## EX.NO:1 CREATING DATABASE TABLE

**AIM:**

To create a DDL to perform creation of table, alter, modify and drop column.

**DDL COMMANDS**

1. The Create Table Command: - it defines each column of the table uniquely. Each column has minimum of three attributes, a name , data type and size.

**Syntax:**

Create table <table name> (<col1> <datatype>(<size>),<col2> <datatype><size>));

Ex:create table emp(empno number(4) primary key, ename char(10));

2. Modifying the structure of tables. a)

Add new columns

**Syntax:**

Alter table <tablename> add(<new col><datatype(size),<new col>datatype(size));

Ex:alter table emp add(sal number(7,2));

3. Dropping a column from a table.

**Syntax:**

Alter table <tablename> drop column <col>;

Ex:alter table emp drop column sal;

4. Modifying existing columns.

**Syntax:**

Alter table <tablename> modify(<col><newdatatype>(<newsize>));

Ex:alter table emp modify(ename varchar2(15));

5. Renaming the tables

**Syntax:**

Rename <oldtable> to <new table>;

Ex:rename emp to emp1;

6. truncating the tables.

**Syntax:**

Truncate table <tablename>;

Ex:trunc table emp1;

7. Destroying tables.

**Syntax:**

Drop table <tablename>;

Ex:drop table emp;

**CREATION OF TABLE:**

**SYNTAX:**

create table<tablename>(column1 datatype,column2 datatype...);

**EXAMPLE:**

SQL>create table std(sno number(5),sname varchar(20),age number(5),sdob date,sm1

number(4,2),sm2 number(4,2),sm3 number(4,4)); Table created.

SQL>insert into std values(101,‟AAA‟,16,‟03-jul-88‟,80,90,98); 1

row created.

```
SQL>insert into std values(102,‟BBB‟,18,‟04-aug-89‟,88,98,90); 1
row created.
```

**OUTPUT:**

```
Select * from std;
SNO   SNAME   AGE   SDOBSM1        SM2   SM3
101   AAA     16    03-jul-88   80    90    98
102   BBB     18    04-aug-89 88      98    90
```

**ALTER TABLE WITH ADD:**

```
SQL>create table student(id number(5),name varchar(10),game varchar(20));
Table created.
SQL>insert into student values(1,‟mercy‟,‟cricket‟); 1
row created.
```

**SYNTAX:**

```
alter table<tablename>add(col1 datatype,col2 datatype..);
```

EXAMPLE:

```
SQL>alter table student add(age number(4));
SQL>insert into student values(2,‟sharmi‟,‟tennis‟,19);
```

**OUTPUT:**

```
ALTER: select * from student;
ID NAME GAME
1   Mercy Cricket


ADD: select * from student;
ID NAME GAME AGE
```

```
1    Mercy cricket
2    Sharmi Tennis 19
```

**ALTER TABLE WITH MODIFY:**

**SYNTAX:**

Alter table<tablename>modify(col1 datatype,col2 datatype..);

**EXAMPLE:**

SQL>alter table student modify(id number(6),game varchar(25));

**OUTPUT:**

MODIFY

desc student;

```
NAME NULL? TYPE
Id       Number(6)
Name Varchar(20)
Game Varchar(25)
Age    Number(4)
```

**DROP:**

**SYNTAX:** drop table<tablename>;

EXAMPLE:

SQL>drop table student;

SQL>Table dropped.

**TRUNCATE TABLE**

**SYNTAX:** TRUNCATE TABLE <TABLE NAME>;

Example: Truncate table stud;

**DESC**

Example: desc emp;

```
Name Null? Type
 -------------------------------- --------
EmpNo NOT NULL number(5)
EName VarChar(15)
Job NOT NULL Char(10)
DeptNo NOT NULL number(3)
PHONE_NO number (10)
```

**Queries:**

Q1. Create a table called EMP with the following structure.

```
Name Type
 ---------- ----------------------
EMPNO NUMBER(6)
ENAME VARCHAR2(20)
JOB VARCHAR2(10)
DEPTNO NUMBER(3)
SAL NUMBER(7,2)
```

Allow NULL for all columns except ename and job.

**Solution:**

1. Understand create table syntax.
2. Use the create table syntax to create the said tables.
3. Create primary key constraint for each table as understand from logical table structure.

Ans:

```
 SQL> create table emp(empno number(6),ename varchar2(20)not null,job varchar2(10) not
null, deptno number(3),sal number(7,2));
Table created.
```

Q2: Add a column experience to the emp table.

experience numeric null allowed.
**Solution:**
1. Learn alter table syntax.
2. Define the new column and its data type.
3. Use the alter table syntax.

Ans: SQL> alter table emp add(experience number(2));
Table altered.

Q3: Modify the column width of the job field of emp table.
**Solution:**
1. Use the alter table syntax.
2. Modify the column width and its data type.

Ans: SQL> alter table emp modify(job varchar2(12));
Table altered.

SQL> alter table emp modify(job varchar(13));
Table altered.

Q4: Create dept table with the following structure.

```
Name Type
------------ ---------------------
DEPTNO NUMBER(2)
DNAME VARCHAR2(10)
LOC VARCHAR2(10)
```

Deptno as the primarykey
**Solution:**
1. Understand create table syntax.
2. Decide the name of the table.
3. Decide the name of each column and its data type.
4. Use the create table syntax to create the said tables.
5. Create primary key constraint for each table as understand from logical table structure.

Ans:
SQL> create table dept(deptno number(2) primary key,dname varchar2(10),loc varchar2(10));
Table created.

Q5: create the emp1 table with ename and empno, add constraints to check the empno value while entering (i.e) empno > 100.
**Solution:**
1. Learn alter table syntax.

2. Define the new constraint [columns name type]
3. Use the alter table syntax for adding constraints.
Ans:
SQL> create table emp1(ename varchar2(10),empno number(6) constraint check(empno>100));
Table created.

Q6: drop a column experience to the emp table.
**Solution:**
1. Learn alter table syntax. Use the alter table syntax to drop the column.

Ans:
SQL> alter table emp drop column experience; Table altered.

Q7: Truncate the emp table and drop the dept table

**Solution:**

1. Learn drop, truncate table syntax.

Ans: SQL> truncate table emp; Table truncated.

**RESULT:**

Thus the DDL commands have been executed successfully.

## EX.NO:2 WORKING WITH DATA MANIPULATION COMMANDS

**AIM**

To study the various DML commands and implement them on the database.

**DML COMMANDS**

DML commands are the most frequently used SQL commands and is used to query and manipulate the existing database objects. Some of the commands are Insert, Select, Update, Delete.

Insert Command This is used to add one or more rows to a table. The values are separated by commas and the data types char and date are enclosed in apostrophes. The values must be entered in the same order as they are defined.

Select Commands It is used to retrieve information from the table. It is generally referred to as querying the table. We can either display all columns in a table or only specify column from the table.

Update Command It is used to alter the column values in a table. A single column may be updated or more than one column could be updated.

Delete command After inserting row in a table we can also delete them if required. The delete command consists of a from clause followed by an optional where clause.

Q1: Insert a single record into dept table.

```
Ans: SQL> insert into dept values (1,'IT','Tholudur');

1 row created.
```

Q2: Insert more than a record into emp table using a single insert command.

```
Ans: SQL> insert into emp values(&empno,'&ename','&job',&deptno,&sal);

Enter value for empno: 1

Enter value for ename: Mathi

Enter value for job: AP

Enter value for deptno: 1

Enter value for sal: 10000

old 1: insert into emp values(&empno,'&ename','&job',&deptno,&sal)
new 1: insert into emp values(1,'Mathi','AP',1,10000)
```

```
1 row created.

SQL> / Enter value for empno: 2

Enter value for ename: Arjun

Enter value for job: ASP

Enter value for deptno: 2

Enter value for sal: 12000

old 1: insert into emp values(&empno,'&ename','&job',&deptno,&sal)

new 1: insert into emp values(2,'Arjun','ASP',2,12000)

1 row created.


SQL> / Enter value for empno: 3

Enter value for ename: Gugan

Enter value for job: ASP

Enter value for deptno: 1

Enter value for sal: 12000

old 1: insert into emp values(&empno,'&ename','&job',&deptno,&sal)

new 1: insert into emp values(3,'Gugan','ASP',1,12000)

1 row created.
```

Q3: Update the emp table to set the salary of all employees to Rs15000/- who are working as ASP

Ans: SQL> select * from emp;

```
EMPNO ENAME JOB DEPTNO SAL

 ---------- -------------------- ------------- ---------- ----------

1 Mathi AP 1 10000

 2 Arjun ASP 2 12000
```

```
3 Gugan ASP 1 12000
SQL> update emp set sal=15000 where job='ASP'; 2 rows updated.

SQL> select * from emp;

EMPNO ENAME JOB DEPTNO SAL

 ---------- -------------------- ------------- ---------- ----------

1 Mathi AP 1 10000

2 Arjun ASP 2 15000

3 Gugan ASP 1 15000
```

Q4: Create a pseudo table employee with the same structure as the table emp and insert rows into the table using select clauses.

```
Ans: SQL> create table employee as select * from emp;

Table created.

SQL> desc employee;

Name Null? Type

----------------------------------------- -------- ----------------------------

EMPNO NUMBER(6)

ENAME NOT NULL VARCHAR2(20)

JOB NOT NULL VARCHAR2(13)

DEPTNO NUMBER(3)

SAL NUMBER(7,2)
```

Q5: select employee name, job from the emp table

```
Ans: SQL> select ename, job from emp;
ENAME JOB
-------------------- -------------
Mathi AP
Arjun ASP
Gugan ASP
Karthik Prof
```

```
Akalya AP
suresh lect
6 rows selected.
```

**RESULT:**

Thus the DML commands have been executed successfully.

## EX.NO:3 BASIC SELECT STATEMENTS

**AIM**

To study the various Basic Select statement on the database.

Q1: Delete only those who are working as lecturer

```
Ans: SQL> select * from emp;
EMPNO ENAME JOB DEPTNO SAL
---------- ------------------- ------------- ---------- ----------
1 Mathi AP 1 10000
2 Arjun ASP 2 15000
3 Gugan ASP 1 15000
4 Karthik Prof 2 30000
5 Akalya AP 1 10000
6 suresh lect 1 8000
6 rows selected.
SQL> delete from emp where job='lect';
1 row deleted.
SQL> select * from emp;
EMPNO ENAME JOB DEPTNO SAL
-                --------- ------------------- ------------- ---------- ----------
1 Mathi AP 1 10000
2 Arjun ASP 2 15000
3 Gugan ASP 1 15000
4 Karthik Prof 2 30000
5 Akalya AP 1 10000
```

Q2: List the records in the emp table orderby salary in ascending order.

```
Ans: SQL> select * from emp order by sal;
EMPNO ENAME JOB DEPTNO SAL
---------- ------------------- ------------- ---------- ----------
1 Mathi AP 1 10000
5 Akalya AP 1 10000
2 Arjun ASP 2 15000
3 Gugan ASP 1 15000
4 Karthik Prof 2 30000
```

Q3: List the records in the emp table orderby salary in descending order.

```
Ans: SQL> select * from emp order by sal desc;
EMPNO ENAME JOB DEPTNO SAL
---------- ------------------- ------------- ---------- ----------
```

```
4 Karthik Prof 2 30000
2 Arjun ASP 2 15000
3 Gugan ASP 1 15000
1 Mathi AP 1 10000
5 Akalya AP 1 10000
```

Q4: Display only those employees whose deptno is 30.
Solution: Use SELECT FROM WHERE syntax.

Ans: SQL> select * from emp where deptno=1;

```
EMPNO ENAME JOB DEPTNO SAL
---------- ------------------- ------------- ---------- ----------
1 Mathi AP 1 10000
3 Gugan ASP 1 15000
5 Akalya AP 1 10000
```

Q5: Display deptno from the table employee avoiding the duplicated values.
Solution:
1. Use SELECT FROM syntax.
2.Select should include distinct clause for the deptno.
Ans: SQL> select distinct deptno from emp;

```
DEPTNO
----------
1

2
```

6. To select all stores with sales above $1,000 in Table Store_Information

SQL>SELECT Store_Name FROM Store_Information WHERE Sales > 1000;

```
Store_Name
Los Angeles
```

7. To select all distinct stores in Table Store_Information, we key in,

SQL>SELECT DISTINCT Store_Name FROM Store_Information;

Result:

```
Store_Name

Los Angeles
```

```
San Diego
Boston
```

8. If we want to select all stores with sales greater than $1,000 or all stores with sales less than $500 but greater than $275 in Table Store_Information, we key in,

```
SQL>SELECT Store_Name FROM Store_Information WHERE Sales > 1000 OR (Sales < 500
AND Sales > 275);
```

Result:

```
Store_Name
Los Angeles
San Francisco
```

9.  To select all records for the Los Angeles and the San Diego stores in Table Store_Information, we key in,

```
SQL>SELECT * FROM Store_Information WHERE Store_Name IN ('Los Angeles', 'San
Diego');
```

Result:

```
Store_Name          Sales   Txn_Date
Los Angeles         1500    Jan-05-1999
San Diego           250     Jan-07-1999
```

10. To select view all sales information between January 6, 1999, and January 10, 1999, we key in,

```
SQL>SELECT * FROM Store_Information WHERE Txn_Date BETWEEN 'Jan-06-1999' AND
'Jan-10-1999';
```

Note that date may be stored in different formats in different databases. This tutorial simply choose one of the formats.

Result:

```
Store_Name          Sales   Txn_Date
San Diego           250     Jan-07-1999
San Francisco       300     Jan-08-1999
Boston               700    Jan-08-1999
```

11. We want to find all stores whose name contains 'AN'. To do so, we key in,

```
SQL>SELECT * FROM Store_Information WHERE Store_Name LIKE '%AN%';
```

```
Result:
Store_Name              Sales   Txn_Date
LOS ANGELES             1500    Jan-05-1999
SAN DIEGO               250     Jan-07-1999
SAN FRANCISCO           300     Jan-08-1999
```

12. To list the contents of Table Store_Information by Sales in descending order, we key in,

```
SQL> SELECT Store_Name, Sales, Txn_Date FROM Store_Information ORDER BY Sales
DESC;
```

```
Result:
Store_Name              Sales   Txn_Date
Los Angeles             1500    Jan-05-1999
Boston                  700     Jan-08-1999
San Francisco           300     Jan-08-1999
San Diego               250     Jan-07-1999
```

13. To see only the stores with sales over $1,500, we would type,

```
SELECT Store_Name, SUM(Sales)
FROM Store_Information GROUP BY Store_Name HAVING SUM(Sales) > 1500;
```

```
Result:
Store_Name              SUM(Sales)
Los Angeles             1800
```

**RESULT:**

Thus the Basic select commands have been executed successfully.

## EX.NO:4 ADVANCED SELECT STATEMENTS

**AIM**

To study the various Advanced Select statement on the database.

1. The LIMIT clause restricts the number of results returned from a SQL statement. It is available in MySQL.

Syntax
The syntax for LIMIT is as follows:

```
[SQL Statement 1]
LIMIT [N];
```

To retrieve the two highest sales amounts in Table Store_Information, we key in:

```
SELECT Store_Name, Sales, Txn_Date FROM Store_Information
ORDER BY Sales DESC
LIMIT 2;
```

| Store_Name | Sales | Txn_Date |
|---|---|---|
| Los Angeles | 1500 | Jan-05-1999 |
| Boston | 700 | Jan-08-1999 |

2. The TOP keyword restricts the number of results returned from a SQL statement in Microsoft SQL Server.
Syntax

The syntax for TOP is as follows:

```
SELECT TOP [TOP argument] "column_name"
FROM "table_name";
```

where [TOP argument] can be one of two possible types:

1. [N]: The first N records are returned.

2. [M] PERCENT: The number of records corresponding to M% of all qualifying records are returned.

Examples

We use the following table for our examples.

Table Store_Information

| Store_Name | Sales | Txn_Date |
|---|---|---|
| Los Angeles | 1500 | Jan-05-1999 |
| San Diego | 250 | Jan-07-1999 |
| San Francisco | 300 | Jan-08-1999 |

|  |  |  |
|---|---|---|
| Boston | 700 | Jan-08-1999 |

[TOP argument] is an integer

To show the two highest sales amounts in Table Store_Information, we key in,

```
SELECT TOP 2 Store_Name, Sales, Txn_Date
FROM Store_Information
ORDER BY Sales DESC;
```

Result:

| Store_Name | Sales | Txn_Date |
|---|---|---|
| Los Angeles | 1500 | Jan-05-1999 |
| Boston | 700 | Jan-08-1999 |

TOP argument] is a percentage

To show the top 25% of sales amounts from Table Store_Information, we key in,

```
SELECT TOP 25 PERCENT Store_Name, Sales, Txn_Date
FROM Store_Information
ORDER BY Sales DESC;
```

Result:

| Store_Name | Sales | Txn_Date |
|---|---|---|
| Los Angeles | 1500 | Jan-05-1999 |

3. EXISTS is a Boolean operator used in a subquery to test whether the inner query returns any row.
If it does, then the outer query proceeds. If not,
the outer query does not execute, and the entire SQL statement returns nothing.

The syntax for EXISTS is:

```
SELECT "column_name1"
FROM "table_name1"
WHERE EXISTS
(SELECT *
FROM "table_name2"
WHERE "condition");
```

The following SQL query,

```
SELECT SUM(Sales) FROM Store_Information
WHERE EXISTS
(SELECT * FROM Geography
WHERE Region_Name = 'West');
```
produces the result below:

```
SUM(Sales)
2750
```

4. CASE is used to provide if-then-else type of logic to SQL. There are two formats: The first is a Simple CASE expression,
 where we compare an expression to static values. The second is a Searched CASE expression,
 where we compare an expression to one or more logical conditions.
Simple CASE Expression Syntax

The syntax for a simple CASE expression is:

```
SELECT CASE ("column_name")
  WHEN "value1" THEN "result1"
  WHEN "value2" THEN "result2"
  ...
  [ELSE "resultN"]
  END
FROM "table_name";
```
The ELSE clause is optional.
```
SELECT Store_Name, CASE Store_Name
  WHEN 'Los Angeles' THEN Sales * 2
  WHEN 'San Diego' THEN Sales * 1.5
  ELSE Sales
  END
"New Sales",
Txn_Date
FROM Store_Information;
```

```
Store_Name              New Sales     Txn_Date
Los Angeles             3000    Jan-05-1999
San Diego               375     Jan-07-1999
San Francisco           300     Jan-08-1999
Boston                  700     Jan-08-1999
```
```
ELECT CASE
  WHEN "condition1" THEN "result1"
  WHEN "condition2" THEN "result2"
  ...
  [ELSE "resultN"]
  END
FROM "table_name";
```
The ELSE clause is optional. "Condition" can consist of one or more logical statements.

Searched CASE Expression Example

We use the same Store_Information above. If we want to define the status of a store's sale based on the following rules:

If Sales >= 1,000, it's a "Good Day"
If Sales >= 500 and < 1,000, it's an "OK Day"
If Sales < 500, it's a "Bad Day"
We can use the following searched CASE expression:

```
SELECT Store_Name, Txn_Date, CASE
  WHEN Sales >= 1000 THEN 'Good Day'
  WHEN Sales >= 500 THEN 'OK Day'
  ELSE 'Bad Day'
  END
"Sales Status"
FROM Store_Information;
```

Result:

| Store_Name | Txn_Date | Sales Status |
|---|---|---|
| Los Angeles | Jan-05-1999 | Good Day |
| San Diego | Jan-07-1999 | Bad Day |
| San Francisco | Jan-08-1999 | Bad Day |
| Boston | Jan-08-1999 | OK Day |

Note that a simple CASE expression is a special case of a searched CASE expression.

As an example, the following two CASE expressions are identical:

Simple CASE Expression:

```
SELECT Store_Name, CASE Store_Name
  WHEN 'Los Angeles' THEN Sales * 2
  WHEN 'San Diego' THEN Sales * 1.5
  ELSE Sales
  END
"New Sales",
Txn_Date
FROM Store_Information;
```

Searched CASE Expression:

```
SELECT Store_Name, CASE
  WHEN Store_Name = 'Los Angeles' THEN Sales * 2
  WHEN Store_Name = 'San Diego' THEN Sales * 1.5
  ELSE Sales
  END
"New Sales",
Txn_Date
```

```
FROM Store_Information;
```

5. AUTO_INCREMENT is used in MySQL to create a numerical primary key value for each additional row of data.5

The syntax for AUTO_INCREMENT is as follows:

```
CREATE TABLE TABLE_NAME
(PRIMARY_KEY_COLUMN INT NOT NULL AUTO_INCREMENT
...
PRIMARY KEY (PRIMARY_KEY_COLUMN));
```

```
CREATE TABLE USER_TABLE
(Userid int NOT NULL AUTO_INCREMENT,
Last_Name varchar(50),
First_Name varchar(50),
PRIMARY KEY (Userid));
```

```
INSERT INTO USER_TABLE VALUES ('Perry', 'Jonathan');
```

Table USER_TABLE

```
Userid                  Last_Name     First_Name
1                       Perry         Jonathan
```

6. he purpose of the SQL UNION query is to combine the results of two queries together while removing duplicates. In other words, when using UNION, only unique values are returned (similar to SELECT DISTINCT).

Syntax

The syntax of UNION in SQL is as follows:

```
[SQL Statement 1]
UNION
[SQL Statement 2];
```

```
ELECT Txn_Date FROM Store_Information
UNION
SELECT Txn_Date FROM Internet_Sales;
```

Result:

```
Txn_Date
Jan-05-1999
Jan-07-1999
Jan-08-1999
```

```
Jan-10-1999
Jan-11-1999
Jan-12-1999
```

7. The INTERSECT command in SQL combines the results of two SQL statement and returns only data that are present in both SQL statements.

INTERSECT can be thought of as an AND operator (value is selected only if it appears in both statements), while UNION and UNION ALL can be thought of as an OR operator (value is selected if it appears in either the first or the second statement).

Syntax

The syntax for INTERSECT is as follows:

```
[SQL Statement 1]
INTERSECT
[SQL Statement 2];
```

```
SELECT Txn_Date FROM Store_Information
INTERSECT
SELECT Txn_Date FROM Internet_Sales;
```
Result:

```
Txn_Date
Jan-07-1999
```

8. The MINUS command operates on two SQL statements. It takes all the results from the first SQL statement, and then subtract out the ones that are present in the second SQL statement to get the final result set. If the second SQL statement includes results not present in the first SQL statement, such results are ignored.

Syntax

The syntax for MINUS is as follows:

```
[SQL Statement 1]
MINUS
[SQL Statement 2];
```

```
SELECT Txn_Date FROM Store_Information
MINUS
SELECT Txn_Date FROM Internet_Sales;
```
Result:

```
Txn_Date
Jan-05-1999
Jan-08-1999
```

**RESULT:**

Thus the advanced Select Statements have been executed successfully

## EX.NO:5 INTEGRITY AND CONSTRAINTS

**AIM**

To study the various Integrity and Constraints on the database.

Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table. If there is any violation between the constraint and the data action, the action is aborted.

Constraints can be column level or table level. Column level constraints apply to a column, and table level constraints apply to the whole table.

The following constraints are commonly used in SQL:

- **NOT NULL** - Ensures that a column cannot have a NULL value
- **UNIQUE** - Ensures that all values in a column are different
- **PRIMARY KEY** - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
- **FOREIGN KEY** - Uniquely identifies a row/record in another table
- **CHECK** - Ensures that all values in a column satisfies a specific condition
- **DEFAULT** - Sets a default value for a column when no value is specified
- **INDEX** - Use to create and retrieve data from the database very quickly

Q1. Sql>Create Table Employee(Empno Number(4)
**Primary Key,**
Ename Varchar2(10),
Job Varchar2(6),
Sal Number(5),
Deptno Number(7));

Column Level Constraints Using Primary Key With Naming Convention
Q2. Sql>Create Table Employee (Empno Number(4)
Constraint **Emp_Empno_Pk Primary Key**,
Ename Varchar2(10),
Job Varchar2(6),
Sal Number(5),
Deptno Number(7));
Q3: Write a query to create foreign key constraints with Table level with alter command.

Sql>Create Table Dept (Deptno Number (2) **Primary Key**,

Dname Varchar2 (20),
Location Varchar2 (15));

Sql>Create Table Emp4 (Empno Number (3),
Deptno Number (2) **References Dept (Deptno)**,
Design Varchar2 (10));
Column Level Foreign Key Constraint With Naming Conversions:

Q4. Sql>Create Table Dept
(Deptno Number (2) **Primary Key,**
Dname Varchar2 (20),
Location Varchar2 (15));

Sql>Create Table Emp5
(Empno Number (3),
Deptno Number (2),
Design Varchar2 (10) **Constraint Enp2_Deptno_Fk Foreign Key**
(Dept No) Referencesdept (Deptno));

Q5. Write a query to create Check constraints with table level using alter command

Sql>Create Table Emp7
(Empno Number (3),
Ename Varchar2 (20),
Design Varchar2 (15),
Sal Number (5) **Constraint Emp7_Sal_Ck Check (Sal>500 And Sal <10001**
Dweptno Number (2));

Q6. Write a query to create unique constraints with column level

Sql>Create Table Emp10 (Empno Number (3),
Ename Varchar2 (20),
Desgin Varchar2 (15) **Constraint Emp10_Design_Uk Unique**,
Sal Number (5));

Q7. Write a query to create Not Null constraints with column level

Sql>Create Table Emp13
(Empno Number (4),
Ename Varchar2 (20) **Constraint Emp13_Ename_Nn Not Null,**
Design Varchar2 (20),
Sal Number (3));

Q8. Write a query to create Null constraints with column level.

```
 Sql>Create Table Emp13
 (Empno Number (4),
 Ename Varchar2 (20) Constraint Emp13_Ename_Nn Null,
Design Varchar2 (20),
Sal Number (3));
```

Q9. Write a query to disable the constraints

```
Sql>Alter Table Emp13 Disable Constraint Emp13_Ename_Nn Null;
```

Q10. Write a query to enable the constraints

```
 Sql>Alter Table Emp13 Enable Constraint Emp13_Ename_Nn Null;
```

**RESULT:**

Thus the Integrity and Constraints have been executed successfully

## EX.NO:6 JOINING TABLES

**AIM**

To study the various Join operations on the database.

SQL joins are used to query data from two or more tables, based on a relationship between certain columns in these tables.

The INNER JOIN keyword return rows when there is at least one match in both tables.

The LEFT JOIN keyword returns all rows from the left table (table_name1), even if there are no matches in the right table (table_name2).

The RIGHT JOIN keyword Return all rows from the right table (table_name2), even if there are no matches in the left table (table_name1).

The FULL JOIN keyword return rows when there is a match in one of the tables.

**LEFT JOIN or LEFT OUTTER JOIN**

Table:1 - ORDERS

```
SQL> CREATE table orders(O_Id number(5),
Orderno number(5),
P_Id number(3));

Table created.

SQL> DESC orders;
 Name Null? Type
 -------------------- -------- -----------
 O_ID NUMBER(5)
 ORDERNO NUMBER(5)
 P_ID NUMBER(3)
```

INSERTING VALUES INTO ORDERS

```
SQL> INSERT into orders values(&O_Id,&Orderno,&P_Id);
Enter value for o_id: 1
Enter value for orderno: 77895
Enter value for p_id: 3
old 1: INSERT into orders values(&O_Id,&Orderno,&P_Id)
new 1: INSERT into orders values(1,77895,3)
```

```
1 row created.

SQL> INSERT into orders values(&O_Id,&Orderno,&P_Id);

Enter value for o_id: 2
Enter value for orderno: 44678
Enter value for p_id: 3
old 1: INSERT into orders values(&O_Id,&Orderno,&P_Id)
new 1: INSERT into orders values(2,44678,3)
1 row created.

SQL> INSERT into orders values(&O_Id,&Orderno,&P_Id);

Enter value for o_id: 3
Enter value for orderno: 22456
Enter value for p_id: 1
old 1: INSERT into orders values(&O_Id,&Orderno,&P_Id)
new 1: INSERT into orders values(3,22456,1)

1 row created.

SQL> INSERT into orders values(&O_Id,&Orderno,&P_Id);

Enter value for o_id: 4
Enter value for orderno: 24562
Enter value for p_id: 1
old 1: INSERT into orders values(&O_Id,&Orderno,&P_Id)
new 1: INSERT into orders values(4,24562,1)

1 row created.

SQL> INSERT into orders values(&O_Id,&Orderno,&P_Id);

Enter value for o_id: 5
Enter value for orderno: 34764
Enter value for p_id: 15
old 1: INSERT into orders values(&O_Id,&Orderno,&P_Id)
new 1: INSERT into orders values(5,34764,15)

1 row created.
```

TABLE SECTION:

```
SQL> SELECT * FROM orders;
 O_ID ORDERNO P_ID
 ---------- ---------- ----------
 1 77895 3
 2 44678 3
 3 22456 1
 4 24562 1
 5 34764 15
```

TABLE -2: PERSONS

```
SQL> CREATE table persons(p_Id number(5),
LASTNAME varchar2(10),
Firstname varchar2(15), Address varchar2(20),
city varchar2(10));

Table created.

SQL> INSERT into persons values(&p_Id,'&Lastname','&firstname','&Address','&city');
Enter value for p_id: 1
Enter value for lastname: Hansen
Enter value for firstname: Ola
Enter value for address: Timoteivn 10
Enter value for city: sadnes

old 1: INSERT into persons values(&p_Id,'&Lastname','&firstname','&Address','&city')
new 1: INSERT into persons values(1,'Hansen','Ola','Timoteivn 10','sadnes')

1 row created.

SQL> INSERT into persons values(&p_Id,'&Lastname','&firstname','&Address','&city');
Enter value for p_id: 2
Enter value for lastname: Svendson
Enter value for firstname: Tove
Enter value for address: Borgn 23
Enter value for city: Sandnes

old 1: INSERT into persons values(&p_Id,'&Lastname','&firstname','&Address','&city')
new 1: INSERT into persons values(2,'Svendson','Tove','Borgn 23','Sandnes')

1 row created.

SQL> INSERT into persons values(&p_Id,'&Lastname','&firstname','&Address','&city');
Enter value for p_id: 3
Enter value for lastname: Pettersen
Enter value for firstname: Kari
Enter value for address: Storgt 20
Enter value for city: Stavanger
old 1: INSERT into persons values(&p_Id,'&Lastname','&firstname','&Address','&city')
```

```
new 1: INSERT into persons values(3,'Pettersen','Kari','Storgt 20','Stavanger')

1 row created.

SQL> SELECT * FROM persons;
 P_ID LASTNAME FIRSTNAME ADDRESS CITY
 ---------- ---------- --------------- ------------------- ----------
 1 Hansen Ola Timoteivn 10 sandnes
 2 Svendson Tove Borgn 23 Sandnes
 3 Pettersen Kari Storgt 20 Stavanger
```

LEFT JOIN SYNTAX

```
SQL> SELECT column_name(s)
FROM table_name1
LEFT JOIN table_name2
ON table_name1.column_name=table_name2.column_name
```

LEFT JOIN EXAMPLE

```
SQL> SELECT persons.lastname,persons.firstname,orders.orderno
 FROM persons
 LEFT JOIN orders
 ON persons.p_Id = orders.p_Id
 ORDER BY persons.lastname;
```

OUTPUT

```
LASTNAME FIRSTNAME ORDERNO
------------------ ------------------ ---------------
Hansen Ola 22456
Hansen Ola 24562
Pettersen Kari 77895
Pettersen Kari 44678
Svendson Tove
```

FULL OUTTER JOIN

```
SQL> SELECT * FROM persons;
 P_ID LASTNAME FIRSTNAME ADDRESS CITY
---------- --------------- ------------------- ---------------- ----------
 1 Hansen Ola Timoteivn 10 sandnes
 2 Svendson Tove Borgn 23 Sandnes
 3 Pettersen Kari Storgt 20 Stavanger
SQL> SELECT * FROM orders;
 O_ID ORDERNO P_ID
---------- ---------- ----------
 1 77895 3
 2 44678 3
 3 22456 1

 4 24562 1
 5 34764 15
```

FULL OUTER JOIN SYNTAX

```
SQL>SELECT column_name(s)
FROM table_name1
FULL JOIN table_name2
ON table_name1.column_name=table_name2.column_name
```

FULL OUTER JOIN EXAMPLE

```
SQL> SELECT persons.lastname,persons.firstname,orders.orderno
 FROM persons
 FULL OUTER JOIN orders
 ON persons.p_Id = orders.p_Id
 ORDER BY persons.lastname;
```

RIGHT OUTTER JOIN

RIGHT OUTTER JOIN SYNTAX

```
SQL>SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo
FROM Persons
RIGHT JOIN Orders
ON Persons.P_Id=Orders.P_Id
ORDER BY Persons.LastName
```

RIGHT OUTTER JOIN EXAMPLE

```
SQL> SELECT persons.lastname,persons.firstname,orders.orderno
 FROM persons
 RIGHT OUTER JOIN orders
 ON persons.p_Id = orders.p_Id
 ORDER BY persons.lastname;

LASTNAME FIRSTNAME ORDERNO
------------------- ------------------ ---------------
Hansen Ola 24562
Hansen Ola 22456
Pettersen Kari 44678
Pettersen Kari 77895
```

INNER JOIN

INNTER JOIN SYNTAX

```
SQL>SELECT column_name(s)
FROM table_name1
INNER JOIN table_name2
ON table_name1.column_name=table_name2.column_name
```

INNTER JOIN EXAMPLE

```
SQL> SELECT persons.lastname,persons.firstname,orders.orderno
 2 FROM persons
 3 INNER JOIN orders
 4 ON persons.p_Id = orders.p_Id
 5 ORDER BY persons.lastname;

LASTNAME FIRSTNAME ORDERNO
------------------ ------------------ ---------------
Hansen Ola 22456
Hansen Ola 24562
Pettersen Kari 77895
Pettersen Kari 44678
```

```
LASTNAME FIRSTNAME ORDERNO
------------- --------------- ----------
Hansen Ola 22456
Hansen Ola 24562
Pettersen Kari 77895
Pettersen Kari 44678
Svendson Tove 34764
6 rows selected.
```

**RESULT:**

Thus the joining tables have been executed successfully.

## EX.NO:7 SQL FUNCTIONS

**AIM**

To study the various SQL Functions operations on the database.

**CHARACTER/STRING FUNCTION:**

```
SQL> select upper('welcome') from dual;
-----------
WELCOME

SQL> select upper('hai') from dual;
---
HAI

SQL> select lower('HAI') from dual;
LOW
---
hai

SQL> select initcap(„hello world') from dual;
INITCAP('Hello
--------------
Hello World

SQL> select ltrim('    hai') from dual;
LTR
---
hai

SQL> select rtrim('hai     ')from dual;
RTR
---
hai

SQL> select rtrim('    hai    ')from dual;
RTRIM('
-------
    hai
SQL> select concat('SRM',' university')from dual;
-----------------------
SRM university
```

```
SQL> select length('SRM‟)from dual;
LENGTH('SRM')
----------------------
12

SQL> select replace('SRM university', 'SRM','Anna')from dual;
----------------
Anna university

SQL> select substr('SRM', 7,6)from dual;
SUBSTR
------
lingam
SQL> select rpad('hai',3,'*')from dual;

RPAD('
------
hai***

SQL> select lpad('hai',3,'*')from dual;
LPAD('
------
***hai

SQL> select replace('Dany','y','ie')from dual;
REPLACE
-------
Danie

SQL> select translate('cold','ld','ol')from dual;
TRANSL
------
cool
```

## DATE & TIME FUNCTION

```
SQL> select sysdate from dual;
```

```
SYSDATE
---------
07-APR-10

SQL> select round(sysdate)from dual;
ROUND(SYS
---------
07-APR-10

SQL> select add_months(sysdate,3)from dual;
ADD_MONTH
---------
07-JUL-10

SQL> select last_day(sysdate)from dual;
LAST_DAY(
---------
30-APR-10

SQL> select sysdate+20 from dual;
SYSDATE+2
---------
27-APR-10

SQL> select next_day(sysdate,'tuesday')from dual;
NEXT_DAY(
---------
13-APR-10
```

## NUMERIC FUNCTION

```
SQL> select round(15.6789)from dual;
```

```
ROUND(15.6789)

--------------
16

SQL> select ceil(23.20)from dual;

CEIL(23.20)

-----------
24

SQL> select floor(34.56)from dual;

FLOOR(34.56)
-----------
34

SQL> select trunc(15.56743)from dual;

TRUNC(15.56743)
---------------
15


SQL> select sign(-345)from dual;

SIGN(-345)
----------
-1


SQL> select abs(-70)from dual;
 ABS(-70)
---------
70
```

## MATH FUNCTION:

```
SQL> select abs(45) from dual;

  ABS(45)
---------
45



SQL> select power(10,12) from dual;

POWER(10,12)
------------
```

```
1.000E+12

SQL> select mod(11,5) from dual;
MOD(11,5)
---------
1

SQL> select exp(10) from dual;
EXP(10)
---------
22026.466

SQL> select sqrt(225) from dual;
SQRT(225)
---------
15
```

**RESULT:**

Thus the SQL Functions have been executed successfully.

## EX.NO:8 SUB QUERIES

**AIM**

To study the various SQL sub queries operations on the database.

Sub Query can have more than one level of nesting in one single query. A SQL nested query is a SELECT query that is nested inside a SELECT, UPDATE, INSERT, or DELETE SQL query.

1. Select Command Is Used To Select Records From The Table.
2. Where Command Is Used To Identify Particular Elements.
3. Having Command Is Used To Identify Particular Elements.
4. Min (Sal) Command Is Used To Find Minimum Salary.

**Syntax For Creating A Table:**

```
 Sql: Create <Obj.Type> <Obj.Name> (Column Name.1 <Datatype>
(Size), Column Name.1 <Datatype> (Size) ……………………………);

Sql> Create Table Emp2(Empno Number(5),
Ename Varchar2(20),
Job Varchar2(20),
Sal Number(6),
Mgrno Number(4),
Deptno Number(3));
```

**Syntax For Insert Records In To A Table:**

```
 Sql :> Insert Into <Table Name> Values< Val1, ‘Val2’,…..);
```

Insertion

```
Sql> Insert Into Emp2 Values(1001,'Mahesh','Programmer',15000,1560,200);
1 Row Created.
Sql> Insert Into Emp2 Values (1002,'Manoj','Tester',12000,1560,200);
1 Row Created.
Sql> Insert Into Emp2 Values(1003,'Karthik','Programmer',13000,1400,201);
1 Row Created.
 Sql> Insert Into Emp2 Values(1004,'Naresh','Clerk',1400,1400,201);
1 Row Created.
Sql> Insert Into Emp2 Values(1005,'Mani','Tester',13000,1400,200);
1 Row Created.
Sql> Insert Into Emp2 Values(1006,'Viki','Designer',12500,1560,201);
1 Row Created.
Sql> Insert Into Emp2 Values(1007,'Mohan','Designer',14000,1560,201);1 Row Created.
Sql> Insert Into Emp2 Values(1008,'Naveen','Creation',20000,1400,201);
1 Row Created.
```

```
Sql> Insert Into Emp2 Values(1009,'Prasad','Dir',20000,1560,202);
1 Row Created.
 Sql> Insert Into Emp2 Values(1010,'Agnesh','Dir',15000,1400,200);
1 Row Created.
```

**Syntax For Select Records From The Table:**

```
 Sql> Select * From <Table Name>;
Sql> Select *From Emp2;
Empno Ename Job Sal Mgrno Dptno
---------- ---------- ---------- ---------- ---------- ----------
 1001 Mahesh Programmer 15000 1560 200
 1002 Manoj Tester 12000 1560 200
 1003 Karthik Programmer 13000 1400 201
 1004 Naresh Clerk 1400 1400 201
 1005 Mani Tester 13000 1400 200
 1006 Viki Designer 12500 1560 201
 1007 Mohan Designer 14000 1560 201
 1008 Naveen Creation 20000 1400 201
 1009 Prasad Dir 20000 1560 202
 1010 Agnesh Dir 15000 1400 200
```

Table- 2

**Syntax For Creating A Table:**

```
 Sql: Create <Obj.Type> <Obj.Name> (Column Name.1 <Datatype>
(Size), Column Name.1 <Datatype> (Size) ……………………………);
Sql> Create Table Dept2(Deptno Number(3),
Deptname Varchar2(10),
Location Varchar2(15));
Table Created.
```

**Syntax For Insert Records In To A Table:**

```
 Sql :> Insert Into <Table Name> Values< Val1, ‘Val2’,…..);
```

**Insertion**

```
Sql> Insert Into Dept2 Values(107,'Develop','Adyar');
1 Row Created.
Sql> Insert Into Dept2 Values(201,'Debug','Uk');
1 Row Created.
Sql> Insert Into Dept2 Values(200,'Test','Us');
Sql> Insert Into Dept2 Values(201,'Test','Ussr');
1 Row Created.
Sql> Insert Into Dept2 Values(108,'Debug','Adyar');
1 Row Created.
 Sql> Insert Into Dept2 Values(109,'Build','Potheri');
```

```
1 Row Created.
```

Syntax For Select Records From The Table:

```
 Sql> Select * From <Table Name>;
Sql> Select *From Dept2;

 Deptno Deptname Location
 ---------- ---------- ---------------
 107 Develop Adyar
 201 Debug Uk
 200 Test Us
 201 Test Ussr
 108 Debug Adyar
 109 Build Potheri
6 Rows Selected.
```

**General Syntax For Nested Query:**

```
Select "Column_Name1"
From "Table_Name1"
Where "Column_Name2" [Comparison Operator]
(Select "Column_Name3"
From "Table_Name2"
Where [Condition])
```

**Syntax Nested Query Statement:**

```
Sql> Select <Column_Name> From Frorm <Table _1> Where
 <Column_Name> <Relational _Operation> ‘Value’
 (Select (Aggrecate Function) From <Table_1> Where <Column
 Name> = ‘Value’
 (Select <Column_Name> From <Table_2> Where <Column_Name=
 ‘Value’));
```

Nested Query Statement:

```
Sql> Select Ename From Emp2 Where Sal>
(Select Min(Sal) From Emp2 Where Dptno=
(Select Deptno From Dept2 Where Location='Uk'));
```

**Nested Query Output:**

```
Ename
----------
Mahesh
Manoj
```

```
Karthik
Mani
Viki
Mohan
Naveen
Prasad
Agnesh
```

**RESULT:**

Thus the SQL sub query has been executed successfully.

## EX.NO:9 VIEWS

**AIM**

To study the various SQL view operations on the database.

1. CREATE VIEW command is used to define a view.
2.  INSERT command is used to insert a new row into the view.
3. DELETE command is used to delete a row from the view.
4. UPDATE command is used to change a value in a tuple without changing all values in the tuple.
5. DROP command is used to drop the view table

Commands Execution

Creation Of Table

```
--------------------------------
Sql> Create Table Employee (
Employee_Namevarchar2(10),
Employee_Nonumber(8),
Dept_Name Varchar2(10),
Dept_No Number (5),Date_Of_Join Date);
Table Created.
```

Table Description

```
-------------------------------
Sql> Desc Employee;
 Name Null? Type
 ------------------------------ -------- ------------------------
 Employee_Name Varchar2(10)
 Employee_No Number(8)
 Dept_Name Varchar2(10)
 Dept_No Number(5)
 Date_Of_Join Date
```

Suntax For Creation Of View

```
--------------------------------------------------
Sql> Create <View> <View Name> As Select
 <Column_Name_1>, <Column_Name_2> From <Table Name>;
```

Creation Of View

```
------------------------------
Sql> Create View Empview As Select
Employee_Name,Employee_No,Dept_Name,Dept_No,Date_Of_Join From
Employee;
View Created.
```

Description Of View

```
--------------------------------
Sql> Desc Empview;
Name Null? Type
 ----------------------------------------- -------- ----------------------------
 Employee_Name Varchar2(10)
 Employee_No Number(8)
 Dept_Name Varchar2(10)
 Dept_No Number(5)
```

Display View:

```
----------------------
Sql> Select * From Empview;
Employee_N Employee_No Dept_Name Dept_No
---------- ----------- ---------- ----------
Ravi 124 Ece 89
Vijay 345 Cse 21
Raj 98 It 22
Giri 100 Cse 67
```

Insertion Into View

```
----------------------------------
```

Insert Statement:

Syntax:

```
Sql> Insert Into <View_Name> (Column Name1,………)
Values(Value1,….);
Sql> Insert Into Empview Values ('Sri', 120,'Cse', 67,'16-Nov-1981');
1 Row Created.
Sql> Select * From Empview;
Employee_N Employee_No Dept_Name Dept_No
---------- ----------- ---------- ----------
Ravi 124 Ece 89
Vijay 345 Cse 21
Raj 98 It 22
Giri 100 Cse 67
Sri 120 Cse 67
Sql> Select * From Employee;
Employee_N Employee_No Dept_Name Dept_No Date_Of_J
---------- ----------- ---------- ---------- ---------
```

```
Ravi 124 Ece 89 15-Jun-05
Vijay 345 Cse 21 21-Jun-06
Raj 98 It 22 30-Sep-06
Giri 100 Cse 67 14-Nov-81
Sri 120 Cse 67 16-Nov-81
```

Deletion Of View:

Delete Statement:

Syntax:

```
Sql> Delete <View_Nmae>Where <Column Nmae> =’Value’;
Sql> Delete From Empview Where Employee_Name='Sri';
1 Row Deleted.
Sql> Select * From Empview;
Employee_N Employee_No Dept_Name Dept_No
---------- ----------- ---------- ----------
Ravi 124 Ece 89
Vijay 345 Cse 21
Raj 98 It 22
Giri 100 Cse 67
```

Update Statement:

Syntax:

```
Aql>Update <View_Name> Set< Column Name> = <Column Name>
+<View> Where <Columnname>=Value;
Sql> Update Empkaviview Set Employee_Name='Kavi' Where
Employee_Name='Ravi';
1 Row Updated.
Sql> Select * From Empkaviview;
Employee_N Employee_No Dept_Name Dept_No
---------- ----------- ---------- ----------
Kavi 124 Ece 89
Vijay 345 Cse 21
Raj 98 It 22
Giri 100 Cse 67
```

Drop A View:

Syntax:

```
Sql> Drop View <View_Name>
```

Example

```
Sql>Drop View Empview;

 View Droped
```

Create A View With Selected Fields:

Syntax:
Sql>Create [Or Replace] View <View Name>As Select <Column
Name1>…..From <Table Anme>;
Example-2:
Sql> Create Or Replace View Empl_View1 As Select Empno, Ename,
Salary From Empl;

Sql> Select * From Empl_View1;
Example-3:
Sql> Create Or Replace View Empl_View2 As Select * From Empl Where
Deptno=10;
Sql> Select * From Empl_View2;
Note:
Replace Is The Keyboard To Avoid The Error "Ora_0095:Name Is Already Used By An
Existing
Abject".
Changing The Column(S) Name M The View During As Select
Statement:
Type-1:
Sql> Create Or Replace View Emp_Totsal(Eid,Name,Sal) As Select
Empno,Ename,Salary From Empl;
View Created.

```
 Empno Ename Salary
---------- ------------------- ---------- ----------
 7369 Smith 1000
 7499 Mark 1050
 7565 Will 1500
 7678 John 1800
 7578 Tom 1500
 7548 Turner 1500
6 Rows Selected.
View Created.
 Empno Ename Salary Mgrno Deptno
---------- ------------------- ---------- ---------- ----------------------------
 7578 Tom 1500 7298 10
 7548 Turner 1500 7298 10
View Created.
```

Sql> Select * From Emp_Totsal;
Type-2:

```
Sql> Create Or Replace View Emp_Totsal As Select Empno "Eid",Ename
"Name",Salary "Sal" From Empl;
Sql> Select * From Emp_Totsal;
Example For Join View:
Type-3:
Sql> Create Or Replace View Dept_Emp As Select A.Empno "Eid",A.Ename
"Empname",A.Deptno "Dno",B.Dnam
E "D_Name",B.Loc "D_Loc" From Empl A,Depmt B Where
A.Deptno=B.Deptno;
Sql> Select * From Dept_Emp;
Eid Name Sal
---------- -------------------- ---------- ----------
 7369 Smith 1000
 7499 Mark 1050
 7565 Will 1500
 7678 John 1800
 7578 Tom 1500
 7548 Turner 1500
6 Rows Selected.
View Created.
Eid Name Sal
---------- -------------------- ---------- ----------
 7369 Smith 1000
 7499 Mark 1050
 7565 Will 1500
 7678 John 1800
 7578 Tom 1500
 7548 Turner 1500
6 Rows Selected.
View Created.
 Eid Empname Dno D_Name D_Loc
---------- -------------------- ---------- ---------- ------------------------
 7578 Tom 10 Account New York
 7548 Turner 10 Account New York
 7369 Smith 20 Sales Chicago
 7678 John 20 Sales Chicago
 7499 Mark 30 Research Zurich
 7565 WILL 30 RESEARCH ZURICH
```

**RESULT:**
Thus the SQL views have been executed successfully.

## EX.NO:10 BASICS OF PL/SQL

**AIM**
To study the various basic PL/SQL view operations on the database.

**1. PL/SQL CODING FOR ADDITION OF TWO NUMBERS**

```
SQL> declare
a number;
 b number;
c number;
begin
a:=&a;
b:=&b;
c:=a+b;
dbms_output.put_line('sum of'||a||'and'||b||'is'||c);
end;
 /
```

INPUT:

```
Enter value for a: 23
old 6: a:=&a;
new 6: a:=23;
Enter value for b: 12
old 7: b:=&b;
new 7: b:=12;
```

OUTPUT:

```
sum of23and12is35

PL/SQL procedure successfully completed.
```

**2. PL/ SQL GENERAL SYNTAX FOR IF CONDITION:**

```
SQL> DECLARE
 <VARIABLE DECLARATION>;
 BEGIN
 IF(CONDITION)THEN
 <EXECUTABLE STATEMENT >;
 END;
```

Coding for If Statement:

```
DECLARE
b number;
c number;
BEGIN
B:=10;
C:=20;
if(C>B) THEN
dbms_output.put_line('C is maximum');
end if;
```

```
end;
/
```

OUTPUT:

```
C is maximum

PL/SQL procedure successfully completed.
```

**3. PL/ SQL GENERAL SYNTAX FOR IF AND ELSECONDITION:**

```
SQL> DECLARE
 <VARIABLE DECLARATION>;
 BEGIN
 IF (TEST CONDITION) THEN
 <STATEMENTS>;
 ELSE
 <STATEMENTS>;
 ENDIF;
 END;
```

******************Less then or Greater Using IF ELSE **********************

```
SQL> declare
 n number;
 begin
 dbms_output. put_line('enter a number');
 n:=&number;
 if n<5 then
 dbms_output.put_line('entered number is less than 5');
 else
 dbms_output.put_line('entered number is greater than 5');

end if;
 end;
 /
```

Input

```
Enter value for number: 2
old 5: n:=&number;
new 5: n:=2;
```

Output:

```
entered number is less than 5

PL/SQL procedure successfully completed.
```

**4.PL/ SQL GENERAL SYNTAX FOR NESTED IF:**

```
SQL> DECLARE
 <VARIABLE DECLARATION>;
 BEGIN
 IF (TEST CONDITION) THEN
 <STATEMENTS>;
 ELSEIF (TEST CONDITION) THEN
 <STATEMENTS>;
 ELSE
 <STATEMENTS>;
 ENDIF;
 END;
```

*********** GREATEST OF THREE NUMBERS USING IF ELSEIF************

```
SQL> declare
 a number;
b number;
c number;
d number;
begin
a:=&a;
b:=&b;
 c:=&b;
if(a>b)and(a>c) then
dbms_output.put_line('A is maximum');
 elsif(b>a)and(b>c)then
dbms_output.put_line('B is maximum');
else
dbms_output.put_line('C is maximum');
end if;
end;
 /
```

INPUT:

```
Enter value for a: 21
old 7: a:=&a;
new 7: a:=21;
Enter value for b: 12
old 8: b:=&b;
new 8: b:=12;
Enter value for b: 45
old 9: c:=&b;
new 9: c:=45;
```

OUTPUT:

```
C is maximum
```

PL/SQL procedure successfully completed.

**5.PL/ SQL GENERAL SYNTAX FOR LOOPING STATEMENT:**

```
SQL> DECLARE
 <VARIABLE DECLARATION>;
 BEGIN
 LOOP
 <STATEMENT>;
 END LOOP;
 <EXECUTAVLE STATEMENT>;
 END;
```

***********SUMMATION OF ODD NUMBERS USING FOR LOOP***********

```
SQL> declare
n number;
sum1 number default 0;
endvalue number;
begin
endvalue:=&endvalue;
 n:=1;
for n in 1..endvalue
loop
 if mod(n,2)=1
then
sum1:=sum1+n;
end if;
 end loop;
dbms_output.put_line('sum ='||sum1);
end;
 /
```

INPUT:

```
Enter value for endvalue: 4
old 6: endvalue:=&endvalue;
new 6: endvalue:=4;
```

OUTPUT:

```
 sum =4

PL/SQL procedure successfully completed.
```

**6.PL/ SQL GENERAL SYNTAX FOR LOOPING STATEMENT:**

```
SQL> DECLARE
 <VARIABLE DECLARATION>;
 BEGIN
 WHILE <condition>
 LOOP
 <STATEMENT>;
 END LOOP;
 <EXECUTAVLE STATEMENT>;
 END;
```

*********SUMMATION OF ODD NUMBERS USING WHILE LOOP**********

```
SQL> declare
n number;
sum1 number default 0;
endvalue number;
begin
endvalue:=&endvalue;
n:=1;
while(n<endvalue)
loop
sum1:=sum1+n;
n:=n+2;
end loop;

dbms_output.put_line('sum of odd no. bt 1 and' ||endvalue||'is'||sum1);
end;
/
```

INPUT:

```
Enter value for endvalue: 4
old 6: endvalue:=&endvalue;
new 6: endvalue:=4;
```

OUTPUT:

```
sum of odd no. bt 1 and4is4
PL/SQL procedure successfully completed.
```

## 7. TRIGGER

### TYPE 1- TRIGGER AFTER UPDATE

```
SQL> CREATE OR REPLACE TRIGGER VIJAY
 AFTER UPDATE OR INSERT OR DELETE ON EMP
 FOR EACH ROW
 BEGIN
IF UPDATING THEN
 DBMS_OUTPUT.PUT_LINE('TABLE IS UPDATED');
ELSIF INSERTING THEN
DBMS_OUTPUT.PUT_LINE('TABLE IS INSERTED');
ELSIF DELETING THEN
```

```
DBMS_OUTPUT.PUT_LINE('TABLE IS DELETED');
END IF;
END;
/


Trigger created.
SQL> update emp set income =900 where empname='kumar';
TABLE IS UPDATED
1 row updated.
SQL> insert into emp values ( 4,'Chandru',700,250,80);
TABLE IS INSERTED
1 row created.
SQL> DELETE FROM EMP WHERE EMPID = 4;
TABLE IS DELETED
1 row deleted.

TYPE 2 - TRIGGER BEFORE UPDATE
------------------------------------------------------
SQL> CREATE OR REPLACE TRIGGER VASANTH
BEFORE UPDATE OR INSERT OR DELETE ON EMPLOYEE
FOR EACH ROW
BEGIN
IF UPDATING THEN
DBMS_OUTPUT.PUT_LINE('TABLE IS UPDATED');
ELSIF INSERTING THEN
DBMS_OUTPUT.PUT_LINE('TABLE IS INSERTED');
ELSIF DELETING THEN
DBMS_OUTPUT.PUT_LINE('TABLE IS DELETED');
END IF;
END;
/
Trigger created.
SQL> INSERT INTO EMP VALUES (4,'SANKAR',700,98,564);
TABLE IS INSERTED
1 row created.
SQL> UPDATE EMP SET EMPID = 5 WHERE EMPNAME = 'SANKAR';
TABLE IS UPDATED
1 row updated.
SQL> DELETE EMP WHERE EMPNAME='SANKAR';
TABLE IS DELETED
1 row deleted
```

**Create a Trigger to check the age valid or not Using Message Alert:**

```
PROGRAM:
SQL> SET SERVEROUTPUT ON;
SQL> CREATE TRIGGER TRIGNEW
AFTER INSERT OR UPDATE OF AGE ON TRIG
FOR EACH ROW
BEGIN
IF(:NEW.AGE<0) THEN
DBMS_OUTPUT.PUT_LINE('INVALID AGE');
ELSE
DBMS_OUTPUT.PUT_LINE('VALID AGE');
END IF;
END;
/
Trigger created.
SQL> insert into trig values('abc',15);
Valid age
1 row created.
SQL> insert into trig values('xyz',-12);
Invalid age
1 row created.
NAME AGE
---------- ----------
abc 15
xyz -12
```

3. Create a Trigger to check the age valid and Raise appropriate error code and error message.

```
SQL> create table data(name char(10),age number(3));
Table created.
SQL> desc data;
Name Null? Type
 ----------------------------------------- -------- ------------------------

NAME CHAR(10)
 AGE NUMBER(3)
SQL> CREATE TRIGGER DATACHECK
AFTER INSERT OR UPDATE OF AGE ON DATA
FOR EACH ROW
BEGIN
IF(:NEW.AGE<0) THEN
RAISE_APPLICATION_ERROR(-20000,'NO NEGATIVE AGE ALLOWED');
END IF;
END;
/
Trigger created.
SQL> INSERT INTO DATA VALUES('ABC',10);
1 ROW CREATED.
```

```
SQL> INSERT INTO DATA VALUES ('DEF',-15)
 *
ERROR at line 1:
ORA-20000: No negative age allowed
ORA-06512: at "4039.DATACHECK", line 3
ORA-04088: error during execution of trigger '4039.DATACHECK'
NAME AGE
---------- ----------
abc 10
```

4. Create a Trigger for EMP table it will update another table SALARY while inserting values.

```
SQL> CREATE TABLE SRM_EMP2(INAME VARCHAR2(10),
IID NUMBER(5),
SALARY NUMBER(10));
Table created.
SQL> CREATE TABLE SRM_SAL2(INAME VARCHAR2(10),
TOTALEMP NUMBER(5),
TOTALSAL NUMBER(10));
Table created.
```

**8. IMPLEMENTATION OF FACTORIAL USING FUNCTION**

I) PROGRAM:

```
 SQL>create function fnfact(n number)
return number is
b number;
begin
b:=1;
for i in 1..n
loop
b:=b*i;
end loop;
return b;
end;
/
 SQL>Declare
n number:=&n;
y number;
begin
y:=fnfact(n);
dbms_output.put_line(y);
end;
/
Function created.
Enter value for n: 5
```

```
old 2: n number:=&n;
new 2: n number:=5;
120
PL/SQL procedure successfully completed.
```

**9. PROCEDURE USING POSITIONAL PARAMETERS:**

```
PROCEDURE USING POSITIONAL PARAMETERS:
SQL> SET SERVEROUTPUT ON
SQL> CREATE OR REPLACE PROCEDURE PROC1 AS
 2 BEGIN
 3 DBMS_OUTPUT.PUT_LINE('Hello from procedure...');
 4 END;
 5 /
Output:
Procedure created.
SQL> EXECUTE PROC1
Hello from procedure...

PL/SQL procedure successfully completed.
II) PROGRAM:
PROCEDURE USING NOTATIONAL PARAMETERS:
SQL> CREATE OR REPLACE PROCEDURE PROC2
 2 (N1 IN NUMBER,N2 IN NUMBER,TOT OUT NUMBER) IS
 3 BEGIN
 4 TOT := N1 + N2;
 5 END;
 6 /
Output:
Procedure created.
SQL> VARIABLE T NUMBER
SQL> EXEC PROC2(33,66,:T)
PL/SQL procedure successfully completed.
SQL> PRINT T
 T
----------
 99
```

**RESULT:**
Thus the pl/sql have been executed successfully.

## **EX.NO:11** DESIGN AND DEVELOP APPLICATIONS

## SAMPLE: DESIGN AND IMPLEMENTATION OF LIBRARY MANAGEMENT SYSTEM

**STEPS:**

1. Create a database for library which request the using SQL

2. Establish ODBC connection

3. In the administrator tools open data source ODBC

4. Click add button and select oracle in ORA home 90, click finish

5. A window will appear given the data source home as oracle and select TNS source name as lion and give the used id as SWTT

6. ADODC CONTROL FOR library FORM:-

7. The above procedure must be follow except the table , A select the table as library

8. Write appropriate Program in form each from created in VB from each from created in VB form project.

Relational Database Schema

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Status | code | description | | | | | |
| Media | media_id | code | | | | | |
| Book | ISBN | title | author | year | dewey | price | |
| BookMedia | media_id | ISBN | | | | | |
| Customer | ID | name | addr | DOB | phone | username | password |
| Card | num | fines | ID | | | | |
| Checkout | media_id | num | since | until | | | |
| Location | name | addr | phone | | | | |
| Hold | media_id | num | name | until | queue | | |
| Stored_In | media_id | name | | | | | |
| Librarian | eid | ID | Pay | name | since | | |
| Video | title | year | director | rating | price | | |
| VideoMedia | media_id | title | year | | | | |

```
CREATE TABLE Status ( code INTEGER, description CHAR(30), PRIMARY KEY
(code) );
```

```
CREATE TABLE Media( media_id INTEGER, code INTEGER, PRIMARY KEY
(media_id),

FOREIGN KEY (code) REFERENCES Status );

CREATE TABLE Book(ISBNCHAR(14), title CHAR(128), author CHAR(64), year

INTEGER, dewey INTEGER, price REAL, PRIMARY KEY (ISBN) );

CREATE TABLE BookMedia( media_id INTEGER, ISBN CHAR(14), PRIMARY
KEY (media_id),

FOREIGN KEY (media_id) REFERENCES Media,

FOREIGN KEY (ISBN) REFERENCES Book);

CREATE TABLE Customer( ID INTEGER, name CHAR(64), addr CHAR(256),
DOB CHAR(10),

phone CHAR(30), username CHAR(16), password CHAR(32), PRIMARY KEY
(ID),

UNIQUE (username) );

CREATE TABLE Card( num INTEGER, fines REAL, ID INTEGER, PRIMARY
KEY (num),

FOREIGN KEY (ID) REFERENCES Customer );

CREATE TABLE Checkout( media_id INTEGER, num INTEGER, since CHAR(10),

until CHAR(10), PRIMARY KEY (media_id),

FOREIGN KEY (media_id) REFERENCES Media,

FOREIGN KEY (num) REFERENCES Card );

CREATE TABLE Location( name CHAR(64), addr CHAR(256), phone CHAR(30),

PRIMARY KEY (name) );

CREATE TABLE Hold( media_id INTEGER, num INTEGER, name CHAR(64),
until CHAR(10),

queue INTEGER, PRIMARY KEY (media_id, num),

FOREIGN KEY (name) REFERENCES Location,
FOREIGN KEY (num) REFERENCES Card,

FOREIGN KEY (media_id) REFERENCES Media );

CREATE TABLE Stored_In( media_id INTEGER, name char(64), PRIMARY KEY
(media_id),

FOREIGN KEY (media_id) REFERENCES Media ON DELETE CASCADE,

FOREIGN KEY (name) REFERENCES Location );

CREATE TABLE Librarian( eid INTEGER, ID INTEGER NOT NULL, Pay REAL,

Loc_name CHAR(64) NOT NULL, PRIMARY KEY (eid),

FOREIGN KEY (ID) REFERENCES Customer ON DELETE CASCADE,

FOREIGN KEY (Loc_name) REFERENCES Location(name) );

CREATE TABLE Video( title CHAR(128), year INTEGER, director CHAR(64),

rating REAL, price REAL, PRIMARY KEY (title, year) );

CREATE TABLE VideoMedia( media_id INTEGER, title CHAR(128), year
INTEGER,

PRIMARY KEY (media_id), FOREIGN KEY (media_id) REFERENCES Media,

FOREIGN KEY (title, year) REFERENCES Video );

INSERT INTO Customer(ID, name, addr, DOB, phone, username, password)
VALUES

(60201, 'Jason L. Gray', '2087 Timberbrook Lane, Gypsum, CO 81637',

'09/09/1958', '970-273-9237', 'jlgray', 'password1');

INSERT INTO Customer(ID, name, addr, DOB, phone, username, password)
VALUES

(89682, 'Mary L. Prieto', '1465 Marion Drive, Tampa, FL 33602',

'11/20/1961', '813-487-4873', 'mlprieto', 'password2');

INSERT INTO Customer(ID, name, addr, DOB, phone, username, password)
VALUES
```

(64937, 'Roger Hurst', '974 Bingamon Branch Rd, Bensenville, IL 60106',

'08/22/1973', '847-221-4986', 'rhurst', 'password3');

INSERT INTO Customer(ID, name, addr, DOB, phone, username, password) VALUES

(31430, 'Warren V. Woodson', '3022 Lords Way, Parsons, TN 38363',

'03/07/1945', '731-845-0077', 'wvwoodson', 'password4');

INSERT INTO Customer(ID, name, addr, DOB, phone, username, password) VALUES

(79916, 'Steven Jensen', '93 Sunny Glen Ln, Garfield Heights, OH 44125',

'12/14/1968', '216-789-6442', 'sjensen', 'password5');

INSERT INTO Customer(ID, name, addr, DOB, phone, username, password) VALUES

(93265, 'David Bain', '4356 Pooh Bear Lane, Travelers Rest, SC 29690',

'08/10/1947', '864-610-9558', 'dbain', 'password6');

INSERT INTO Customer(ID, name, addr, DOB, phone, username, password) VALUES

(58359, 'Ruth P. Alber', '3842 Willow Oaks Lane, Lafayette, LA 70507',

'02/18/1976', '337-316-3161', 'rpalber', 'password7');

INSERT INTO Customer(ID, name, addr, DOB, phone, username, password) VALUES

(88564, 'Sally J. Schilling', '1894 Wines Lane, Houston, TX 77002',

'07/02/1954', '832-366-9035', 'sjschilling', 'password8');

INSERT INTO Customer(ID, name, addr, DOB, phone, username, password) VALUES

(57054, 'John M. Byler', '279 Raver Croft Drive, La Follette, TN 37766',

'11/27/1954', '423-592-8630', 'jmbyler', 'password9');

INSERT INTO Customer(ID, name, addr, DOB, phone, username, password) VALUES

```
(49312, 'Kevin Spruell', '1124 Broadcast Drive, Beltsville, VA 20705',
'03/04/1984', '703-953-1216', 'kspruell', 'password10');
INSERT INTO Card(num, fines, ID) VALUES ( 5767052, 0.0, 60201);
INSERT INTO Card(num, fines, ID) VALUES ( 5532681, 0.0, 60201);
INSERT INTO Card(num, fines, ID) VALUES ( 2197620, 10.0, 89682);
INSERT INTO Card(num, fines, ID) VALUES ( 9780749, 0.0, 64937);
INSERT INTO Card(num, fines, ID) VALUES ( 1521412, 0.0, 31430);
INSERT INTO Card(num, fines, ID) VALUES ( 3920486, 0.0, 79916);
INSERT INTO Card(num, fines, ID) VALUES ( 2323953, 0.0, 93265);
INSERT INTO Card(num, fines, ID) VALUES ( 4387969, 0.0, 58359);
INSERT INTO Card(num, fines, ID) VALUES ( 4444172, 0.0, 88564);
INSERT INTO Card(num, fines, ID) VALUES ( 2645634, 0.0, 57054);
INSERT INTO Card(num, fines, ID) VALUES ( 3688632, 0.0, 49312);
INSERT INTO Location(name, addr, phone) VALUES ('Texas Branch',
'4832 Deercove Drive, Dallas, TX 75208', '214-948-7102');
INSERT INTO Location(name, addr, phone) VALUES ('Illinois Branch',
'2888 Oak Avenue, Des Plaines, IL 60016', '847-953-8130');
INSERT INTO Location(name, addr, phone) VALUES ('Louisiana Branch',
'2063 Washburn Street, Baton Rouge, LA 70802', '225-346-0068'); INSERT
INTO Status(code, description) VALUES (1, 'Available'); INSERT INTO
Status(code, description) VALUES (2, 'In Transit'); INSERT INTO
Status(code, description) VALUES (3, 'Checked Out'); INSERT INTO
Status(code, description) VALUES (4, 'On Hold'); INSERT INTO Media(
media_id, code) VALUES (8733, 1); INSERT INTO Media( media_id, code)
VALUES (9982, 1);
```

```
INSERT INTO Media( media_id, code) VALUES (3725, 1);

INSERT INTO Media( media_id, code) VALUES (2150, 1);

INSERT INTO Media( media_id, code) VALUES (4188, 1);

INSERT INTO Media( media_id, code) VALUES (5271, 2);

INSERT INTO Media( media_id, code) VALUES (2220, 3);

INSERT INTO Media( media_id, code) VALUES (7757, 1);
INSERT INTO Media( media_id, code) VALUES (4589, 1);

INSERT INTO Media( media_id, code) VALUES (5748, 1);

INSERT INTO Media( media_id, code) VALUES (1734, 1);

INSERT INTO Media( media_id, code) VALUES (5725, 1);

INSERT INTO Media( media_id, code) VALUES (1716, 4);

INSERT INTO Media( media_id, code) VALUES (8388, 1);

INSERT INTO Media( media_id, code) VALUES (8714, 1);

INSERT INTO Book(ISBN, title, author, year, dewey, price) VALUES

('978-0743289412', 'Lisey''s Story', 'Stephen King',

2006, 813, 10.0);

INSERT INTO Book(ISBN, title, author, year, dewey, price) VALUES

('978-1596912366', 'Restless: A Novel', 'William Boyd', 2006, 813, 10.0);

INSERT INTO Book(ISBN, title, author, year, dewey, price) VALUES

('978-0312351588', 'Beachglass', 'Wendy Blackburn', 2006, 813, 10.0);


INSERT INTO Book(ISBN, title, author, year, dewey, price) VALUES

('978-0156031561', 'The Places In Between', 'Rory Stewart', 2006, 910,

10.0);
```

```
INSERT INTO Book(ISBN, title, author, year, dewey, price) VALUES
('978-0060583002', 'The Last Season', 'Eric Blehm', 2006, 902, 10.0);
INSERT INTO Book(ISBN, title, author, year, dewey, price) VALUES
('978-0316740401', 'Case Histories: A Novel', 'Kate Atkinson', 2006, 813,
10.0);
INSERT INTO Book(ISBN, title, author, year, dewey, price) VALUES ('978-
0316013949', 'Step on a Crack', 'James Patterson, et al.',
2007, 813, 10.0);
INSERT INTO Book(ISBN, title, author, year, dewey, price) VALUES
('978-0374105235', 'Long Way Gone: Memoirs of a Boy Soldier', 'Ishmael
Beah', 2007, 916, 10.0);
INSERT INTO Book(ISBN, title, author, year, dewey, price) VALUES ('978-
0385340229', 'Sisters', 'Danielle Steel', 2006, 813, 10.0);
INSERT INTO BookMedia(media_id, ISBN) VALUES (8733, '978-0743289412');
INSERT INTO BookMedia(media_id, ISBN) VALUES (9982, '978-1596912366');
INSERT INTO BookMedia(media_id, ISBN) VALUES (3725, '978-1596912366');
INSERT INTO BookMedia(media_id, ISBN) VALUES (2150, '978-0312351588');
INSERT INTO BookMedia(media_id, ISBN) VALUES (4188, '978-0156031561');
INSERT INTO BookMedia(media_id, ISBN) VALUES (5271, '978-0060583002');
INSERT INTO BookMedia(media_id, ISBN) VALUES (2220, '978-0316740401');
INSERT INTO BookMedia(media_id, ISBN) VALUES (7757, '978-0316013949');
INSERT INTO BookMedia(media_id, ISBN) VALUES (4589, '978-0374105235');
INSERT INTO BookMedia(media_id, ISBN) VALUES (5748, '978-0385340229');
INSERT INTO Checkout(media_id, num, since, until) VALUES (2220, 9780749,
'02/15/2007', '03/15/2007');
```

```
INSERT INTO Video(title, year, director, rating, price) VALUES
('Terminator 2: Judgment Day', 1991, 'James Cameron', 8.3, 20.0);
INSERT INTO Video(title, year, director, rating, price) VALUES
('Raiders of the Lost Ark', 1981, 'Steven Spielberg', 8.7, 20.0);
INSERT INTO Video(title, year, director, rating, price) VALUES
('Aliens', 1986, 'James Cameron', 8.3, 20.0);
INSERT INTO Video(title, year, director, rating, price) VALUES
('Die Hard', 1988, 'John McTiernan', 8.0, 20.0);
INSERT INTO VideoMedia(media_id, title, year) VALUES
( 1734, 'Terminator 2: Judgment Day', 1991);
INSERT INTO VideoMedia(media_id, title, year) VALUES (
5725, 'Raiders of the Lost Ark', 1981);
INSERT INTO VideoMedia(media_id, title, year) VALUES (
1716, 'Aliens', 1986);
INSERT INTO VideoMedia(media_id, title, year) VALUES (
8388, 'Aliens', 1986);
INSERT INTO VideoMedia(media_id, title, year) VALUES (
8714, 'Die Hard', 1988);
INSERT INTO Hold(media_id, num, name, until, queue) VALUES
(1716, 4444172, 'Texas Branch', '02/20/2008', 1);
INSERT INTO Librarian(eid, ID, pay, Loc_name) Values
(2591051, 88564, 30000.00, 'Texas Branch');
INSERT INTO Librarian(eid, ID, pay, Loc_name) Values
(6190164, 64937, 30000.00, 'Illinois Branch');
INSERT INTO Librarian(eid, ID, pay, Loc_name) Values
```

```
    (1810386, 58359, 30000.00, 'Louisiana Branch');
    INSERT INTO Stored_In(media_id, name) VALUES(8733, 'Texas Branch');
    INSERT INTO Stored_In(media_id, name) VALUES(9982, 'Texas Branch');
    INSERT INTO Stored_In(media_id, name) VALUES(1716, 'Texas Branch');
    INSERT INTO Stored_In(media_id, name) VALUES(1734, 'Texas Branch');
    INSERT INTO Stored_In(media_id, name) VALUES(4589, 'Texas Branch');
    INSERT INTO Stored_In(media_id, name) VALUES(4188, 'Illinois Branch');
    INSERT INTO Stored_In(media_id, name) VALUES(5271, 'Illinois Branch');
    INSERT INTO Stored_In(media_id, name) VALUES(3725, 'Illinois Branch');
    INSERT INTO Stored_In(media_id, name) VALUES(8388, 'Illinois Branch');
    INSERT INTO Stored_In(media_id, name) VALUES(5748, 'Illinois Branch');
    INSERT INTO Stored_In(media_id, name) VALUES(2150, 'Louisiana Branch');
    INSERT INTO Stored_In(media_id, name) VALUES(8714, 'Louisiana Branch');
    INSERT INTO Stored_In(media_id, name) VALUES(7757, 'Louisiana Branch');
    INSERT INTO Stored_In(media_id, name) VALUES(5725, 'Louisiana Branch');

SELECT C.ID, C.name, C.addr, C.DOB, C.phone, C.username,
nvl((SELECT 'Librarian'
FROM Librarian L
WHERE L.ID = C.ID), 'Customer') AS role
FROM Customer C
WHERE C.username = <user input> AND C.password = <user input>; /*
Book search for customers */
SELECT B.ISBN, B.title, B.author, B.year,
(SELECT COUNT(*)
FROM BookMedia BM

WHERE BM.ISBN = B.ISBN AND BM.code = 1) AS num_available

FROM Book B

WHERE B.title LIKE '%<user input>%' AND B.author LIKE '%<user input>%' AND

B.year <= <user input> AND B.year >= <user input>;

/* Find all copies of a book (used for placing holds or viewing detailed

information). */

SELECT BM.media_id, S.description,

nvl((SELECT SI.name

FROM Stored_In SI

WHERE SI.media_id = BM.media_id), 'none') AS name

FROM BookMedia BM, Media M, Status S

WHERE BM.ISBN = <user input> AND M.media_id = BM.media_id AND S.code =
M.code;

/* Video search for customers */

SELECT V.title, V.year, V.director, V.rating

(SELECT COUNT(*)

FROM VideoMedia VM

WHERE VM.ID = V.ID AND VM.code = 1) AS num_available

FROM Video V

WHERE V.title LIKE '%<user input>%' AND V.year <= <user input> AND V.year <= <user
input>

AND V.director LIKE '%<user input>%' AND V.rating >= <user input>; /*

Find all copies of a video (used for placing holds or viewing detailed

information). */

SELECT VM.media_id, S.description,
nvl((SELECT SI.name
FROM Stored_In SI
WHERE SI.media_id = VM.media_id), 'none') AS name
FROM VideoMedia VM, Media M, Status S
WHERE VM.title = <user input> AND VM.year = <user input> AND
M.media_id = VM.media_id AND S.code = M.code; /* Find the status of
a given media item */
SELECT S.description
FROM Status S, Media M
WHERE S.code = M.code AND M.media_id = <user input>; /*
Create a new Hold */
INSERT INTO Hold(media_id, num, name, until, queue) VALUES
(<user input>, <user input>, <user input>, <user input>, nvl((SELECT
MAX(H.queue)
FROM Hold H
WHERE H.media_id = <user input>), 0) + 1 );
/* Cancel Hold, Step 1: Remove the entry from hold */
DELETE FROM Hold
WHERE media_id = <user input> AND num = <user input> /*
Cancel Hold, Step 2: Update queue for this item */ UPDATE
Hold
SET queue = queue-1
WHERE media_id = <user input> AND queue > <user input>; /*
Functions needed to view information about a customer */ /*
View the customer's card(s) */ SELECT CR.num, CR.fines
```

FROM Card CR

WHERE CR.ID = <user input>;

/* View media checked out on a given card */

SELECT B.title, B.author, B.year, BM.media_id, CO.since, CO.until

FROM Checkout CO, BookMedia BM, Book B

WHERE CO.num = <user input> AND CO.media_id = BM.media_id AND B.ISBN = BM.ISBN

UNION

SELECT V.title, V.director, V.year, VM.media_id, CO.since, CO.until

FROM Checkout CO, VideoMedia VM, Book B

WHERE CO.num = <user input> AND CO.media_id = VM.media_id AND

VM.title = V.title AND VM.year = V.year;

/* View media currently on hold for a given card */

SELECT B.title, B.author, B.year, BM.media_id, H.until, H.queue, SI.name

FROM Hold H, BookMedia BM, Book B, Stored_In SI

WHERE H.num = <user input> AND H.media_id = BM.media_id AND B.ISBN = BM.ISBN

AND SI.media_id = H.media_id

UNION

SELECT V.title, V.director, V.year, VM.media_id, H.until, H.queue, SI.name

FROM Hold H, VideoMedia VM, Book B, Stored_In SI

WHERE H.num = <user input> AND H.media_id = VM.media_id AND

VM.title = V.title AND VM.year = V.year AND SI.media_id = H.media_id; /*

View the total amount of fines the customer has to pay */ SELECT

SUM(CR.fines)

FROM Card CR

```
WHERE CR.ID = <user input>;

/* *\

Functions reserved for librarians

\* */

/* Add new customer */

INSERT INTO Customer(ID, name, addr, DOB, phone, username, password) VALUES

(<user input>, <user input>, <user input>, <user input>, <user input>, <user input>, <user

input>, );

/* Find a customer */

SELECT C.ID, C.name, C.addr, C.DOB, C.phone, C.username,

nvl((SELECT 'Librarian'

FROM Librarian L

WHERE L.ID = C.ID), 'Customer') AS role

FROM Customer C

WHERE C.username = <user input> AND C.name LIKE '%<user input>%'; /*

Add new card and assign it to a customer */

INSERT INTO Card(num, fines, ID) VALUES ( <user input>, 0, <user input>); /*

Create an entry in Checkout */

INSERT INTO Checkout(media_id, num, since, until) VALUES

(<user input>, <user input>, <user input>, <user input>); /*

Remove the entry for Stored_In */

DELETE FROM Stored_In

WHERE media_id = <user input>;

/* Change the status code of the media */

UPDATE Media
```

SET code = <user input>

WHERE media_id = <user input>;

/* Remove the entry from Checkout */

DELETE FROM Checkout

WHERE media_id = <user input>;

/* Create the entry in Stored_In */

INSERT INTO Stored_In(media_id, name) VALUES (<user input>, <user input>); /*

Find the next Hold entry for a given media */ SELECT H.num, H.name, H.until

FROM Hold H

WHERE H.queue = 1 AND H.media_id = <user input>;

/* Change the Stored_In entry to the target library branch */

UPDATE Stored_In

SET name = <user input>

WHERE media_id = <user input>;

/* Find the customer that should be notified about book arrival */

SELECT C.name, C.phone, CR.num FROM Customer C, Card CR,

Hold H

WHERE H.queue = 1 AND H.name = <user input> AND H.media_id = <user input> AND

CR.num = H.num AND C.ID = CR.ID;

/* Add a new entry into the Book table */

INSERT INTO Book(ISBN, title, author, year, dewey, price) VALUES

(<user input>, <user input>, <user input>, <user input>, <user input>,

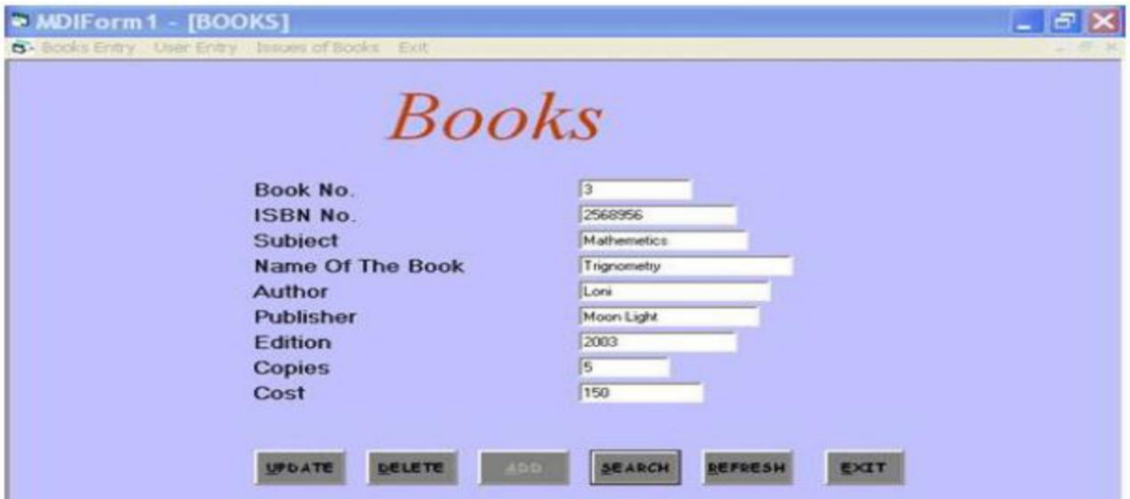
<user input>);

/* Add a new entry into the Video table */

```
INSERT INTO Video(title, year, director, rating, price) VALUES (<user
input>, <user input>, <user input>, <user input>, <user input>); /* Add a
new Media object */
INSERT INTO Media( media_id, code) VALUES (<user input>, 1); /*
Add a new BookMedia object */
INSERT INTO BookMedia(media_id, ISBN) VALUES (<user input>, <user input>); /*
Add a new VideoMedia object */
INSERT INTO VideoMedia(media_id, title, year) VALUES
(<user input>, <user input>, <user input>);
/* Remove an entry from the BookMedia table */
DELETE FROM BookMedia WHERE media_id =
<user input>;
/* Remove an entry from the VideoMedia table */
DELETE FROM VideoMedia WHERE media_id =
<user input>;
/* Remove an entry from the Media table */
DELETE FROM Media
WHERE media_id = <user input>;
/* Remove an entry from the Book table */
DELETE FROM Book
WHERE ISBN = <user input>;
/* Remove an entry from the Video table */
DELETE FROM Video
WHERE title = <user input> AND year = <user input>; /*
Update the customer's fines */ UPDATE Card
```

```
SET fines = <user input>

WHERE num = <user input>
```



Books

| | |
|---|---|
| Book No. | 3 |
| ISBN No. | 2568956 |
| Subject | Mathemetics |
| Name Of The Book | Trignometry |
| Author | Loni |
| Publisher | Moon Light |
| Edition | 2003 |
| Copies | 5 |
| Cost | 150 |