EXP1

1. Student(IDCardnumber,RollNumber,Stud_Name,Class,Branch)
2. Book(BookID,Book_Name,Author,Price)
3. Issue(IDCardnumber,BookID,IssueDate,SubmissionDate)

Queries:

1. Find the total number of students issued books from library from “CSE” Department

```
SELECT COUNT(DISTINCT Student.IDCardnumber) AS Total_CSE_Students_Issued FROM
Student JOIN Issue ON Student.IDCardnumber = Issue.IDCardnumber WHERE Student.Branch =
'CSE';
```

OR

```
SELECT COUNT(*) FROM Issue NATURAL JOIN Student WHERE Branch=“CSE”
```

Or

```
SELECT COUNT(IDCardnumber) FROM Issue NATURAL JOIN Student WHERE Branch=“CSE”
```

2. Find the list of students not issued books from the library.

```
SELECT * FROM  Student WHERE IDCardnumber NOTIN(SELECT IDCardnumber FROM Student
NATURAL JOIN Issue )
```

Or

```
SELECT * FROM  Student WHERE IDCardnumber NOTIN(SELECT IDCardnumber FROM Student
INNER JOIN Issue ON Student. IDCardnumber==Issue. IDCardnumber)
```

3. Find the details of the book with the second highest price.

```
SELECT BookID, Book_Name, Author, Price FROM Book ORDER BY Price DESC LIMIT 1 OFFSET
1;
```

4. Find branch wise count of students issued books from the library.

```
SELECT Student.Branch, COUNT(DISTINCT Student.IDCardnumber) AS Students_Issued
FROM Student JOIN Issue ON Student.IDCardnumber = Issue.IDCardnumber
GROUP BY Student.Branch;
```

Or

```
SELECT Branch, COUNT(IDCardnumber) AS Students_Issued
FROM Student NATURAL JOIN Issue GROUP BY Student.Branch;
```

5. Find the following details, i.e Book name , student name and issue & submission Dates

```
SELECT Book.Book_Name, Student.Stud_Name, Issue.IssueDate, Issue.SubmissionDate

FROM Issue JOIN Student ON Issue.IDCardnumber = Student.IDCardnumber JOIN Book ON
Issue.BookID = Book.BookID;
```

OR

```
SELECT Book_Name, Stud_Name, IssueDate, SubmissionDate FROM Student NATURAL JOIN
Issue NATURAL JOIN Book
```

EXP2

Implement the following PL/SQL Programs
1) Find the sum of “N” numbers.

```
DECLARE
   N NUMBER;
   sum_result NUMBER := 0;
   i NUMBER;
BEGIN
   N := &N;
   FOR i IN 1..N LOOP
      sum_result := sum_result + i;
   END LOOP;
   DBMS_OUTPUT.PUT_LINE('The sum of numbers from 1 to ' || N || ' is: ' ||sum_result);
END;
/
```

2) Find the Factorial of a given number.

```
DECLARE

   num NUMBER;

   factorial_result NUMBER := 1;

   i NUMBER;

BEGIN

      num := &num;

   FOR i IN 1..num LOOP

      factorial_result := factorial_result * i;

   END LOOP;

   DBMS_OUTPUT.PUT_LINE('The factorial of ' || num || ' is: ' || factorial_result);

END;

/
```

EXP3

Implement the following PL/SQL Program to demonstrate the use of switch case statements.
1) Find the area of the Circle.
2) Find the area of Triangle.

```
DECLARE
   area NUMBER;
   radius NUMBER := 5;
   base NUMBER := 8;
   height NUMBER := 10;
   choice NUMBER;
BEGIN
   choice := &choice;
   CASE choice
   WHEN 1 THEN
         Area := 3.14*radius*radius;
         DBMS_OUTPUT.PUT_LINE('Area of Circle is: ' || area);
   WHEN 2 THEN
         Area := 0.5*base*height;
         DBMS_OUTPUT.PUT_LINE('Area of Triangle is: ' || area);
   WHEN OTHER THEN
         DBMS_OUTPUT.PUT_LINE('INVALID CHOICE');
   END CASE;
END;
/
```

EXP4

Implement & Object Oriented Database.
1) Create a Rectangle Object with operations Area & Perimeter.

EXP5

Demonstrate the implementation of PL/SQL Function at Schema level .
1) Implement a PL/SQL Function to find the number of employees working for “TCS” & getting salary more than Rs.50,000/-.

First create employee table contain empid name company salary

```
create or replace function resultemp

return number

is

        v_count number;

begin

        select count(*) into v_count from employee where company = ‘TCS’ and
        salary>50000;

        return v_count;

end resultemp;

declare

        t_count number;

begin

        t_count=resultemp;

        DBMS_OUTPUT.PUT_LINE('no of tcs employee with salary >50000'||t_count);

end;
```

EXP6

Implement a PL/SQL Procedure(Inside a PL/SQL block)
1) Find the square of a number using the “IN OUT” Parameter.

```
create or replace procedure findsquare(n inout number)
is
begin
        n:=n*n;
end;
declare
        num number:=5;
begin
        findmax(num);
        DBMS_OUTPUT.PUT_LINE(num);
end;
```

2) Find the Maximum of three numbers use IN , IN & IN OUT for the three parameters respectively. Use the third parameter to hold the result.

```
create or replace procedure findmax(n1 in number,n2 in number,n3 in number,maxn
out number)
is
begin
        if n1>n2 and n1>n3 then
                maxn:=n1;
        elsif n2>n1 and n2>n3 then
                maxn:=n2;
        else
                maxn:=n3;
        end if;
end;
declare
        maxnum number;
begin
        findmax(10,20,30,maxnum);
        DBMS_OUTPUT.PUT_LINE(maxnum);
end;
```

or

```
create or replace procedure findmax(n1 in number,n2 in number,n3 inout number)
is
        maxn number:=0;
begin
        if n1>n2 and n1>n3 then
                maxn:=n1;
        elsif n2>n1 and n2>n3 then
                maxn:=n2;
        else
                maxn:=n3;
```

```
        end if;
end;
declare
        maxnum number;
begin
        maxnum :=findmax(10,20,30);
        DBMS_OUTPUT.PUT_LINE(maxnum);
end;
```

EXP7

Demonstrate the concept of sequences.
1) Create sequence to generate EmloyeeID's of
Employee(EmloyeeID,Ename,Salary,City)
2) Demonstrate Insert, Select , Delete Operations

```
CREATE TABLE Employee(EmloyeeID NUMBER PRIMARY KEY,Ename
VARCHAR(50),Salary NUMBER,City VARCHAR(50));

CREATE SEQUENCE EMPSEQ
START WITH 1
INCREMENT BY 1
MINVALUE 1
MAXVALUE 100
NOCYCLE
CACHE 20;

INSERT INTO Employee(EmloyeeID,Ename,Salary,City)
VALUES(EMPSEQ.NEXTVAL,'yash',50000,'satara');
```

All operation are same as it is we write.

EXP8

Implementation of OLAP queries using star schema.

Star Schema

1. Location(LID,City,State,Country)
2. Product(PID,Pname,Price,Category)
3. Customer(CID,Cname,Cust_City,Cust_State)
4. TimeLine(TID,SaleDate,SaleDay,Month,Year)
5. Sales(CID,LID,TID,PID,Quantity)

```
--- Create tables
CREATE TABLE Location (
  LID INT PRIMARY KEY,
  City VARCHAR(50),
  State VARCHAR(50),
  Country VARCHAR(50)
);

CREATE TABLE Product (
  PID INT PRIMARY KEY,
  Pname VARCHAR(100),
  Price DECIMAL(10, 2),
  Category VARCHAR(50)
);

CREATE TABLE Customer (
  CID INT PRIMARY KEY,
  Cname VARCHAR(100),
  Cust_City VARCHAR(50),
  Cust_State VARCHAR(50)
);

CREATE TABLE TimeLine (
  TID INT PRIMARY KEY,
  SaleDate DATE,
  SaleDay VARCHAR(10),
  Month VARCHAR(10),
  Year INT
);

CREATE TABLE Sales (
  CID INT,
  LID INT,
  TID INT,
  PID INT,
  Quantity INT,
  FOREIGN KEY (CID) REFERENCES Customer(CID),
  FOREIGN KEY (LID) REFERENCES Location(LID),
  FOREIGN KEY (TID) REFERENCES TimeLine(TID),
  FOREIGN KEY (PID) REFERENCES Product(PID)
);
```

```
-- Insert sample data
INSERT INTO Location VALUES
  (1, 'New York', 'NY', 'USA'),
  (2, 'Los Angeles', 'CA', 'USA'),
  (3, 'Chicago', 'IL', 'USA');

INSERT INTO Product VALUES
  (1, 'Product A', 10.99, 'Electronics'),
  (2, 'Product B', 9.99, 'Clothing'),
  (3, 'Product C', 12.99, 'Home Goods');

INSERT INTO Customer VALUES
  (1, 'John Doe', 'New York', 'NY'),
  (2, 'Jane Smith', 'Los Angeles', 'CA'),
  (3, 'Bob Johnson', 'Chicago', 'IL');

INSERT INTO TimeLine VALUES
  (1, '2022-01-01', 'Monday', 'January', 2022),
  (2, '2022-02-01', 'Tuesday', 'February', 2022),
  (3, '2022-03-01', 'Wednesday', 'March', 2022);

INSERT INTO Sales VALUES
  (1, 1, 1, 1, 5),
  (2, 2, 2, 2, 10),
  (3, 3, 3, 3, 15);


-----1. Total Sales by Product Category

SELECT P.Category, SUM(S.Quantity) AS Total_Sales
FROM Sales S JOIN Product P ON S.PID = P.PID
GROUP BY P.Category;

-----2. Sales by Location and Time

SELECT L.City, T.Year, SUM(S.Quantity) AS Sales
FROM Sales S JOIN Location L ON S.LID = L.LID
JOIN TimeLine T ON S.TID = T.TID
GROUP BY L.City, T.Year;

-----3. Sales Trend by Month

SELECT T.Month, SUM(S.Quantity) AS Sales
FROM Sales S JOIN TimeLine T ON S.TID = T.TID
GROUP BY T.Month
ORDER BY T.Month;
```

EXP9

Demonstrate ROLLUP and CUBE Operations on a Sample Sales Database.

CREATE TABLE Sales (Region VARCHAR(50),Country VARCHAR(50),Product VARCHAR(50),SalesDate DATE,Quantity INT,Amount DECIMAL(10, 2));

INSERT INTO Sales VALUES
('North', 'USA', 'Product A', '2022-01-01', 10, 100.00),
('North', 'USA', 'Product B', '2022-01-15', 20, 200.00),
('North', 'Canada', 'Product A', '2022-02-01', 15, 150.00),
('South', 'Brazil', 'Product B', '2022-03-01', 30, 300.00),
('South', 'Argentina', 'Product A', '2022-04-01', 25, 250.00);

----ROLLUP Operation

SELECT Region,Country,SUM(Quantity) AS Total_Quantity,SUM(Amount) AS Total_Amount FROM Sales GROUP BY ROLLUP (Region, Country);

----CUBE Operation

SELECT Region,Country,SUM(Quantity) AS Total_Quantity, SUM(Amount) AS Total_Amount FROM Sales GROUP BY CUBE (Region, Country);

EXP10

Demonstrate the Arrays in PL/SQL
1. Create arrays of Name, Grade & Class/Semester
2. Display the students names, Grades & their classes/Semesters

```
DECLARE
  TYPE name_array IS TABLE OF VARCHAR2(50);
  TYPE grade_array IS TABLE OF VARCHAR2(10);
  TYPE class_array IS TABLE OF VARCHAR2(20);

  names name_array := name_array();
  grades grade_array := grade_array();
  classes class_array := class_array();

  BEGIN
    names.extend(5);
    grades.extend(5);
    classes.extend(5);

    names(1) := 'John Doe';
    grades(1) := 'A';
    classes(1) := 'Sophomore';

    names(2) := 'Jane Smith';
    grades(2) := 'B+';
    classes(2) := 'Junior';

    names(3) := 'Bob Johnson';
    grades(3) := 'A-';
```

```
    classes(3) := 'Senior';

    names(4) := 'Alice Brown';
    grades(4) := 'B';
    classes(4) := 'Freshman';

    names(5) := 'Mike Davis';
    grades(5) := 'A+';
    classes(5) := 'Graduate';
  END;

  -- Display students' information
  BEGIN
    DBMS_OUTPUT.PUT_LINE('Student Information:');
    FOR i IN 1..names.COUNT LOOP
      DBMS_OUTPUT.PUT_LINE('Name: ' || names(i) || ', Grade: ' || grades(i) || ', Class/Semester: ' ||
classes(i));
    END LOOP;
  END;
/
```