Redundant Functional Dependencies

Direct - F3 = { A →B, A →C, B → C}

Indirect - F3 = { A →B , B → C}

Therefore, the Canonical Cover F' = { A →B , B → C}

4. Consider the Relation R (A, B, C, D) and Functional dependencies F = {A →B, B→C, C→D}. Check whether the given relation is 2NF or not.

Refer Classwork Notes

5. **Define Functional dependencies with example.**

Functional Dependency (FD) is a constraint that determines the relation of one attribute to another attribute. Functional Dependency helps to maintain the quality of data in the database. It plays a vital role to find the difference between good and bad database design.

A functional dependency is denoted by an arrow "→".

The functional dependency of X on Y is represented by X → Y

Example:

Employee number	Employee Name	Salary	City
1	Dana	50000	San Francisco
2	Francis	38000	London
3	Andrew	25000	Tokyo

If we know the value of Employee number, we can obtain Employee Name, city, salary, etc. By this, we can say that the city, Employee Name, and salary are functionally depended on Employee number.

6. **List the set operations of SQL.**

The SQL Set operation is used to combine the two or more SQL SELECT statements.

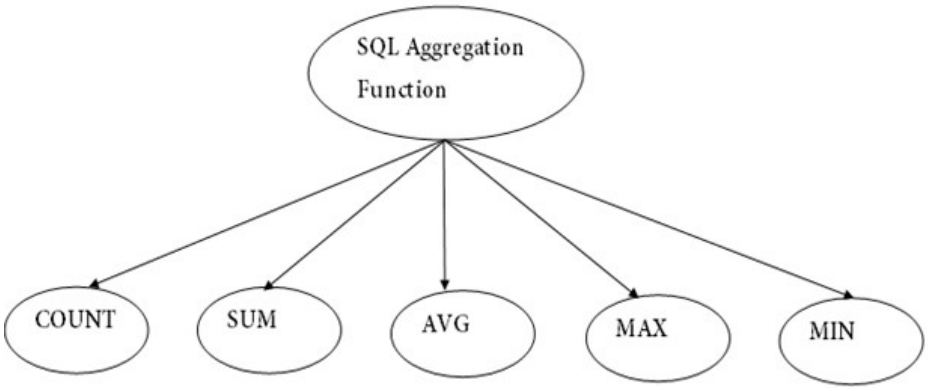
Types of Set Operation

1. Union
2. UnionAll
3. Intersect
4. Minus

7. **What are aggregate functions? And list the aggregate functions supported by SQL**

SQL aggregation function is used to perform the calculations on multiple rows of a single column of a table. It returns a single value.

It is also used to summarize the data.



8. Define Boyce Codd Normal Form.

1. BCNF is the advance version of 3NF. It is stricter than 3NF.
2. A table is in BCNF if every functional dependency X → Y, X is the super key of the table.
3. For BCNF, the table should be in 3NF, and for every FD, LHS is super key.

9. Write down the steps for Closure set of FD's.

Closure of an Attribute can be defined as a set of attributes that can be functionally determined from it.

If "F" is a functional dependency then closure of functional dependency can be denoted using "{F}+".

Steps

Step 1: Add the attributes contained in the attribute set X to the result set X^+.

Step 2: Add the attributes to the result set X^+ which can be functionally determined from the attributes already contained in the result set.

Step 3: Repeat step 2 until no more attributes can be added to the result set X^+.

10. Define Multi-valued Functional Dependency.

- Multi-valued dependency for a relation A −> B exists when for a single value of A, multiple values of B exist.
- Multivalued dependency occurs when two attributes in a table are independent of each other but, both depend on a third attribute.
- A multivalued dependency consists of at least two attributes that are dependent on a third attribute that's why it always requires at least three attributes.

PART B — (2 x 13 = 26 Marks, 1x 14 = 14 Marks)

1 i. Explain with example in SQL the following 7 Marks

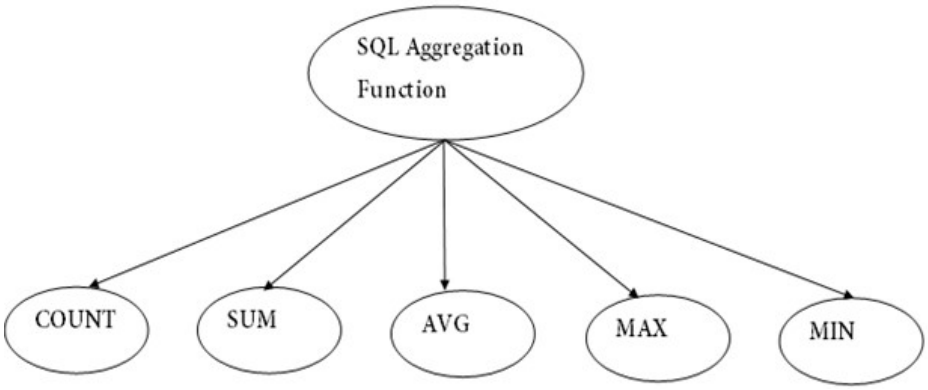
- **Aggregate Function**
- **Group by and Order by**

Aggregate Function

SQL Aggregate Functions

- SQL aggregation function is used to perform the calculations on multiple rows of a single column of a table. It returns a single value.
- It is also used to summarize the data.

Types of SQL Aggregation Function



1. COUNT FUNCTION

- COUNT function is used to Count the number of rows in a database table. It can work on both numeric and non-numeric data types.
- COUNT function uses the COUNT(*) that returns the count of all the rows in a specified table. COUNT(*) considers duplicate and Null.

Syntax

COUNT(*)
or
COUNT([ALL|DISTINCT] expression)

Sample table:

PRODUCT_MAST

PRODUCT	COMPANY	QTY	RATE	COST
Item1	Com1	2	10	20
Item2	Com2	3	25	75
Item3	Com1	2	30	60
Item4	Com3	5	10	50
Item5	Com2	2	20	40
Item6	Cpm1	3	25	75
Item7	Com1	5	30	150
Item8	Com1	3	10	30
Item9	Com2	2	25	50
Item10	Com3	4	30	120

Example: COUNT()

SELECT COUNT(*) FROM PRODUCT_MAST;

Output: 10

Example: COUNT with WHERE

SELECT COUNT(*) FROM PRODUCT_MAST WHERE RATE>=20;

Output: 7

Example: COUNT() with DISTINCT

SELECT COUNT(DISTINCT COMPANY) FROM PRODUCT_MAST;

Output: 3

Example: COUNT() with GROUP BY

SELECT COMPANY, COUNT(*) FROM PRODUCT_MAST GROUP BY COMPANY;

Output:

Com1 5

Com2 3
Com3 2

Example: COUNT() with HAVING

```
SELECT COMPANY, COUNT(*)  FROM PRODUCT_MAST  GROUP BY COMPANY
HAVING COUNT(*)>2;
```

Output:

Com1 5
Com2 3

2. SUM Function

Sum function is used to calculate the sum of all selected columns. It works on numeric fields only.

Syntax

SUM()
or
SUM([ALL|DISTINCT] expression)

Example: SUM()

```
SELECT SUM(COST)  FROM PRODUCT_MAST;
```

Output: 670

Example: SUM() with WHERE

```
SELECT SUM(COST)  FROM PRODUCT_MAST  WHERE QTY>3;
```

Output: 320

Example: SUM() with GROUP BY

```
SELECT SUM(COST)  FROM PRODUCT_MAST  WHERE QTY>3
GROUP BY COMPANY;
```

Output:

Com1 150
Com2 170

Example: SUM() with HAVING

```
SELECT COMPANY, SUM(COST)FROM PRODUCT_MAST GROUP BY
COMPANY HAVING SUM(COST)>=170;
```

Output:

Com1 335
Com3 170

3. AVG function

The AVG function is used to calculate the average value of the numeric type. AVG function returns the average of all non-Null values.

Syntax

1. AVG()
2. or
3. AVG([ALL|DISTINCT] expression)

Example:

```
SELECT AVG(COST) FROM PRODUCT_MAST;
```

Output: 67.00

4. MAX Function

MAX function is used to find the maximum value of a certain column. This function determines the largest value of all selected values of a column.

Syntax

MAX()
or
MAX([ALL|DISTINCT] expression)

Example:

SELECT MAX(RATE) FROM PRODUCT_MAST;
Output: 30

5. MIN Function

MIN function is used to find the minimum value of a certain column. This function determines the smallest value of all selected values of a column.

Syntax

MIN()
or
MIN([ALL|DISTINCT] expression)

Example:

SELECT MIN(RATE) FROM PRODUCT_MAST;

Output: 10

ii. Write a PL/SQL program to arrange the number of two variable in such a way that the small number will store in num_small variable and large number will store in num_large variable.

```
DECLARE
num_small NUMBER := 8;
num_large NUMBER := 5;
num_temp NUMBER;
BEGIN
IF num_small > num_large THEN
num_temp := num_small;
num_small := num_large;
num_large := num_temp;
END IF;
DBMS_OUTPUT.PUT_LINE ('num_small = '||num_small);
DBMS_OUTPUT.PUT_LINE ('num_large = '||num_large);
END;
```

2. Explain about Advanced and Embedded SQL with an example

Accessing SQL from a Programming Language

A database programmer must have access to a general-purpose programming language for at least two reasons

- Not all queries can be expressed in SQL, since SQL does not provide the full expressive power of a general-purpose language.
- Non-declarative actions -such as printing a report, interacting with a user, or sending the results of a query to a graphical user interface -- cannot be done from within SQL.

- A general-purpose program can connect to and communicate with a database server using a collection of functions

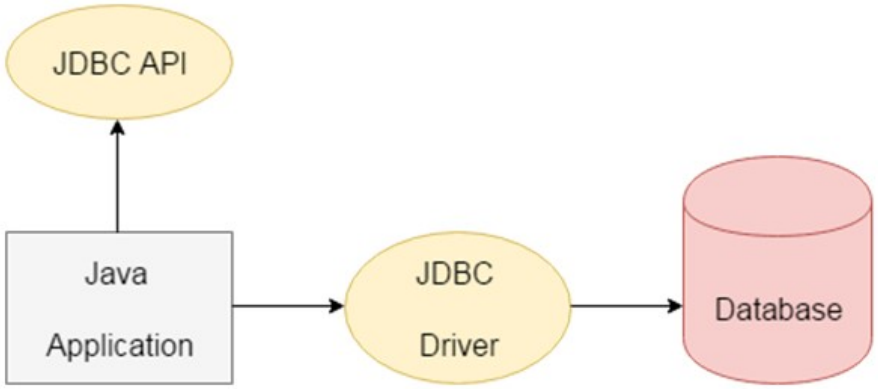
There are two approaches to accessing SQL from a general-purpose programming language

- Embedded SQL - Embedded SQL statements are preprocessed by the SQL processor before the application program is compiled.
- Dynamic SQL - Dynamic SQL is a programming technique that could be used to write SQL queries during runtime. Dynamic SQL could be used to create general and flexible SQL queries.

JDBC

JDBC stands for Java Database Connectivity. JDBC is a Java API to connect and execute the query with the database. It is a part of JavaSE (Java Standard Edition). JDBC API uses JDBC drivers to connect with the database. There are four types of JDBC drivers:

- JDBC-ODBC Bridge Driver,
- Native Driver,
- Network Protocol Driver, and
- Thin Driver



JDBC API to handle database using Java program and can perform the following activities:

1. Connect to the database
2. Execute queries and update statements to the database
3. Retrieve the result received from the database.

Program

```
package com.sa.jdbc;
import java.sql.*;
public class JDBCDemo {
public static void main(String args[]) throws SQLException, ClassNotFoundException
{
String driverClassName = "sun.jdbc.odbc.JdbcOdbcDriver";
String url = "jdbc:odbc:XE";
  String username = "scott";
  String password = "tiger";
  String query = "insert into students values(109, 'bhatt')";
// Load driver class
      Class.forName(driverClassName);
```

```
        // Obtain a connection
        Connection con = DriverManager.getConnection(
           url, username, password);

        // Obtain a statement
        Statement st = con.createStatement();

        // Execute the query
        int count = st.executeUpdate(query);
        System.out.println(
           "number of rows affected by this query= "
           + count);

        // Closing the connection as per the
        // requirement with connection is completed
        con.close();
    }
}
```

Embedded SQL

- The SQL standard defines embeddings of SQL in a variety of programming languages such as C, C++, Java, Fortran, and PL/1,
- The basic form of these languages follows that of the System R embedding of SQL into PL/1.
- EXEC SQL statement is used in the host language to identify embedded SQL request to the preprocessor
  EXEC SQL <embedded SQL statement >;
- Before executing any SQL statements, the program must first connect to the database. This is done using:
  EXEC-SQL connect to server user *user-name* using *password*;
  Here, *server* identifies the server to which a connection is to be established.
- Variables used as above must be declared within DECLARE section,
  EXEC-SQL BEGIN DECLARE SECTION}
  int *credit-amount* ;
  EXEC-SQL END DECLARE SECTION;

Example:

EXEC SQL
declare *c* cursor for
select *ID, name* from *student* where tot_cred > *:credit_amount*
END_EXEC

Functions and Procedure

- The function program has a block of code that performs some specific tasks or functions.
- Particular set of instructions or commands along known as a procedure.

Function Syntax

CREATE [OR REPLACE] FUNCTION function_name
[(parameter_name type [, …])]
// this statement is must for functions
RETURN return_datatype
{IS | AS}

```
BEGIN
  // program code
[EXCEPTION
  exception_section;
END [function_name];
```

Example

```
create function MultiplyNumbers(@int1 as int,@int2 as int)
As
BEGIN
Return (@int1 * @int2)
end
```

Procedure Syntax

```
CREATE or REPLACE PROCEDURE name(parameters)
IS
variables;
BEGIN
//statements;
END;
```

Example

```
CREATE or REPLACE PROCEDURE INC_SAL(eno IN NUMBER, up_sal OUT
NUMBER)
IS
BEGIN
UPDATE emp_table SET salary = salary+1000 WHERE emp_no = eno;
COMMIT;
SELECT sal INTO up_sal FROM emp_table WHERE emp_no = eno;
END;
```

Trigger

- 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 or when certain table columns are being updated.

Syntax

```
create trigger [trigger_name] [before | after]
{insert | update | delete}
on [table_name]
[for each row]
[trigger_body]
```

Example

```
mysql> desc Student;
+-------+-------------+------+-----+---------+----------------+
| Field | Type        | Null | Key | Default | Extra          |
+-------+-------------+------+-----+---------+----------------+
| tid   | int(4)      | NO   | PRI | NULL    | auto_increment |
| name  | varchar(30) | YES  |     | NULL    |                |
| subj1 | int(2)      | YES  |     | NULL    |                |
| subj2 | int(2)      | YES  |     | NULL    |                |
| subj3 | int(2)      | YES  |     | NULL    |                |
| total | int(3)      | YES  |     | NULL    |                |
| per   | int(3)      | YES  |     | NULL    |                |
+-------+-------------+------+-----+---------+----------------+
7 rows in set (0.00 sec)
```

create trigger stud_marks
before
INSERT
on Student
for each row
set Student.total = Student.subj1 + Student.subj2 + Student.subj3, Student.per = Student.total * 60 / 100;

```
mysql> insert into Student values(0, "ABCDE", 20, 20, 20, 0, 0);
Query OK, 1 row affected (0.09 sec)

mysql> select * from Student;
+-----+-------+-------+-------+-------+-------+------+
| tid | name  | subj1 | subj2 | subj3 | total | per  |
+-----+-------+-------+-------+-------+-------+------+
| 100 | ABCDE |    20 |    20 |    20 |    60 |   36 |
+-----+-------+-------+-------+-------+-------+------+
1 row in set (0.00 sec)
```

3. Define and illustrate the 1NF,2NF,3NF and BCNF with your own example.

Normalization

- Normalization is the process of organizing the data in the database.
- Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate undesirable characteristics like Insertion, Update, and Deletion Anomalies.
- Normalization divides the larger table into smaller and links them using relationships.
- The normal form is used to reduce redundancy from the database table.

Why do we need Normalization?

- The main reason for normalizing the relations is removing these anomalies.
- Failure to eliminate anomalies leads to data redundancy and can cause data integrity and other problems as the database grows.
- Normalization consists of a series of guidelines that helps to guide you in creating a good database structure.

Data modification anomalies can be categorized into three types:

- **Insertion Anomaly:** Insertion Anomaly refers to when one cannot insert a new tuple into a relationship due to lack of data.
- **Deletion Anomaly:** The delete anomaly refers to the situation where the deletion of data results in the unintended loss of some other important data.
- **Updatation Anomaly:** The update anomaly is when an update of a single data value requires multiple rows of data to be updated.

Types of Norml Forms

Normal Form	Description
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.
BCNF	A stronger definition of 3NF is known as Boyce Codd's normal form.
4NF	A relation will be in 4NF if it is in Boyce Codd's normal form and has no multi-valued dependency.
5NF	A relation is in 5NF. If it is in 4NF and does not contain any join dependency, joining should be lossless.

Refer Classwork Notes for Rules and Examples

4. i. Consider the Relation R(ABCDEG) and Functional Dependencies F= {AB →C, AC→B, AD→E, B→D, BC→A, E→G}and Decomposition D = {ABC, ACDE, ADG}. Find it is lossy and Lossless Decomposition.

Refer Classwork Notes

4.ii. Explain in detail about 4NF and 5NF with your own example.

Fourth Normal Form (4NF)

A relation will be in 4NF if it is in Boyce Codd normal form and has no multi-valued dependency.

For a dependency A → B, if for a single value of A, multiple values of B exist, then the relation will be a multi-valued dependency.

Properties – A relation R is in 4NF if and only if the following conditions are satisfied:

- It should be in the Boyce-Codd Normal Form (BCNF).
- the table should not have any Multi-valued Dependency.

Example

STUDENT

STU_ID	COURSE	HOBBY
21	Computer	Dancing
21	Math	Singing
34	Chemistry	Dancing

74	Biology	Cricket
59	Physics	Hockey

The given STUDENT table is in 3NF, but the COURSE and HOBBY are two independent entity. Hence, there is no relationship between COURSE and HOBBY.

In the STUDENT relation, a student with STU_ID, **21** contains two courses, **Computer** and **Math** and two hobbies, **Dancing** and **Singing**. So there is a Multi-valued dependency on STU_ID, which leads to unnecessary repetition of data.
So to make the above table into 4NF, we can decompose it into two tables:

STUDENT_COURSE

STU_ID	COURSE
21	Computer
21	Math
34	Chemistry
74	Biology
59	Physics

STUDENT_HOBBY

STU_ID	HOBBY
21	Dancing
21	Singing
34	Dancing
74	Cricket
59	Hockey

Fifth normal form (5NF)

- A relation is in 5NF if it is in 4NF and not contains any join dependency and joining should be lossless.
- 5NF is satisfied when all the tables are broken into as many tables as possible in order to avoid redundancy.
- 5NF is also known as Project-join normal form (PJ/NF).

Example

SUBJECT	LECTURER	SEMESTER
Computer	Anshika	Semester 1
Computer	John	Semester 1
Math	John	Semester 1

Math	Akash	Semester 2
Chemistry	Praveen	Semester 1

In the above table, John takes both Computer and Math class for Semester 1 but he doesn't take Math class for Semester 2. In this case, combination of all these fields required to identify a valid data.

Suppose we add a new Semester as Semester 3 but do not know about the subject and who will be taking that subject so we leave Lecturer and Subject as NULL. But all three columns together acts as a primary key, so we can't leave other two columns blank.

So to make the above table into 5NF, we can decompose it into three relations P1, P2 & P3:

P1

SEMESTER	SUBJECT
Semester 1	Computer
Semester 1	Math
Semester 1	Chemistry
Semester 2	Math

P2

SUBJECT	LECTURER
Computer	Anshika
Computer	John
Math	John
Math	Akash
Chemistry	Praveen

P3

SEMSTER	LECTURER
Semester 1	Anshika
Semester 1	John
Semester 1	John
Semester 2	Akash
Semester 1	Praveen

5. i. Normalize the given table into Fourth Normal Form and explain it.

Subject	Lecture	Semester
Computer	Anu	1
Computer	John	1
Maths	John	1
Maths	Akash	2

Refer Q.no 4.ii.

5. ii. Normalize the given table into Fifth Normal Form and explain it.

Std_Id	Course	Hobby
121	Computer	Reading
121	Maths	Playing
123	Chemistry	Designing
124	Biology	Drawing

Refer Q.no 4.ii.

6. Explain the Dependencies and types used in database design with a suitable example

The functional dependency is a relationship that exists between two attributes. It typically exists between the primary key and non-key attribute within a table.

X → Y , The left side of FD is known as a determinant, the right side of the production is known as a dependent.

Types of Functional dependencies

1. Trivial functional dependency
2. Non-Trivial functional dependency
3. Multivalued functional dependency
4. Transitive functional dependency

1. Trivial Functional Dependency

In Trivial Functional Dependency, a dependent is always a subset of the determinant. i.e. If X → Y and Y is the subset of X, then it is called trivial functional dependency

Example,

roll_no	name	age
42	abc	17

roll_no	name	age
43	pqr	18
44	xyz	18

Here, {roll_no, name} → name is a trivial functional dependency, since the dependent name is a subset of determinant set {roll_no, name}. Similarly, roll_no → roll_no is also an example of trivial functional dependency.

2. Non-trivial Functional Dependency

In Non-trivial functional dependency, the dependent is strictly not a subset of the determinant.
i.e. If X → Y and Y is not a subset of X, then it is called Non-trivial functional dependency.
Example,

roll_no	name	age
42	abc	17
43	pqr	18
44	xyz	18

Here, roll_no → name is a non-trivial functional dependency, since the dependent name is not a subset of determinant roll_no
Similarly, {roll_no, name} → age is also a non-trivial functional dependency, since age is not a subset of {roll_no, name}

3. Multivalued Functional Dependency

In Multivalued functional dependency, entities of the dependent set are not dependent on each other.
i.e. If a → {b, c} and there exists no functional dependency between b and c, then it is called a multivalued functional dependency.
Example,

roll_no	name	age
42	abc	17
43	pqr	18
44	xyz	18
45	abc	19

Here, roll_no → {name, age} is a multivalued functional dependency, since the dependents name & age are not dependent on each other(i.e. name → age or age → name doesn't exist !)

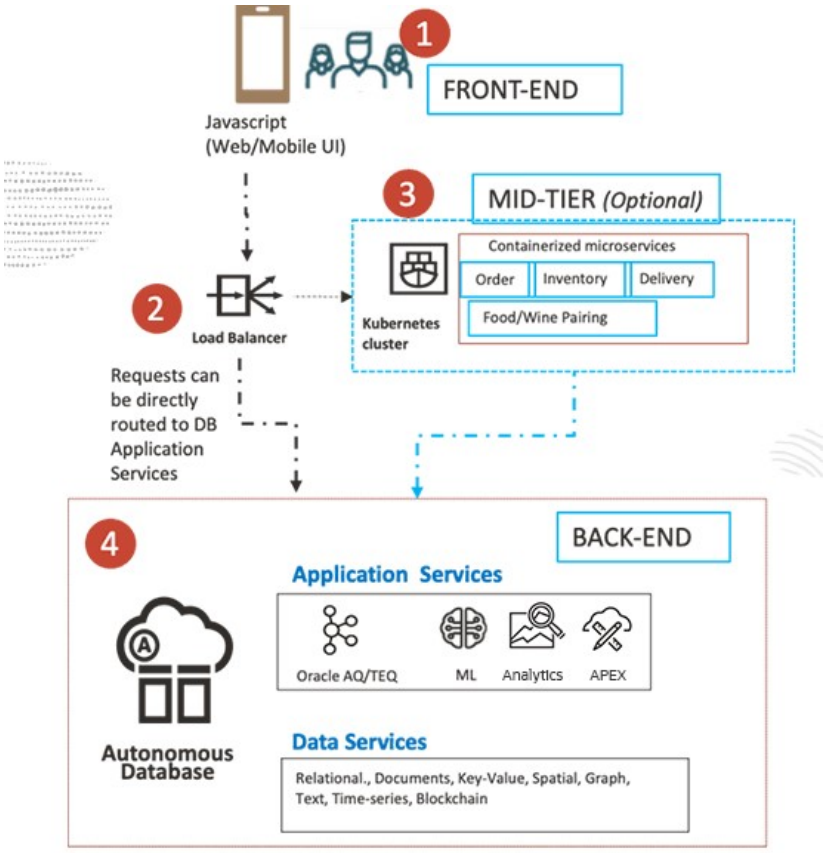
4. Transitive Functional Dependency

In transitive functional dependency, dependent is indirectly dependent on determinant. i.e. If a → b & b → c, then according to axiom of transitivity, a → c. This is a transitive functional dependency
Example,

enrol_no	name	dept	building_no
42	abc	CO	4
43	pqr	EC	2
44	xyz	IT	1
45	abc	EC	2

Here, enrol_no → dept and dept → building_no,
Hence, according to the axiom of transitivity, enrol_no → building_no is a valid functional dependency. This is an indirect functional dependency, hence called Transitive functional dependency.

7. Build a mobile food delivery app with a data-centric architecture pattern using Kubernetes engine, Docker containers, and Oracle Autonomous Database.using Modern AppDev



PART B (14 Marks)

1.The following tables form part of a database held in a relational DBMS:

Employee (empno, name, office, age)
Books (isbn, title, authors, publisher)
Loan (empno, isbn, date)
Write the following queries in SQL

1. **Find the name of all employees who have borrowed a book published by McGraw-Hill.**
2. **Find the name of all employees who have borrowed all book published by McGraw-Hill.**
3. **Find the names of employees who have borrowed more than five different books published by McGraw-Hill.**
4. **For each publisher, find the name of employees who have borrowed more than five books of that publisher**

Ans:

a. select name from employee e, books b, loan l where e.empno = l.empno and l.isbn = b.isbn and b.publisher = 'McGrawHill'

b. select name from employee e join loan l on e.empno=l.empno join (select isbn from books where publisher = 'McGrawHill') x on l.isbn=x.isbn group by e.empno,name having count(*)= (select count(*) from books where publisher='McGrawHill')

c. select name from employee,loan where employee.empno=loan.empno and isbn in (select distinct isbn from books where publisher='McGraw-Hill') group by employee.empno,name having count(isbn) >=5

d. select name from employee,loan,books where employee.empno=loan.empno and books.isbn=loan.isbn group by employee.empno, name,books.publisher having count(loan.isbn) >=5

2.i. Convert the Relation R(A,B,C,D) and functional Dependencies (A→B, B →C) from INF to 2NF.

Refer Classwork Notes

2.ii. Consider the Relation R(A,B,C,D,E) and Functional Dependencies F= {A →BCDE, BC→ACE, D→E}. Find out the highest normal from given in the relation

Refer Classwork Notes

3. **Consider the following relation schema**
Works (Pname,Cname,salary)
Lives (Pname,Street,City)
located in (Cname, city)
Manager (Pname,Mgrname)
Write the SQL queries for the following

i) **List the names of the people who work for company Wipro along with the cities they live in**
ii) **Find the people who work for the company "Infosys" with salary more than Rs. 50000/-. List the names of the people, along with the streets and city addresses.**
iii) **Find the names of the persons who live and work in the same city**
iv) **Find the names of the person who do not work for "Infosys"**
v) **Find the average salary of "Infosys" persons**

1. SELECT L.PName, City FROM Works W, Lives L Where CName = "Wipro" and W. PName = L.Pname;

2. SELECT L.PName, City, Street FROM WORKS W, Lives L Where W. CName = 'Infosys' AND W.PName = L. CName AND Salary > 50000;

3. SELECT W.PName FROM Works W, Lives L, LocatedIn I Where W.CName = I.CName AND W.PName = L.PName AND I.City = L.City

4. SELECT PName FROM Works Where CName ! = 'Infoysy';

5. SELECT Avg(salary) FROM Works Where CNAME = "Infosys";

4.Derive the secondary rules for the given axioms

1. **Union**

   If A → B holds and A → C holds, then A →BC holds. If X →Y and X → Z then X → YZ

   Proof

   1. X → Y (given)
   2. X → Z (given)
   3. X → XY (using IR_2 on 1 by augmentation with X. Where XX = X)
   4. XY → YZ (using IR_2 on 2 by augmentation with Y)
   5. X → YZ (using IR_3 on 3 and 4)

2. **Composition**

   If A → B and X → Y holds, then AX → BY holds.

   Proof

   A→B_____(i)
   X→Y_____(ii)
   AX→BX_____(iii) (Augmentation of i and C)
   AX→B_____(iv) Decomposition of iii)
   AX→AY_____(v) (Augmentation of ii and A)
   AX→Y_____(vi) (Decomposition of v)
   AX→BY_____ (Union iv and vi)

3. **Decomposition**

   If A → BC holds then A → B and A → C hold. If X → YZ then X → Y and X → Z

   Proof

   1. A → BC (given)
   2. BC → B (using IR1 Rule)
   3. A → B (using IR3 on 1 and 2)