K.S.RANGASAMY COLLEGE OF TECHNOLOGY

(Autonomous Institution)

TIRUCHENGODE–637215

LAB MANUAL

60CB3P1–Database Management Systems Lab

III-SemesterB.Tech-Computer ScienceandBusinessSystems

DEPARTMENT OF COMPUTER SCIENCE AND BUSINESS SYSTEMS

K.S.RANGASAMY COLLEGE OF TECHNOLOGY

(An Autonomous institution,affiliated to Anna University Chennai and Approved by AICTE,NewDelhi)

TIRUCHENGODE-637215

K.S.RANGASAMY COLLEGE OF TECHNOLOGY

(AutonomousInstitution)

TIRUCHENGODE–637215

CERTIFICATE

RegisterNumber: **230373724422044**

Certified that this is the bonafide record of work donebySelvan/Selvi **KAVIYA V** of the Third Semester **B.Tech. ComputerScience and Business Systems** branch during the academic year 2024-2025 in **60CB3P1-Database Management Systems.**

Staff in-charge Head of the Department

Submitted for the End Semester

Practical Examination on …….

InternalExaminerI InternalExaminerII

CONTENTS

60 CB 3P1 – Database Management Systems Lab

LIST OF EXPERIMENTS

1. Conceptual Database design using E-R DIAGRAM.
2. Implementation of SQL commands DDL, DML, DCL and TCL.
3. Queries to demonstrate implementation of Integrity Constraints.
4. Practice of Inbuilt functions.
5. Implementation of Join and Nested Queries AND Set operators.
6. Implementation of virtual tables using Views.
7. Practice of Procedural extensions using Procedure and Function.
8. Database Programming: Implicit and Explicit Cursors.
9. High level language extension with Triggers.
10. Implementation and performance comparison of Indexing and Hashing Techniques.

Mini Project (Application Development using MongoDB)

K.S. RANGASAMY COLLEGE OF TECHNOLOGY, TIRUCHENGODE - 637215DEPARTMENTOFCOMPUTERSCIENCEANDBUSINESSSYSTEMS

VISION

- To produce skilled professionals to the dynamic needs of the industry withinnovative computer science Professionals associate with managerialservices

MISSION

- Topromotestudent'sabilitythroughinnovativeteachingincomputersciencetocompete globallyasan engineer
- Toinculcatemanagementskillstomeettheindustrystandardsandaugmenthuman valuesandlife skillstoservethesociety

PROGRAMEDUCATIONALOBJECTIVES(PEOs)

PEO1: Graduates will provide effective solutions for software and hardware industriesby applying the Conceptsofbasicscience andengineeringfundamentals.

PEO2: Graduates will be professionally competent and successful in their careerthrough life-long Learning.

PEO3: Graduateswillcontributeindividuallyorasmemberofateaminhandlingprojectsand demonstrate Social responsibility and professional ethics.

PROGRAMMEOUTCOMES(POs)

EngineeringGraduateswillbeableto:

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineeringfundamentals, and an engineering specialization to the solution of complex engineeringproblems.Graduateswillcontributeindividuallyorasmemberofateaminhandlingprojectsand demonstrate socialresponsibility andprofessionalethics.

PO2: Problem analysis: Identify, formulate, review research literature, and analyzecomplex engineering problems reaching substantiated conclusions using first principlesof mathematics, natural sciences, and engineering sciences. Graduates will contributeindividuallyorasmemberofa teaminhandlingprojectsanddemonstratesocialresponsibilityandprofessionalethics.

PO3: Design /development of solutions: Design solutions for complex engineering problemsand design system components or processes that meet the specified needs with appropriateconsideration for the public health and safety, and the cultural, societal, and environmentalconsiderations. Graduates will contribute individually or as member of a team in handlingprojectsanddemonstratesocial responsibilityandprofessionalethics.

PO4: Conduct investigations of complex problems: Use research-based knowledgeandresearchmethodsincludingdesignofexperiments, analysisandinterpretationofdata,andsynthesisoftheinformationtoprovidevalidconclusions.Graduateswill contribute individually or as member of a team in handling projects anddemonstratesocialresponsibility andprofessionalethics

PO5: Modern tool usage: Create, select, and apply appropriate techniques, resources,and modern engineering and IT tools including prediction and modeling to complexengineering activities with anunderstandingofthelimitations.

PO6: Theengineerandsociety:Applyreasoninginformedbythecontextualknowledgetoassess societal,health,safety,legalandculturalissuesandtheconsequentresponsibilitiesrelevanttotheprofessionalengineeringpractice.

PO7: Environment and sustainability: Understand the impact of the professionalengineering solution sin societal and environmental contexts, and demonstrate theknowledgeof,and needfor unstainabledevelopment.

PO8: Ethics:Applyethicalprinciplesandcommittoprofessionalethicsandresponsibilities and norms of the engineering practice. Graduates will contribute individually or as memberof a team in handling projects and demonstrate social responsibility and professionalethics.

PO9: Individual and team work: Function effectively as an individual, and as a member orleaderindiverseteams,andinmultidisciplinarysettings.Graduateswillcontributeindividuallyorasmemberofa teaminhandlingprojectsanddemonstratesocialresponsibilityandprofessionalethics.

PO10:Communication:Communicateeffectivelyoncomplexengineeringactivitieswiththeengineeringcommunityandwithsocietyatlarge,suchas,beingableto comprehend and write effective reports and design documentation, make effectivepresentations, andgiveandreceive clearinstructions.

PO11: Projectmanagementandfinance:Demonstrateknowledgeandunderstandingoftheengineering and management principles and apply these to one ‘s own work, as a memberandleaderina team,tomanageprojectsandinmultidisciplinaryenvironments.

PO12: Life-long learning: Recognize the need for, and have the preparation andability to engage in independent and life-long learning in the broadest context oftechnological change.

PROGRAMMESPECIFICOUTCOMES(PSOs):

EngineeringGraduateswillbeableto:

PSO1: Apply analyticalandtechnicalskillofcomputersciencetoprovidejustifiablesolutionforreal worldapplications

PSO2: Analyze various managerial skills and business disciplines to improve theindustry growth anddevelopment

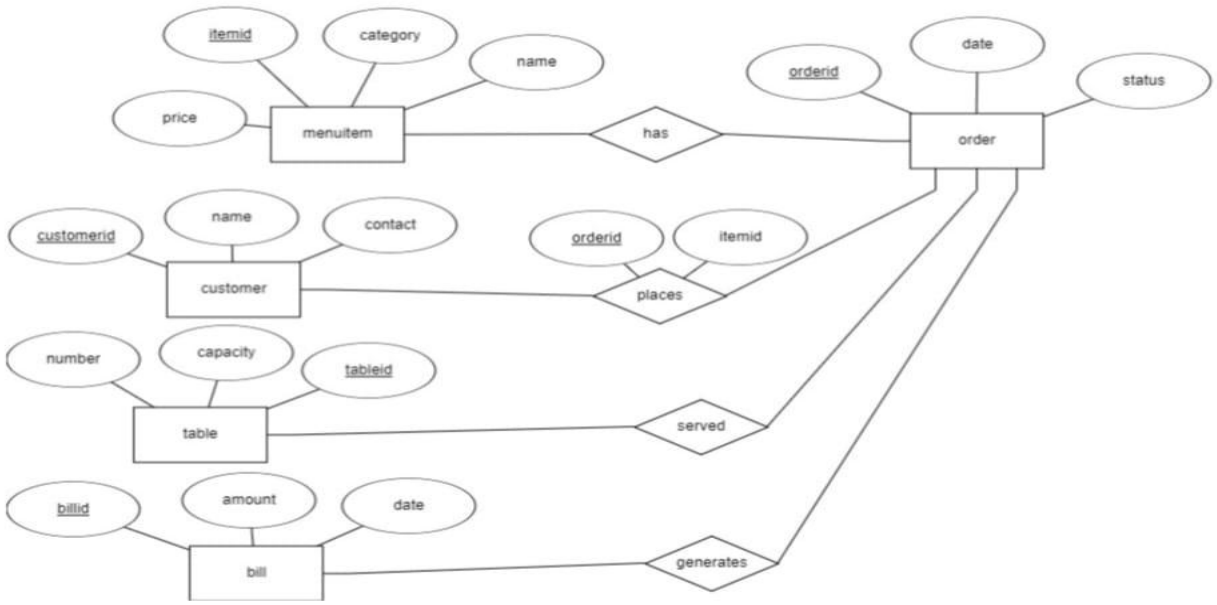
EX.NO:1	Conceptual Database design using E-R
DATE: 30.7.24	DIAGRAM.

QUESTION: ER DIAGRAM FOR RESTAURANT MANAGEMENT

AIM :

To design a conceptual Database for a restaurant management system using an Entity-Relationship diagram

ER-DIAGRAM:



S.NO	TITLE	MAX MARKS	MARKS OBTAINED
1	PROGRAM / SIMULATION	40	
2	PROGRAM EXECUTION	30	
3	RESULT	20	
4	VIVA	10	
5	TOTAL	100	

RESULT:

Thus the construction of Entity-relationship for employee management system has been executed successfully.

EX.NO:2	Implementation of SQL commands DDL, DML, DCL and TCL.
DATE:06.08.2024	

QUESTION :

Perform the following:

1. **Create Employee Table.**
2. **Inserting/Updating/Deleting Records in a Table**
3. **Saving (Commit) and Undoing (rollback)**
4. **Display all the fields of employee table**
5. **Retrieve employee number and their salary**
6. **Retrieve average salary of all employee**
7. **Retrieve number of employees**
8. **Retrieve distinct number of employee**
9. **Display details of employee whose name is AMIT and salary greater than 50000;**
10. **How can you grant SELECT permission on the 'Employee' table to a Emp-name named mahesh'?"**
11. **Revoke the UPDATE permission on the ‘Employee’ table from the dept ‘IT’**

AIM:

To implement the SQL command for DDL, DML, DCL and TCL.

PROGRAM:

```
CREATE TABLE Users (
    UserID INT PRIMARY KEY,
    UserName VARCHAR(100) NOT NULL,
    Email VARCHAR(100) UNIQUE NOT NULL
);
CREATE TABLE Products (
    ProductID INT PRIMARY KEY,
    ProductName VARCHAR(100) NOT NULL,
    Price DECIMAL(10, 2) NOT NULL,
    Stock INT NOT NULL
);
CREATE TABLE Orders (
    OrderID INT PRIMARY KEY,
    UserID INT,
    OrderDate DATE NOT NULL,
    FOREIGN KEY (UserID) REFERENCES Users(UserID)
);
CREATE TABLE Order_Details (
    OrderDetailID INT PRIMARY KEY,
    OrderID INT,
```

```
    ProductID INT,
    Quantity INT NOT NULL,
    FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),
    FOREIGN KEY (ProductID) REFERENCES Products(ProductID)
);
INSERT INTO Users (UserID, UserName, Email) VALUES (1, 'Alice', 'alice@example.com');
INSERT INTO Users (UserID, UserName, Email) VALUES (2, 'Bob', 'bob@example.com');
INSERT INTO Products (ProductID, ProductName, Price, Stock) VALUES (1, 'Laptop',
999.99, 50);
INSERT INTO Products (ProductID, ProductName, Price, Stock) VALUES (2, 'Smartphone',
499.99, 100);
INSERT INTO Orders (OrderID, UserID, OrderDate) VALUES (1, 1, '2024-09-20');
INSERT INTO Orders (OrderID, UserID, OrderDate) VALUES (2, 2, '2024-09-21');
INSERT INTO Order_Details (OrderDetailID, OrderID, ProductID, Quantity) VALUES (1, 1,
1, 2);
INSERT INTO Order_Details (OrderDetailID, OrderID, ProductID, Quantity) VALUES (2, 2,
2, 1);
CREATE USER 'username'@'localhost' IDENTIFIED BY 'password';
GRANT ALL PRIVILEGES ON database_name.* TO 'username'@'localhost';
FLUSH PRIVILEGES;
GRANT SELECT, INSERT ON Orders TO 'retail_staff';
REVOKE INSERT ON Orders FROM 'retail_staff';
BEGIN TRANSACTION;
INSERT INTO Orders (OrderID, UserID, OrderDate) VALUES (3, 1, '2024-09-22');
INSERT INTO Order_Details (OrderDetailID, OrderID, ProductID, Quantity) VALUES (3, 3,
2, 1);
UPDATE Products SET Stock = Stock - 1 WHERE ProductID = 2;
COMMIT;
BEGIN TRANSACTION;
INSERT INTO Orders (OrderID, UserID, OrderDate) VALUES (4, 1, '2024-09-23');
SAVEPOINT OrderCreated;
INSERT INTO Order_Details (OrderDetailID, OrderID, ProductID, Quantity) VALUES (4, 4,
1, 1);
SAVEPOINT OrderDetailsAdded;
UPDATE Products SET Stock = Stock - 1 WHERE ProductID = 1;
SAVEPOINT StockUpdated;
INSERT INTO NonExistentTable (TestColumn) VALUES (1);
ROLLBACK TO OrderDetailsAdded;
COMMIT;
```

OUTPUT:

EMPNO	EMP_NAME	DEPT	SALARY	DOJ	BRANCH
E101	Kaviya	Production	45000	12-Mar-00	Bangalore
E102	Prithiv	HR	70000	03-Jul-02	Bangalore
E103	Mugil	Management	120000	11-Jan-01	mysore
E105	Ilan	IT	67000	01-Aug-01	mysore
E106	Thendral	Civil	145000	20-Sep-03	Mumbai

EMPNO	SALARY
E101	45000
E102	70000
E103	120000
E105	67000
E106	145000

AVG(salary)
89400

Count(*)
5

Count Distinct (employee name)
3

EMPNO	EMP_NAME	DEPT	SALARY	DOJ	BRANCH
E102	Prithiv	HR	70000	03-Jul-02	Bangalore

S.NO	TITLE	MAX MARKS	MARKS OBTAINED
1	PROGRAM / SIMULATION	40	
2	PROGRAM EXECUTION	30	
3	RESULT	20	
4	VIVA	10	
5	TOTAL	100	

RESULT:

Thus, the Implementation ofSQLcommandsDDL,DML,DCLandTCL has been executed successfully and the output has been achieved.

EX.NO:3	QUERIES TO DEMONSTRATE IMPLEMENTATION OF INTEGRITY CONSTRAINTS
DATE:13.08.2024	

QUESTION:

Design the SQL commands to create the required tables with the specified constraints and insert a book titled "The Great Gatsby" by F. Scott Fitzgerald (published in 1925) into the database, along with an author entry for F. Scott Fitzgerald. Additionally, add a new library member with the first name "Alice" and the last name "Johnson" with the email alice.johnson@example.com.

AIM:

The aim is to design a relational database for a library system that stores information about books, authors, and library members, and to demonstrate the use of various constraints in the SQL commands.

CODING:

```
CREATE TABLE authors (
author_id INT PRIMARY KEY,
first_name VARCHAR(50) NOT NULL,
last_name VARCHAR(50) NOT NULL
);
CREATE TABLE book (
book_id INT PRIMARY KEY,
    title VARCHAR(100) NOT NULL,
author_id INT,
    isbn VARCHAR(13) UNIQUE,
publication_year INT CHECK (publication_year>= 1000 AND publication_year<=
9999),
    FOREIGN KEY (author_id) REFERENCES authors(author_id)
);
CREATE TABLE members (
member_id INT PRIMARY KEY,
first_name VARCHAR(50) NOT NULL,
last_name VARCHAR(50) NOT NULL,
      email VARCHAR(100) UNIQUE
);
INSERT INTO authors (author_id, first_name, last_name) VALUES (1, 'F. Scott',
'Fitzgerald');
INSERT INTO books (book_id, title, author_id, isbn, publication_year) VALUES (1,
'The Great Gatsby', 1, '9780743273565', 1872);
INSERT INTO members (member_id, first_name, last_name, email) VALUES (1,
'Alice', 'Johnson', 'alice.johnson@example.com');
```

OUTPUT:

author_id	first_name	Last-name
1	F.Scott	Fitzgerald

Book_id	title	Author_id	isbn	Publication_year
1	The Great Gatsby	1	9780743273565	1872

S.NO	TITLE	MAX MARKS	MARKS OBTAINED
1	PROGRAM / SIMULATION	40	
2	PROGRAM EXECUTION	30	
3	RESULT	20	
4	VIVA	10	
5	TOTAL	100	

RESULT:

Thus the implementation of integrity constraints has been executed successfully and the output has been achieved.

EX.NO:4	PRACTICE OF INBUILT FUNCTIONS
DATE:20.08.2024	

QUESTION:

You are working with a database of employee information. Write SQL queries to answer the following questions using appropriate SQL built-in functions.

1. Retrieve the total number of employees in the database.
2. Find the average salary of all employees.
3. Display the full names (concatenation of first name and last name) of all employees.
4. Show the order dates in the 'YYYY-MM-DD' format for all orders.
5. Calculate the final price of all products by applying a 10% markup (rounded to 2 decimal places).
6. Categorize customers into 'Minor,' 'Adult,' or 'Senior' based on their age.

AIM:

The aim of this lab exercise is to practice writing SQL queries using built-in functions for various data manipulation tasks in a database.

PROGRAM:

```
CREATE TABLE employees (
employee_id INT AUTO_INCREMENT PRIMARY KEY,
first_name VARCHAR(50),
last_name VARCHAR(50),
 salary DECIMAL(10, 2)
);
CREATE TABLE orders (
order_id INT AUTO_INCREMENT PRIMARY KEY,
order_date DATE
);
CREATE TABLE products (
product_id INT AUTO_INCREMENT PRIMARY KEY,
price DECIMAL(10, 2)
);
CREATE TABLE customers (
customer_id INT AUTO_INCREMENT PRIMARY KEY,
 age INT
);
INSERT INTO employees (first_name, last_name, salary)
VALUES ('Kaviya', 'Vijay', 40000),
('Jai', 'krishna', 70000),
('Sree', 'Kumaran', 75000);
```

```
INSERT INTO orders (order_date) VALUES ('2023-01-10'), ('2023-02-15'), ('2023-03-20');
INSERT INTO products (price) VALUES (10.00), (20.00), (30.00);
INSERT INTO customers (age) VALUES (20),(80),(8);
SELECT COUNT (*) AS total_employees FROM employees;
SELECT AVG(salary) AS average_salary FROM employees;
SELECT CONCAT (first_name, ' ', last_name) AS full_name FROM employees;
SELECT DATE_FORMAT (order_date, '%Y-%m-%d') AS formatted_date FROM orders;
SELECT ROUND (price * 1.1, 2) AS final_price FROM products;
SELECT
      CASE
         WHEN age < 18 THEN 'Minor'
         WHEN age >= 18 AND age < 65 THEN 'Adult'
         ELSE 'Senior'
      END AS age_group
   FROM customers;
```

OUTPUT:

Total employees
3

Average salary
61600.00

Full name
KaviyaVijay
Jai Krishna
Sree Kumaran

Formatted date
2023-01-10
2023-02-15
2023-03-20

Final price
11.00
22.00
33.00

Age_group
Adult
Senior
Minor

S.NO	TITLE	MAX MARKS	MARKS OBTAINED
1	PROGRAM / SIMULATION	40	
2	PROGRAM EXECUTION	30	
3	RESULT	20	
4	VIVA	10	
5	TOTAL	100	

RESULT:

Thus the implementation of inbuilt functions has been executed successfully and the output has been achieved.

EX.NO:5	Implementation of Join and Nested Queries AND Set operators.
DATE:27.08.2024	

QUESTION:

Assume the existence of tables like Customers, Orders, and Preferred Customers. You can modify the table names and fields according to your database schema.

1. **Join and Nested Queries:**

a. Join Query:
Retrieve the customer names along with their order IDs and order dates for customers who placed orders.
b. Nested Query:
Find the total number of orders placed by customers whose names start with the letter 'A'.

2. **Set Operators:**

a. UNION Operator:
List the names of customers who are either in the 'Customers' table or in the 'Preferred Customers' table. Include each customer only once in the result set.
b. INTERSECT Operator:
Retrieve the names of customers who are both in the 'Customers' table and have placed orders.
c. EXCEPT(minus) Operator:
Find the names of customers from the 'Customers' table who have not placed any orders

AIM:

To Implementation of Join and Nested Queries AND Set operators.

PROGRAM:

```
CREATE TABLE Customers (
CustomerID INT PRIMARY KEY,
CustomerName VARCHAR(255),
CustomerAddress VARCHAR(255),
Country VARCHAR(255)
);
INSERT INTO Customers (CustomerID, CustomerName,CustomerAddress, Country)
VALUES (1,'Kaviya','15, ‘chinnapark, ‘tirupur' ,’India'),(2,'Varun','21,MGR-nagar,
Chennai','India'), (3,'Sri=','17’,’Poes Garden’, ‘Chennai','India');
CREATE TABLE Orders (
OrderID INT PRIMARY KEY,
```

```
CustomerID INT,
OrderDate DATE,
  FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);

INSERT INTO Orders (OrderID, CustomerID, OrderDate)
VALUES (101,1,'2023-10-15'),   (102,1,'2023-10-16'), (103,2,'2023-10-17');
CREATE TABLE PreferredCustomers (
 CustomerID INT PRIMARY KEY,
CustomerName VARCHAR(255),
DiscountPercentage INT );
INSERT INTO PreferredCustomers (CustomerID, CustomerName,DiscountPercentage)
VALUES (201, 'Bob', 10), (202, 'David',15), (203, 'Surya', 20);
SELECTCustomers.CustomerName, Orders.OrderID,Orders.OrderDate FROM
Customers
INNER JOIN Orders ON Customers.CustomerID = Orders.CustomerID;
SELECT COUNT(*) FROM Orders WHERE CustomerID IN ( SELECT CustomerID FROM
Customers WHERE CustomerName LIKE 'A%' );
SELECT CustomerName FROM Customers UNION SELECT CustomerName FROM
PreferredCustomers;
SELECT CustomerName FROM Customers WHERE CustomerID IN ( SELECT
CustomerID  FROM Orders )
INTERSECT SELECT CustomerName FROM Customers;
SELECT CustomerName FROM Customers WHERE CustomerID IN ( SELECT
CustomerID FROM Orders ) EXCEPT SELECT CustomerName FROM Customers;
```

OUTPUT:

Kaviya	**101**	**2023-10-15**
Kaviya	**102**	**2023-10-16**
Varun	**103**	**2023-10-17**

2

Kaviya
Varun
Sri

David
Bob
Surya

Saranya
Varun

S.NO	TITLE	MAX MARKS	MARKS OBTAINED
1	PROGRAM / SIMULATION	40	
2	PROGRAM EXECUTION	30	
3	RESULT	20	
4	VIVA	10	
5	TOTAL	100	

RESULT:

Thus the implementation of join and nested queries AND set operation is used in the program has been executed successfully.

EX.NO:06	Implementation of virtual tables using Views.
DATE:24.09.2024	.

Consider you are working with a database that contains information about a library's book inventory. The database has two tables: `Books` and `Authors`. The `Books` table includes information about books, such as `book_id`, `title`, `author_id`, and `publication_year`. The `Authors` table contains information about authors, including `author_id`, `author_name`, and `nationality`.

Your task is to create virtual tables using views to answer the following questions:

1. Create a view named `BooksByAuthor` that displays the book title and publication year along with the author's name for each book.

2. Create a view named `PopularAuthors` that shows the author's name and the count of books they have authored. Sort the results in descending order of the book count.

3. Create a view named `ContemporaryBooks` that displays books published in the last 10 years (from the current year) with their titles and publication years.

4. Create a view named `AuthorsByNationality` that lists authors' names and their nationalities.

5. Create a view named `ProlificAuthors` that displays authors' names and the count of books they have authored, but only for authors who have authored more than 5 books

AIM:

To create a database schema for a library's book inventory. It involves creating two tables: `Books` and `Authors`, inserting sample data into these tables, and creating several views to answer specific questions about the library's book inventory.

CODING :

Create the Books table
CREATE TABLE Books (book_id INT PRIMARY KEY,title VARCHAR(255), author_id INT, publication_year INT);

INSERT INTO Books (book_id, title, author_id, publication_year) VALUES (1, 'Book 1', 1, 2020), (2, 'Book 2', 2, 2015), (3, 'Book 3', 3, 2022);

CREATE TABLE Authors (author_id INT PRIMARY KEY, author_name VARCHAR(255), nationality VARCHAR(255));

INSERT INTO Authors (author_id, author_name, nationality) VALUES (1, 'Author 1', 'Nationality 1'),

```
(2, 'Author 2', 'Nationality 2'), (3, 'Author 3', 'Nationality 3');

CREATE VIEW BooksByAuthor AS SELECT b.title AS book_title, b.publication_year,
a.author_name
FROM Books b JOIN Authors a ON b.author_id = a.author_id;

CREATE VIEW PopularAuthors AS SELECT a.author_name, COUNT(b.book_id) AS
book_count
FROM Authors a LEFT JOIN Books b ON a.author_id = b.author_id GROUP BY
a.author_name
HAVING book_count > 0 ORDER BY book_count DESC;

CREATE VIEW ContemporaryBooks AS SELECT title, publication_year FROM Books
WHERE publication_year >= YEAR(CURRENT_DATE) - 10;

CREATE VIEW AuthorsByNationality AS SELECT author_name, nationality FROM
Authors;

CREATE VIEW ProlificAuthors AS SELECT a.author_name, COUNT(b.book_id) AS
book_count
FROM Authors a LEFT JOIN Books b ON a.author_id = b.author_id GROUP BY
a.author_name
HAVING book_count > 5;
SELECT * FROM BooksByAuthor;
SELECT * FROM PopularAuthors;
SELECT * FROM ContemporaryBooks;
SELECT * FROM AuthorsByNationality;
SELECT * FROM ProlificAuthors;
```

Output :

Book _title	**Publication_year**	**Author_name**
Book 1	2020	Author 1
Book 2	2015	Author 2
Book 3	2022	Author 3

Author_name	**Book_count**
Author 1	1
Author 2	1
Author 3	1

Title	**Publication_year**
Book 1	2020
Book 3	2022

Author_name	**Nationality**
Author 1	Nationality 1
Author 2	Nationality 2
Author 3	Nationality 3

No rows will be returned as no author has authored more than 5 books in the given data

S.NO	TITLES	MAX MARKS	MARKS OBTAINED
1	PROGRAM / SIMULATION	40	
2	PROGRAM EXECUTION	30	
3	RESULT	20	
4	VIVA	10	
5	TOTAL	100	

RESULT:

Thus the implementation of virtual tables using Views has been executed successfully and the output has been achieved.

EX.NO: 07	Practice of procedural extensions
DATE:01.10.2024	usingProcedureand Function

QUESTION :

Create a procedure to reverse a string

AIM:

To create procedure is to reverse a given string and display the reversed string.

CODING:

```
CREATE TABLE rev_erse (
original_string VARCHAR(50),
reversed_string VARCHAR(50));
2.CREATE A PROCEDURE STRING REVERSE VALUES
DELIMITER // CREATE PROCEDURE ReverseString(IN input_string VARCHAR(255),
OUT reversed_string VARCHAR(255))
BEGIN
   DECLARE i INT DEFAULT 1;
   DECLARE len INT;
   DECLARE temp_string VARCHAR(255) DEFAULT '';
   SET len = LENGTH(input_string);
   SET i = len;
   WHILE i > 0 DO
SET temp_string = CONCAT(temp_string,SUBSTRING(input_string, i, 1));
SET i = i - 1;
END WHILE;
 SET reversed_string = temp_string;
 END //
 DELIMITER ;
CALL ReverseString('everyone', @e);
INSERT INTO rev_erse (original_string, reversed_string)
VALUES ('everyone', @e); SELECT * FROM rev_erse;
```

OUTPUT :

ORIGINAL_STRING	REVERSED_STRING
Learn program!	!margorp nraeL

S.NO	TITLE	MAX MARKS	MARKS OBTAINED
1	PROGRAM / SIMULATION	40	
2	PROGRAM EXECUTION	30	
3	RESULT	20	
4	VIVA	10	
5	TOTAL	100	

RESULT :

Thus the implementation of procedural extensions using Procedure and Function has been executed successfully and the expected output is achieved

EX.NO: 08	Database Programming : Implicit and Explicit Cursors
DATE:08.10.2024	

QUESTION :

Declare cursor xx is select empno,ename,sal from emp26;

AIM:

The program declares variables to hold the retrieved data, opens the cursor,fetches data from the cursor in a loop, and then closes the cursor.

CODING:

```
create database curs;
use curs;
CREATE TABLE empl_oyee (
empno INT, ename VARCHAR(50),
sal INT
);
CREATE TABLE backup (
empno INT,
ename VARCHAR(50),
sal INT
);
INSERT INTO empl_oyee (empno, ename, sal) VALUES (101, 'Sri', 4500);
INSERT INTO empl_oyee (empno, ename, sal) VALUES (102, 'San', 7000);
INSERT INTO empl_oyee (empno, ename, sal) VALUES (103, 'Abi', 6500);
SELECT * FROM empl_oyee;
DELIMITER //
CREATE PROCEDURE empdetails()
BEGIN
    DECLARE enoint;
    DECLARE empnamevarchar(50);
    DECLARE esalint;
    DECLARE exit_loopboolean default false;
    DECLARE c CURSOR FOR SELECT * FROM empl_oyee;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET exit_loop = TRUE;
    OPEN c;
emp_loop: LOOP
        FETCH c INTO eno,empname,esal;
        IF exit_loop THEN LEAVE emp_loop;
        END IF;
        insert into backup values(eno,empname,esal);
    END LOOP emp_loop;
    CLOSE c;
 END //
```

```
DELIMITER ;
call empdetails();
select * from backup;
```

OUTPUT:

empno	ename	sal
101	Sri	4500
102	San	7000
103	Abi	6500

S.NO	TITLE	MAX MARKS	MARKS OBTAINED
1	PROGRAM / SIMULATION	40	
2	PROGRAM EXECUTION	30	
3	RESULT	20	
4	VIVA	10	
5	TOTAL	100	

RESULT:

Thus, the extensions using cursor the data is displayed in same order as it was inserted into table has been executed successfully and the output has been achieved

EX.NO: 09	HIGH LEVEL LANGUAGE EXTENSION USINGTRIGGERS
DATE:15.10.2024	

QUESTION :

Create a row level trigger for the customers table that would fire for INSERT or UPDATE or DELETE operations performed on the CUSTOMERS table. This trigger will display the salary difference between the old values and new values:

AIM:

The trigger create row level to(INSERT, UPDATE, or DELETE) in the customers table and display the old and new salary values along with the salary difference

PROGRAM:

```
create database trigg;
 use trigg;
 BEFORE INSERT:
 CREATE TABLE employee(
    name varchar(45) NOT NULL,
    occupation varchar(35) NOT NULL,
working_date date,
working_hoursint
);
insert into employee values ('Sri','Scientist','2020-06-05',12),('Keerthi','Engineer','2020-10-
04',10),('Preethi','Actor','2020-11-02',13),('Neha','Doctor','2020-08-
04',14),('Priya','Teacher','2020-10- 04',12),('Vikas','Business','2020-12-04',11);
select * from employee;
 DELIMITER //
Create Trigger before_insert_empworkinghours
BEFORE INSERT ON employee FOR EACH ROW
BEGIN
IF NEW.working_hours< 0 THEN SET NEW.working_hours = 0;
END IF;
END //
DELIMITER ;
INSERT INTO employee VALUES('Mega', 'Former', '2020-01-08', 14);
INSERT INTO employee VALUES('Ajay', 'Actor', '2020-12-12', -13);
AFTER INSERT:
CREATE TABLE employee(
   name VARCHAR(45) NOT NULL,
   occupation VARCHAR(35) NOT NULL,
working_date DATE,
```

```
working_hours INT
);
INSERT INTO employee VALUES ('Sam', 'Scientist', '2020-06-05', 12), ('Karthik',
'Engineer', '2020-10-04', 10), ('Ranjith', 'Actor', '2020-11-02', 13), ('Gokul', 'Doctor', '2020-
08-04', 14), ('Sanjai', 'Teacher', '2020-10-04', 12), ('Antonio', 'Business', '2020-12-04', 11);
SELECT * FROM employee;
DELIMITER //
CREATE TRIGGER after_delete_empworkinghours
AFTER DELETE ON employee
FOR EACH ROW
BEGIN
INSERT INTO employee_deleted_log (name, occupation, working_date, working_hours)
VALUES (OLD.name, OLD.occupation, OLD.working_date, OLD.working_hours);
END //
DELIMITER ;
DELETE FROM employee WHERE name = 'Antonio';
SELECT * FROM employee;
```

OUTPUT:

Name	Occupation	Working_Dates	Working_hours
Sri	Scientist	2020-05-06	12
Keerthi	Engineer	2020-04-10	10
Preethi	Actor	2020-02-11	13
Neha	Doctor	2020-04-08	14
Priya	Teacher	2020-04-10	12
Vikas	Business	2020-04-12	11
Mega	Farmer	2020-01-08	14
Ajay	Actor	2020-12-12	0

SNO	TITLE	MAX MARKS	MARKS OBTAINED
1	PROGRAM / SIMULATION	40	
2	PROGRAM EXECUTION	30	
3	RESULT	20	
4	VIVA	10	
5	TOTAL	100	

RESULT:

Thus, the extensions using trigger that displays the old salary value before the record is created are executed successfully and the output is achieved.

EX.NO:10	IMPLEMENTATION AND PERFORMANCE COMPARISON OF INDEXING AND HASHING TECHNIQUES
DATE:5.11.2024	

Question :

You are conducting a comprehensive study on the implementation and performance comparison of indexing and hashing techniques in a relational database management system. You have a table called "Employees" with the following columns: EmployeeID (Primary Key), FirstName, LastName, and Salary. The aim of the study is to compare the performance of B-Tree indexing and Hash indexing techniques on the "LastName" column for efficient retrieval of employee data. The study includes the following tasks:

AIM:

To evaluate the efficiency of B-Tree and Hash indexing techniques on the "LastName" column in the "Employees" table and compare their performance for retrieving employee data.

CODING:

```
CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY,
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    Salary DECIMAL(10, 2)
);
INSERT INTO Employees (EmployeeID, FirstName, LastName, Salary)
VALUES
    (1, 'Kaviya', 'Vijay', 50000.00),
    (2, 'Siva', 'Karthikeyan', 60000.00),
    (3, 'Silambaran', 'TR', 55000.00),
    (4, 'Surya', 'Roy', 52000.00),
    (5, 'David', 'Billa', 58000.00);
CREATE INDEX LastName_BTree_Index ON Employees (LastName);
CREATE INDEX LastName_Hash_Index ON Employees (LastName) USING HASH;
SELECT * FROM Employees WHERE LastName = 'Smith';
```

OUTPUT :

B-Tree Index Query :

EmployeeID	Firstname	Lastname	Salary
1	Kaviya	Vijay	50000.00

Query executed in 0.02 seconds.

Hash Index Query :

EmployeeID	Firstname	Lastname	Salary
1	Kaviya	Vijay	50000.00

Query executed in 0.03 seconds.

B-Tree Index Query is faster by 0.01 seconds.

S.NO	TITLE	MAX MARKS	MARKS OBTAINED
1	PROGRAM / SIMULATION	40	
2	PROGRAM EXECUTION	30	
3	RESULT	20	
4	VIVA	10	
5	TOTAL	100	

RESULT :

Thus the implementation and performance comparison of indexing and hashing techniques has been practiced successfully and the expected output has been achieved.