FEDERAL INSTITUTE OF SCIENCE AND TECHNOLOGY (FISAT)®

HORMIS NAGAR, MOOKKANNOOR, ANGAMALY-683577

FOCUS ON EXCELLENCE

20MCA134 ADVANCED DBMS LAB

……

LABORATORY RECORD

Name: ALEX LISON

Branch: MASTER OF COMPUTER APPLICATIONS

Semester: 2 Batch: A Roll No: 11

University Register Number: FIT24MCA-2011

May 2025

FEDERAL INSTITUTE OF SCIENCE AND TECHNOLOGY (FISAT)™

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 **ALEX LISON (FIT24MCA-2011)** *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
Ms.Anju L

Signature of HOD
Dr. Deepa Mary Mathews

Date of University practical examination …….

Signature of
Internal Examiner

Signature of
External Examiner

CONTENT

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

Creation of a database using DDL commands including integrity

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

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

```
CREATE TABLE STUDENT11 (REGD_NO INTEGER PRIMARY KEY ,NAME
VARCHAR(20) NOT NULL,BRANCH VARCHAR(20) NOT NULL);

INSERT INTO STUDENT11 VALUES(0001,'Ram','CSE');

INSERT INTO STUDENT11 VALUES(0002,'Hari','MECH')

INSERT INTO STUDENT11 VALUES(0003,'Pradeep','EEE');

INSERT INTO STUDENT11 VALUES(0004,'Deepak','ETC');

SELECT * FROM STUDENT11;
```

Output

REGD_NO	NAME	BRANCH
104	Deepak	ETC
101	Ram	CSE
102	Hari	MECH
103	Pradeep	EEE

2. 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 EMPLOYEE11(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));

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

ALTER TABLE EMPLOYEE11 ADD HIREDATE DATE;

Output

TABLE EMPLOYEE11

Column	Null?	Type
EMP_NO	-	NUMBER
E_NAME	-	VARCHAR2(20)
E_ADDRESS	-	VARCHAR2(50)
E_PH_NO	-	NUMBER(10,0)
DEPT_NO	-	NUMBER
DEPT_NAME	-	VARCHAR2(20)
JOB_ID	-	VARCHAR2(20)
SALARY	-	NUMBER(10,2)
HIREDATE	-	DATE

b) Change the datatype of JOB_ID from varchar to integer.

ALTER TABLE EMPLOYEE11 MODIFY(Job_id INTEGER);

Output

TABLE EMPLOYEE11

Column	Null?	Type
EMP_NO	-	NUMBER
E_NAME	-	VARCHAR2(20)
E_ADDRESS	-	VARCHAR2(50)
E_PH_NO	-	NUMBER(10,0)
DEPT_NO	-	NUMBER
DEPT_NAME	-	VARCHAR2(20)
JOB_ID	-	NUMBER
SALARY	-	NUMBER(10,2)
HIREDATE	-	DATE

c) Change the name of column/field Emp_no to E_no.

ALTER TABLE EMPLOYEE11 RENAME COLUMN Emp_no TO E_no;

Output

TABLE EMPLOYEE11

Column	Null?	Type
E_NO	-	NUMBER
E_NAME	-	VARCHAR2(20)
E_ADDRESS	-	VARCHAR2(50)
E_PH_NO	-	NUMBER(10,0)
DEPT_NO	-	NUMBER

DEPT_NAME	-	VARCHAR2(20)
JOB_ID	-	NUMBER
SALARY	-	NUMBER(10,2)
HIREDATE	-	DATE

d) Modify the column width of the Employee name field of emp table.

ALTER TABLE EMPLOYEE11 MODIFY(E_name VARCHAR(30));

Output

TABLE EMPLOYEE11

Column	Null?	Type
E_NO	-	NUMBER
E_NAME	-	VARCHAR2(30)
E_ADDRESS	-	VARCHAR2(50)
E_PH_NO	-	NUMBER(10,0)
DEPT_NO	-	NUMBER
DEPT_NAME	-	VARCHAR2(20)
JOB_ID	-	NUMBER
SALARY	-	NUMBER(10,2)
HIREDATE	-	DATE

3. 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_11(empno INTEGER, ename VARCHAR(20), deptno INTEGER, job VARCHAR(20), hiredate DATE);

CREATE TABLE Department_11(deptno INTEGER,dname VARCHAR(20),loc VARCHAR(50));

Include the following constraints on column of emp table.

a) to make the empno as primary key of the table.

ALTER TABLE Employee_11 MODIFY(empno PRIMARY KEY);

Output

TABLE EMPLOYEE_11

Column	Null?	Type
EMPNO	NOT NULL	NUMBER
ENAME	-	VARCHAR2(20)
DEPTNO	-	NUMBER
JOB	-	VARCHAR2(20)
HIREDATE	-	DATE

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

ALTER TABLE Employee_11 MODIFY(ename NOT NULL);

Output

TABLE EMPLOYEE_11

Column	Null?	Type
EMPNO	NOT NULL	NUMBER
ENAME	NOT NULL	VARCHAR2(20)

DEPTNO	-	NUMBER
JOB	-	VARCHAR2(20)
HIREDATE	-	DATE

c) the job column to have only UPPERCASE entries.

ALTER TABLE Employee_11 MODIFY(job VARCHAR(20) CHECK(job = UPPER(job)));

Output

```
Table altered.
```

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

ALTER TABLE Employee_11 MODIFY(hiredate DEFAULT(CURRENT_DATE));

Output

```
Table altered.
```

Include the following constraints on column of Department table.

a) To make deptno as primary key.

ALTER TABLE Department_11 MODIFY(deptno PRIMARY KEY);

Output

TABLE DEPARTMENT_11

Column	Null?	Type
DEPTNO	NOT NULL	NUMBER
DNAME	-	VARCHAR2(20)
LOC	-	VARCHAR2(50)

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

ALTER TABLE Department_11 MODIFY dname NOT NULL;

ALTER TABLE Department_11 MODIFY loc NOT NULL;

Output

TABLE DEPARTMENT_11

Column	Null?	Type
DEPTNO	NOT NULL	NUMBER
DNAME	NOT NULL	VARCHAR2(20)
LOC	NOT NULL	VARCHAR2(50)

c) 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_11 ADD FOREIGN KEY (deptno) REFERENCES Department_11(deptno);

Output

```
Table altered.
```

Experiment 2: Implementation of DML commands

4. 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_11(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_11 VALUES (10, 'John', 'Pune', '9876543210', 'D10', 'SALES', 'J01', 5000);
INSERT INTO EMPLOYEE_11 VALUES (11, 'James', 'Mumbai', '9876543211', 'D12', 'MECH', 'J02', 4500);
INSERT INTO EMPLOYEE_11 VALUES (12, 'Adam', 'Nagpur', '9876543212', 'D10', 'SALES', 'J03', 4800);
INSERT INTO EMPLOYEE_11 VALUES (13, 'Mary', 'Chennai', '9876543213', 'D12', 'MECH', 'J04', 5200);
INSERT INTO EMPLOYEE_11 VALUES (14, 'Robert', 'Delhi', '9876543214', 'D20', 'IT', 'J05', 4700);
```

2. Display all the information of EMP table.

```
SELECT * FROM EMPLOYEE_11;
```

Output

EMP_NO	E_NAME	E_ADDRESS	E_PH_NO	DEPT_NO	DEPT_NAME	JOB_ID	SALARY
10	John	Pune	9876543210	D10	SALES	J01	5000
11	James	Mumbai	9876543211	D12	MECH	J02	4500
12	Adam	Pune	9876543212	D10	SALES	J03	4800
14	Robert	Delhi	9876543214	D20	IT	J05	4700
13	Mary	Chennai	9876543213	D12	MECH	J04	5200

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

SELECT * FROM EMPLOYEE_11 WHERE Dept_no = 'D10';

Output

EMP_NO	E_NAME	E_ADDRESS	E_PH_NO	DEPT_NO	DEPT_NAME	JOB_ID	SALARY
10	John	Pune	9876543210	D10	SALES	J01	5000
12	Adam	Pune	9876543212	D10	SALES	J03	4800

4. Update the city of Emp_no-12 with current city as Nagpur.

UPDATE EMPLOYEE_11 SET E_address = 'Nagpur' WHERE Emp_no = 12;

Output

EMP_NO	E_NAME	E_ADDRESS	E_PH_NO	DEPT_NO	DEPT_NAME	JOB_ID	SALARY
12	Adam	Nagpur	9876543212	D10	SALES	J03	4800

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

SELECT * FROM EMPLOYEE_11 WHERE Dept_name = 'MECH';

Output

EMP_NO	E_NAME	E_ADDRESS	E_PH_NO	DEPT_NO	DEPT_NAME	JOB_ID	SALARY
11	James	Mumbai	9876543211	D12	MECH	J02	4500
13	Mary	Chennai	9876543213	D12	MECH	J04	5200

6. Delete the email_id of employee James.

ALTER TABLE EMPLOYEE_11 ADD email_id VARCHAR(20);

UPDATE EMPLOYEE_11 SET email_id = 'james@gmail.com' WHERE E_name = 'James';

UPDATE EMPLOYEE_11 SET email_id = 'NULL' WHERE E_name = 'James';

Output

EMP_NO	E_NAME	E_ADDRESS	E_PH_NO	DEPT_NO	DEPT_NAME	JOB_ID	SALARY	EMAIL_ID
11	James	Mumbai	9876543211	D12	MECH	J02	4500	NULL

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

SELECT * FROM EMPLOYEE_11 WHERE Dept_name = 'SALES';

Output

EMP_NO	E_NAME	E_ADDRESS	E_PH_NO	DEPT_NO	DEPT_NAME	JOB_ID	SALARY	EMAIL_ID
10	John	Pune	9876543210	D10	SALES	J01	5000	-
12	Adam	Nagpur	9876543212	D10	SALES	J03	4800	-

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

SELECT Emp_no, E_name, Salary FROM EMPLOYEE_11;

Output

EMP_NO	E_NAME	SALARY
10	John	5000
11	James	4500
12	Adam	4800
14	Robert	4700
13	Mary	5200

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

SELECT E_name FROM EMPLOYEE_11 WHERE Salary >= 4800;

Output

E_NAME
John
Adam
Mary

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));

a) 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);

b) Find the names of clients whose bal_due> 5000

Output

NAME
John
Adam
Eve

c) Change the bal_due of ClientNO " C123" to Rs. 5100

UPDATE Client_master SET bal_due = 5100 WHERE ClientNO = 'C123';

Output

CLIENTNO	NAME	ADDRESS	CITY	STATE	BAL_DUE
C123	Adam	HK House	Aloor	THRISSUR	5100

d) Change the name of Client_master to Client12

ALTER TABLE Client_master RENAME TO Client12;

Output

```
Table altered.
```

e) Display the bal_due heading as "BALANCE"

SELECT bal_due AS BALANCE FROM Client12;

Output

BALANCE
6000
5000
5100
4000
9000

6. (Rollback and Commit commands)

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

CREATE TABLE TEACHER_11(Name VARCHAR(10) PRIMARY KEY, DeptNo VARCHAR(6), Date_of_joining DATE, DeptName VARCHAR(20), Location VARCHAR(15),Salary NUMERIC(6,2));

a) Insert five records

INSERT INTO TEACHER_11 VALUES ('John','D101','08-JAN-01', 'Mathematics', 'THRISSUR', 6000);

INSERT INTO TEACHER_11 VALUES ('Varun','D102','04-JAN-03', 'Commerce', 'Idukki', 7000);

INSERT INTO TEACHER_11 VALUES ('Vinu','D101','05-JAN-08', 'Mathematics', 'THRISSUR', 8000);

INSERT INTO TEACHER_11 VALUES ('Jenna','D103','09-JAN-06', 'English', 'Idukki', 6000);

INSERT INTO TEACHER_11 VALUES ('Eve','D104','07-JAN-05', 'Commerce', 'THRISSUR', 8000);

b) Give Increment of 25% salary for Mathematics Department.

UPDATE TEACHER_11 SET Salary = Salary + (Salary * 0.25) WHERE DeptName = 'Mathematics';

Output

NAME	DEPTNO	DATE_OF_JOINING	DEPTNAME	LOCATION	SALARY
John	D101	08-JAN-01	Mathematics	THRISSUR	7500
Vinu	D101	05-JAN-08	Mathematics	THRISSUR	10000

c) Perform Rollback command

ROLLBACK;

Output

NAME	DEPTNO	DATE_OF_JOINING	DEPTNAME	LOCATION	SALARY
John	D101	08-JAN-01	Mathematics	THRISSUR	6000
Vinu	D101	05-JAN-08	Mathematics	THRISSUR	8000

d) Give Increment of 15% salary for Commerce Department

UPDATE TEACHER_11 SET Salary = Salary + (Salary * 0.15) WHERE DeptName = 'Commerce';

Output

NAME	DEPTNO	DATE_OF_JOINING	DEPTNAME	LOCATION	SALARY
Varun	D102	04-JAN-03	Commerce	Idukki	8050
Eve	D104	07-JAN-05	Commerce	THRISSUR	9200

e) Perform commit command

COMMIT;

Output

NAME	DEPTNO	DATE_OF_JOINING	DEPTNAME	LOCATION	SALARY
Varun	D102	04-JAN-03	Commerce	Idukki	8050
Eve	D104	07-JAN-05	Commerce	THRISSUR	9200

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);

a) 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');

b) Calculate total salesamount in each branch

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

Output

BRANCH	TOTALSALES
East	25000
North	33000
South	41000

c) Calculate average salesamount in each branch .

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

Output

BRANCH	AVERAGESALES
East	25000
North	16500
South	20500

d) 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

SALESNAME	FORMATTEDDOB
David Wilson	30-DEC-87
John Smith	15-DEC-85
Mike Brown	21-DEC-88

e) 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

SALESNAME	FORMATTEDDOB
Emily Davis	12-AUG-92
David Wilson	30-DEC-87
John Smith	15-DEC-85
Mike Brown	21-DEC-88
Sarah Johnson	03-MAR-90

Experiment - 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));
```

a) 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);
```

b) 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

EMPNO	EMPNAME	JOB	BASIC	DA	HRA	PF	GROSSPAY	NETPAY	DEPARTMENT
103	Mike Brown	Analyst	15000	4500	6000	1500	25500	24000	HR
104	Lisa Davis	Designer	9000	2700	3600	900	15300	14400	Creative
105	David Wilson	Tester	5000	1500	2000	500	8500	8000	IT

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

UPDATE Emp

SET NetPay = NetPay + 1200

WHERE NetPay < 10000;

Output

EMPNO	EMPNAME	JOB	BASIC	DA	HRA	PF	GROSSPAY	NETPAY	DEPARTMENT
101	John Smith	Manager	25000	7500	10000	2500	42500	40000	HR
102	Sarah Johnson	Developer	18000	5400	7200	1800	30600	28800	IT
103	Mike Brown	Analyst	15000	4500	6000	1500	25500	24000	HR
104	Lisa Davis	Designer	9000	2700	3600	900	15300	14400	Creative
105	David Wilson	Tester	5000	1500	2000	500	8500	9200	IT

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

SELECT * FROM Emp WHERE GrossPay BETWEEN 10000 AND 20000;

Output

EMPNO	EMPNAME	JOB	BASIC	DA	HRA	PF	GROSSPAY	NETPAY	DEPARTMENT
104	Lisa Davis	Designer	9000	2700	3600	900	15300	14400	Creative

e) Display all the employees who earn maximum salary Employee Table.

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

Output

EMPNO	EMPNAME	JOB	BASIC	DA	HRA	PF	GROSSPAY	NETPAY	DEPARTMENT
101	John Smith	Manager	25000	7500	10000	2500	42500	40000	HR

Experiment – 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

EMP_NO	E_NAME	SALARY
103	Ajay	52000
101	John	50000

2. 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

EMP_NO	E_NAME	E_ADDRESS	E_PH_NO	DEPT_NO	DEPT_NAME	JOB_ID	DESIGNATION	SALARY	HIREDATE
103	Ajay	Thrissur	9812345678	30	Finance	J103	MANAGER	52000	20-JAN-80

3. 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

EMP_NO	E_NAME	E_ADDRESS	E_PH_NO	DEPT_NO	DEPT_NAME	JOB_ID	DESIGNATION	SALARY	HIREDATE
105	David	Alappuzha	9765432187	20	IT	J105	ANALYST	25000	12-JAN-85
106	Seira	Alappuzha	9705432187	20	IT	J106	IT PROFF	51000	11-MAY-95

4. 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

E_NAME	EXPERIENCE_YEARS	DAILY_SALARY
John	43	1666.67
Ajay	45	1733.33
Hiran	44	933.33
David	40	833.33
Seira	29	1700

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

```
SELECT * FROM EMPLOYEE
WHERE Designation IN ('CLERK', 'ANALYST');
```

Output

EMP_NO	E_NAME	E_ADDRESS	E_PH_NO	DEPT_NO	DEPT_NAME	JOB_ID	DESIGNATION	SALARY	HIREDATE
104	Hiran	Ernakulam	9785612345	40	Creative	J104	CLERK	28000	05-MAY-80
105	David	Alappuzha	9765432187	20	IT	J105	ANALYST	25000	12-JAN-85

6. 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

EMP_NO	E_NAME	E_ADDRESS	E_PH_NO	DEPT_NO	DEPT_NAME	JOB_ID	DESIGNATION	SALARY	HIREDATE
101	John	Kochi	9876543210	10	HR	J101	MANAGER	50000	03-DEC-81
103	Ajay	Thrissur	9812345678	30	Finance	J103	MANAGER	52000	19-JAN-80

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

```
SELECT * FROM EMPLOYEE WHERE Dept_no IN (10, 20);
```

Output

EMP_NO	E_NAME	E_ADDRESS	E_PH_NO	DEPT_NO	DEPT_NAME	JOB_ID	DESIGNATION	SALARY	HIREDATE
101	John	Kochi	9876543210	10	HR	J101	MANAGER	50000	03-DEC-81
105	David	Alappuzha	9765432187	20	IT	J105	ANALYST	25000	12-JAN-85
106	Seira	Alappuzha	9705432187	20	IT	J106	IT PROFF	51000	11-MAY-95

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

```
SELECT E_name FROM EMPLOYEE WHERE E_name LIKE 'S%';
```

Output

E_NAME
Seira

9. 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

E_NAME	FIRST_FIVE_CHARS
Hiran	Hiran

10. 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

EMP_NO	E_NAME	E_ADDRESS	E_PH_NO	DEPT_NO	DEPT_NAME	JOB_ID	DESIGNATION	SALARY	HIREDATE
105	David	Alappuzha	9765432187	20	IT	J105	ANALYST	25000	12-JAN-85
104	Hiran	Ernakulam	9785612345	40	Creative	J104	CLERK	28000	05-MAY-80
106	Seira	Alappuzha	9705432187	20	IT	J106	IT PROFF	51000	11-MAY-95

10. Consider Employee table

EMPNO	EMP_NAME	DEPT	SALARY	DOJ	BRANCH
E101	Amit	roduction	45000	12-Mar-00	Bangalore
E102	Amit	HR	70000	03-Jul-02	Bangalore
E103	sunita	anagemer	120000	11-Jan-01	mysore
E105	sunita	IT	67000	01-Aug-01	mysore
E106	mahesh	Civil	145000	20-Sep-03	Mumbai

Perform the following

1. Display all the fields of employee table.

SELECT * FROM Employee;

Output

EMPNO	EMP_NAME	DEPT	SALARY	DOJ	BRANCH
E101	Amit	Production	45000	12-MAR-00	Bangalore
E102	Amit	HR	70000	03-JUL-02	Bangalore
E103	Sunita	Management	120000	11-JAN-01	Mysore
E105	Sunita	IT	67000	01-AUG-01	Mysore
E106	Mahesh	Civil	145000	20-SEP-03	Mumbai

2. Retrieve employee number and their salary.

SELECT EMPNO, SALARY FROM Employee;

Output

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

3. Retrieve average salary of all employee.

SELECT AVG(SALARY) AS Average_Salary FROM Employee;

Output

AVERAGE_SALARY
89400

4. Retrieve number of employee.

SELECT COUNT(*) AS Total_Employees FROM Employee;

Output

TOTAL_EMPLOYEES
5

5. Retrieve distinct number of employee.

SELECT COUNT(DISTINCT EMPNO) AS Distinct_Employees FROM Employee;

Output

DISTINCT_EMPLOYEES
5

6. 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

EMP_NAME	TOTAL_SALARY	NAME_COUNT
Amit	115000	2
Sunita	187000	2
Mahesh	145000	1

7. 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

TOTAL_SALARY
187000
145000

8. Display name of employee in descending order.

```
SELECT EMP_NAME FROM Employee ORDER BY EMP_NAME DESC;
```

Output

EMP_NAME
Sunita
Sunita
Mahesh
Amit
Amit

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

```
SELECT * FROM Employee WHERE EMP_NAME = 'Amit' AND SALARY > 50000;
```

Output

EMPNO	EMP_NAME	DEPT	SALARY	DOJ	BRANCH
E102	Amit	HR	70000	03-JUL-02	Bangalore

11. Create a table called Employee with the following structure.

Name	Type
Empno	Number
Ename	Varchar2(20)
Job	Varchar2(20)
Mgr	Number
Sal	Number

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));

a) 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

EMPNO	ENAME	JOB	MGR	SAL	DEPTNO
7934	MILLER	CLERK	7782	1300	10
7369	SMITH	CLERK	7902	800	20
7900	JAMES	MANAGER	7698	950	30

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

DEPTNO	DNAME	EMPLOYEE_COUNT
10	ACCOUNTING	2
20	RESEARCH	4
30	SALES	3

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

DEPTNO	DEPT_NAME	EMPLOYEE_COUNT
10	ADMIN	2
20	RESEARCH	4
30	SALES	3

d) List all employees which start with either B or C.

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

Output

EMPNO	ENAME	JOB	MGR	SAL	DEPTNO	DEPT_NAME
7944	BEN	SALESMAN	7784	1400	30	SALES
7954	CIBIN	SALESMAN	7794	1200	30	SALES

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

ENAME
KING

f) Calculate the average salary for each different job.

SELECT Job, AVG(Sal) AS Avg_Salary FROM Employee GROUP BY Job;

Output

JOB	AVG_SALARY
SALESMAN	1362.5
CLERK	1066.66666666666666666666666666666666666667
ANALYST	3000
MANAGER	1962.5
PRESIDENT	5000

g) 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

JOB	AVG_SALARY
SALESMAN	1362.5
CLERK	1066.66666666666666666666666666666667
ANALYST	3000
PRESIDENT	5000

h) 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

DEPTNO	AVG_SALARY
30	1280
20	1968.75

i) How many days between day of birth to current date.

SELECT Empno, Ename, DOB, TRUNC(SYSDATE - DOB) AS Days_Lived FROM Employee;

Output

EMPNO	ENAME	DOB	DAYS_LIVED
7369	SMITH	01-JAN-80	16584
7499	ALLEN	15-FEB-81	16173
7566	JONES	10-MAR-82	15785
7654	MARTIN	20-JUL-83	15288

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

```
SELECT Ename,Sal AS Current_Salary,Sal * 1.15
AS Salary_After_Rise FROM Employee;
```

Output

ENAME	CURRENT_SALARY	SALARY_AFTER_RISE
BEN	1400	1610
CIBIN	1200	1380
SMITH	800	920
ALLEN	1600	1840
JONES	2975	3421.25
MARTIN	1250	1437.5
ADAM	3000	3450
KING	5000	5750
ADAMS	1100	1265
JAMES	950	1092.5
MILLER	1300	1495

k) Display lowest paid emp details under each manager.

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

EMPNO	ENAME	JOB	MGR	SAL	DEPTNO	DEPT_NAME
7944	BEN	SALESMAN	7784	1400	30	SALES
7954	CIBIN	SALESMAN	7794	1200	30	SALES
7369	SMITH	CLERK	7902	800	20	RESEARCH
7566	JONES	MANAGER	7839	2975	20	RESEARCH
7788	ADAM	ANALYST	7566	3000	20	RESEARCH
7876	ADAMS	CLERK	7788	1100	20	RESEARCH
7900	JAMES	MANAGER	7698	950	30	SALES
7934	MILLER	CLERK	7782	1300	10	ADMIN

l) Display the average monthly salary bill for each deptno.

SELECT Deptno,Dept_name, AVG(Sal) AS Avg_Monthly_Salary

FROM Employee GROUP BY Deptno,Dept_name;

Output

DEPTNO	DEPT_NAME	AVG_MONTHLY_SALARY
30	SALES	1280
20	RESEARCH	1968.75
10	ADMIN	3150

m) 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

```
no data found
```

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

SELECT Deptno, Empno, Sal, (SELECT AVG(Sal) FROM Employee WHERE Deptno = 5) AS Avg_Salary FROM Employee WHERE Deptno = 5;

Output

DEPTNO	EMPNO	SAL	AVG_SALARY
5	7940	2000	2500
5	7941	3000	2500

o) Count the number of employees in department 20.

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

Output

EMPLOYEE_COUNT
4

p) 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

MIN_SALARY	MAX_SALARY	AVG_SALARY
800	5000	1870.4545454545454545454545454545454545

q) 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

JOB	MIN_SALARY	MAX_SALARY
SALESMAN	1200	1600
CLERK	800	1300
ANALYST	3000	3000
MANAGER	950	2975
PRESIDENT	5000	5000

r) List the employee names in descending order.

SELECT Ename FROM EmployeeORDER BY Ename DESC;

Output

ENAME
SMITH
MILLER
MARTIN
KING
JONES

JAMES
CIBIN
BEN
ALLEN
ADAMS
ADAM

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

```
SELECT Empno, Ename
 FROM Employee
 ORDER BY Empno ASC;
```

Output

EMPNO	ENAME
7369	SMITH
7499	ALLEN
7566	JONES
7654	MARTIN
7788	ADAM
7839	KING
7876	ADAMS

7900	JAMES
7934	MILLER
7944	BEN
7954	CIBIN

Experiment – 5 : Implementation of Join,Views,Set operations

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

DEPT_NO
10
20
30
40

2. 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

DEPT_NO
10
10
20
30
40

3. 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

DEPT_NO
40

DEPT_NO
50

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));

a) 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

SID	SNAME	RATING	AGE
1	Bob	8	25
3	Charlie	9	30

b) 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

BNAME
Sea Queen

c) 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

SNAME
Bob
Eve
Charlie

d) 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

SNAME
Dave
Bob
Charlie
Alice
Eve

e) 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

SID	SNAME
5	Eve

f) 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

SID
1
2
5
3

g) Find the name and the age of the youngest sailor.

```
SELECT sname, age FROM Sailors
WHERE age = (SELECT MIN(age) FROM Sailors);
```

Output

SNAME	AGE
Dave	19

h) Count the number of different sailor names.

```
SELECT COUNT(DISTINCT sname) AS unique_names
FROM Sailors;
```

Output

UNIQUE_NAMES
5

i) Find the average age of sailors for each rating level.

```
SELECT rating, AVG(age) AS avg_age
FROM Sailors
GROUP BY rating;
```

Output

RATING	AVG_AGE
6	28
7	22
8	25
5	19
9	30

j) 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

```
no data found
```

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

```
View created.
```

EMPLOYEE_ID	NAME	SALARY
1	Alice John	55000
2	Bob Mathew	62000
3	Catherine Joy	48000
4	Daniel Roy	75000
5	Eva Thomas	51000

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

```
View created.
```

FULL_NAME	EMAIL
John Doe	john.doe@gmail.com
David Wilson	david.wilson@gmail.com

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

```
View created.
```

EMPLOYEE_ID	NAME	SALARY	GRADE
3	Catherine Joy	45000	1
1	Alice John	56000	2
5	Eva Thomas	62000	2
2	Bob Mathew	72000	3
4	Daniel Roy	98000	4

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);

a) 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

NAME
Alice Johnson
Bob Williams
David Wilson
Emily Davis
John Smith
Michael Brown
Robert Taylor
Sarah Miller

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

```
SELECT Employees.name FROM Employees
INTERSECT
SELECT Managers.name FROM Managers;
```

Output

NAME
Alice Johnson
Emily Davis

c) 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

NAME
Bob Williams
Michael Brown
John Smith

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

NAME	ROLE
Alice Johnson	Employee
Alice Johnson	Manager
Bob Williams	Employee
David Wilson	Manager
Emily Davis	Employee
Emily Davis	Manager
John Smith	Employee
Michael Brown	Employee
Robert Taylor	Manager
Sarah Miller	Manager

Experiment – 6 : Apply PL/SQL for processing databases.

1) 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

```
SQL> @c2q1.sql
 18  /
Enter value for n1: 5
old   6: n1:=&n1;
new   6: n1:=5;
Enter value for n2: 10
old   7: n2:=&n2;
new   7: n2:=10;
n1=10
n2=5

PL/SQL procedure successfully completed.

Commit complete.
SQL> @c2q1.sql
 19  /
Enter value for n1: 5
old   6: n1:=&n1;
new   6: n1:=5;
Enter value for n2: 10
old   7: n2:=&n2;
new   7: n2:=10;
======== After Swapping =========
n1=10
n2=5

PL/SQL procedure successfully completed.
```

2) 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

```
SQL> @c2q2.sql
Enter value for n1: 2
old   6:     n1 := &n1;
new   6:     n1 := 2;
Enter value for n2: 7
old   7:     n2 := &n2;
new   7:     n2 := 7;
Enter value for n3: 5
old   8:     n3 := &n3;
new   8:     n3 := 5;
---------------------
Greatest is =7
---------------------
```

3) 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

```
SQL> @c2q3.sql
Enter value for n: 123
old   7: n:=&n;
new   7: n:=123;
---------------------
Sum = 6
---------------------
```

4) 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

```
SQL> @c2q4.sql
Enter value for n: 321
old   6: n:=&n;
new   6: n:=321;
---------------------
Reversed Number = 123
---------------------
```

5) 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 number;
   da number;
   hra number;
   pf number;
   net_salary number;
   annual_salary number;
begin
   basic:= &basic;
   da := 0.3 * basic;
   hra := 0.1 * basic;

   if basic < 8000 then
      pf := 0.07 * basic;
   elsif basic <= 16000 then
      pf := 0.10 * basic;
   else
      pf := 0.12 * basic;
   end if;
   net_salary := basic + da + hra - pf;
   annual_salary := net_salary * 12;
   dbms_output.put_line('given basic: ' || basic);
   dbms_output.put_line('net salary: ' || net_salary);
   dbms_output.put_line('annual salary: ' || annual_salary);
end;
```

Output

SQL> @c2q5.sql

Enter the value for basic: 9000

given basic: 9000

net salary: 11700

annual salary:140400

6) 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> @c2q6.sql

ACCT_NO	BALANCE
101	4500
102	8000
103	1000

```
Account_no:103
Minimum balance required: Rs.1500/-
Changes made in account. Rs.100 deducted.


PL/SQL procedure successfully completed.
```

ACCT_NO	BALANCE
101	4500
102	8000
103	900

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

Program Code

```
CREATE OR REPLACE FUNCTION get_max(a NUMBER,b NUMBER)
RETURN NUMBER
IS
BEGIN
    IF a > b
    THEN
        RETURN a;
    ELSE
        RETURN b;
    END IF;
END;
/
DECLARE
n1 NUMBER;
n2 NUMBER;
result NUMBER;
BEGIN
    n1:=&n1;
    n2:=&n2;
    result:= get_max(n1,n2);
    dbms_output.put_line('Maximum value is: ' || result);
END;
/
```

Output

```
Function created.
Statement processed.
Enter value for a: 25
Enter value for b: 35
Maximum value is: 35
```

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

Program Code

```
CREATE OR REPLACE FUNCTION is_palindrome(input_string IN VARCHAR)
RETURN VARCHAR2
IS
   reversed VARCHAR(10) := '';
BEGIN
   FOR i IN REVERSE 1 .. LENGTH(input_string)
   LOOP
      reversed := reversed || SUBSTR(input_string, i, 1);
   END LOOP;

   IF input_string = reversed
   THEN
      RETURN 'YES';
   ELSE
      RETURN 'NO';
   END IF;
END;
/
DECLARE
input_string VARCHAR(10);
result VARCHAR(3);
BEGIN
    input_string:='&input_string';
    result:=is_palindrome(input_string);
    dbms_output.put_line('----------------------------');
    dbms_output.put_line('Is '||input_string||' Palindrome ?: '||result);
END;
/
```

Output

```
SQL> @is_palindrome.sql

Function created.

SQL> @c3q7.sql
Enter value for input_string: malayalam
old   5:      input_string:='&input_string';
new   5:      input_string:='malayalam';
----------------------------
Is malayalam Palindrome ?: YES

PL/SQL procedure successfully completed.
```

9) 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
  cust_count NUMBER;
BEGIN
  cust_count := customer_count();
  DBMS_OUTPUT.PUT_LINE('Total customers: ' || cust_count);
END;
/
```

Output

ID	NAME	AGE
1	Adam	22
2	John	21
3	Vinod	19
4	Jeena	21

```
Function created.
Statement processed.
Total customers: 4
```

10) 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 36 and 24 = 60
```

11) 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(101,Adam)
```

12) 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	mca	anu
2	ece	arun
3	mca	deepa
4	eee	varun

```
Statement processed.
Instructors in mca: 2
```

13) 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: Alice, City: Ernakulam
ID: 2, Name: Bob, City: Kottayam
ID: 3, Name: Carol, City: Kollam
```

14) 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: 1, Name: Anu

ID: 2, Name: Deepa
```

15) 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

```
Trigger TRG_CHECK_JOIN_DATE compiled

Elapsed: 00:00:00.017

SQL> INSERT INTO Teacher VALUES (1, 'John', current_date, 'MCA')


1 row inserted.

SQL> INSERT INTO Teacher VALUES (2, 'Anu', TO_DATE('2024-05-01', 'YYYY-MM-DD'), 'MBA')

ORA-20001: Joining date cannot be earlier than today.
ORA-06512: at "SQL_KWFL2H41G6E1C4VBO8LXOAE73K.TRG_CHECK_JOIN_DATE", line 3
ORA-04088: error during execution of trigger 'SQL_KWFL2H41G6E1C4VBO8LXOAE73K.TRG_CHECK_JOIN_DATE'
```

Experiment – 7 : Configuration of NoSQL database

Comparison between relational and non-relational (NoSQL) databases and the configuration of NoSQL Databases.

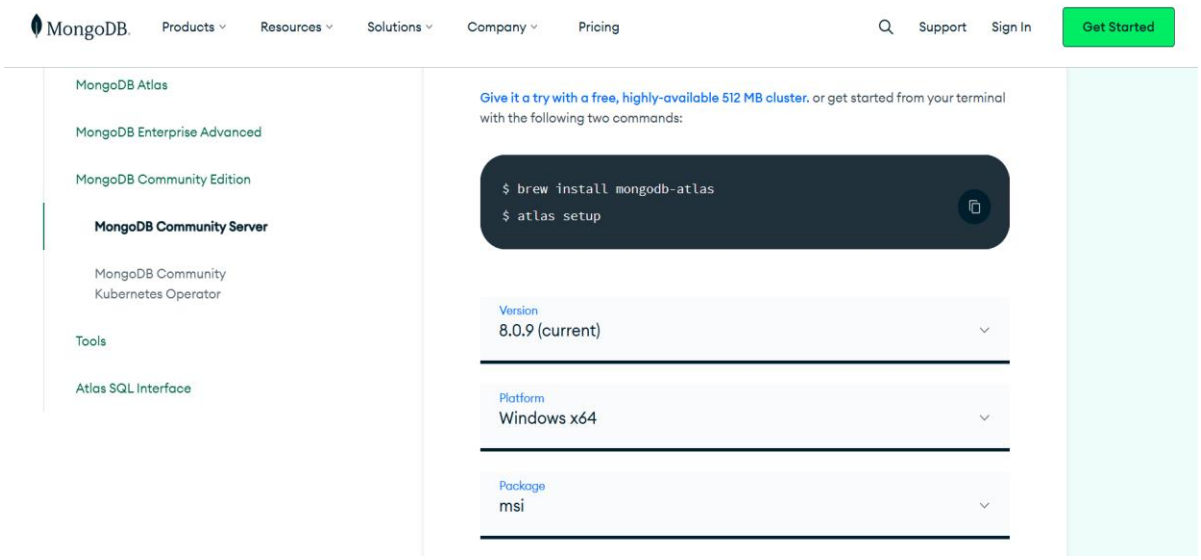
Output

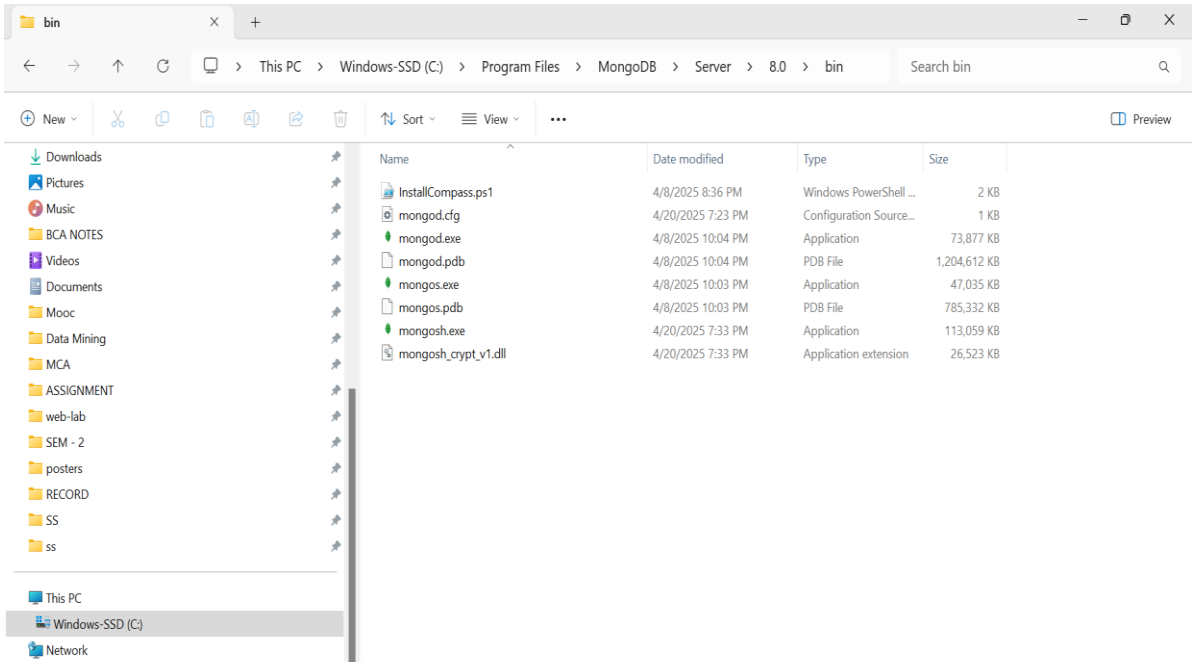
Feature	Relational (SQL) Databases	Non-Relational (NoSQL) Databases
Data Model	Structured tables with rows and columns	Flexible models: document, key-value, wide-column, graph
Schema	Fixed schema; predefined structure	Dynamic schema; allows for unstructured or semi-structured data
Scalability	Vertical 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

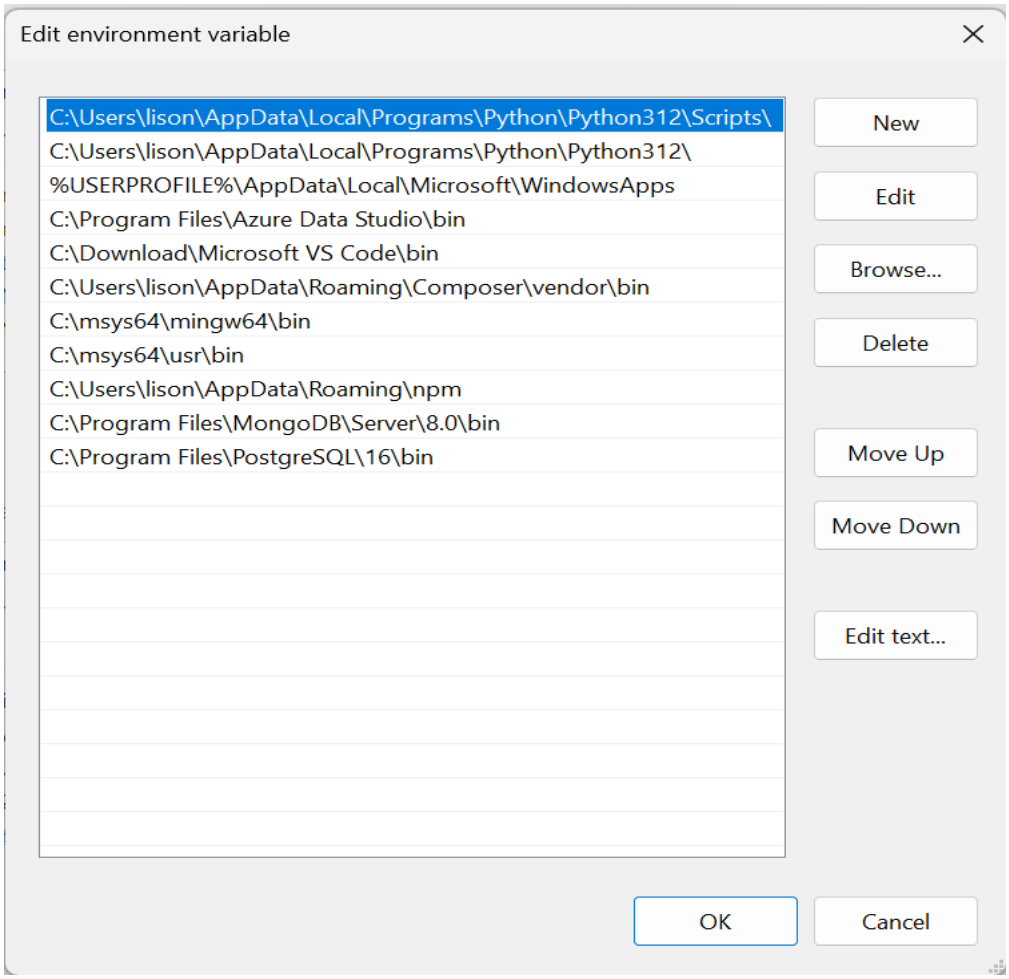
Experiment – 8 : Programs Using MongoDB

1) Install the MongoDB and configure it.

Output






Edit environment variable
C:\Users\lison\AppData\Local\Programs\Python\Python312\Scripts\
C:\Users\lison\AppData\Local\Programs\Python\Python312\
%USERPROFILE%\AppData\Local\Microsoft\WindowsApps
C:\Program Files\Azure Data Studio\bin
C:\Download\Microsoft VS Code\bin
C:\Users\lison\AppData\Roaming\Composer\vendor\bin
C:\msys64\mingw64\bin
C:\msys64\usr\bin
C:\Users\lison\AppData\Roaming\npm
C:\Program Files\MongoDB\Server\8.0\bin
C:\Program Files\PostgreSQL\16\bin
New
Edit
Browse...
Delete
Move Up
Move Down
Edit text...
OK
Cancel


mongosh mongodb://127.0.0.
Microsoft Windows [Version 10.0.22631.5189]
(c) Microsoft Corporation. All rights reserved.
C:\Users\lison>mongosh
Current Mongosh Log ID: 681a1b6505efdb522bb5f898
Connecting to: mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2
.5.0
Using MongoDB: 8.0.8
Using Mongosh: 2.5.0
For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/
The server generated these startup warnings when booting
2025-05-06T18:11:07.434+05:30: Access control is not enabled for the database. Read and write access to data and conf
iguration is unrestricted
test>

2) Create a collection student consists of details like rollno, name, phoneno, marks, address, year of course etc.

db.createCollection("Student")

db.Student.insertOne({rollno:1,name:"Adam",phoneno:8392020029,marks:78,address:"aloor",year_of_course:3})

Output

```
mydb> db.createCollection("Student")
{ ok: 1 }
mydb> db.Student.insertOne({rollno:1,name:"Adam",phoneno:8392020029,marks:78,address:"RN Street,Kottayam",year_of_course
:3,city:"Kottayam"})
{
  acknowledged: true,
  insertedId: ObjectId('6818e51b754842093eb5f89a')
}
```

3) Insert the details of the multiple students (atleast 5) in the form of documents in the student collection.

```
db.Student.insertMany([
    {
        rollno: 2,
        name: "Ajay",
        phoneno: "9234567890",
        marks: 85,
        address: "XYZ Street, Thrissur, Kerala",
        year_of_course: 2,
        city: "Thrissur"
    },
    {
        rollno: 3,
        name: "John",
        phoneno: "9876543210",
        marks: 92,
        address: "ABC Lane, Ernakulam, Kerala",
        year_of_course: 3,
        city: "Ernakulam"
```

```
  },
   {
      rollno: 4,
      name: "Maya",
      phoneno: "9998765432",
      marks: 78,
      address: "PQR Road, Thrissur, Kerala",
      year_of_course: 1,
      city: "Thrissur"
   },
   {
      rollno: 5,
      name: "Sujith",
      phoneno: "+91-9123456789",
      marks: 88,
      address: "STU Avenue, Kochi, Kerala",
      year_of_course: 4,
      city: "Kochi"
   },
   {
      rollno: 6,
      name: "Abhilash",
      phoneno: "8765432109",
      marks: 95,
      address: "LMN Street, Palakkad, Kerala",
      year_of_course: 2,
      city: "Palakkad"
   }
]);
```

Output

```
mydb> db.Student.insertMany([
...     {
...         rollno: 2,
...         name: "Ajay",
...         phoneno: "9234567890",
...         marks: 85,
...         address: "XYZ Street, Thrissur, Kerala",
...         year_of_course: 2,
...         city: "Thrissur"
...     },
...     {
...         rollno: 3,
...         name: "John",
...         phoneno: "9876543210",
...         marks: 92,
...         address: "ABC Lane, Ernakulam, Kerala",
...         year_of_course: 3,
...         city: "Ernakulam"
...     },
...     {
...         rollno: 4,
...         name: "Maya",
...         phoneno: "9998765432",
...         marks: 78,
...         address: "PQR Road, Thrissur, Kerala",
...         year_of_course: 1,
...         city: "Thrissur"
...     },
...     {
...         rollno: 5,
...         name: "Sujith",
...         phoneno: "+91-9123456789",
...         marks: 88,
...         address: "STU Avenue, Kochi, Kerala",
...         year_of_course: 4,
...         city: "Kochi"
...     },
...     {
...         rollno: 6,
...         name: "Abhilash",
...         phoneno: "8765432109",
...         marks: 95,
...         address: "LMN Street, Palakkad, Kerala",
...         year_of_course: 2,
...         city: "Palakkad"
...     }
... ]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('6818e6b9754842093eb5f89b'),
    '1': ObjectId('6818e6b9754842093eb5f89c'),
    '2': ObjectId('6818e6b9754842093eb5f89d'),
    '3': ObjectId('6818e6b9754842093eb5f89e'),
    '4': ObjectId('6818e6b9754842093eb5f89f')
  }
}
```

4) Retrieve the fields rollno, name, phoneno, marks, city for all the documents in the collection student.

db.Student.find()

Output

```
mydb> db.Student.find()
[
  {
    _id: ObjectId('6818e51b754842093eb5f89a'),
    rollno: 1,
    name: 'Adam',
    phoneno: 8392020029,
    marks: 78,
    address: 'RN Street,Kottayam',
    year_of_course: 3,
    city: 'Kottayam'
  },
  {
    _id: ObjectId('6818e6b9754842093eb5f89b'),
    rollno: 2,
    name: 'Ajay',
    phoneno: '9234567890',
    marks: 85,
    address: 'XYZ Street, Thrissur, Kerala',
    year_of_course: 2,
    city: 'Thrissur'
  },
  {
    _id: ObjectId('6818e6b9754842093eb5f89c'),
    rollno: 3,
    name: 'John',
    phoneno: '9876543210',
    marks: 92,
    address: 'ABC Lane, Ernakulam, Kerala',
    year_of_course: 3,
    city: 'Ernakulam'
  },
  {
    _id: ObjectId('6818e6b9754842093eb5f89d'),
    rollno: 4,
    name: 'Maya',
    phoneno: '9998765432',
    marks: 78,
    address: 'PQR Road, Thrissur, Kerala',
    year_of_course: 1,
    city: 'Thrissur'
  },
  {
    _id: ObjectId('6818e6b9754842093eb5f89e'),
    rollno: 5,
    name: 'Sujith',
    phoneno: '+91-9123456789',
    marks: 88,
    address: 'STU Avenue, Kochi, Kerala',
    year_of_course: 4,
    city: 'Kochi'
  },
```

```
  {
    _id: ObjectId('6818e6b9754842093eb5f89f'),
    rollno: 6,
    name: 'Abhilash',
    phoneno: '8765432109',
    marks: 95,
    address: 'LMN Street, Palakkad, Kerala',
    year_of_course: 2,
    city: 'Palakkad'
  }
]
mydb>
```

5) Display the details of students who achieved a score more than 90 and are from 'Thrissur'.

db.Student.find({marks:{$gt:90},city:"Thrissur"})

Output

```
mydb> db.Student.find({marks:{$gt:90},city:"Thrissur"})
[
  {
    _id: ObjectId('6818ead7754842093eb5f8a0'),
    rollno: 7,
    name: 'varun',
    phoneno: 9992015678,
    marks: 93,
    address: 'MKG Street,Thrissur,kerala',
    year_of_course: 3,
    city: 'Thrissur'
  }
]
```

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

db.Student.updateOne({name:"Sujith"},{$set:{phoneno:8383838290}})
db.Student.find({name:"Sujith"})

Output

```
mydb> db.Student.updateOne({name:"Sujith"},{$set:{phoneno:8383838290}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

```
mydb> db.Student.find({name:"Sujith"})
[
  {
    _id: ObjectId('6818e6b9754842093eb5f89e'),
    rollno: 5,
    name: 'Sujith',
    phoneno: 8383838290,
    marks: 88,
    address: 'STU Avenue, Kochi, Kerala',
    year_of_course: 4,
    city: 'Kochi'
  }
```

7) Update the year of course in all the documents in the student collection to 2021. Also retrieve the updated information.

db.Student.updateMany({},{$set:{year_of_course:2021}})
db.Student.find({})

Output

```
mydb> db.Student.updateMany({},{$set:{year_of_course:2
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 7,
  modifiedCount: 7,
  upsertedCount: 0
}
mydb> db.Student.find({})
[
  {
    _id: ObjectId('6818e51b754842093eb5f89a'),
    rollno: 1,
    name: 'Adam',
    phoneno: 8392020029,
    marks: 78,
    address: 'RN Street,Kottayam',
    year_of_course: 2021,
    city: 'Kottayam'
  },
  {
    _id: ObjectId('6818e6b9754842093eb5f89b'),
    rollno: 2,
    name: 'Ajay',
    phoneno: '9234567890',
    marks: 85,
    address: 'XYZ Street, Thrissur, Kerala',
    year_of_course: 2021,
    city: 'Thrissur'
  },
  {
    _id: ObjectId('6818e6b9754842093eb5f89c'),
    rollno: 3,
    name: 'John',
    phoneno: '9876543210',
    marks: 92,
    address: 'ABC Lane, Ernakulam, Kerala',
    year_of_course: 2021,
    city: 'Ernakulam'
  },
```

```
  },
  {
    _id: ObjectId('6818e6b9754842093eb5f89d'),
    rollno: 4,
    name: 'Maya',
    phoneno: '9998765432',
    marks: 78,
    address: 'PQR Road, Thrissur, Kerala',
    year_of_course: 2021,
    city: 'Thrissur'
  },
  {
    _id: ObjectId('6818e6b9754842093eb5f89e'),
    rollno: 5,
    name: 'Sujith',
    phoneno: 8383838290,
    marks: 88,
    address: 'STU Avenue, Kochi, Kerala',
    year_of_course: 2021,
    city: 'Kochi'
  },
  {
    _id: ObjectId('6818e6b9754842093eb5f89f'),
    rollno: 6,
    name: 'Abhilash',
    phoneno: '8765432109',
    marks: 95,
    address: 'LMN Street, Palakkad, Kerala',
    year_of_course: 2021,
    city: 'Palakkad'
  },
  {
    _id: ObjectId('6818ead7754842093eb5f8a0'),
    rollno: 7,
    name: 'varun',
    phoneno: 9992015678,
    marks: 93,
    address: 'MKG Street,Thrissur,kerala',
    year_of_course: 2021,
    city: 'Thrissur'
  }
]
mydb>
```

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

db.Student.deleteOne({ name: "Abhilash" });

Output

```
mydb> db.Student.deleteOne({name:"Abhilash"})
{ acknowledged: true, deletedCount: 1 }
```

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

```
db.Student.aggregate([
 {
  $group: {
   _id: "$department",
   total_students: { $sum: 1 }
  }
 }
]);
```

Output

```
mydb> db.Student.aggregate([
...   {
...     $group: {
...       _id: "$department",
...       total_students: { $sum: 1 }
...     }
...   }
... ]);
[
  { _id: null, total_students: 1 },
  { _id: 'EEE', total_students: 2 },
  { _id: 'CS', total_students: 2 },
  { _id: 'MECH', total_students: 1 }
]
```

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

```
db.Student.find().sort({name:1})
```

Output

```
mydb> db.Student.find().sort({name:1})
[
  {
    _id: ObjectId('6818e51b754842093eb5f89a'),
    rollno: 1,
    name: 'Adam',
    phoneno: 8392020029,
    marks: 78,
    address: 'RN Street,Kottayam',
    year_of_course: 2021,
    city: 'Kottayam',
    department: 'CS'
  },
```

```
  {
    _id: ObjectId('6818e6b9754842093eb5f89b'),
    rollno: 2,
    name: 'Ajay',
    phoneno: '9234567890',
    marks: 85,
    address: 'XYZ Street, Thrissur, Kerala',
    year_of_course: 2021,
    city: 'Thrissur',
    department: 'MECH'
  },
  {
    _id: ObjectId('6818e6b9754842093eb5f89c'),
    rollno: 3,
    name: 'John',
    phoneno: '9876543210',
    marks: 92,
    address: 'ABC Lane, Ernakulam, Kerala',
    year_of_course: 2021,
    city: 'Ernakulam',
    department: 'CS'
  },
  {
    _id: ObjectId('6818e6b9754842093eb5f89d'),
    rollno: 4,
    name: 'Maya',
    phoneno: '9998765432',
    marks: 78,
    address: 'PQR Road, Thrissur, Kerala',
    year_of_course: 2021,
    city: 'Thrissur',
    department: 'EEE'
  },
  {
    _id: ObjectId('6818e6b9754842093eb5f89e'),
    rollno: 5,
    name: 'Sujith',
    phoneno: 8383838290,
    marks: 88,
    address: 'STU Avenue, Kochi, Kerala',
    year_of_course: 2021,
    city: 'Kochi',
    department: 'EEE'
  },
  {
    _id: ObjectId('6818ead7754842093eb5f8a0'),
    rollno: 7,
    name: 'varun',
    phoneno: 9992015678,
    marks: 93,
    address: 'MKG Street,Thrissur,kerala',
    year_of_course: 2021,
    city: 'Thrissur'
  }
]
```

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

```
db.Student.updateMany(
   {},
   {
      $rename: { "city": "town" },
      $set: { "address": { apartment_no: "123", street_name: "XYZ Street", pin: "686001" } }
   }
);
```

Output

```
mydb> db.Student.updateMany(
...     {},
...     {
...         $rename: { "city": "town" },
...         $set: { "address": { apartment_no: "123", street_name: "XYZ Street", pin: "686001" } }
...     }
... );
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 6,
  modifiedCount: 6,
  upsertedCount: 0
}
```

12) Display the contact address of ‘Abhilash’.

```
db.student.find({ name: "Abhilash Reddy" }, { address: 1 });
```

Output

```
mydb> db.Student.find({ name: "Abhilash" }, { address: 1 });
[
  {
    _id: ObjectId('6818ead7754842093eb5f8a0'),
    address: { apartment_no: '123', street_name: 'XYZ Street', pin: '686001' }
  }
]
```