KG COLLEGE OF ARTS AND SCIENCE
Autonomous Institution | Affiliated to Bharathiar University
Accredited with A++ Grade by NAAC
ISO 9001:2015 Certified Institution
KGiSL Campus, Saravanampatti, Coimbatore – 641 035

# LAB MANUAL

# DATABASE MANAGEMENT SYSTEM

## B.Sc. CS / BCA / B.Sc. CT / B.Sc. IT / B.Sc. AI & DS

## 2024 BATCH

## LIST OF EXPERIMENTS

**COURSE TITLE:** DATABASE MANAGEMENT SYSTEM LAB

| S.No | Name of the Experiment |
|---|---|
| | Sample Program |
| 1. | Create a table for Employee details following fields: Name, Designation, Gender, Age, Date of Joining and Salary. Insert at least ten rows and perform various queries using any one Comparison, Logical, Set Operators |
| 2. | Create a table EMP with following fields Empno,Ename, Job, Mgr,Hiredate,Sal , Comm , Deptno. Insert records and perform Transaction Control Statements. |
| 3. | Write queries to get customized output using different SQL Single row functions. |
| 4. | Grouping data and perform aggregation using Aggregate functions & Clauses. |
| 5. | Create necessary tables for bus reservation system application with required constraints**.** View the constraints that were added for the tables. |
| 6. | Create Sailors, Boats & Reserves table and implement Simple Join, Self Join, Outer Join, Inner Join, Left and Right Join. |
| 7. | Create tables for library management system .<br>Master table should have the following fields: ACCNO, TITLE, AUTHOR AND RATE.<br>Transaction table should have the following fields: USER ID, ACCNO, DATE OF ISSUE AND DATE OF RETURN. Create a report with fields Accno, Title, DateofIssue for the given Date of Return with column formats. |
| 8 | Write a PL/SQL to update the rate field by 20% more than the current rate in inventory table which has the following fields: Prono, ProName and Rate. After updating the table a new field (Alter) called for Number of item and place for values for the new field without using PL/SQL block. |
| 9 | a) Write a PL/SQL program to find the factorial of a given number.<br>b) Write a program to accept a number and find the sum of the digits |
| 10 | a) Write a PL/SQL program to find the Reverse String<br>b) Write a PL/SQL program to find the Fibonacci Series |
| 11 | Write a PL/SQL to split the student table into two tables based on result (One table for ―Pass and another for ―Fail). Use cursor for handling records of student table. Assume necessary fields and create a student details table. |
| 12 | Write a PL/SQL to raise the following Exception in Bank Account Management table when deposit amount is zero. |

| | |
|---|---|
| 13 | Write a PL/SQL program to find the total and average of four subjects and display the grade. |
| 14 | Develop a Stored Procedure for Inserting, Updating & Deleting records in TBMARKS table. |
| 15 | Create a database trigger to implement on master and transaction tables which are based on inventory management system for checking data validity. Assume the necessary fields for both tables. |

## Sample Program

### Create department table with the following structure.

| Name | Type |
|---|---|
| Deptno | Number |
| Deptname | Varchar2(10) |
| Location | Varchar2(10) |

a. Add column designation to the department table.
b. Insert values into the table.
c. To Save the record
d. To Read the record
e. Update the record
f. Delete the record
g. Drop the Column
h. Drop the table

**Solution :**

```
SQL> create table  department
(deptno number,deptname varchar2(10),location varchar2(10));

Table created.

SQL> desc department;
 Name                      Null?    Type
 ------------------------- -------- ------
 DEPTNO                             NUMBER
 DEPTNAME                           VARCHAR2(10)
 LOCATION                           VARCHAR2(10)
```

### a. Add column designation to the department table.

```
SQL> alter table department add(designation varchar2(10));

Table altered.
```

```
SQL> DESC DEPARTMENT
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------------------------------
 DEPTNO                                             NUMBER
 DEPTNAME                                           VARCHAR2(10)
 LOCATION                                           VARCHAR2(10)
 DESIGNATION                                        VARCHAR2(10)
```

**b. Insert values into the table.**

```
SQL> insert into department values(&deptno,'&deptname','&location','&designation');
Enter value for deptno: 10
Enter value for deptname: ACCOUNTING
Enter value for location: HYDERABAD
Enter value for designation: MANAGER
old   1: insert into department values(&deptno,'&deptname','&location','&designation')
new   1: insert into department values(10,'ACCOUNTING','HYDERABAD','MANAGER')

1 row created.

SQL> /
Enter value for deptno: 11
Enter value for deptname: SALES
Enter value for location: CHENNAI
Enter value for designation: SALESMAN
old   1: insert into department values(&deptno,'&deptname','&location','&designation')
new   1: insert into department values(11,'SALES','CHENNAI','SALESMAN')

1 row created.

SQL> /
Enter value for deptno: 12
Enter value for deptname: OPERATIONS
Enter value for location: BANGALORE
Enter value for designation: OPERATOR
old   1: insert into department values(&deptno,'&deptname','&location','&designation')
new   1: insert into department values(12,'OPERATIONS','BANGALORE','OPERATOR')

1 row created.
```

**c. To Save the Record**

```
SQL> Commit;
```

**d. To Read the Records**

```
SQL> SELECT * FROM DEPARTMENT;

    DEPTNO    DEPTNAME              LOCATION              DESIGNATIO
```

| | | | |
|---|---|---|---|
| 10 | ACCOUNTING | HYDERABAD | MANAGER |
| 11 | SALES | CHENNAI | SALESMAN |
| 12 | OPERATIONS | BANGALORE | OPERATOR |

**e. Update the record**

SQL> Update DEPARTMENT set LOCATION='COIMBATORE' where Deptno=10;

1 row updated.

SQL> commit;

Commit complete.

SQL> SELECT * FROM DEPARTMENT;

| DEPTNO | DEPTNAME | LOCATION | DESIGNATIO |
|---|---|---|---|
| **10** | **ACCOUNTING** | **COIMBATORE** | **MANAGER** |
| 11 | SALES | CHENNAI | SALESMAN |
| 12 | OPERATIONS | BANGALORE | OPERATOR |

**f. To Delete the record**

SQL> Delete from department where deptno=10;

1 row deleted.

SQL> commit;

Commit complete.

SQL> SELECT * FROM DEPARTMENT;

| DEPTNO | DEPTNAME | LOCATION | DESIGNATIO |
|---|---|---|---|
| 10 | ACCOUNTING | COIMBATORE | MANAGER |
| 12 | OPERATIONS | BANGALORE | OPERATOR |

**g. To Remove all records & undo the deleted records**

SQL> Delete from department;

2 rows deleted.

```
SQL> SELECT * FROM DEPARTMENT;

no rows selected

SQL> ROLLBACK;

Rollback complete.

SQL> SELECT * FROM DEPARTMENT;
```

```
SQL> SELECT * FROM DEPARTMENT;

DEPTNO     DEPTNAME          LOCATION              DESIGNATIO
----------  ---------- -     ---------             ----------
        10     ACCOUNTING    COIMBATORE    MANAGER
        12     OPERATIONS    BANGALORE     OPERATOR
```

**h. To remove the column**

```
SQL> ALTER TABLE DEPARTMENT DROP (DESIGNATION);

Table altered.

SQL>  SELECT * FROM DEPARTMENT;

   DEPTNO DEPTNAME   LOCATION
---------- ---------- ----------
        11 SALES      CHENNAI
        12 OPERATIONS BANGALORE
```

**i. To Remove the table**

```
SQL> DROP TABLE DEPARTMENT;

Table dropped.

SQL> SELECT * FROM DEPARTMENT;
SELECT * FROM DEPARTMENT
         *
ERROR at line 1:
     ORA-00942: table or view does not exist
```

# PROGRAM 1 - EMPLOYEE DATABASE

**AIM:**

Create a table for Employee details with Employee Number as primary key and following fields: Name, Designation, Gender, Age, Date of Joining and Salary. Insert at least ten rows and perform various queries using any one Comparison, Logical, Set, Sorting operators.

**Program :**

**1.CREATE A TABLE FOR EMPLOYEE DETAILS:**

```
create table employee(empno number primary key,empname varchar2(20),designation varchar2(30),gender
varchar2(6),age number,dojdate,salary number);
```

Table created.

**2. DESCRIBE A TABLE:**

```
SQL>desc employee;
Name                                      Null?    Type
----------------------------------------- -------- ----------------------------
EMPNO                                     NOT NULL NUMBER
EMPNAME                                            VARCHAR2(20)
DESIGNATION                                        VARCHAR2(30)
GENDER                                             VARCHAR2(6)
AGE                                                NUMBER
DOJ                                                DATE
SALARY                                             NUMBER
```

**3.INSERT A VALUES FOR A EMPLOYEE TABLE:**

```
SQL> insert into employee values(&empno,'&empname','&designation','&gender',&age,'&doj',&salary);
```

**4.SELECT ALL THE ROWS FROM THE EMPLOYEE TABLE:**

```
SQL>select * from employee;
```

| EMPNO | EMPNAME | DESIGNATION | GENDER | AGE | DOJ | SALARY |
|---|---|---|---|---|---|---|
| 101 | arjun | manager | male | 30 | 12-JAN-14 | 35000 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 103 | manoj | clerk | male | 26 | 28-MAR-15 | 28000 |
| 102 | john | hr | male | 32 | 17-APR-15 | 30000 |
| 105 | peter | marketing manager | male | 35 | 01-FEB-15 | 35000 |
| 104 | pooja | project developer | female | 27 | 14-APR-15 | 25000 |
| 107 | aishu | tester | female | 25 | 20-JUL-15 | 24000 |
| 108 | yamuna | clerk | female | 30 | 11-JAN-15 | 31000 |

7 rows selected.

**5.COMPARISON:**

(i) SQL> select * from employee where salary>30000;

| EMPNO | EMPNAME | DESIGNATION | GENDER | AGE | DOJ | SALARY |
|---|---|---|---|---|---|---|
| 101 | arjun | manager | male | 30 | 12-JAN-14 | 35000 |
| 105 | peter | marketing manager | male | 35 | 01-FEB-15 | 35000 |
| 108 | yamuna | clerk | female | 30 | 11-JAN-15 | 31000 |

(ii) SQL> select * from employee where age between 25 and 30;

| EMPNO | EMPNAME | DESIGNATION | GENDER | AGE | DOJ | SALARY |
|---|---|---|---|---|---|---|
| 101 | arjun | manager | male | 30 | 12-JAN-14 | 35000 |
| 103 | manoj | clerk | male | 26 | 28-MAR-15 | 28000 |
| 104 | pooja | project developer | female | 27 | 14-APR-15 | 25000 |
| 107 | aishu | tester | female | 25 | 20-JUL-15 | 24000 |
| 108 | yamuna | clerk | female | 30 | 11-JAN-15 | 31000 |

(iii) SQL> select * from employee where empname like '%a';

| EMPNO | EMPNAME | DESIGNATION | GENDER | AGE | DOJ | SALARY |
|---|---|---|---|---|---|---|
| 104 | pooja | project developer | female | 27 | 14-APR-15 | 25000 |
| 108 | yamuna | clerk | female | 30 | 11-JAN-15 | 31000 |

(iv) SQL> select * from employee where salary in(35000,30000**,28000);**

| EMPNO | EMPNAME | DESIGNATION | GENDER | AGE | DOJ | SALARY |
|---|---|---|---|---|---|---|
| 101 | arjun | manager | male | 30 | 12-JAN-14 | 35000 |
| 103 | manoj | clerk | male | 26 | 28-MAR-15 | 28000 |
| 102 | john | hr | male | 32 | 17-APR-15 | 30000 |
| 105 | peter | marketing manager | male | 35 | 01-FEB-15 | 35000 |

(v) SQL>select * from employee where empno=103;

| EMPNO | EMPNAME | DESIGNATION | GENDER | AGE | DOJ | SALARY |
|---|---|---|---|---|---|---|
| 103 | manoj | clerk | male | 26 | 28-MAR-15 | 28000 |

**5.LOGICAL:**

(i) SQL> select * from employee where salary<50000 and salary>30000;

| EMPNO | EMPNAME | DESIGNATION | GENDER | AGE | DOJ | SALARY |
|---|---|---|---|---|---|---|
| 101 | arjun | manager | male | 30 | 12-JAN-14 | 35000 |
| 105 | peter | marketing manager | male | 35 | 01-FEB-15 | 35000 |
| 108 | yamuna | clerk | female | 30 | 11-JAN-15 | 31000 |

(ii) SQL> select * from employee where designation='manager' or designation='admin';

| EMPNO | EMPNAME | DESIGNATION | GENDER | AGE | DOJ | SALARY |
|---|---|---|---|---|---|---|
| 101 | arjun | manager | male | 30 | 12-JAN-14 | 35000 |

(iii) SQL> select * from employee where not salary<30000;

| EMPNO | EMPNAME | DESIGNATION | GENDER | AGE | DOJ | SALARY |
|---|---|---|---|---|---|---|
| 101 | arjun | manager | male | 30 | 12-JAN-14 | 35000 |
| 102 | john | hr | male | 32 | 17-APR-15 | 30000 |
| 105 | peter | marketing manager | male | 35 | 01-FEB-15 | 35000 |
| 108 | yamuna | clerk | female | 30 | 11-JAN-15 | 31000 |

4 rows selected.

**6.SORTING:**

(i) SQL> select * from employee order by empno;

| EMPNO | EMPNAME | DESIGNATION | GENDER | AGE | DOJ | SALARY |
|---|---|---|---|---|---|---|
| 101 | arjun | manager | male | 30 | 12-JAN-14 | 35000 |
| 102 | john | hr | male | 32 | 17-APR-15 | 30000 |
| 103 | manoj | clerk | male | 26 | 28-MAR-15 | 28000 |
| 104 | pooja | project developer | female | 27 | 14-APR-15 | 25000 |
| 105 | peter | marketing manager | male | 35 | 01-FEB-15 | 35000 |
| 107 | aishu | tester | female | 25 | 20-JUL-15 | 24000 |
| 108 | yamuna | clerk | female | 30 | 11-JAN-15 | 31000 |

7 rows selected.

(ii) SQL> select * from employee order by empno desc;

| EMPNO | EMPNAME | DESIGNATION | GENDER | AGE | DOJ | SALARY |
|---|---|---|---|---|---|---|
| 108 | yamuna | clerk | female | 30 | 11-JAN-15 | 31000 |
| 107 | aishu | tester | female | 25 | 20-JUL-15 | 24000 |
| 105 | peter | marketing manager | male | 35 | 01-FEB-15 | 35000 |
| 104 | pooja | project developer | female | 27 | 14-APR-15 | 25000 |
| 103 | manoj | clerk | male | 26 | 28-MAR-15 | 28000 |
| 102 | john | hr | male | 32 | 17-APR-15 | 30000 |
| 101 | arjun | manager | male | 30 | 12-JAN-14 | 35000 |

7 rows selected.

**7.SET OPERATION:**

(i) SQL> select * from employee where salary>30000 union select * from employee where age between 25 and 30;

| EMPNO | EMPNAME | DESIGNATION | GENDER | AGE | DOJ | SALARY |
|---|---|---|---|---|---|---|
| 101 | arjun | manager | male | 30 | 12-JAN-14 | 35000 |
| 103 | manoj | clerk | male | 26 | 28-MAR-15 | 28000 |
| 104 | pooja | project developer | female | 27 | 14-APR-15 | 25000 |
| 105 | peter | marketing manager | male | 35 | 01-FEB-15 | 35000 |
| 107 | aishu | tester | female | 25 | 20-JUL-15 | 24000 |
| 108 | yamuna | clerk | female | 30 | 11-JAN-15 | 31000 |

6 rows selected.

(ii) SQL> select * from employee where salary>30000 union all select * from employee where age between 25 and 30;

| EMPNO | EMPNAME | DESIGNATION | GENDER | AGE | DOJ | SALARY |
|---|---|---|---|---|---|---|
| 101 | arjun | manager | male | 30 | 12-JAN-14 | 35000 |
| 105 | peter | marketing manager | male | 35 | 01-FEB-15 | 35000 |
| 108 | yamuna | clerk | female | 30 | 11-JAN-15 | 31000 |
| 101 | arjun | manager | male | 30 | 12-JAN-14 | 35000 |
| 103 | manoj | clerk | male | 26 | 28-MAR-15 | 28000 |
| 104 | pooja | project developer | female | 27 | 14-APR-15 | 25000 |
| 107 | aishu | tester | female | 25 | 20-JUL-15 | 24000 |
| 108 | yamuna | clerk | female | 30 | 11-JAN-15 | 31000 |

8 rows selected.

(iii) SQL> select * from employee where salary>30000 intersect select * from employee where age between 25 and 30;

| EMPNO | EMPNAME | DESIGNATION | GENDER | AGE | DOJ | SALARY |
|---|---|---|---|---|---|---|
| 101 | arjun | manager | male | 30 | 12-JAN-14 | 35000 |
| 108 | yamuna | clerk | female | 30 | 11-JAN-15 | 31000 |

(iv) SQL> select * from employee where salary>30000 minus select * from employee where age between 25 and 30;

| EMPNO | EMPNAME | DESIGNATION | GENDER | AGE | DOJ | SALARY |
|---|---|---|---|---|---|---|
| 105 | peter | marketing manager | male | 35 | 01-FEB-15 | 35000 |

## PROGRAM 2: IMPLEMENTATION OF TRANSACTION CONTROL STATEMENTS

**Aim :**

Create a table EMP with following fields EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO. Store 5 records and use ROLLBACK, SAVEPOINT and COMMIT commands

**Program :**

```
SQL> Create table EMP(EMPNO number(4) primary key,Ename varchar(10),JOB Varchar(9),
     MGR number(4),Hiredate date,SAL number(7,2),Comm Number(7,2),Deptno number(2));

Table created.
```

**SQL> -- Sturcture of the table**

```
SQL> DESC EMP
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMPNO                                     NOT NULL NUMBER(4)
 ENAME                                              VARCHAR2(10)
 JOB                                                VARCHAR2(9)
 MGR                                                NUMBER(4)
 HIREDATE                                           DATE
 SAL                                                NUMBER(7,2)
 COMM                                               NUMBER(7,2)
 DEPTNO                                             NUMBER(2)
```

**SQL> -- To Store Records**

```
SQL> insert into EMP values(7369,'SMITH','CLERK',7902,'17-Dec-80',800,null,20);

1 row created.

SQL> insert into EMP values(7499,'ALLEN','SALESMAN',7698,'2-FEB-81',1600,300,30);

1 row created.
```

**SQL> -- To Save the Record**

```
SQL> Commit;

Commit complete.
```

**SQL> -- To Read the Records**

```
SQL> select * from emp;
```

```
    EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
     7369 SMITH      CLERK            7902 17-DEC-80        800                    20
     7499 ALLEN      SALESMAN         7698 02-FEB-81       1600        300         30
```

SQL> insert into emp values(1234,'Miller','Manager',7778,'12-Dec-90',10000,null,10);

1 row created.

**SQL> -- To Read the records**
**SQL> -- Display Committed & Uncommitted Records**
SQL>
SQL> select * from emp;

```
    EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
     7369 SMITH      CLERK            7902 17-DEC-80        800                    20
     7499 ALLEN      SALESMAN         7698 02-FEB-81       1600        300         30
     1234 Miller     Manager          7778 12-DEC-90      10000                    10
```

**SQL> -- To UNDO the changes**
**SQL> Rollback;**

Rollback complete.

**SQL> -- Display Committed Records**

SQL> select * from emp;

```
    EMPNO     ENAME      JOB              MGR                 HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
     7369         SMITH      CLERK            7902                 17-DEC-80        800                    20
     7499         ALLEN      SALESMAN         7698   02-FEB-81       1600        300         30
```

SQL> insert into emp values(7521,'WARD','SALESMAN',7698,'22-FEB-81',1250,500,30);

1 row created.

SQL> select * from emp;

```
    EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
     7369 SMITH      CLERK            7902 17-DEC-80        800                    20
     7499 ALLEN      SALESMAN         7698 02-FEB-81       1600        300         30
```

```
     7521 WARD       SALESMAN        7698 22-FEB-81       1250        500         30

SQL> -- 3rd record is uncommitted
SQL>
SQL> SAVEPOINT A;

Savepoint created.

SQL>  insert into emp values(1251,'MILLER','PRESIDENT',7499, '31-dec-90',10000,null,10);

1 row created.

SQL> -- 4th Record is Uncommited
SQL> select * from emp;

    EMPNO ENAME      JOB           MGR HIREDATE        SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
     7369 SMITH      CLERK          7902 17-DEC-80        800                    20
     7499 ALLEN      SALESMAN        7698 02-FEB-81       1600        300         30
     7521 WARD       SALESMAN        7698 22-FEB-81       1250        500         30
     1251 MILLER     PRESIDENT       7499 31-DEC-90      10000                    10

SQL> -- Have to Save 3rd Record & Undo the 4th Record
SQL>
SQL> Rollback to A;

Rollback complete.

SQL>  select * from emp;

    EMPNO ENAME      JOB           MGR HIREDATE        SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
     7369 SMITH      CLERK          7902 17-DEC-80        800                    20
     7499 ALLEN      SALESMAN        7698 02-FEB-81       1600        300         30
     7521 WARD       SALESMAN        7698 22-FEB-81       1250        500         30

SQL> -- 4th Record not listed
SQL>
SQL> -- Save the the 3rd Record
SQL>
SQL> commit;

Commit complete.

SQL> select * from emp;
```

```
    EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
      7499 ALLEN      SALESMAN        7698 02-FEB-81       1600        300         30
      7521 WARD       SALESMAN        7698 22-FEB-81       1250        500         30

SQL>
SQL> insert into emp values(&EMPNO,'&ENAME','&JOB','&HIREDATE',&SAL,&COMM,&DEPTNO)
  2
SQL> insert into emp
values(&EMPNO,'&ENAME','&JOB',&MGR,'&HIREDATE',&SAL,&COMM,&DEPTNO);
Enter value for empno: 7566
Enter value for ename: JONES
Enter value for job: MANAGER
Enter value for mgr: 7839
Enter value for hiredate: 02-APR-81
Enter value for sal: 2975
Enter value for comm: null
Enter value for deptno: 20
old   1: insert into emp
values(&EMPNO,'&ENAME','&JOB',&MGR,'&HIREDATE',&SAL,&COMM,&DEPTNO)
new   1: insert into emp values(7566,'JONES','MANAGER',7839,'02-APR-81',2975,null,20)

1 row created.



SQL> /
Enter value for empno: 7654
Enter value for ename: MARTIN
Enter value for job: SALESMAN
Enter value for mgr: 7698
Enter value for hiredate: 28-SEP-81
Enter value for sal: 1250
Enter value for comm: 1400
Enter value for deptno: 30
old   1: insert into emp
values(&EMPNO,'&ENAME','&JOB',&MGR,'&HIREDATE',&SAL,&COMM,&DEPTNO)
new   1: insert into emp values(7654,'MARTIN','SALESMAN',7698,'28-SEP-81',1250,1400,30)

1 row created.
```

```
SQL> /
Enter value for empno: 7698
Enter value for ename: BLAKE
Enter value for job: MANAGER
Enter value for mgr: 7839
Enter value for hiredate: 01-MAY-81
Enter value for sal: 2450
Enter value for comm: null
Enter value for deptno: 30
old   1: insert into emp
values(&EMPNO,'&ENAME','&JOB',&MGR,'&HIREDATE',&SAL,&COMM,&DEPTNO)
new   1: insert into emp values(7698,'BLAKE','MANAGER',7839,'01-MAY-81',2450,null,30)

1 row created.

SQL> /
Enter value for empno: 7782
Enter value for ename: CLARK
Enter value for job: MANAGER
Enter value for mgr: 7839
Enter value for hiredate: 09-JUN-81
Enter value for sal: 2850
Enter value for comm: null
Enter value for deptno: 10
old   1: insert into emp
values(&EMPNO,'&ENAME','&JOB',&MGR,'&HIREDATE',&SAL,&COMM,&DEPTNO)
new   1: insert into emp values(7782,'CLARK','MANAGER',7839,'09-JUN-81',2850,null,10)

1 row created.

SQL> /
Enter value for empno: 7788
Enter value for ename: SCOTT
Enter value for job: ANALYST
Enter value for mgr: 7566
Enter value for hiredate: 19-APR-87
Enter value for sal: 3000
Enter value for comm: null
Enter value for deptno: 20
old   1: insert into emp
values(&EMPNO,'&ENAME','&JOB',&MGR,'&HIREDATE',&SAL,&COMM,&DEPTNO)
new   1: insert into emp values(7788,'SCOTT','ANALYST',7566,'19-APR-87',3000,null,20)
1 row created.

SQL> commit;
```

**SQL> SELECT * FROM EMP;**

| EMPNO | ENAME | JOB | MGR | HIREDATE | SAL | COMM | DEPTNO |
|---|---|---|---|---|---|---|---|
| 7839 | KING | PRESIDENT | | 17-NOV-81 | 5000 | | 10 |
| 7698 | BLAKE | MANAGER | 7839 | 01-MAY-81 | 2850 | | 30 |
| 7782 | CLARK | MANAGER | 7839 | 09-JUN-81 | 2450 | | 10 |
| 7566 | JONES | MANAGER | 7839 | 02-APR-81 | 2975 | | 20 |
| 7654 | MARTIN | SALESMAN | 7698 | 28-SEP-81 | 1250 | 1400 | 30 |
| 7499 | ALLEN | SALESMAN | 7698 | 20-FEB-81 | 1600 | 300 | 30 |
| 7844 | TURNER | SALESMAN | 7698 | 08-SEP-81 | 1500 | 0 | 30 |
| 7900 | JAMES | CLERK | 7698 | 03-DEC-81 | 950 | | 30 |
| 7521 | WARD | SALESMAN | 7698 | 22-FEB-81 | 1250 | 500 | 30 |
| 7902 | FORD | ANALYST | 7566 | 03-DEC-81 | 3000 | | 20 |
| 7369 | SMITH | CLERK | 7902 | 17-DEC-80 | 800 | | 20 |
| 7788 | SCOTT | ANALYST | 7566 | 09-DEC-82 | 3000 | | 20 |
| 7876 | ADAMS | CLERK | 7788 | 12-JAN-83 | 1100 | | 20 |
| 7934 | MILLER | CLERK | 7782 | 23-JAN-82 | 1300 | | 10 |

# PROGRAM 3 : SQL SINGLE ROW FUNCTIONS

**Aim :**

Write queries to get customized output using different SQL Single row functions.
a) Write a query to display the current date. Label the column DATE.
b) For each employee, display the EMPNO,ENAME,SAL and SAL increased by 15% and expressed as a whole number. Label the column New Salary.
c) Write a query that displays the ENAME with the first letter capitalized and all other letters lowercase and the length of the name for all employees.
d) Display the employee's name, and calculate the number of months between today and the date the employees hired. Label the column MONTHS_WORKED. Order your results by the number of months employed.

**Program :**

a) Write a query to display the current date. Label the column DATE.

```
SQL> select SYSDATE from dual;

SYSDATE
---------
02-DEC-25
```

b) For each employee, display the EMPNO,ENAME,SAL and SAL increased by 15% and expressed as a whole number. Label the column New Salary.

```
SQL> SELECT   EMPNO,ENAME,SAL, ROUND(sal * 1.15, 0) "New Salary"    FROM     emp;

    EMPNO     ENAME          SAL          New Salary
----------    ----------     ---------- -    ---------
     7369     SMITH           800          920
     7499     ALLEN          1600          1840
     7521     WARD           1250          1438
     7566      JONES         2975          3421
     7654     MARTIN         1250          1438
     7698     BLAKE          2450          2818
     7782     CLARK          2850          3278
     7788     SCOTT          3000          3450
```

c) Write a query that displays the ENAME with the first letter capitalized and all other letters lowercase and the length of the name for all employees.

SQL> SELECT ENAME,INITCAP(ENAME),LENGTH(ENAME) from EMP;

| ENAME | INITCAP(EN | LENGTH(ENAME) |
|---|---|---|
| SMITH | Smith | 5 |
| ALLEN | Allen | 5 |
| WARD | Ward | 4 |
| JONES | Jones | 5 |
| MARTIN | Martin | 6 |
| BLAKE | Blake | 5 |
| CLARK | Clark | 5 |
| SCOTT | Scott | 5 |

d) Display the employee's name, and calculate the number of months between today and the date the employees hired. Label the column MONTHS_WORKED. Order your results by the number of months employed.

```
SQL> SELECT ENAME,HIREDATE,TRUNC(MONTHS_BETWEEN(SYSDATE,HIREDATE))
    "MONTHS_WORKED" FROM EMP ORDER BY
 MONTHS_BETWEEN(SYSDATE,HIREDATE);
```

| ENAME | HIREDATE | MONTHS_WORKED |
|---|---|---|
| ADAMS | 12-JAN-83 | 514 |
| SCOTT | 09-DEC-82 | 515 |
| MILLER | 23-JAN-82 | 526 |
| FORD | 03-DEC-81 | 527 |
| JAMES | 03-DEC-81 | 527 |
| KING | 17-NOV-81 | 528 |
| MARTIN | 28-SEP-81 | 530 |
| TURNER | 08-SEP-81 | 530 |
| CLARK | 09-JUN-81 | 533 |
| BLAKE | 01-MAY-81 | 535 |
| JONES | 02-APR-81 | 536 |
| WARD | 22-FEB-81 | 537 |
| ALLEN | 20-FEB-81 | 537 |
| SMITH | 17-DEC-80 | 539 |

## PROGRAM 4 : AGGREGATE FUNCTIONS & CLAUSES

**Aim :**

Write queries using aggregate functions (SUM, AVERAGE, COUNT, MINIMUM, MAXIMUM) & Clauses (WHERE,GROUP BY,HAVING ,ORDER BY)

**Program :**

a) Display the Number of rows in a table

```
SQL> SELECT COUNT(*) FROM EMP;

  COUNT(*)
----------
        14
```

b) Write a query to find the minimum,maximum,average & total salary?

```
SQL> SELECT MIN(SAL),MAX(SAL),AVG(SAL),SUM(SAL) FROM EMP;

  MIN(SAL)   MAX(SAL)   AVG(SAL)   SUM(SAL)
---------- ---------- ---------- ----------
       800       5000 2073.21429      29025
```

c) Display the maximum salary being paid to CLERK

```
SQL> SELECT MAX(SAL) FROM EMP WHERE JOB='CLERK';

  MAX(SAL)
----------
      1300
```

d) Display the total salary drawn by analyst working in deptno 20

```
SQL> SELECT SUM(SAL) FROM EMP WHERE DEPTNO=20 AND JOB='ANALYST';

  SUM(SAL)
----------
      6000
```

e) Display Department numbers and total number of employees working in each Department?

```
SQL> SELECT DEPTNO,COUNT(*) FROM EMP
  2  GROUP BY DEPTNO;
```

```
    DEPTNO   COUNT(*)
---------- ----------
        30          6
        20          5
        10          3
```

f) Display the various jobs and total number of employees working in each job group?

```
SQL> SELECT JOB,COUNT(*) FROM EMP
        GROUP BY JOB;

JOB              COUNT(*)
--------- -      ---------
CLERK                 4
SALESMAN              4
PRESIDENT             1
MANAGER               3
ANALYST               2
```

g) Display each job along with min of salary being paid in each job group?

```
SQL> SELECT JOB,MIN(SAL), FROM EMP
                GROUP BY JOB;

JOB              MIN(SAL)
---------        ----------
CLERK            800
SALESMAN         1250
PRESIDENT        5000
MANAGER          2450
 ANALYST          3000
```

h) Display the minimum,maximum,sum and the average salary for each job type.

```
SQL> SELECT JOB,MIN(SAL),MAX(SAL),AVG(SAL),SUM(SAL) FROM EMP
  GROUP BY JOB;

JOB       MIN(SAL)   MAX(SAL)   AVG(SAL)   SUM(SAL)
--------- ---------- ---------- ---------- ----------
CLERK          800       1300     1037.5       4150
SALESMAN       1250       1600       1400       5600
PRESIDENT      5000       5000       5000       5000
MANAGER        2450       2975 2758.33333       8275
ANALYST        3000       3000       3000       6000
```

g) Display the deptno,job and number of employees working in each deptno & in each job?

```
SQL> SELECT DEPTNO,JOB,COUNT(*) FROM EMP
  2  GROUP BY DEPTNO,JOB;

    DEPTNO     JOB              COUNT(*)
----------     ---------         ----------
        20          CLERK              2
        30          SALESMAN           4
        20          MANAGER            1
        30          CLERK              1
        10          PRESIDENT          1
        30          MANAGER            1
        10          CLERK              1
        10          MANAGER            1
        20          ANALYST            2
```

i) Display the deptno,job and number of employees working each job in deptno 20 ?

```
SQL> SELECT DEPTNO,JOB,COUNT(*) FROM EMP
  2  WHERE DEPTNO=20
  3  GROUP BY DEPTNO,JOB;

    DEPTNO JOB          COUNT(*)
----------     ---------         ----------
        20 CLERK            2
        20 MANAGER                1
        20 ANALYST                2
```

j) Display the deptno,job and number of employees working each job in deptno 20 & restrict the count more than 1 ?

```
SQL> SELECT DEPTNO,JOB,COUNT(*) FROM EMP
                WHERE DEPTNO=20
                GROUP BY DEPTNO,JOB
                HAVING COUNT(*)>1
                ORDER BY JOB;

    DEPTNO JOB          COUNT(*)
---------- ---------          ----------
        20 ANALYST                2
        20 CLERK                  2
```

## PROGRAM 5 : BUS RESERVATION SYSTEM APPLICATION WITH REQUIRED CONSTRAINTS

**Aim :**

Create the tables for bus reservation system application with required constraints**.**

BUS (**ROUTENO**, SOURCE, DESTINATION)

PASSENGER (**PID**, PNAME, DOB, GENDER)

BOOK_TICKET (**PID, ROUTENO, JOURNEY_DATE**, SEAT_NO)

The primary keys are underlined. Identify the foreign keys.

**Program :**

**a) Implement the above schema enforcing primary key and foreign key constraints.**

```
SQL> CREATE TABLE BUS (ROUTENO NUMBER(4) PRIMARY KEY,
  2  SOURCE VARCHAR(15),DESTINATION VARCHAR(15));
Table created.

SQL> DESC BUS
 Name                                      Null?             Type
 ----------------------------------------- -------- -        ----------------------------
 ROUTENO                                   NOT NULL          NUMBER(4)
 SOURCE                                                      VARCHAR2(15)
 DESTINATION                                                 VARCHAR2(15)

SQL> CREATE TABLE PASSENGER (PID  NUMBER(4) PRIMARY KEY,
  2  PNAME VARCHAR(15),DOB DATE,GENDER VARCHAR(6));
```

Table created.

SQL> DESC PASSENGER

| Name | Null? | Type |
|---|---|---|
| PID | NOT NULL | NUMBER(4) |
| PNAME | | VARCHAR2(15) |
| DOB | | DATE |
| GENDER | | VARCHAR2(6) |

SQL> CREATE TABLE BOOK_TICKET(PID NUMBER(4) REFERENCES PASSENGER(PID),
2 ROUTENO NUMBER(4) REFERENCES BUS(ROUTENO),
3 JOURNEY_DATE DATE,SEAT_NO VARCHAR(5),
4 PRIMARY KEY(PID,ROUTENO,JOURNEY_DATE));

Table created.

SQL> DESC BOOK_TICKET

| Name | Null? | Type |
|---|---|---|
| PID | NOT NULL | NUMBER(4) |
| ROUTENO | NOT NULL | NUMBER(4) |
| JOURNEY_DATE | NOT NULL | DATE |
| SEAT_NO | | VARCHAR2(5) |

**b) inserting records in the above relations.**

SQL> insert into bus values(&ROUTENO,'&SOURCE','&DESTINATION');
Enter value for routeno: 10

```
Enter value for source: COIMBATORE
Enter value for destination: SALEM
old   1: insert into bus values(&ROUTENO,'&SOURCE','&DESTINATION')
new   1: insert into bus values(10,'COIMBATORE','SALEM')

1 row created.


SQL> /
Enter value for routeno: 11
Enter value for source: COIMBATORE
Enter value for destination: METTUPALAYAM
old   1: insert into bus values(&ROUTENO,'&SOURCE','&DESTINATION')
new   1: insert into bus values(11,'COIMBATORE','METTUPALAYAM')

1 row created.

SQL> /
Enter value for routeno: 12
Enter value for source: COIMBATORE
Enter value for destination: ERODE
old   1: insert into bus values(&ROUTENO,'&SOURCE','&DESTINATION')
new   1: insert into bus values(12,'COIMBATORE','ERODE')

1 row created.



SQL> COMMIT;

Commit complete.

SQL> insert into PASSENGER VALUES(&PID,'&PNAME','&DOB','&GENDER');
Enter value for pid: 101
Enter value for pname: KUMAR
Enter value for dob: 12-SEP-78
Enter value for gender: M
old   1: insert into PASSENGER VALUES(&PID,'&PNAME','&DOB','&GENDER')
new   1: insert into PASSENGER VALUES(101,'KUMAR','12-SEP-78','M')

1 row created.

SQL> /
Enter value for pid: 102
Enter value for pname: JAYA
```

```
Enter value for dob: 12-DEC-67
Enter value for gender: F
old   1: insert into PASSENGER VALUES(&PID,'&PNAME','&DOB','&GENDER')
new   1: insert into PASSENGER VALUES(102,'JAYA','12-DEC-67','F')

1 row created.

SQL> /
Enter value for pid: 103
Enter value for pname: PRAVEEN
Enter value for dob: 20-MAR-01
Enter value for gender: M
old   1: insert into PASSENGER VALUES(&PID,'&PNAME','&DOB','&GENDER')
new   1: insert into PASSENGER VALUES(103,'PRAVEEN','20-MAR-01','M')

1 row created.

SQL> COMMIT;

Commit complete.

SQL>  insert into BOOK_TICKET
VALUES(&PID,&ROUTENO,'&JOURNEY_DATE','&SEATNO');
Enter value for pid: 101
Enter value for routeno: 10
Enter value for journey_date: 12-APR-12
Enter value for seatno: S1
old   1:  insert into BOOK_TICKET
VALUES(&PID,&ROUTENO,'&JOURNEY_DATE','&SEATNO')
new   1:  insert into BOOK_TICKET VALUES(101,10,'12-APR-12','S1')

1 row created.

SQL> /
Enter value for pid: 102
Enter value for routeno: 11
Enter value for journey_date: 13-SEP-13
Enter value for seatno: S10
old   1:  insert into BOOK_TICKET
VALUES(&PID,&ROUTENO,'&JOURNEY_DATE','&SEATNO')
new   1:  insert into BOOK_TICKET VALUES(102,11,'13-SEP-13','S10')

1 row created.
```

```
SQL> /
Enter value for pid: 102
Enter value for routeno: 12
Enter value for journey_date: 30-JUN-13
Enter value for seatno: S12
old   1:  insert into BOOK_TICKET
VALUES(&PID,&ROUTENO,'&JOURNEY_DATE','&SEATNO')
new   1:  insert into BOOK_TICKET VALUES(102,12,'30-JUN-13','S12')

1 row created.

SQL> /
Enter value for pid: 102
Enter value for routeno: 12
Enter value for journey_date: 31-JUL-13
Enter value for seatno: S13
old   1:  insert into BOOK_TICKET
VALUES(&PID,&ROUTENO,'&JOURNEY_DATE','&SEATNO')
new   1:  insert into BOOK_TICKET VALUES(102,12,'31-JUL-13','S13')

1 row created.

SQL> /
Enter value for pid: 102
Enter value for routeno: 12
Enter value for journey_date: 31-AUG-13
Enter value for seatno: S14
old   1:  insert into BOOK_TICKET
VALUES(&PID,&ROUTENO,'&JOURNEY_DATE','&SEATNO')
new   1:  insert into BOOK_TICKET VALUES(102,12,'31-AUG-13','S14')

1 row created.

SQL> COMMIT;


Commit complete.

SQL> SELECT * FROM BUS;
  ROUTENO SOURCE         DESTINATION
---------- --------------- ---------------
        10 COIMBATORE      SALEM
        11 COIMBATORE      METTUPALAYAM
        12 COIMBATORE      ERODE
```

```
SQL> SELECT * FROM PASSENGER;

       PID  PNAME          DOB           GENDER
----------  ---------------      ---------                ------
       101  KUMAR       12-SEP-78        M
       102  JAYA          12-DEC-67       F
       103  PRAVEEN     20-MAR-01      M

SQL>  SELECT * FROM BOOK_TICKET;

       PID    ROUTENO JOURNEY_DATE  SEAT_NO
----------  ----------         ---------         -----
       101         10           12-APR-12     S1
       102         11           13-SEP-13     S10
       102         12           30-JUN-13     S12
       102         12           31-JUL-13     S13
       102         12           31-AUG-13     S14
```

**c) Confirm with data Dictionary**

```
SQL> SELECT CONSTRAINT_NAME,CONSTRAINT_TYPE FROM USER_CONSTRAINTS
            WHERE TABLE_NAME='BUS';

CONSTRAINT_NAME              CONSTRAINT_TYPE
------------------------------ -                 --------------------------
SYS_C005553                    P

SQL> SELECT CONSTRAINT_NAME,CONSTRAINT_TYPE FROM USER_CONSTRAINTS
            WHERE TABLE_NAME='PASSENGER';

CONSTRAINT_NAME              CONSTRAINT_TYPE
------------------------------                 --------------------------
    SYS_C005554                P

SQL> SELECT CONSTRAINT_NAME,CONSTRAINT_TYPE,R_CONSTRAINT_NAME
       FROM USER_CONSTRAINTS
                          WHERE TABLE_NAME='BOOK_TICKET';
CONSTRAINT_NAME              CONSTRAINT_TYPE              R_CONSTRAINT_NAME
------------------------------                 --------------------         - ------------------------------
SYS_C005555                          P
SYS_C005556                          R                        SYS_C005554
SYS_C005557                          R                        SYS_C005553
```

## PROGRAM 6 – IMPLEMENTATION OF JOINS

**Aim :**

Create Sailors, Boats & Reserves tables and implement Simple Join, Self Join, Outer Join, Inner Join, Left and Right Join.

**Program :**

```
SQL> CREATE TABLE Sailors (
       sid     NUMBER       PRIMARY KEY,
       sname   VARCHAR2(14) NOT NULL,
      rating  NUMBER,
      age     NUMBER   );
```

Table created.

```
SQL> CREATE TABLE Boats (
       bid     NUMBER       PRIMARY KEY,
       bname   VARCHAR2(15) NOT NULL,
     color   VARCHAR2(10)    );
```

Table created.

```
SQL> 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)  );
```

Table created.

```
SQL> --INSERT Records into Boats
SQL> INSERT INTO Boats (bid, bname, color) VALUES (101, 'Ocean Queen', 'Red');
```

1 row created.

```
SQL> INSERT INTO Boats (bid, bname, color) VALUES (102, 'Sea Rider', 'Blue');
```

1 row created.

```
SQL> INSERT INTO Boats (bid, bname, color) VALUES (103, 'Wave Master', 'Green');
```

```
1 row created.

SQL> INSERT INTO Boats (bid, bname, color) VALUES (104, 'Sunset Cruiser', 'Yellow');

1 row created.

SQL> INSERT INTO Boats (bid, bname, color) VALUES (105, 'Storm Runner', 'Black');

1 row created.

SQL>
SQL> --INSERT Records into Reserves
SQL> INSERT INTO Reserves (sid, bid, day)
  2  VALUES (1, 101, TO_DATE('2025-01-10', 'YYYY-MM-DD'));

1 row created.

SQL>
SQL> INSERT INTO Reserves (sid, bid, day)
  2  VALUES (1, 102, TO_DATE('2025-01-12', 'YYYY-MM-DD'));

1 row created.

SQL>
SQL> INSERT INTO Reserves (sid, bid, day)
  2  VALUES (2, 103, TO_DATE('2025-01-11', 'YYYY-MM-DD'));

1 row created.

SQL>
SQL> INSERT INTO Reserves (sid, bid, day)
  2  VALUES (3, 101, TO_DATE('2025-01-15', 'YYYY-MM-DD'));

1 row created.

SQL>
SQL> INSERT INTO Reserves (sid, bid, day)
  2  VALUES (4, 104, TO_DATE('2025-01-16', 'YYYY-MM-DD'));

1 row created.

SQL> commit;

Commit complete.
```

SQL> select * from SAILORS;

| SID | SNAME | RATING | AGE |
|---|---|---|---|
| 1 | John | 7 | 25 |
| 2 | Mary | 5 | 30 |
| 3 | Alex | 8 | 22 |
| 4 | Ravi | 6 | 28 |
| 5 | Sita | 9 | 27 |

SQL> select * from Boats;

| BID | BNAME | COLOR |
|---|---|---|
| 101 | Ocean Queen | Red |
| 102 | Sea Rider | Blue |
| 103 | Wave Master | Green |
| 104 | Sunset Cruiser | Yellow |
| 105 | Storm Runner | Black |

SQL> select * from RESERVES;

| SID | BID | DAY |
|---|---|---|
| 1 | 101 | 10-JAN-25 |
| 1 | 102 | 12-JAN-25 |
| 2 | 103 | 11-JAN-25 |
| 3 | 101 | 15-JAN-25 |
| 4 | 104 | 16-JAN-25 |

**INNER JOIN :** Shows only rows that match in all tables.

```
SQL> SELECT S.sid, S.sname, B.bname, R.day
  2  FROM Sailors S
  3  INNER JOIN Reserves R ON S.sid = R.sid
  4  INNER JOIN Boats B ON B.bid = R.bid;
```

| SID | SNAME | BNAME | DAY |
|---|---|---|---|
| 3 | Alex | Ocean Queen | 15-JAN-25 |
| 1 | John | Ocean Queen | 10-JAN-25 |
| 1 | John | Sea Rider | 12-JAN-25 |
| 2 | Mary | Wave Master | 11-JAN-25 |

```
         4 Ravi          Sunset Cruiser  16-JAN-25
```

**LEFT OUTER JOIN :** Shows all sailors, even those who never reserved a boat.

```
SQL> SELECT S.sid, S.sname, B.bname, R.day
  2  FROM Sailors S
  3  LEFT JOIN Reserves R ON S.sid = R.sid
  4  LEFT JOIN Boats B ON B.bid = R.bid;

     SID SNAME          BNAME          DAY
---------- -------------- -------------- ---------
         3 Alex           Ocean Queen     15-JAN-25
         1 John           Ocean Queen     10-JAN-25
         1 John           Sea Rider       12-JAN-25
         2 Mary           Wave Master     11-JAN-25
         4 Ravi           Sunset Cruiser  16-JAN-25
         5 Sita
```

**RIGHT OUTER JOIN :** Shows all boats even if no sailor reserved them.

```
SQL> SELECT S.sname, B.bname, R.day
  2  FROM Sailors S
  3  RIGHT JOIN Reserves R ON S.sid = R.sid
  4  RIGHT JOIN Boats B ON B.bid = R.bid;

SNAME          BNAME          DAY
-------------- -------------- ---------
John           Sea Rider       12-JAN-25
John           Ocean Queen     10-JAN-25
Mary           Wave Master     11-JAN-25
Alex           Ocean Queen     15-JAN-25
Ravi           Sunset Cruiser  16-JAN-25
               Storm Runner
```

**FULL OUTER JOIN :** Returns all sailors and all boats, with matching and non-matching rows.

```
SQL> SELECT S.sid, S.sname, B.bid, B.bname, R.day
  2  FROM Sailors S
  3  FULL OUTER JOIN Reserves R ON S.sid = R.sid
  4  FULL OUTER JOIN Boats B ON B.bid = R.bid;

     SID SNAME                BID BNAME          DAY
---------- -------------- ---------- -------------- ---------
```

| | | | |
|---|---|---|---|
| 3 | Alex | 101 Ocean Queen | 15-JAN-25 |
| 1 | John | 101 Ocean Queen | 10-JAN-25 |
| 1 | John | 102 Sea Rider | 12-JAN-25 |
| 2 | Mary | 103 Wave Master | 11-JAN-25 |
| 4 | Ravi | 104 Sunset Cruiser | 16-JAN-25 |
| 5 | Sita | | |
| | | 105 Storm Runner | |

**NATURAL JOIN :**

Oracle will join using **columns with identical names** automatically

```
SQL> SELECT sid, S.sname, R.day
  2  FROM Sailors S
  3  NATURAL JOIN Reserves R;
```

| SID | SNAME | DAY |
|---|---|---|
| 1 | John | 10-JAN-25 |
| 1 | John | 12-JAN-25 |
| 2 | Mary | 11-JAN-25 |
| 3 | Alex | 15-JAN-25 |
| 4 | Ravi | 16-JAN-25 |

## PROGRAM 7-LIBRARY MANAGEMENT

**Aim :**

Create tables for library management system .
Master table should have the following fields: ACCNO, TITLE, AUTHOR AND RATE.
Transaction table should have the following fields: USER ID, ACCNO, DATE OF ISSUE AND DATE OF RETURN. Create a report with fields Accno, Title, Date of Issue for the given Date of Return with column formats.

**PROGRAM**

**1. CREATE A LIBRARY MASTER TABLE:**

```
SQL> create table library_master(accno number primary key,title varchar2(25),author varchar2(25),
        rate number);
Table created.
```

**2. CREATE A LIBRARY TRANSACTION TABLE:**

```
SQL> create table library_transaction(userid number,accno number references library_master(accno),
date_of_issue date, date_of_return date);
Table created.
```

**3.INSERT VALUES INTO LIBRARY_MASTER TABLE:**

```
SQL> insert into library_master values(&accno,'&title','&author',&rate);
Enter value for accno: 1001
Enter value for title: c programming
Enter value for author:  balaguruswamy
Enter value for rate: 300
old   1: insert into library_master values(&accno,'&title','&author',&rate)
new   1: insert into library_master values(1001,'c programming',' balaguruswamy',300)
1 row created.
SQL> /
Enter value for accno: 1002
Enter value for title: c++ programming
Enter value for author: balaguru
Enter value for rate: 350
old   1: insert into library_master values(&accno,'&title','&author',&rate)
new   1: insert into library_master values(1002,'c++ programming','balaguru',350)
1 row created.
SQL> /
Enter value for accno: 1004
Enter value for title: data structure
Enter value for author: michael
Enter value for rate: 400
old   1: insert into library_master values(&accno,'&title','&author',&rate)
new   1: insert into library_master values(1004,'data structure','michael',400)
```

```
1 row created.
SQL> /
Enter value for accno: 1005
Enter value for title: oracle & RDBMS
Enter value for author: john peter
Enter value for rate: 320
old   1: insert into library_master values(&accno,'&title','&author',&rate)
new   1: insert into library_master values(1005,'oracle &RDBMS','john peter',320)
1 row created.
```

**4.SELECT ALL ROWS FROM LIBRARY_MASTER:**

```
SQL> select * from library_master;
```

| ACCNO | TITLE | AUTHOR | RATE |
|---|---|---|---|
| 1001 | c programming | balaguruswamy | 300 |
| 1002 | c++ programming | balaguru | 350 |
| 1004 | data structure | michael | 400 |
| 1005 | oracle & RDBMS | john peter | 320 |

**5. INSERT VALUES INTO LIBRARY_TRANSACTION TABLE:**

```
SQL> insert into library_transaction values(&userid,&accno,'&date_of_issue','&date_of_return');
Enter value for userid: 01
Enter value for accno: 1002
Enter value for date_of_issue: 12-jun-16
Enter value for date_of_return: 13-jul-16
old   1: insert into library_transaction values(&userid,&accno,'&date_of_issue','&date_of_return')
new   1: insert into library_transaction values(01,1002,'12-jun-16','13-jul-16')
1 row created.
SQL> /
Enter value for userid: 02
Enter value for accno: 1005
Enter value for date_of_issue: 25-jul-16
Enter value for date_of_return: 24-aug-16
old   1: insert into library_transaction values(&userid,&accno,'&date_of_issue','&date_of_return')
new   1: insert into library_transaction values(02,1005,'25-jul-16','24-aug-16')
1 row created.

SQL> /
Enter value for userid: 03
Enter value for accno: 1004
Enter value for date_of_issue: 01-jun-16
Enter value for date_of_return: 30-jun-16
old   1: insert into library_transaction values(&userid,&accno,'&date_of_issue','&date_of_return')
new   1: insert into library_transaction values(03 ,1004,'01-jun-16','30-jun-16')
1 row created.
```

**6. SELECT ALL ROWS FROM LIBRARY_TRANSACTION:**

```
SQL> select * from library_transaction;
USERID  ACCNO DATE_OF_I DATE_OF_R
---------- ---------- -------------- ----------------
         1       1002    12-JUN-16      13-JUL-16
2        1005    25-JUL-16       24-AUG-16
3          1004    01-JUN-16      30-JUN-16
```

**7. FORMATING THE LIBRARY MANAGEMENT SYSTEM**

```
SQL> set linesize 75
SQL>Ttitle center “LIBRARY MANAGEMENT' skip 1-
>center **********
SQL>  column accno heading 'ACCOUNT NUMBER'
SQL>  column title heading 'BOOK_NAME'
SQL>  column date_of_issue heading 'DATE_OF_ISSUE'
SQL>  column date_of_retrun heading 'DATE_OF_RETURN'
SQL>  set  underline-
SQL>  break on row skip1
SQL>  select m.accno,m.title,t.date_of_issue from library_masterm,library_transaction t
        where m.accno=t.accno and date_of_return='24-aug-16';

                  LIBRARY MANAGEMENT
                     **********
ACCOUNT NUMBER BOOK_NAME              DATE_OF_ISSUE
---------------------- ------------------------   --------------------
               1005   oracle & RDBMS           25-JUL-16
```

## PROGRAM 8 -INVENTORY TABLE

**Aim :**

Write a PL/SQL to update the rate field by 20% more than the current rate in inventory table which has the following fields: Prono, ProName and Rate. After updating the table a new field (Alter) called for Number of item and place for values for the new field without using PL/SQL block.

**1.CREATE A TABLE AS INVENTORY:**

```
SQL> create table inventory(prono number primary key,
2  pronamevarchar(20),
3  rate number);

Table created.
```

**2.INSERT VALUES INTO INVENTORY TABLE:**

```
SQL> insert into inventory values(&prono,'&proname',&rate);
Enter value for prono: 1001
Enter value for proname: soap
Enter value for rate: 30
old   1: insert into inventory values(&prono,'&proname',&rate)
new   1: insert into inventory values(1001,'soap',30)

1 row created.
SQL> /
Enter value for prono: 1002
Enter value for proname: horlicks
16

Enter value for rate: 100
old   1: insert into inventory values(&prono,'&proname',&rate)
new   1: insert into inventory values(1002,'horlicks',100)

1 row created.
SQL> /
Enter value for prono: 1003
Enter value for proname: milkybar
Enter value for rate: 50
old   1: insert into inventory values(&prono,'&proname',&rate)
new   1: insert into inventory values(1003,'milkybar',50)

1 row created.
SQL> /
Enter value for prono: 1004
```

```
Enter value for proname: chocobar
Enter value for rate: 40
old   1: insert into inventory values(&prono,'&proname',&rate)
new   1: insert into inventory values(1004,'chocobar',40)

1 row created.
```

**3.SELECT ALL THE ROWS IN INVENTORY TABLE:**

```
SQL> select * from inventory;

    PRONO PRONAME       RATE
---------- -------------------- ----------
      1001 soap                  30
      1002 horlicks             100
      1003 milkybar              50
      1004 chocobar              40
```

**4.UPDATE THE COLUMN AS RATE:**

```
SQL>   begin
        update inventory set rate=rate+(rate*20/100);
       commit;
       end;
       /

PL/SQL procedure successfully completed.
```

**5. SELECT ALL THE ROWS IN INVENTORY TABLE:**

```
SQL> select * from inventory;
    PRONO PRONAME       RATE
---------- -------------------- ----------
      1001 soap                  36
      1002 horlicks             120
      1003 milkybar              60
      1004 chocobar              48
```

**6.ADDING ANEW COLUMN AS NO_OF_ITEMS IN INVENTORY TABLE:**

```
SQL> alter table inventory add no_of_items number;

Table altered.
```

**7.DESCRIBEA INVENTORY TABLE:**

```
SQL>desc inventory;
 Name                      Null?    Type
```

| | | |
|---|---|---|
| PRONO | NOT | NULL NUMBER |
| PRONAME | | VARCHAR2(20) |
| RATE | | NUMBER |
| NO_OF_ITEMS | | NUMBER |

**8.UPDATE AN EXISTING COLUMNS IN INVENTORY TABLE:**

```
SQL> update inventory set no_of_items=case prono
  when 1001 then 10
  when 1002 then 8
  when 1003 then 6
  when 1004 then 4
 end;

4 rows updated.
```

**9. SELECT ALL THE ROWS IN INVENTORY TABLE:**

```
SQL> select * from inventory;
```

| PRONO | PRONAME | RATE | NO_OF_ITEMS |
|---|---|---|---|
| 1001 | soap | 36 | 10 |
| 1002 | horlicks | 120 | 8 |
| 1003 | milkybar | 60 | 6 |
| 1004 | chocobar | 48 | 4 |

## PROGRAM 9 : FACTORIAL & SUM OF THE DIGITS

**Aim :**

a) Create a PL/SQL program to find the factorial of a given number.
b) Create a program to accept a number and find the sum of the digits

**Program :**

**a) Factorial of a given number.**

```
SQL> declare
     i number(4):=1;
     n number(4):=&n;
     f number(4):=1;
     begin
     for i in 1..n loop
     f:=f*i;
     end loop;
     Dbms_output.put_line('The Factorial of '||n||' is:'||f);
     end;
     /

SQL> /
     Enter value for n: 5
     old   3: n number(4):=&n;
     new   3: n number(4):=5;
     The Factorial of 5 is:120

     PL/SQL procedure successfully completed.
```

**b) Write a program to accept a number and find the sum of the digits**

```
SQL> Declare
        n number:=&n;
        s number:=0;
        r number:=0;
        begin
        while n !=0
        loop
        r:=mod(n,10);
        s:=s+r;
        n:=trunc(n/10);
        end loop;
        dbms_output.put_line('Sum of digits of given number is '||s);
        end;

        SQL> /
        Enter value for n: 1234
        old   2: n number:=&n;
        new   2: n number:=1234;
        Sum of digits of given number is 10

        PL/SQL procedure successfully completed.
```

## PROGRAM 10 : PL/SQL PROGRAM FOR REVERSE STRING & FIBONACCI SERIES

**Aim :**

a) Create a PL/SQL program to find the Reverse String
b) Create a PL/SQL program to find the Fibonacci Series

**Program :**

**a) PL/SQL program to find the Reverse String**

```
SQL > DECLARE
        string1  VARCHAR2(100) := '&String1';
        string2  VARCHAR2(100) := '';
        i  number;
      BEGIN
        FOR i IN REVERSE 1 .. LENGTH(string1) LOOP
          string2 := string2 || SUBSTR(string1, i, 1);
        END LOOP;
        DBMS_OUTPUT.PUT_LINE('String Input: ' || string1);
        DBMS_OUTPUT.PUT_LINE('String Output: ' || string2);
      END;

SQL> /
      Enter value for string1: KGCAS
      old   2:    string1  VARCHAR2(100) := '&String1';
      new   2:    string1  VARCHAR2(100) := 'KGCAS';
      String Input: KGCAS
      String Output: SACGK

      PL/SQL procedure successfully completed.
```

**b) PL/SQL program to find the Fibonacci Series**

```
SQL > DECLARE
        n NUMBER := 10;
        a NUMBER := 0;
        b NUMBER := 1;
        c NUMBER;
      BEGIN
        DBMS_OUTPUT.PUT_LINE('Fibonacci Series (First ' || n || ' Terms):');
        DBMS_OUTPUT.PUT_LINE(a);
        DBMS_OUTPUT.PUT_LINE(b);
        FOR i IN 3 .. n LOOP
          c := a + b;
          DBMS_OUTPUT.PUT_LINE(c);
          a := b;
          b := c;
        END LOOP;
      END;
SQL> /
```

```
Fibonacci Series (First 10 Terms):
0
1
1
2
3
5
8
13
21
34
```

## PROGRAM 11: SPLIT STUDENT TABLE USING CURSORS IN PL/SQL

**AIM:**

Write a PL/SQL to split the student table into two tables based on result (One table for ―Pass and another for ―Fail). Use cursor for handling records of student table. Assume necessary fields and create a student details table.

**Program :**

**i) CREATE A STUDENT TABLE:**

**SQL**>create table student(rollno number,name varchar2(20),totalmarks number,result varchar2(4));

Table created.

**ii) INSERT RECORDS TO STUDENT TABLE:**

**SQL**>insert into Student values(&rollno,'&name',&total,'&result');

Enter value for rollno: 1001
Enter value for name: ARAVIND
Enter value for total: 33
Enter value for result: FAIL
old 1: insert into Student values(&rollno,'&name,',&total,'&FAIL')
new 1: insert into Student values(1001,'ARAVIND',33, 'FAIL')

1 row created.

**iii) SELECT ALL ROWS FROM STUDENT TABLE:**

SQL>select * from Student;

| ROLL_NO | NAME | TOTAL | RESULT |
|---|---|---|---|
| 1001 | ARAVIND | 33 | FAIL |
| 1002 | ARUN | 95 | PASS |
| 1003 | SENTHIL | 25 | FAIL |
| 1004 | THARUN | 44 | PASS |
| 1005 | PRIYA | 98 | PASS |

**iv) SPLIT THE STUDENT TABLE INTO 2 TABLES BASED ON RESULT:**

```
SQL>declare
    query1 varchar2(100):='create table Student_pass as(select * from Student where result="pass")';
    query2 varchar2(100):='create table Student_fail as(select * from Student where result="fail")';
    begin
    execute immediate query1;
    execute immediate query2;
    end;
    /
    PL/SQL procedure successfully completed.
```

**v) USE CURSOR FOR HANDLING RECORDS**

```
    SQL>declare
    cursor cpass is select * from Student_pass;
    cursor cfail is select * from Student_fail;
    begin
    dbms_output.put_line('LIST OF PASSED STUDENTS');
    for passrec in cpass loop
    dbms_output.put_line(passrec.name);
    end loop;
    dbms_output.put_line('LIST OF FAILED STUDENTS');
    for failrec in cfail loop
    dbms_output.put_line(failrec.name);
    end loop;
    end;
    /
```

LIST OF PASSED STUDENTS

| ROLL_NO | NAME | TOTAL | RESULT |
|---|---|---|---|
| 1002 | ARUN | 95 | PASS |
| 1004 | THARUN | 44 | PASS |
| 1005 | PRIYA | 98 | PASS |

LIST OF FAILED STUDENTS

| ROLL_NO | NAME | TOTAL | RESULT |
|---|---|---|---|
| 1001 | ARAVIND | 33 | FAIL |
| 1003 | SENTHIL | 25 | FAIL |

## PROGRAM 12- EXCEPTION HANDLING FOR BANK ACCOUNT MANAGEMENT TABLE

**AIM:**

Write a PL/SQL to raise the following Exception in Bank Account Management table when deposit amount is zero.

**PROGRAM:**

**EXCEPTION HANDLING FOR BANK ACCOUNT MANAGEMENT TABLE**

**i) CREATE BANK ACCOUNT MANAGEMENT TABLE:-**

**SQL>**create table bank_acc(accno varchar(10) primary key,name varchar(20),deposit number);
Table created.

**ii) CREATE AN EXCEPTION FOR BANK ACCOUNT MANAGEMENT TABLE:-**

**SQL>**set serveroutput on

```
SQL > declare
        vaccno varchar(10):= '&vaccno';
        vname varchar2(20):= '&vname';
        vdeposit number:= &vdeposit;
        zerodeposit exception;
        begin
        if vdeposit<=0 then
        raise zerodeposit;
        else
        insert into bank_acc(accno,name,deposit) values (vaccno,vname,vdeposit);
        dbms_output.put_line('Record is successfully inserted');
        end if;
        exception
        when zerodeposit then
        dbms_output.put_line('Invalid deposit amount');
        end;
```

```
SQL> /
        ENTER VALUE FOR ACCNO: SAR001
        OLD   9:   A:='&ACCNO';
        NEW   9:   A:='SAR001';
        ENTER VALUE FOR DEPOSIT: 3500
        OLD  10:   D:='&DEPOSIT';
        NEW  10:   D:='3500';
```

Record is successfully inserted

PL/SQL PROCEDURE SUCCESSFULLY COMPLETED.

SQL> SELECT * FROM BANK_ACC;

| ACCNO | NAME | DEPOSIT |
|---|---|---|
| SAR001 | SURESH | 4000 |
| SAR002 | ANOOP | 80000 |
| SAR003 | SARAN | 90000 |

EXCEPTION HANDLING USING PL/SQL:

SQL> SELECT * FROM BANK_ACC;

| ACCNO | UNAME | ACCTYPE | BAL |
|---|---|---|---|
| SAR001 | SURESH | SAVINGS | 500 |
| SAR002 | ANOOP | CURRENT | 80000 |
| SAR003 | SARAN | SAVINGS | 90000 |

```
SQL > declare
        vaccno varchar(10):= '&vaccno';
        vname varchar2(20):= '&vname';
        vdeposit number:= &vdeposit;
        zerodeposit exception;
        begin
        if vdeposit<=0 then
```

```
        raise zerodeposit;
        else
        insert into bank_acc(accno,name,deposit) values (vaccno,vname,vdeposit);
        dbms_output.put_line('Record is successfully inserted');
        end if;
        exception
        when zerodeposit then
        dbms_output.put_line('Invalid deposit amount');
        end;
SQL> /
ENTER VALUE FOR ACCNO: SAR002
OLD   9:   A:='&ACCNO';
NEW   9:   A:='SAR002';
ENTER VALUE FOR DEPOSIT: 0
OLD  10:   D:='&DEPOSIT';
NEW  10:   D:='0';

Invalid deposit amount

PL/SQL PROCEDURE SUCCESSFULLY COMPLETED.
```

## PROGRAM 13 : PL/SQL PROGRAM FOR CALCULATING THE GRADE

**Aim :**

Write a PL/SQL program to find the total and average of 4 subjects and display the grade.

**PROGRAM :**

```
SQL >  declare
        java number(3);
        dbms number(3);
        co number(3);
        mfcs number(3);
        total number(4);
        per number(6,2);
        begin
        dbms_output.put_line('ENTER THE MARKS');
        java:=&java;
        dbms:=&dbms;
        co:=&co;
        mfcs:=&mfcsl;
        total:=(java+dbms+co+mfcs);
        per:=(total/400)*100;
        if (java<40 or dbms<40 or co<40 or mfcs<40) then
        dbms_output.put_line('FAIL');
        elsif per>75 then
        dbms_output.put_line('GRADE A');
        elsif(per>65 and per<75) then
        dbms_output.put_line('GRADE B');
        elsif(per>55 and per<65) then
        dbms_output.put_line('GRADE C');
        else
        dbms_output.put_line('INVALID INPUT');
        end if;
        dbms_output.put_line('PERCENTAGE IS '||per);
        end;
        /
```

```
SQL> /
Enter value for java: 100
old  10: java:=&java;
new  10: java:=100;
Enter value for dbms: 100
old  11: dbms:=&dbms;
new  11: dbms:=100;
Enter value for co: 100
old  12: co:=&co;
new  12: co:=100;
Enter value for mfcsl: 100
old  13: mfcs:=&mfcsl;
new  13: mfcs:=100;
ENTER THE MARKS
GRADE A
PERCENTAGE IS 100

PL/SQL procedure successfully completed.

SQL> /
Enter value for java: 90
old  10: java:=&java;
new  10: java:=90;
Enter value for dbms: 67
old  11: dbms:=&dbms;
new  11: dbms:=67;
Enter value for co: 89
old  12: co:=&co;
new  12: co:=89;
Enter value for mfcsl: 56
old  13: mfcs:=&mfcsl;
new  13: mfcs:=56;
ENTER THE MARKS
GRADE A
PERCENTAGE IS 75.5

PL/SQL procedure successfully completed.
```

```
SQL> /
Enter value for java: 56
old  10: java:=&java;
new  10: java:=56;
Enter value for dbms: 78
old  11: dbms:=&dbms;
new  11: dbms:=78;
Enter value for co: 45
old  12: co:=&co;
new  12: co:=45;
Enter value for mfcsl: 44
old  13: mfcs:=&mfcsl;
new  13: mfcs:=44;
ENTER THE MARKS
GRADE C
PERCENTAGE IS 55.75

PL/SQL procedure successfully completed.

SQL> /
Enter value for java: 8
old  10: java:=&java;
new  10: java:=8;
Enter value for dbms: 9
old  11: dbms:=&dbms;
new  11: dbms:=9;
Enter value for co: 78
old  12: co:=&co;
new  12: co:=78;
Enter value for mfcsl: 90
old  13: mfcs:=&mfcsl;
new  13: mfcs:=90;
ENTER THE MARKS
FAIL
PERCENTAGE IS 46.25

PL/SQL procedure successfully completed.
```

## PROGRAM 14 : STORED PROCEDURE

**Aim :**

Write a PL/SQL program to implement the concept “Procedures”.

**PROGRAM:**

**Step 1 : Creating table**

```
SQL> create table tbmarks
 (rno number(3),name varchar(15),mark1 number(3),mark2 number(3), total
 number(3),AVERAGE number(5,2),result varchar(10));
```

**Step 2 : Creating procedure for Inserting records in tbmarks table**

```
SQL>Create or replace procedure prmarksins (prno in number,pname in varchar,pmark1 in number,
    pmark2 in number)
    is
    vtotal number;
    vaverage number;
    vresult varchar(10);
    begin
    vtotal:=pmark1+pmark2
    ; vaverage:=vtotal/2;
    if(pmark1>=40 and pmark2>=40) then
    vresult:='Pass';
    else
     vresult:='Fail';
    end if;
    insert into tbmarks values(prno,pname,pmark1,pmark2,vtotal,vaverage,vresult);
    commit;
    end;
```

**Step 3: To Run the Procedure**

```
SQL> exec prmarksins(1,'KING',89,90);

 PL/SQL procedure successfully completed.

 SQL> exec prmarksins(2,'SCOTT',39,56); PL/SQL
```

procedure successfully completed.

SQL> select * from tbmarks;

| RNO | NAME | MARK1 | MARK2 | TOTAL | AVERAGE | RESULT |
|---|---|---|---|---|---|---|
| 1 | KING | 89 | 90 | 179 | 89.5 | Pass |
| 2 | SCOTT | 39 | 56 | 95 | 47.5 | Fail |

**Step 4 : Creating Procedure for Modifying records**

```
SQL>create or replace procedure prmarksupd

(prno in number, pmark1 in number,pmark2 in number)
is
vtotal number;
 vaverage number;
vresult varchar(10);
begin
vtotal:=pmark1+pmark2;
vaverage:=vtotal/2;
if(pmark1>=40 and pmark2>=40) then
vresult:='Pass';
else
vresult:='Fail';
end if;
update tbmarks set
mark1=pmark1,mark2=pmark2,total=vtotal,average=vaverage, result=vresult
where rno=prno;
commit;
end;
```

**Step 5 :**

SQL> select * from tbmarks;

| RNO | NAME | MARK1 | MARK2 | TOTAL | AVERAGE | RESULT |
|---|---|---|---|---|---|---|
| 1 | KING | 89 | 90 | 179 | 89.5 | Pass |
| 2 | SCOTT | 39 | 56 | 95 | 47.5 | Fail |

```
SQL> exec prmarksupd(2,55,56);

PL/SQL procedure successfully completed.

SQL> select * from tbmarks;

    RNO NAME          MARK1    MARK2    TOTAL   AVERAGE RESULT
------------------------------ ----------
      1 KING            89       90      179     89.5 Pass
      2 SCOTT           55       56      111     55.5 Pass
```

**Step 6: Creating Procedure for deleting the record in TBMARKS table**

```
SQL>

create or replace procedure prmarksdel(prno in number) is
begin
delete from tbmarks whererno=prno; commit;
end;


SQL> exec prmarksdel(2);

PL/SQL procedure successfully completed.

SQL> select * from tbmarks;

    RNO NAME          MARK1    MARK2    TOTAL          AVERAGE        RESULT

    1 KING            89          90      179                89.5        Pass
```

## PROGRAM : 15 INVENTORY MANAGEMENT SYSTEM USING DATABASE TRIGGERS

**AIM:**

Create a database trigger to implement on master and transaction tables which are based on inventory management system for checking data validity. Assume the necessary fields for both tables.

**PROGRAM:**

**i) CREATE INVENTORY MASTER TABLE:-**

**SQL>**create table inventory_master1(orderid number primary key,custid number,orderdate date,amount number);

Table created

**ii) CREATE INVENTORY TRANSACTION TABLE:-**

**SQL>**create table inventory_trans1(orderid number references inventory_master1,productid number,productname varchar2(30),quantity number,unitprice number);

Table created

**iii) CREATE TRIGGER ON INVENTORY MASTER:-**

**SQL>** set serveroutput on
SQL> create or replace trigger

check1 before insert on inventory_master1 for each row

declare

begin

if(:new.amount<0)then

raise_application_error(-20040,'invalid amount');

end if;

end;

Trigger created.

**iv) INSERT INTO INVENTORY TABLES:-**

**SQL>**insert into inventory_master1 values(&orderid,&custid,'&orderdate',&amount);

Enter value for orderid: 5001
Enter value for custid: 101
Enter value for orderdate: 12-JUL-20
Enter value for amount:2000
old 1: insert into inventory values(&orderid,&custid,'&orderdate',&amount)
new 1: insert into inventory values(5001,101,12-JUL-20,2000)

1 row inserted

**SQL>**insert into inventory_master1 values(&orderid,&custid,'&orderdate',&amount);

Enter value for orderid: 5002
Enter value for custid: 102
Enter value for orderdate: 13-JUL-20
Enter value for amount:0
old 1: insert into inventory values(&orderid,&custid,'&orderdate',&amount)
new 1: insert into inventory values(5002,102,13-JUL-20,0)
ORA-20202: INVALID AMOUNT
ORA-06512: AT "SURESH.CHECK1", LINE 16
ORA-04088: ERROR DURING EXECUTION OF TRIGGER 'SURESH.CHECK1'

**v) CREATE TRIGGER ON INVENTORY TRANSACTION:-**

**SQL>**create or replace trigger
check2 before insert on inventory_trans1 for each row
declare
begin
if(:new.quantity<0)then
raise_application_error(-20040,'invalid quantity');
end if;
end;

Trigger created.

**vi) INSERT INTO INVENTORY TRANSACTION:-**

SQL>insert into inventory_trans1 values (&orderid,&productid,'&productname',&quantity,&unitprice);

Enter value for orderid: 5001
Enter value for productid: 88

Enter value for productname: Television
Enter value for quantity:10
Ener value for unitprice: 32000
old 1: insert into inventory values(&orderid,&productid,'&productname',&quantity,&unitprice)
new 1: insert into inventory values(5001,88,Television,10,32000)

1 row inserted

SQL>insert into inventory_trans1 values (&orderid,&productid,'&productname',&quantity,&unitprice);

Enter value for orderid: 5002
Enter value for productid: 89
Enter value for productname: Airconditioner
Enter value for quantity:-10
Ener value for unitprice: 40000
old 1: insert into inventory values(&orderid,&productid,'&productname',&quantity,&unitprice)
new 1: insert into inventory values(5002,89,'Airconditioner',-10,40000)
ORA-20040: INVALID QUANTITY
ORA-06512: AT "SURESH.CHECK2", LINE 16
ORA-04088: ERROR DURING EXECUTION OF TRIGGER 'SURESH.CHECK2'