**FEDERAL INSTITUTE OF SCIENCE AND TECHNOLOGY (FISAT)** **®**

**HORMIS NAGAR, MOOKKANNOOR, ANGAMALY-683577**



**FOCUS ON EXCELLENCE**

**20MCA134 ADVANCED DBMS LAB**

**…………………………………………………………………..**

**LABORATORY RECORD**

**Name: ANANTHAKRISHNAN P R**

**Branch: MASTER OF COMPUTER APPLICATIONS**

**Semester: 2 Batch: A Roll No: 19**

**University Register Number: FIT24MCA-2019**

# JANUARY 2025

**FEDERAL INSTITUTE OF SCIENCE AND TECHNOLOGY (FISAT)TM**

**HORMIS NAGAR, MOOKKANNOOR, ANGAMALY-683577**



**FOCUS ON EXCELLENCE**

**CERTIFICATE**

*This is to certify that this is a Bonafide record of the Practical work done by* **ANANTHAKRISHNAN P R (FIT24MCA-2019)** *in the* **20MCA134 ADVANCED DBMS LAB** *Laboratory**towards the partial fulfilment for the award of the Master Of Computer Applications during the academic year 2024-2025.*

Signature of Staff in Charge Signature of H O D

Ms.Anju L Dr. Deepa Mary Mathews

**Date of University practical examination ……………………….**

Signature of Signature of

# Internal Examiner External Examiner

**CONTENT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl. No** | **Date** | **Title** | **Page No** | **Signature of Staff -In**  **-Charge** |
| 1 | 13.02.25 | Creation of a database using DDL commands including integrity constraints | 1 |  |
| 2 | 20.02.25 | Implementation of DML commands | 8 |  |
| 3 | 25.02.25 | Implementation of different types of operators in SQL . | 17 |  |
| 4 | 05.03.25 | Implementation of different types of functions with suitable examples | 20 |  |
| 5 | 11.03.25 | Implementation of different types of functions with suitable examples | 37 |  |
| 6 | 19.03.25 | Implementation of Join,Views,Set operations | 39 |  |
| 7 | 01.04.25 | PLSQL programs | 48 |  |
| 8 | 10.04.25 | Comparison between relational and nonrelational (NoSQL) databases and the configuration of NoSQL Databases. | 69 |  |
| 9 | 22.04.25 | Programs using MongoDB | 70 |  |

**Question – 1 : Creation of a database using DDL commands including integrity constraints**

Creation of a database using DDL commands including integrity

|  |  |  |
| --- | --- | --- |
| REGD.NO | NAME | BRANCH |
| 0001 | Ram | CSE |
| 0002 | Hari | MECH |
| 0003 | Pradeep | EEE |
| 0004 | Deepak | ETC |

1. Create a table called student with the following values and Write a SQL command which will show the entire STUDENT table.

CREATE TABLE STUDENT19 (REGD\_NO INTEGER PRIMARY KEY ,NAME VARCHAR(20) NOT NULL,BRANCH VARCHAR(20) NOT NULL);

INSERT INTO STUDENT19 VALUES(0001,'Ram','CSE');

INSERT INTO STUDENT19 VALUES(0002,'Hari','MECH')

INSERT INTO STUDENT19 VALUES(0003,'Pradeep','EEE');

INSERT INTO STUDENT19 VALUES(0004,'Deepak','ETC');

SELECT \* FROM STUDENT19;

**Output**

**A screenshot of a computer

AI-generated content may be incorrect.**

1. Create a table EMPLOYEE with following schema:

(Emp\_no, E\_name, E\_address, E\_ph\_no, Dept\_no, Dept\_name,Job\_id , Salary)

CREATE TABLE EMPLOYEE19(Emp\_no INTEGER, E\_name VARCHAR(20), E\_address VARCHAR(50), E\_ph\_no NUMERIC(10), Dept\_no INTEGER, Dept\_name VARCHAR(20),Job\_id VARCHAR(20) , Salary NUMERIC(10,2));

1. Add a new column; HIREDATE to the existing relation.

ALTER TABLE EMPLOYEE19 ADD HIREDATE DATE;

**Output**

A white sheet with black text

AI-generated content may be incorrect.

1. Change the datatype of JOB\_ID from varchar to integer.

ALTER TABLE EMPLOYEE19 MODIFY(Job\_id INTEGER);

**Output**

A screenshot of a computer

AI-generated content may be incorrect.

1. Change the name of column/field Emp\_no to E\_no.

ALTER TABLE EMPLOYEE19 RENAME COLUMN Emp\_no TO E\_no;

**Output**

**A white sheet with black text

AI-generated content may be incorrect.**

A white sheet with black text

AI-generated content may be incorrect.

1. Modify the column width of the Employee name field of emp table.

ALTER TABLE EMPLOYEE19 MODIFY(E\_name VARCHAR(30));

**Output**

A white sheet with black text

AI-generated content may be incorrect.

1. Write a query in sql to create a table employee and department.

Employee(empno, ename, deptno, job, hiredate)

Department(deptno,dname,loc)

CREATE TABLE Employee\_19(empno INTEGER, ename VARCHAR(20), deptno INTEGER, job VARCHAR(20), hiredate DATE);

CREATE TABLE Department\_19(deptno INTEGER,dname VARCHAR(20),loc VARCHAR(50));

Include the following constraints on column of emp table.

1. to make the empno as primary key of the table.

ALTER TABLE Employee\_19 MODIFY(empno PRIMARY KEY);

**Output**

A screenshot of a computer

AI-generated content may be incorrect.

1. To ensure that the ename column does not contain NULL values.

ALTER TABLE Employee\_19 MODIFY(ename NOT NULL);

**Output**

A screenshot of a computer

AI-generated content may be incorrect.

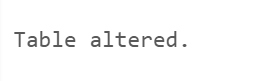
A screenshot of a computer

AI-generated content may be incorrect.

1. the job column to have only UPPERCASE entries.

ALTER TABLE Employee\_19 MODIFY(job VARCHAR(20) CHECK(job = UPPER(job)));

**Output**



1. put the current date as default date in hire date column in case data is not supplied for the

column.

ALTER TABLE Employee\_19 MODIFY(hiredate DEFAULT(CURRENT\_DATE));

**Output**

A close-up of a word

AI-generated content may be incorrect.

Include the following constraints on column of Department table.

1. To make deptno as primary key.

ALTER TABLE Department\_19 MODIFY(deptno PRIMARY KEY);

**Output**

A screenshot of a computer

AI-generated content may be incorrect.

1. To ensure dname,loc coloumns does not contain NULL values.

ALTER TABLE Department\_19 MODIFY dname NOT NULL;

ALTER TABLE Department\_19 MODIFY loc NOT NULL;

**Output**

A screenshot of a computer

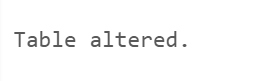
AI-generated content may be incorrect.

1. Also enforce REFERENTIAL INTEGRITY, declare deptno field of dept table as

primary key and deptno field of emp table as foreign key.

ALTER TABLE Employee\_19 ADD FOREIGN KEY (deptno) REFERENCES Department\_19(deptno);

**Output**



**Question 2: Implementation of DML commands**

1. Create a table EMPLOYEE with following schema:

(Emp\_no, E\_name, E\_address, E\_ph\_no, Dept\_no, Dept\_name,Job\_id , Salary).

CREATE TABLE EMPLOYEE\_19(Emp\_no INTEGER, E\_name VARCHAR(20), E\_address VARCHAR(100), E\_ph\_no NUMERIC(10), Dept\_no VARCHAR(5), Dept\_name VARCHAR(20),Job\_id VARCHAR(5) , Salary NUMERIC(6,2));

**Write SQL queries for following question:**

1. Insert aleast 5 rows in the table.

INSERT INTO EMPLOYEE\_19 VALUES (10, 'John', 'Pune', '9876543210', 'D10', 'SALES', 'J01', 5000);

INSERT INTO EMPLOYEE\_19 VALUES (19, 'James', 'Mumbai', '9876543219', 'D12', 'MECH', 'J02', 4500);

INSERT INTO EMPLOYEE\_19 VALUES (12, 'Adam', 'Nagpur', '9876543212', 'D10', 'SALES', 'J03', 4800);

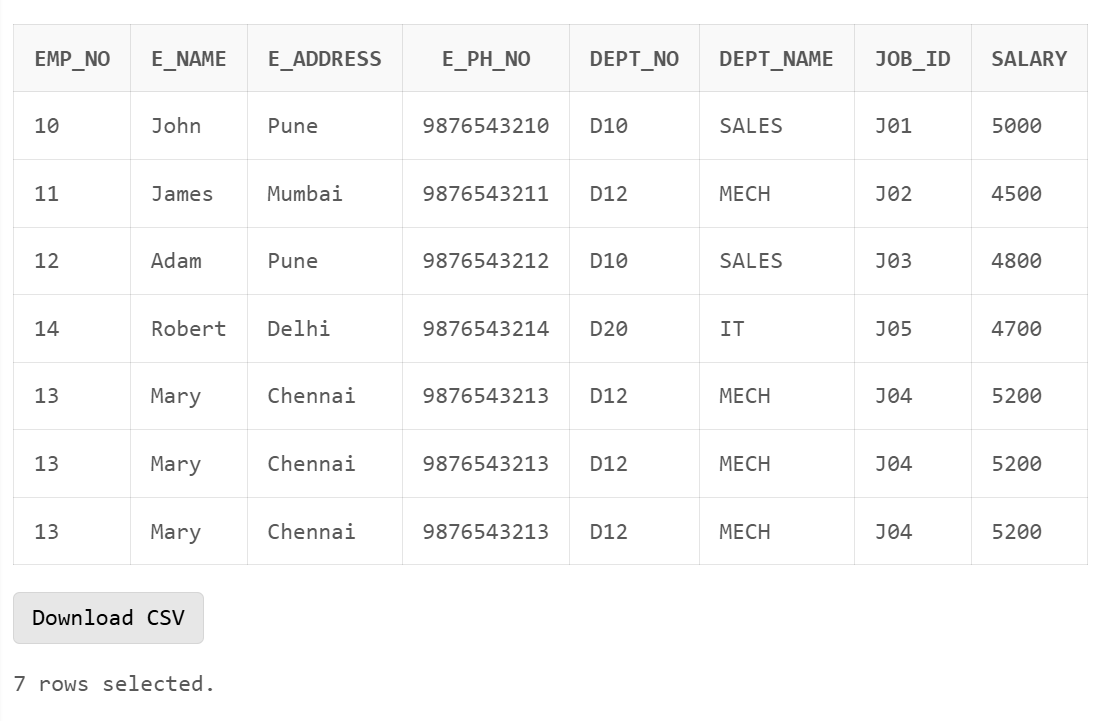
INSERT INTO EMPLOYEE\_19 VALUES (13, 'Mary', 'Chennai', '9876543213', 'D12', 'MECH', 'J04', 5200);

INSERT INTO EMPLOYEE\_19 VALUES (14, 'Robert', 'Delhi', '9876543214', 'D20', 'IT', 'J05', 1900);

1. Display all the information of EMP table.

SELECT \* FROM EMPLOYEE\_19;

**Output**



1. Display the record of each employee who works in department D10.

SELECT \* FROM EMPLOYEE\_19 WHERE Dept\_no = 'D10';

**Output**

A screenshot of a computer

AI-generated content may be incorrect.

1. Update the city of Emp\_no-12 with current city as Nagpur.

UPDATE EMPLOYEE\_19 SET E\_address = 'Nagpur' WHERE Emp\_no = 12;

**Output**

A close-up of a number

AI-generated content may be incorrect.

1. Display the details of Employee who works in department MECH.

SELECT \* FROM EMPLOYEE\_19 WHERE Dept\_name = 'MECH';

**Output**

A screenshot of a computer

AI-generated content may be incorrect.

6. Delete the email\_id of employee James.

ALTER TABLE EMPLOYEE\_19 ADD email\_id VARCHAR(20);

UPDATE EMPLOYEE\_19 SET email\_id = 'james@gmail.com' WHERE E\_name = 'James';

UPDATE EMPLOYEE\_19 SET email\_id = 'NULL' WHERE E\_name = 'James';

**Output**

A close-up of a number

AI-generated content may be incorrect.

7. Display the complete record of employees working in SALES Department.

SELECT \* FROM EMPLOYEE\_19 WHERE Dept\_name = 'SALES';

**Output**

A close-up of a number

AI-generated content may be incorrect.

8. Find out the employee id, names, salaries of all the employees

SELECT Emp\_no, E\_name, Salary FROM EMPLOYEE\_19;

**Output**

A screenshot of a table

AI-generated content may be incorrect.

9. Find the names of the employees who have a salary greater than or equal to 4800

SELECT E\_name FROM EMPLOYEE\_19

WHERE Salary >= 4800;

**Output**

A screenshot of a phone

AI-generated content may be incorrect.

**5.** **(Exercise on updating records in table)**

Create Client\_master with the following fields(ClientNO, Name, Address, City, State, bal\_due).

CREATE TABLE Client\_master(ClientNO VARCHAR(6), Name VARCHAR(10), Address VARCHAR(20), City VARCHAR(10), State VARCHAR(10), bal\_due NUMERIC(6,2));

1. Insert five records

INSERT INTO Client\_master VALUES ('C121', 'John','Rk House', 'Aloor', 'THRISSUR', 6000);

INSERT INTO Client\_master VALUES ('C122', 'Vinod','Mk House', 'Kattapana', 'Idukki', 5000);

INSERT INTO Client\_master VALUES ('C123', 'Adam', 'HK House','Aloor', 'THRISSUR', 5700);

INSERT INTO Client\_master VALUES ('C124', 'Hari','PK House', 'Aloor', 'THRISSUR', 4000);

INSERT INTO Client\_master VALUES ('C125', 'Eve','JK House', 'Aloor', 'THRISSUR', 9000);

1. Find the names of clients whose bal\_due> 5000

**Output**

A screenshot of a phone

AI-generated content may be incorrect.

1. Change the bal\_due of ClientNO “ C123” to Rs. 5100

UPDATE Client\_master SET bal\_due = 5100 WHERE ClientNO = 'C123';

**Output**

A white rectangular sign with black text

AI-generated content may be incorrect.

1. Change the name of Client\_master to Client12

ALTER TABLE Client\_master RENAME TO Client12;

**Output**

A close-up of a word

AI-generated content may be incorrect.

1. Display the bal\_due heading as “BALANCE”

SELECT bal\_due AS BALANCE FROM Client12;

**Output**

A screenshot of a cell phone

AI-generated content may be incorrect.

**6. (Rollback and Commit commands )**

Create Teacher table with the following fields(Name, DeptNo, Date of joining, DeptName, Location,Salary)

CREATE TABLE TEACHER\_19(Name VARCHAR(10) PRIMARY KEY, DeptNo VARCHAR(6), Date\_of\_joining DATE, DeptName VARCHAR(20), Location VARCHAR(15),Salary NUMERIC(6,2));

1. Insert five records

INSERT INTO TEACHER\_19 VALUES ('John','D101','08-JAN-01', 'Mathematics', 'THRISSUR', 6000);

INSERT INTO TEACHER\_19 VALUES ('Varun','D102','04-JAN-03', 'Commerce', 'Idukki', 7000);

INSERT INTO TEACHER\_19 VALUES ('Vinu','D101','05-JAN-08', 'Mathematics', 'THRISSUR', 8000);

INSERT INTO TEACHER\_19 VALUES ('Jenna','D103','09-JAN-06', 'English', 'Idukki', 6000);

INSERT INTO TEACHER\_19 VALUES ('Eve','D104','07-JAN-05', 'Commerce', 'THRISSUR', 8000);

1. Give Increment of 25% salary for Mathematics Department.

UPDATE TEACHER\_19 SET Salary = Salary + (Salary \* 0.25) WHERE DeptName = 'Mathematics';

**Output**

A screenshot of a computer

AI-generated content may be incorrect.

1. Perform Rollback command

ROLLBACK;

**Output**

A screenshot of a computer

AI-generated content may be incorrect.

1. Give Increment of 15% salary for Commerce Department

UPDATE TEACHER\_ SET Salary = Salary + (Salary \* 0.15) WHERE DeptName = 'Commerce';

**Output**

A screenshot of a computer

AI-generated content may be incorrect.

1. Perform commit command

COMMIT;

**Output**

A screenshot of a computer

AI-generated content may be incorrect.

**7.(Exercise on order by and group by clauses)**

Create Sales table with the following fields( Sales No, Salesname, Branch, Salesamount, DOB)

CREATE TABLE Sales (SalesNo NUMBER PRIMARY KEY,Salesname VARCHAR2(50),Branch VARCHAR2(30),Salesamount NUMBER(10,2),DOB DATE);

1. Insert five records

INSERT INTO Sales VALUES (1, 'John Smith', 'North', 15000, '15-DEC-85');

INSERT INTO Sales VALUES (2, 'Sarah Johnson', 'South', 22000, '03-MAR-90');

INSERT INTO Sales VALUES (3, 'Mike Brown', 'North', 18000, '21-DEC-88');

INSERT INTO Sales VALUES (4, 'Emily Davis', 'East', 25000, '12-AUG-92');

INSERT INTO Sales VALUES (5, 'David Wilson', 'South', 19000, '30-DEC-87');

1. Calculate total salesamount in each branch

SELECT Branch,SUM(Salesamount) AS TotalSales FROM Sales GROUP BY Branch ORDER BY Branch;

**Output**

A screenshot of a white rectangular box with black text

AI-generated content may be incorrect.

1. Calculate average salesamount in each branch .

SELECT Branch, AVG(Salesamount) AS AverageSales FROM Sales GROUP BY Branch ORDER BY Branch;

**Output**

A screenshot of a white table

AI-generated content may be incorrect.

1. Display all the salesmen, DOB who are born in the month of December as day in character format i.e. 21-Dec-09.

SELECT Salesname, TO\_CHAR(DOB, 'DD-MON-YY') AS Formatted DOB FROM Sales WHERE EXTRACT(MONTH FROM DOB) = 12 ORDER BY Salesname;

**Output**

A white rectangular box with black text

AI-generated content may be incorrect.

1. Display the name and DOB of salesman in alphabetical order of the month.

SELECT Salesname, TO\_CHAR(DOB, 'DD-MON-YY') AS FormattedDOB FROM Sales ORDER BY TO\_CHAR(DOB, 'MON'), Salesname;

**Output**

A screenshot of a computer

AI-generated content may be incorrect.

**Question - 3 : Implementation of different types of operators in SQL**

**8.** Create an Emp table with the following fields:(EmpNo, EmpName, Job, Basic, DA,

HRA,PF, GrossPay, NetPay) Hint:( PF is calculated as 10% of basic salary) (Calculate DA

as 30% of Basic and HRA as 40% of Basic).

CREATE TABLE Emp (EmpNo INT PRIMARY KEY,EmpName VARCHAR(50),Job VARCHAR(30),Basic DECIMAL(10,2),DA DECIMAL(10,2),HRA DECIMAL(10,2),PF DECIMAL(10,2),GrossPay DECIMAL(10,2),NetPay DECIMAL(10,2),Department VARCHAR(30));

1. Insert Five Records and calculate GrossPay and NetPay.

INSERT INTO Emp (EmpNo, EmpName, Job, Basic, Department)

VALUES (101, 'John Smith', 'Manager', 25000.00, 'HR');

INSERT INTO Emp (EmpNo, EmpName, Job, Basic, Department)

VALUES (102, 'Sarah Johnson', 'Developer', 18000.00, 'IT');

INSERT INTO Emp (EmpNo, EmpName, Job, Basic, Department)

VALUES (103, 'Mike Brown', 'Analyst', 15000.00, 'HR');

INSERT INTO Emp (EmpNo, EmpName, Job, Basic, Department)

VALUES (104, 'Lisa Davis', 'Designer', 9000.00, 'Creative');

INSERT INTO Emp (EmpNo, EmpName, Job, Basic, Department)

VALUES (105, 'David Wilson', 'Tester', 5000.00, 'IT');

UPDATE Emp SET

DA = Basic \* 0.30,

HRA = Basic \* 0.40,

PF = Basic \* 0.10,

GrossPay = Basic + (Basic \* 0.30) + (Basic \* 0.40),

NetPay = Basic + (Basic \* 0.30) + (Basic \* 0.40) - (Basic \* 0.10);

1. Display the employees whose Basic is lowest in each department.

SELECT Emp.\* FROM Emp

INNER JOIN (

SELECT Department, MIN(Basic) as MinBasic

FROM Emp

GROUP BY Department

) dept ON Emp.Department = dept.Department AND Emp.Basic = dept.MinBasic;

**Output**

A table of numbers and letters

AI-generated content may be incorrect.

1. If NetPay is less than <Rs. 10,000 add Rs. 1200 as special allowances.

UPDATE Emp

SET NetPay = NetPay + 1200

WHERE NetPay < 10000;

**Output**

A table with numbers and letters

AI-generated content may be incorrect.

1. Display the employees whose GrossPay lies between 10,000 & 20,000.

SELECT \* FROM Emp WHERE GrossPay BETWEEN 10000 AND 20000;

**Output**

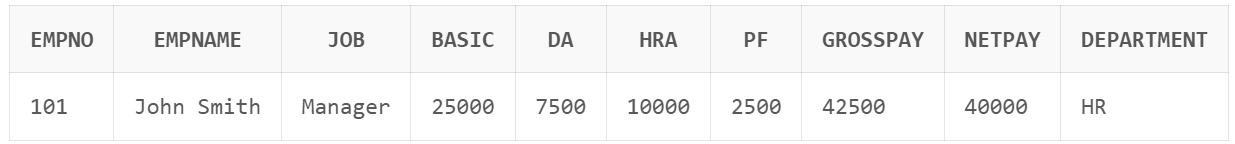
A group of numbers and symbols

AI-generated content may be incorrect.

1. Display all the employees who earn maximum salary Employee Table.

SELECT \* FROM Emp WHERE NetPay = (SELECT MAX(NetPay) FROM Emp);

**Output**



**Question – 4 : Implementation of different types of functions with suitable examples**

**9.** Create a table EMPLOYEE with following schema:

(Emp\_no, E\_name, E\_address, E\_ph\_no, Dept\_no, Dept\_name,Job\_id, Designation , Salary)

CREATE TABLE EMPLOYEE (Emp\_no INT PRIMARY KEY,E\_name VARCHAR(50) NOT NULL,E\_address VARCHAR(100),E\_ph\_no VARCHAR(15),Dept\_no INT,Dept\_name VARCHAR(30),Job\_id VARCHAR(20),Designation VARCHAR(30),Salary DECIMAL(10,2),HireDate DATE);

Write SQL statements for the following query.

1. List the E\_no, E\_name, Salary of all employees working for MANAGER.

SELECT Emp\_no, E\_name, Salary FROM EMPLOYEE WHERE Designation = 'MANAGER';

**Output**

A screenshot of a computer

AI-generated content may be incorrect.

1. Display all the details of the employee whose salary is more than the Sal of any IT PROFF.

SELECT \* FROM EMPLOYEE

WHERE Salary > ANY (

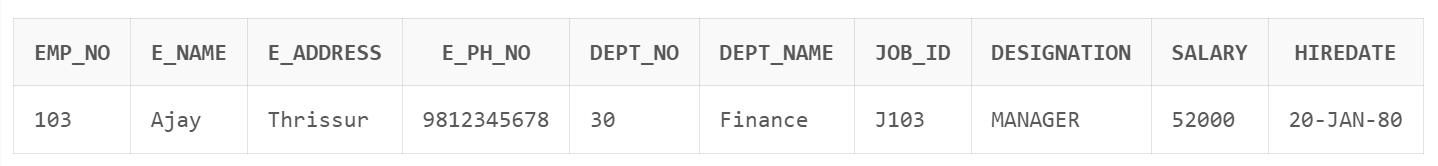
SELECT Salary

FROM EMPLOYEE

WHERE Designation = 'IT PROFF'

);

**Output**



1. List the employees in the ascending order of Designations of those joined after 1981.

SELECT \* FROM EMPLOYEE

WHERE EXTRACT(YEAR FROM HireDate) > 1981

ORDER BY Designation ASC;

**Output**

A screen shot of a computer

AI-generated content may be incorrect.

1. List the employees along with their Experience and Daily Salary.

SELECT

E\_name,

FLOOR(MONTHS\_BETWEEN(SYSDATE, HireDate)/12) AS Experience\_Years,

ROUND(Salary/30, 2) AS Daily\_Salary

FROM EMPLOYEE;

**Output**

A screenshot of a white sheet

AI-generated content may be incorrect.

1. List the employees who are either ‘CLERK’ or ‘ANALYST’.

SELECT \* FROM EMPLOYEE

WHERE Designation IN ('CLERK', 'ANALYST');

**Output**

A white rectangular object with black text

AI-generated content may be incorrect.

1. List the employees who joined on 1-MAY-81, 3-DEC-81, 17-DEC-81,19-JAN-80.

SELECT \* FROM EMPLOYEE

WHERE HireDate IN ('01-MAY-81', '03-DEC-81', '17-DEC-81', '19-JAN-80');

**Output**

A close up of a list

AI-generated content may be incorrect.

1. List the employees who are working for the Deptno 10 or 20.

SELECT \* FROM EMPLOYEE WHERE Dept\_no IN (10, 20);

**Output**

A screenshot of a computer

AI-generated content may be incorrect.

1. List the Enames those are starting with ‘S’.

SELECT E\_name FROM EMPLOYEE WHERE E\_name LIKE 'S%';

**Output**

A close up of a name

AI-generated content may be incorrect.

1. Dislay the name as well as the first five characters of name(s) starting with ‘H.

SELECT

E\_name,

SUBSTR(E\_name, 1, 5) AS First\_Five\_Chars

FROM EMPLOYEE

WHERE E\_name LIKE 'H%';

**Output**

A close-up of a white box

AI-generated content may be incorrect.

1. List all the emps except ‘PRESIDENT’ & ‘MANAGR” in asc order of Salaries.

SELECT \* FROM EMPLOYEE

WHERE Designation NOT IN ('PRESIDENT', 'MANAGER')

ORDER BY Salary ASC;

**Output**

A screenshot of a computer

AI-generated content may be incorrect.

**10**. Consider Employee table

A table with numbers and text

AI-generated content may be incorrect.

**Perform the following**

1. Display all the fields of employee table.

SELECT \* FROM Employee;

**Output**

A white rectangular grid with black text

AI-generated content may be incorrect.

1. Retrieve employee number and their salary.

SELECT EMPNO, SALARY FROM Employee;

**Output**

A screenshot of a number

AI-generated content may be incorrect.

1. Retrieve average salary of all employee.

SELECT AVG(SALARY) AS Average\_Salary FROM Employee;

**Output**

A screenshot of a number

AI-generated content may be incorrect.

1. Retrieve number of employee.

SELECT COUNT(\*) AS Total\_Employees FROM Employee;

**Output**

A screenshot of a computer

AI-generated content may be incorrect.

1. Retrieve distinct number of employee.

SELECT COUNT(DISTINCT EMPNO) AS Distinct\_Employees FROM Employee;

**Output**

A close-up of a message

AI-generated content may be incorrect.

1. Retrieve total salary of employee group by employee name and count similar names.

SELECT EMP\_NAME, SUM(SALARY) AS Total\_Salary, COUNT(\*) AS Name\_Count

FROM Employee GROUP BY EMP\_NAME;

**Output**

A white rectangular grid with black text

AI-generated content may be incorrect.

1. Retrieve total salary of employee which is greater than >120000.

SELECT SUM(SALARY) AS Total\_Salary FROM Employee

GROUP BY EMP\_NAME

HAVING SUM(SALARY) > 120000;

**Output**

A screenshot of a phone

AI-generated content may be incorrect.

1. Display name of employee in descending order.

SELECT EMP\_NAME FROM Employee ORDER BY EMP\_NAME DESC;

**Output**

**A screenshot of a phone

AI-generated content may be incorrect.**

1. Display details of employee whose name is AMIT and salary greater than 50000;

SELECT \* FROM Employee WHERE EMP\_NAME = 'Amit' AND SALARY > 50000;

**Output**

A close-up of a number

AI-generated content may be incorrect.

**19.** Create a table called Employee with the following structure.

A table of names and numbers

AI-generated content may be incorrect.

CREATE TABLE Employee (Empno NUMBER(4) PRIMARY KEY,Ename VARCHAR2(20) NOT NULL,Job VARCHAR2(20),Mgr NUMBER(4),Sal NUMBER(7,2),Deptno NUMBER(2));

1. Display lowest paid employee details under each department.

SELECT Employee.\*

FROM Employee

WHERE (Employee.Deptno, Employee.Sal) IN (

SELECT Deptno, MIN(Sal)

FROM Employee

GROUP BY Deptno

)

ORDER BY Employee.Deptno;

**Output**

A screenshot of a white grid

AI-generated content may be incorrect.

b) Display number of employees working in each department and their department

number.

SELECT Deptno, COUNT(\*) AS Employee\_Count

FROM Employee GROUP BY Deptno

ORDER BY Deptno ASC;

**Output**

A white grid with black text

AI-generated content may be incorrect.

c) Using built-in functions, display number of employees working in each department

and their department name from dept table. Insert deptname to dept table and insert deptname for each row, do the required thing specified above.

ALTER TABLE Employee ADD Dept\_name VARCHAR2(20);

UPDATE Employee SET Dept\_name = 'ADMIN' WHERE Deptno = 10;

UPDATE Employee SET Dept\_name = 'RESEARCH' WHERE Deptno = 20;

UPDATE Employee SET Dept\_name = 'SALES' WHERE Deptno = 30;

SELECT Deptno,Dept\_name, COUNT(\*) AS Employee\_Count

FROM Employee

GROUP BY Deptno,Dept\_name

ORDER BY Deptno ASC;

**Output**

A white grid with black text

AI-generated content may be incorrect.

1. List all employees which start with either B or C.

SELECT \* FROM Employee WHERE Ename LIKE 'B%' OR Ename LIKE 'C%';

**Output**

A screenshot of a number

AI-generated content may be incorrect.

e) Display only these ename of employees where the maximum salary is greater than

or equal to 5000.

SELECT Ename FROM Employee GROUP BY Ename

HAVING MAX(Sal) >= 5000;

**Output**

**A white rectangular sign with black text

AI-generated content may be incorrect.**

1. Calculate the average salary for each different job.

SELECT Job, AVG(Sal) AS Avg\_Salary FROM Employee GROUP BY Job;

**Output**

A screenshot of a computer

AI-generated content may be incorrect.

1. Show the average salary of each job excluding manager.

SELECT Job, AVG(Sal) AS Avg\_Salary FROM Employee WHERE Job != 'MANAGER' GROUP BY Job;

**Output**

**A white card with numbers and numbers

AI-generated content may be incorrect.**

1. Show the average salary for all departments employing more than three people.

SELECT Deptno, AVG(Sal) AS Avg\_Salary FROM Employee

GROUP BY Deptno HAVING COUNT(\*) > 3;

**Output**

**A screenshot of a calculator

AI-generated content may be incorrect.**

1. How many days between day of birth to current date.

SELECT Ename,Sal AS Current\_Salary,Sal \* 1.15

AS Salary\_After\_Rise FROM Employee;

**Output**

**A table of numbers and numbers

AI-generated content may be incorrect.**

1. List all employee names, salary and 15% rise in salary.

SELECT Employee.\* FROM Employee

INNER JOIN (

SELECT Mgr, MIN(Sal) AS MinSal

FROM Employee

WHERE Mgr IS NOT NULL

GROUP BY Mgr

) mgr ON Employee.Mgr = mgr.Mgr AND Employee.Sal = mgr.MinSal;

**Output**

**A table of numbers and names

AI-generated content may be incorrect.**

1. Display lowest paid emp details under each manager.

SELECT Deptno,Dept\_name, AVG(Sal) AS Avg\_Monthly\_Salary

FROM Employee

GROUP BY Deptno,Dept\_name;

**Output**

A screenshot of a white table

AI-generated content may be incorrect.

1. Display the average monthly salary bill for each deptno.

SELECT Deptno, AVG(Sal) AS Avg\_Salary FROM Employee

GROUP BY Deptno

HAVING COUNT(\*) > 2;

**Output**

A screenshot of a white table

AI-generated content may be incorrect.

1. Show the average salary for all departments employing more than two people.

SELECT Empno, AVG(Sal) AS Avg\_Salary FROM Employee WHERE Deptno = 5

GROUP BY Empno;

**Output**

A close-up of a text

AI-generated content may be incorrect.

n) By using the group by clause, display the eid who belongs to deptno 05 along with

average salary.

SELECT COUNT(\*) AS Employee\_Count FROM Employee WHERE Deptno = 20;

**Output**

A screenshot of a computer

AI-generated content may be incorrect.

1. Count the number of employees in department 20.

SELECT MIN(Sal) AS Min\_Clerk\_Salary FROM Employee WHERE Job = 'CLERK';

**Output**

A screenshot of a computer

AI-generated content may be incorrect.

1. Find the minimum salary earned by clerk.

SELECT MIN(Sal) AS Min\_Salary,MAX(Sal) AS Max\_Salary,AVG(Sal) AS Avg\_Salary FROM Employee;

**Output**

A number on a white background

AI-generated content may be incorrect.

1. Find minimum, maximum, average salary of all employees.

SELECT Job,MIN(Sal) AS Min\_Salary,MAX(Sal) AS Max\_Salary FROM Employee

GROUP BY Job;

**Output**

A table of jobs

AI-generated content may be incorrect.

1. List the employee names in descending order.

SELECT Ename FROM Employee

ORDER BY Ename DESC;

**Output**

A white rectangular object with black text

AI-generated content may be incorrect.

1. List the employee id, names in ascending order by empid.

SELECT Empno, Ename

FROM Employee

ORDER BY Empno ASC;

**Output**

A screenshot of a cellphone

AI-generated content may be incorrect.

**Question – 5 : Implementation of different types of functions with suitable examples**

**12.** Create a table EMPLOYEE with following schema:

CREATE TABLE EMPLOYEE (Emp\_no NUMBER PRIMARY KEY,E\_name VARCHAR2(50),E\_address VARCHAR2(100),E\_ph\_no VARCHAR2(15),Dept\_no NUMBER,Dept\_name VARCHAR2(30),Job\_id VARCHAR2(20),Salary NUMERIC(10,2));

CREATE TABLE DEPARTMENT(Dept\_no NUMBER,Dept\_name VARCHAR2(30),LOC VARCHAR(10) );

1. Display all the dept numbers available with the dept and emp tables avoiding duplicates.

SELECT Dept\_no FROM EMPLOYEE UNION SELECT Dept\_no FROM DEPARTMENT ORDER BY Dept\_no;

**Output**

A screenshot of a cell phone

AI-generated content may be incorrect.

1. Display all the dept numbers available with the dept and emp tables.

SELECT Dept\_no FROM EMPLOYEE UNION ALL SELECT Dept\_no FROM DEPARTMENT ORDER BY Dept\_no;

**Output**

A screenshot of a calendar

AI-generated content may be incorrect.

1. Display all the dept numbers available in emp and not in dept tables and vice versa.

SELECT Dept\_no FROM EMPLOYEE MINUS SELECT Dept\_no FROM DEPARTMENT;

SELECT Dept\_no FROM DEPARTMENT MINUS SELECT Dept\_no FROM EMPLOYEE;

**Output**

A white rectangular object with black text

AI-generated content may be incorrect.

A white rectangular object with black text

AI-generated content may be incorrect.

**Question – 6 : Implementation of Join,Views,Set operations**

**13.** Consider the following schema:

Sailors (sid, sname, rating, age)

Boats (bid, bname, color)

Reserves (sid, bid, day(date))

CREATE TABLE Sailors(sid NUMBER PRIMARY KEY,sname VARCHAR2(50),rating NUMBER,age NUMBER);

CREATE TABLE Boats(bid NUMBER PRIMARY KEY,bname VARCHAR2(50),color VARCHAR2(20));

CREATE TABLE Reserves(sid NUMBER,bid NUMBER,day DATE,PRIMARY KEY (sid, bid, day),FOREIGN KEY (sid) REFERENCES Sailors(sid),FOREIGN KEY (bid) REFERENCES Boats(bid));

1. Find all the information of sailors who have reserved boat number 101.

SELECT Sailors.\* FROM Sailors

JOIN

Reserves ON Sailors.sid = Reserves.sid

WHERE Reserves.bid = 101;

**Output**

A screenshot of a computer

AI-generated content may be incorrect.

1. Find the name of boat reserved by Bob.

SELECT Boats.bname FROM Boats

JOIN

Reserves ON Boats.bid = Reserves.bid

JOIN

Sailors ON Sailors.sid = Reserves.sid

WHERE Sailors.sname = 'Bob';

**Output**

A white rectangular sign with black text

AI-generated content may be incorrect.

1. Find the names of sailors who have reserved a red boat, and list in order of age.

SELECT sname

FROM (

SELECT DISTINCT Sailors.sname, Sailors.age

FROM Sailors

JOIN Reserves ON Sailors.sid = Reserves.sid

JOIN Boats ON Reserves.bid = Boats.bid

WHERE Boats.color = 'red'

)

ORDER BY age;

**Output**

A screenshot of a phone

AI-generated content may be incorrect.

1. Find the names of sailors who have reserved at least one boat.

SELECT DISTINCT Sailors.sname

FROM Sailors

JOIN Reserves

ON Sailors.sid =Reserves.sid;

**Output**

A screenshot of a phone

AI-generated content may be incorrect.

1. Find the ids and names of sailors who have reserved two different boats on the same day.

SELECT DISTINCT s.sid, s.sname

FROM Sailors s

JOIN Reserves r1 ON s.sid = r1.sid

JOIN Reserves r2 ON s.sid = r2.sid

WHERE r1.bid <> r2.bid AND r1.day = r2.day;

**Output**

A screenshot of a computer

AI-generated content may be incorrect.

1. Find the ids of sailors who have reserved a red boat or a green boat.

SELECT DISTINCT Sailors.sid

FROM Sailors

JOIN Reserves ON

Sailors.sid = Reserves.sid

JOIN Boats ON

Reserves.bid = Boats.bid

WHERE Boats.color IN ('red', 'green');

**Output**

**A white rectangular object with black numbers

AI-generated content may be incorrect.**

1. Find the name and the age of the youngest sailor.

SELECT sname, age FROM Sailors

WHERE age = (SELECT MIN(age) FROM Sailors);

**Output**

A screenshot of a cellphone

AI-generated content may be incorrect.

1. Count the number of different sailor names.

SELECT COUNT(DISTINCT sname) AS unique\_names

FROM Sailors;

**Output**

A close-up of a text

AI-generated content may be incorrect.

1. Find the average age of sailors for each rating level.

SELECT rating, AVG(age) AS avg\_age

FROM Sailors

GROUP BY rating;

**Output**

A screenshot of a cell phone

AI-generated content may be incorrect.

1. Find the average age of sailors for each rating level that has at least two sailors.

SELECT rating, AVG(age) AS avg\_age

FROM Sailors

GROUP BY rating

HAVING COUNT(\*) >= 2;

**Output**

**A close up of a text

AI-generated content may be incorrect.**

**14.**  **Original Table:** Employees (employee\_id, name, salary, department\_id)

**Question:** Create a view named EmployeeDetails that displays the employee ID, name, and

salary from the Employees table.

CREATE TABLE Employees (employee\_id INTEGER PRIMARY KEY,name VARCHAR(100),salary NUMERIC(10, 2),department\_id INTEGER);

CREATE VIEW EmployeeDetails AS SELECT employee\_id, name, salary

FROM Employees;

**Output**

**A close-up of a logo

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**15.**  **Original Table:** Customers (customer\_id, first\_name, last\_name, email)

**Question:** Write a SQL query to create a view called CustomerContacts that combines the

customer's first name, last name, and email address from the Customers table.

**Output**

A close-up of a logo

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**16. Original Tables:**

Employees (employee\_id, name, salary\_grade\_id),

SalaryGrades (salary\_grade\_id, min\_salary, max\_salary)

CREATE TABLE SalaryGrades (salary\_grade\_id INTEGER PRIMARY KEY,min\_salary NUMERIC(10, 2),max\_salary NUMERIC(10, 2));

CREATE TABLE Employees (employee\_id INTEGER PRIMARY KEY,name VARCHAR(20),salary NUMERIC(10, 2),salary\_grade\_id INTEGER,

FOREIGN KEY (salary\_grade\_id) REFERENCES SalaryGrades(salary\_grade\_id)

);

Create a view named EmployeeSalaries that shows the employee ID, name, and salary along with the salary grade from the Employees and SalaryGrades tables.

CREATE VIEW EmployeeSalaries AS

SELECT

Employees.employee\_id,

Employees.name,

Employees.salary,

SalaryGrades.salary\_grade\_id AS grade

FROM Employees

JOIN SalaryGrades

ON Employees.salary BETWEEN SalaryGrades.min\_salary AND SalaryGrades.max\_salary;

**Output**

**A close-up of a logo

AI-generated content may be incorrect.**

A screenshot of a computer

AI-generated content may be incorrect.

**17. Create tables**

Employees (employee\_id , name )

Managers ( manager\_id, name )

CREATE TABLE Employees (employee\_id INT PRIMARY KEY,name VARCHAR(50) NOT NULL);

CREATE TABLE Managers (manager\_id INT PRIMARY KEY,name VARCHAR(50) NOT

NULL);

1. Write a SQL query to retrieve the names of all employees and managers, ensuring that duplicate names are removed.

SELECT name FROM Employees

UNION

SELECT name FROM Managers;

**Output**

A screenshot of a cell phone

AI-generated content may be incorrect.

1. Create a query to find the common names between employees and managers.

SELECT Employees.name FROM Employees

INTERSECT

SELECT Managers.name FROM Managers;

**Output**

A screenshot of a phone

AI-generated content may be incorrect.

1. Write a query to find the names of employees who are not managers.

SELECT Employees.name

FROM Employees

WHERE Employees.name NOT IN (SELECT name FROM Managers);

**Output**

A screenshot of a phone

AI-generated content may be incorrect.

d) Write a query to find the distinct names of all employees and managers, along with their

respective roles (employee/manager).

SELECT name, 'Employee' AS role FROM Employees

UNION

SELECT name, 'Manager' AS role FROM Managers ORDER BY name;

**Output**

A screenshot of a phone

AI-generated content may be incorrect.

**Question – 18**

Write a PL/SQL program to swap the values of two numbers.

**Program Code**

DECLARE

n1 NUMBER;

n2 NUMBER;

temp NUMBER;

BEGIN

n1:=&n1;

n2:=&n2;

temp:=n1;

n1:=n2;

n2:=temp;

dbms\_output.put\_line('======== After Swapping =========');

dbms\_output.put\_line('n1='||n1);

dbms\_output.put\_line('n2='||n2);

END;

**Output**

A screenshot of a computer program

AI-generated content may be incorrect.

**Question –19**

Write a PL/SQL program to determine the largest among three given numbers.

**Program Code**

DECLARE

n1 NUMBER;

n2 NUMBER;

n3 NUMBER;

BEGIN

    n1 := &n1;

    n2 := &n2;

    n3 := &n3;

    IF n1 > n2 AND n1 > n3

    THEN

        dbms\_output.put\_line('Greatest is =' || n1);

    ELSIF n2 > n1 AND n2 > n3

    THEN

        dbms\_output.put\_line('Greatest is =' || n2);

    ELSE

        dbms\_output.put\_line('Greatest is =' || n3);

    END IF;

END;

/

**Output**

A screenshot of a computer

AI-generated content may be incorrect.

**Question – 20**

Write a PL/SQL program to compute the sum of digits of a given number.

**Program Code**

DECLARE

n NUMBER;

s NUMBER;

r NUMBER;

BEGIN

n:=&n;

s:=0;

WHILE n>0 LOOP

r:= MOD(n,10);

s:=s + r;

n:=TRUNC(n / 10);

END LOOP;

dbms\_output.put\_line('Sum = '||s);

END;

/

**Output**

A white paper with black text and numbers

AI-generated content may be incorrect.

**Question – 21**

Write a PL/SQL program to display a given number in reverse order.

**Program Code**

DECLARE

n NUMBER;

rev NUMBER;

r NUMBER;

BEGIN

n:=&n;

rev:=0;

WHILE n>0 LOOP

r:= MOD(n,10);

rev:=rev \* 10 + r;

n:=TRUNC(n / 10);

END LOOP;

dbms\_output.put\_line('Reversed Number = '||rev);

END;

/

**Output**

A white background with black text

AI-generated content may be incorrect.

QUESTION 22

Write a PL/SQL program to calculate the net salary and annual salary, considering DA as 30% of basic, HRA as 10% of basic, and PF as:7% if the basic salary is less than 8000 10% if the basic salary is between 8000 and 16000.

PROGRAM CODE

DECLARE

basic\_salary NUMBER := 17000;

da\_percent NUMBER := 30;

hra\_percent NUMBER := 10;

pf\_percent\_low NUMBER := 7;

pf\_percent\_high NUMBER := 10;

da NUMBER := basic\_salary \* (da\_percent / 100);

hra NUMBER := basic\_salary \* (hra\_percent / 100);

pf\_percent NUMBER;

pf NUMBER;

net\_salary NUMBER;

yearly\_salary NUMBER;

BEGIN

IF basic\_salary < 8000 THEN

pf\_percent := pf\_percent\_low;

ELSIF basic\_salary >= 8000 AND basic\_salary <= 16000 THEN

pf\_percent := pf\_percent\_high;

ELSE

pf\_percent := 0;

END IF;

pf := basic\_salary \* (pf\_percent / 100);

net\_salary := basic\_salary + da + hra - pf;

yearly\_salary := net\_salary \* 12;

DBMS\_OUTPUT.PUT\_LINE('Net Salary: ' || net\_salary); DBMS\_OUTPUT.PUT\_LINE('Yearly Salary: ' || yearly\_salary);

END;

/

OUTPUT

SQL> @q5.sql

Net Salary: 23800

Yearly Salary: 285600

QUESTION 23

Write a PL/SQL program that accepts an account number, checks if the balance is below the minimum required balance, and deducts Rs.100/- from the balance if necessary. The program should be applied to the acct table.

PROGRAM CODE

declare

v\_acct\_no number ;

v\_balance number;

begin

v\_acct\_no:=&v\_acct\_no;

dbms\_output.put\_line('account\_no:'||v\_acct\_no);

dbms\_output.put\_line('minimum balance required: rs.1500/-');

select balance into v\_balance from account where acct\_no = v\_acct\_no;

if v\_balance < 1500 then

update account set balance = balance - 100 where acct\_no = v\_acct\_no;

dbms\_output.put\_line('changes made in account. rs.100 deducted.');

else

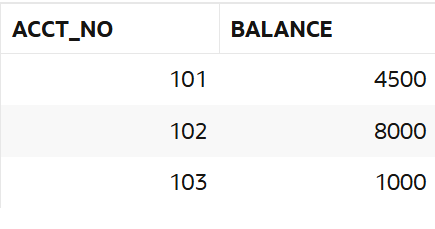
dbms\_output.put\_line('balance sufficient. no changes needed.');

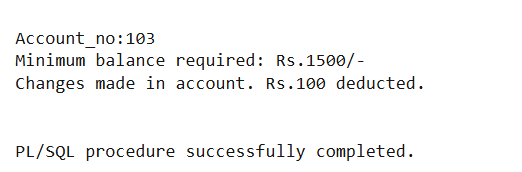
end if;

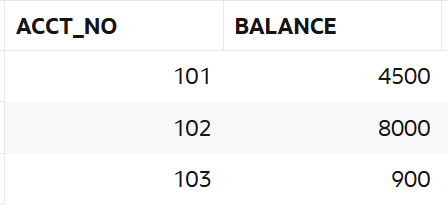
end;

OUTPUT

SQL>@bal.sql







QUESTION 24

Write a PL/SQL function that computes and returns the maximum of two given values.

PROGRAM CODE

create or replace function max\_two(a number, b number)

return number

as

begin

return greatest(a, b);

end;

/

declare

result number;

a number;

b number;

begin

a:=&a;

b:=&b;

result := max\_two(a, b);

dbms\_output.put\_line('maximum of '||a|| ' and '|| b || ' is: '|| result);

end;

OUTPUT

Function created.

Statement processed.

Enter value for a: 20

Ener value for b: 45  
maximum of 25 and 45 is: 45

QUESTION 25

Write a PL/SQL function to check whether a given string is a palindrome.

PROGRAM CODE

create or replace function is\_palindrome(str in varchar)

return varchar2

as

rev\_str varchar(20) := '';

i integer;

begin

for i in reverse 1..length(str) loop

rev\_str := rev\_str || substr(str, i, 1);

end loop;

if lower(str) = lower(rev\_str) then

return 'palindrome';

else

return 'not palindrome';

end if;

end;

/

declare

result varchar(20);

string varchar(20);

begin

string := '&string';

dbms\_output.put\_line('given string: ' || string);

result := is\_palindrome(string);

dbms\_output.put\_line('result: ' || result);

end;

OUTPUT

Statement processed.  
Given string: civic  
Result: palindrome

QUESTION 26

Write a PL/SQL function that returns the total count of customers in the customers table.

PROGRAM CODE

create or replace function customer\_count

return number

as

total number;

begin

select count(\*) into total from customers;

return total;

end;

/

declare

count number;

begin

count := customer\_count;

dbms\_output.put\_line('total customers: ' || count);

end;

/

OUTPUT

|  |  |  |  |
| --- | --- | --- | --- |
| **CUSTOMER\_ID** | **FIRST\_NAME** | **LAST\_NAME** | **EMAIL** |
| 1 | ram | pk | ram123@gmail.com |
| 2 | raj | nath | rajss11@gmail.com |
| 3 | Albert | anto | albert97@gmail.com |

Statement processed.  
total customers: 3

QUESTION 27

Write a PL/SQL procedure to compute and display the sum of two numbers.

PROGRAM CODE

create or replace procedure sum\_two(a in number, b in number)

as

total number;

begin

total := a + b;

dbms\_output.put\_line('sum of ' ||a|| ' and ' || b || ' =' || total);

end;

/

declare

a number;

b number;

begin

a:=&a;

b:=&b;

sum\_two(a, b);

end;

/

OUTPUT

Statement processed.  
Sum of 46 and 34 = 80

QUESTION 28

Write a PL/SQL procedure to insert a student's roll number and name into the student table.

PROGRAM CODE

create or replace procedure insert\_student(p\_roll in number, p\_name in varchar2)

as

begin

insert into student(rollno, name) values (p\_roll, p\_name);

dbms\_output.put\_line('student inserted('||p\_roll || ',' || p\_name || ')');

end;

/

declare

roll number;

name varchar2(20);

begin

roll:=&roll;

name:='&name';

insert\_student(roll, name);

end;

OUTPUT

Statement processed.  
student inserted(4,basil)

QUESTION 29

Write a PL/SQL procedure to retrieve and display the count of instructors in a specified department.

PROGRAM CODE

create or replace procedure instructor\_count(p\_dept in varchar2)

as

cnt number;

begin

select count(\*) into cnt from instructor where dept = p\_dept;

dbms\_output.put\_line('instructors in ' || p\_dept || ': ' || cnt);

end;

/

declare

begin

instructor\_count('mca');

end;

/

OUTPUT

|  |  |  |
| --- | --- | --- |
| **FID** | **DEPT** | **FNAME** |
| 1 | btech | Ram |
| 2 | btech | basil |
| 3 | mca | ravi |
| 4 | btech | arun |

Statement processed.  
Instructors in mca: 1

QUESTION 30

Create a Customers table with attributes (CustId (Primary Key), CustName, City). Then, write a PL/SQL program using an explicit cursor to display all details from the Customers table.

PROGRAM CODE

declare

cursor cust\_cursor is select \* from customers;

rec customers%rowtype;

begin

open cust\_cursor;

loop

fetch cust\_cursor into rec;

exit when cust\_cursor%notfound;

dbms\_output.put\_line('id: ' || rec.custid ||

', name: ' || rec.custname ||

', city: ' || rec.city);

end loop;

close cust\_cursor;

end;

/

OUTPUT

ID: 1, Name: Ajay, City: Thrissur

ID: 2, Name: sam, City: Kottayam

ID: 3, Name: ram, City: Kollam

QUESTION 31

Write a PL/SQL program using an explicit cursor to display details of employees working in the MCA department.

PROGRAM CODE

declare

cursor emp\_cursor is select \* from instructor where dept = 'mca';

rec instructor%rowtype;

begin

open emp\_cursor;

loop

fetch emp\_cursor into rec;

exit when emp\_cursor%notfound;

dbms\_output.put\_line('id: ' || rec.fid || ', name: ' || rec.fname);

end loop;

close emp\_cursor;

end;

OUTPUT

ID: 3, Name: ravi

QUESTION 32

Create a table Teacher with following attributes

Teacher(T\_id, T\_name, Join\_date, Department). Write a trigger that verifies the joining date when a new row is inserted in the ‘teacher’ table. Joining date should be greater than or equal to current date.

PROGRAM CODE

create table teacher ( t\_id number primary key, t\_name varchar2(20), join\_date date,department varchar2(20));

create or replace trigger trg\_check\_join\_date

before insert on teacher

for each row

begin

if :new.join\_date < trunc(sysdate) then

raise\_application\_error(-20001, 'joining date cannot be earlier than today.');

end if;

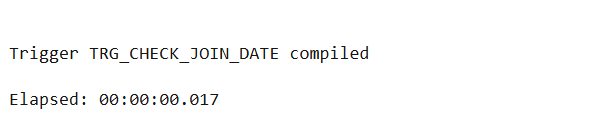
end;

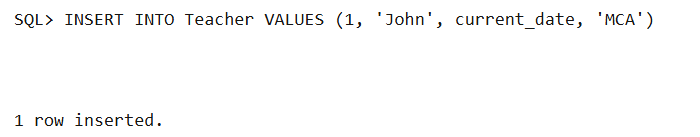
/

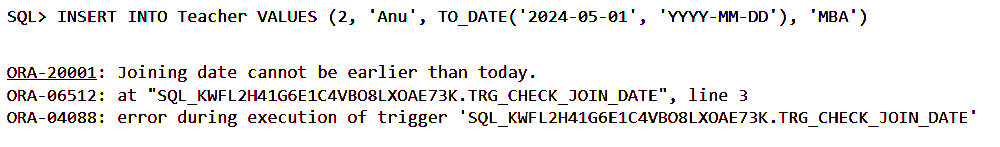
insert into teacher values (1, 'john', current\_date, 'mca');

insert into teacher values (2, 'anu', to\_date('2024-05-01', 'yyyy-mm-dd'), 'mba');

OUTPUT







**EXPERIMENT 8**

Configuration of NoSQL database

QUESTION 33

Comparison between relational and non-relational (NoSQL) databases and the

configuration of NoSQL Databases.

OUTPUT

| **Feature** | **Relational (SQL) Databases** | **Non-Relational (NoSQL) Databases** |
| --- | --- | --- |
| **Data Model** | Structured tables with rows and columns | Flexible models: document, key-value, wide-column, graph |
| **Schema** | Fixed schema; predefined structure | Dynamic schema; allows for unstructured or semi-structured data |
| **Scalability** | Vertical scaling (adding more power to a single server) | Horizontal scaling (adding more servers to distribute the load) |
| **Query Language** | Structured Query Language (SQL) | Varies by database (e.g., MongoDB uses its own query language) |
| **Transactions** | Strong ACID compliance (Atomicity, Consistency, Isolation, Durability) | Some support for ACID; others favor BASE (Basically Available, Soft state, Eventual consistency) |
| **Examples** | MySQL, PostgreSQL, Oracle, Microsoft SQL Server | MongoDB, Cassandra, Redis, Couchbase, Neo4j |
| **Use Cases** | Complex queries, multi-row transactions, structured data | Large volumes of diverse data, real-time analytics, content management, IoT, big data applications |
| **Data Integrity** | Enforced through constraints and relationships | Application-level enforcement; less emphasis on strict data integrity |
| **Flexibility** | Less flexible; changes require altering the schema | Highly flexible; easy to add new fields or data types without affecting existing data |
| **Performance** | Optimized for complex queries and joins | Optimized for high-speed read/write operations and large-scale data handling |

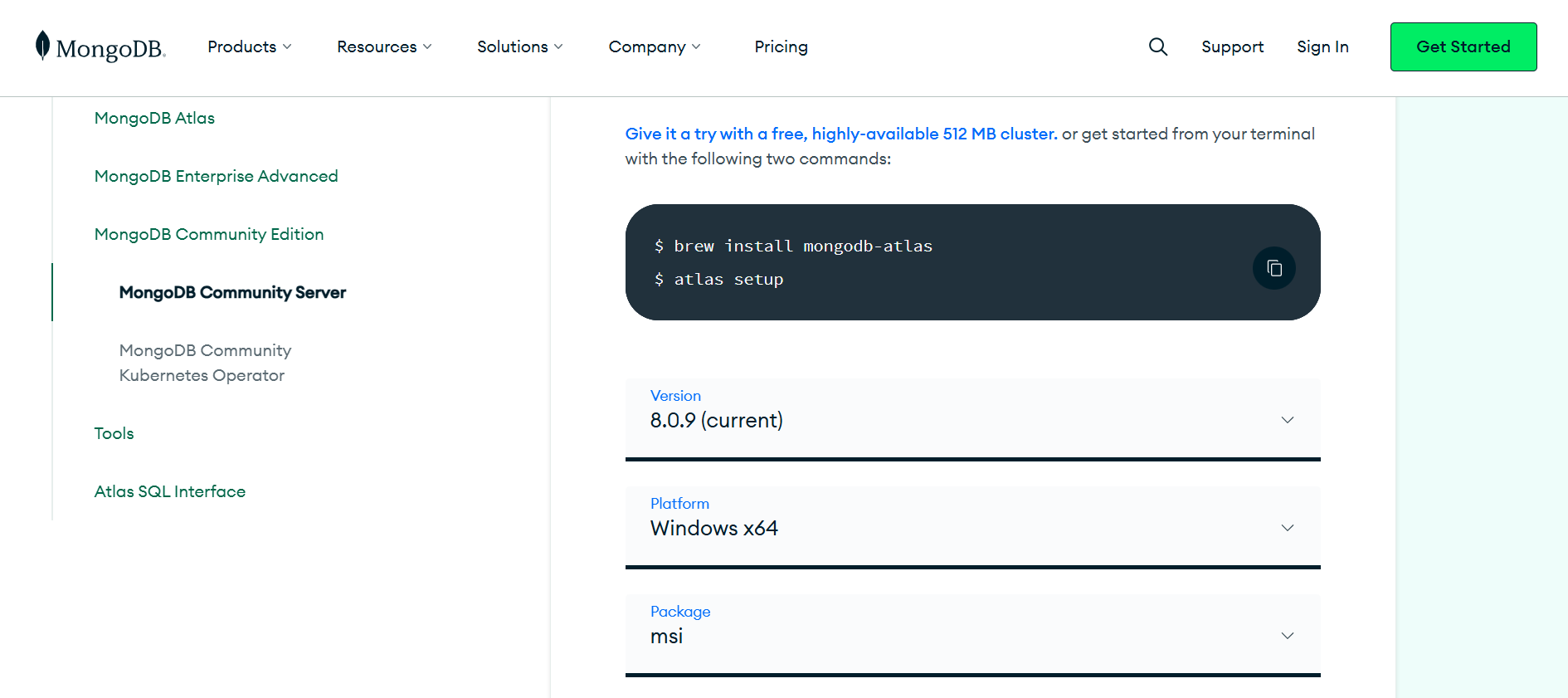
**EXPERIMENT 9**

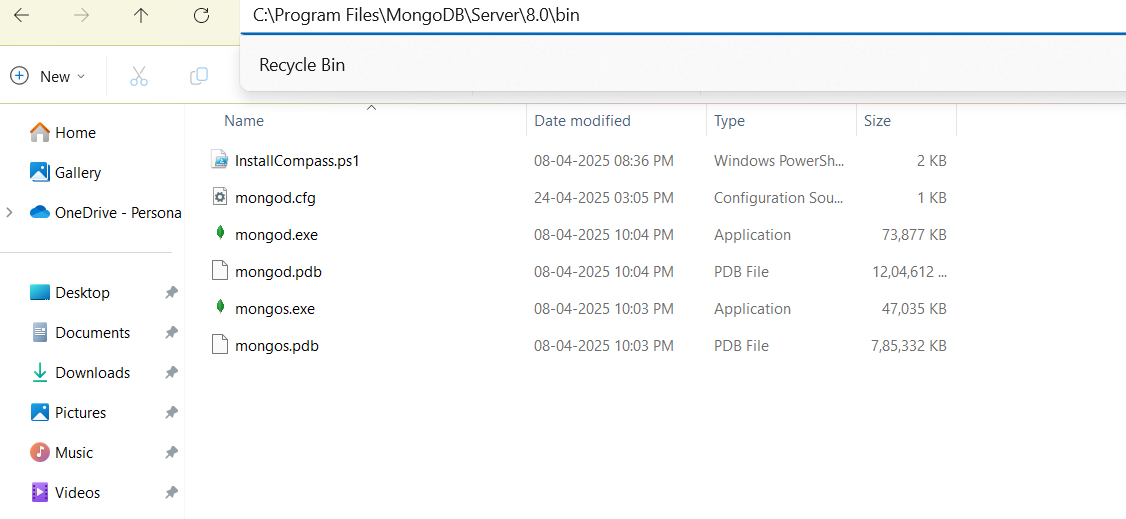
PROGRAMS USING MONGODB

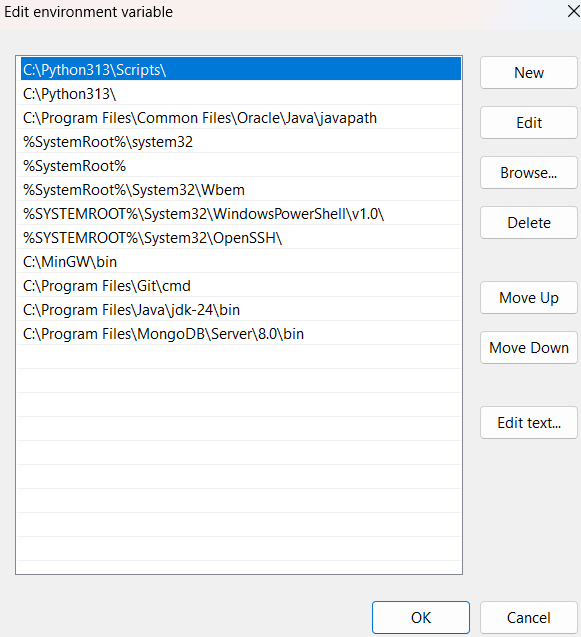
QUESTION 34

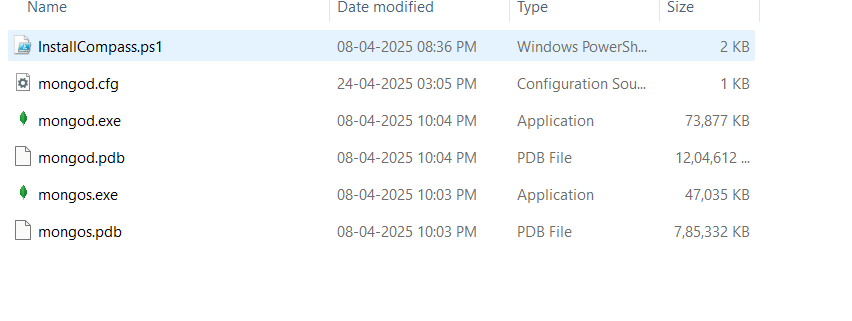
Install the MongoDB and configure it.

OUTPUT











QUESTION 35

Create a collection student consists of details like rollno, name, phoneno, marks,

address, year of course etc.

PROGRAM CODE

test> use college

switched to db college

college> db.createCollection("student19")

{ ok: 1 }

college> db.student19.find()

QUESTION 36

Insert the details of the multiple students (atleast 5) in the form of documents in the

student collection.

PROGRAM CODE

college> db.student19.insertOne({rollno:1,name:"gourav",phoneno:8590451893,marks:94,address:"Kottayam",year:2024})

{

acknowledged: true,

insertedId: ObjectId('6818b16f0859ba1fbdb5f89e')

}

college> db.student19.insertOne({rollno:2,name:"abhi",phoneno:8547459618,marks:90,address:"Thrissur",year:2024})

{

acknowledged: true,

insertedId: ObjectId('6818b1d70859ba1fbdb5f89f')

}

college> db.student19.insertOne({rollno:3,name:"Sujith",phoneno:9605080027,marks:85,address:"Kollam",year:2024})

{

acknowledged: true,

insertedId: ObjectId('6818b20e0859ba1fbdb5f8a0')

}

college> db.student19.insertOne({rollno:4,name:"Abhilash",phoneno:9446491760,marks:94,address:"Kannur",year:2022})

{

acknowledged: true,

insertedId: ObjectId('6818b2520859ba1fbdb5f8a1')

}

college> db.student19.insertOne({rollno:5,name:"Alfred",phoneno:8547210352,marks:92,address:"Thrissur",year:2024})

{

acknowledged: true,

insertedId: ObjectId('6818b2830859ba1fbdb5f8a3')

}

QUESTION 37

Retrieve the fields rollno, name, phoneno, marks, city for all the documents in the

collection student.

PROGRAM CODE

college> db.student19.find()

OUTPUT

[

{

\_id: ObjectId('6818b16f0859ba1fbdb5f89e'),

rollno: 1,

name: 'gourav',

phoneno: 8590451893,

marks: 94,

address: 'Kottayam',

year: 2024

},

{

\_id: ObjectId('6818b1d70859ba1fbdb5f89f'),

rollno: 2,

name: 'abhi',

phoneno: 8547459618,

marks: 90,

address: 'Thrissur',

year: 2024

},

{

\_id: ObjectId('6818b20e0859ba1fbdb5f8a0'),

rollno: 3,

name: 'Sujith',

phoneno: 9605080027,

marks: 85,

address: 'Kollam',

year: 2024

},

{

\_id: ObjectId('6818b2520859ba1fbdb5f8a1'),

rollno: 4,

name: 'Abhilash',

phoneno: 9446491760,

marks: 94,

address: 'Kannur',

year: 2022

},

{

\_id: ObjectId('6818b2830859ba1fbdb5f8a3'),

rollno: 5,

name: 'Alfred',

phoneno: 8547210352,

marks: 92,

address: 'Thrissur',

year: 2024

}

]

QUESTION 38

Display the details of students who achieved a score more than 90 and are from

‘Thrissur’.

PROGRAM CODE

college> db.student19.find({ marks: { $gt: 90 }, address: "Thrissur" })

OUTPUT

[

{

\_id: ObjectId('6818b2830859ba1fbdb5f8a3'),

rollno: 5,

name: 'Alfred',

phoneno: 8547210352,

marks: 92,

address: 'Thrissur',

year: 2024

}

]

QUESTION 39

Update the phone number of Sujith in the student collection. Retrieve the updated

Information.

PROGRAM CODE

college> db.student19.updateOne({name:"Sujith"},{ $set: {phoneno:9876542130}})

college> db.student19.find({ name: "Sujith" })

OUTPUT

[

{

\_id: ObjectId('6818b20e0859ba1fbdb5f8a0'),

rollno: 3,

name: 'Sujith',

phoneno: 9876542130,

marks: 85,

address: 'Kollam',

year: 2024

}]

QUESTION 40

Update the year of course in all the documents in the student collection to 2021. Also

retrieve the updated information.

PROGRAM CODE

college> db.student19.updateMany({},{$set:{year:2021}})

college> db.student19.find()

OUTPUT

[

{

\_id: ObjectId('6818b16f0859ba1fbdb5f89e'),

rollno: 1,

name: 'gourav',

phoneno: 8590451893,

marks: 94,

address: 'Kottayam',

year: 2021

},

{

\_id: ObjectId('6818b1d70859ba1fbdb5f89f'),

rollno: 2,

name: 'abhi',

phoneno: 8547459618,

marks: 90,

address: 'Thrissur',

year: 2021

},

{

\_id: ObjectId('6818b20e0859ba1fbdb5f8a0'),

rollno: 3,

name: 'Sujith',

phoneno: 9876542130,

marks: 85,

address: 'Kollam',

year: 2021

},

{

\_id: ObjectId('6818b2520859ba1fbdb5f8a1'),

rollno: 4,

name: 'Abhilash',

phoneno: 9446491760,

marks: 94,

address: 'Kannur',

year: 2021

},

{

\_id: ObjectId('6818b2830859ba1fbdb5f8a3'),

rollno: 5,

name: 'Alfred',

phoneno: 8547210352,

marks: 92,

address: 'Thrissur',

year: 2021

}

]

QUESTION 41

Display the contact address of ‘Abhilash’.

PROGRAM CODE

college> db.student19.find({name:"Abhilash"},{\_id:0,phoneno:1})

OUTPUT

[ { phoneno: 9446491760 } ]

QUESTION 42

Delete the details of the student whose name is ‘Abhilash’ from the student collection.

PROGRAM CODE

college> db.student19.deleteOne({ name: "Abhilash" })

QUESTION 43

Retrieve the number of students per department from the student collection.

PROGRAM CODE

college> db.student19.aggregate([{$group:{\_id:"$department", count:{$sum:1}}}])

OUTPUT  
[ { \_id: null, count: 5 } ]

QUESTION 44

Arrange the name of the students in ascending order along with all the columns.

PROGRAM CODE

college> db.student19.find().sort({ name: 1 })

OUTPUT

[

{

\_id: ObjectId('6818b1d70859ba1fbdb5f89f'),

rollno: 2,

name: 'Abhi',

phoneno: 8547459618,

marks: 90,

address: 'Thrissur',

year: 2021

},

{

\_id: ObjectId('6818b2830859ba1fbdb5f8a3'),

rollno: 5,

name: 'Alfred',

phoneno: 8547210352,

marks: 92,

address: 'Thrissur',

year: 2021

},

{

\_id: ObjectId('6818b16f0859ba1fbdb5f89e'),

rollno: 1,

name: 'Gourav',

phoneno: 8590451893,

marks: 94,

address: 'Kottayam',

year: 2021

},

{

\_id: ObjectId('6818b20e0859ba1fbdb5f8a0'),

rollno: 3,

name: 'Sujith',

phoneno: 9876542130,

marks: 85,

address: 'Kollam',

year: 2021

}

]

QUESTION 45

Rename city as town and add the detail of address consists of apartment no, street name

and PIN.

PROGRAM CODE

college>db.student19.updateMany({},[{$set:{town:{apartment\_no:"101",street:"Gandhinagar",pin:"680001"}}}])

college> db.student19.updateMany({},{$unset:{address:""}})

college> db.student19.find({rollno:1})

OUTPUT

[ {

\_id: ObjectId('6818b16f0859ba1fbdb5f89e'),

rollno: 1,

name: 'Gourav',

phoneno: 8590451893,

marks: 94,

year: 2021,

town: { apartment\_no: '101', street: 'Gandhinagar', pin: '680001' }

}]

