G. H. Raisoni College of Engineering & Management,
Wagholi, Pune – 412207

RAISONI GROUP
a vision beyond

Department of Master of Computer Applications

Lab Manual (2023-24)

Pattern-2020

Class : FY Term: II

Oracle &PL/SQL - (MCAP203)

Faculty Name: 1.Mrs. Supriya Bankar
2.Mrs. Anita Pisote

G. H. Raisoni College of Engineering & Management,
Wagholi, Pune – 412207

RAISONI GROUP
a vision beyond

Department: MCA

Course Details

Course : Oracle &PL/SQL Lab (MCAP203)

Class: MCA | **Division: A& B**
Internal Marks:25 | **External marks:25**
Credits : 2 | **Pattern : 2020**

ORACLE and PL/SQLLAB(MCAP203)	
Course Outcome	
CO1	1. Understanding basic database security and object privileges.
CO2	2. Using the powerful set of built-in SQL functions
CO3	3. Protecting the database and extending the basic data model using declarative constraints.
CO4	4. Performing advanced SQL queries such as grouping and Join, Sub queries, PL/SQL Blocks, functions.
CO5	5. Use the PL/SQL code constructs of IF-THEN-ELSE and LOOP types as well as syntax and command functions.

List of Experiment

Sr.No.	Name of Experiment	CO Mapped
Week 1 **1**	1. Create a table called Employee with the following structure. Name Type Empno Number ,Ename Varchar2(20), Job Varchar2(20) Mgr Number ,Sal Number. a. Add a column commission with domain to the Employee table. b. Insert any five records into the table. c. Update the column details of job d. Rename the column of Employ table using alter command. e. Delete the employee whose empno is19.	
2	2. Create department table with the following structure. Deptno Number, Deptname Varchar2(20), location Varchar2(20) a. Add column designation to the department table. b. Insert values into the table. c. List the records of emp table grouped by deptno. d. Update the record where deptno is 9. e. Delete any column data from the table	
3	3. Create a table called Customer table Name Type Cust name Varchar2(20) ,Cust street Varchar2(20), Cust city varchar2(20) a. Insert records into the table. b. Add salary column to the table. c. Alter the table column domain. d. Drop salary column of the customer table. e. Delete the rows of customer table whose custcity is 'hyd'	
4	4. Create a table called branch table. Name Type Branch name Varchar2(20) ,Branch city Varchar2(20) asserts Number. a. Increase the size of data type for asserts to the branch. b. Add and drop a column to the branch table. c. Insert values to the table. d. Update the branch name column	

	e. Delete any two columns from the table	
5	5. Create a table called sailor table Name Type Sid Number, Sname Varchar2(20) ,rating Varchar2(20) a. Add column age to the sailor table. b. Insert values into the sailortable. c. Delete the row with rating>8. d. Update the column details of sailor. e. Insert null values into the table.	
6	6. Create a table called reserves table name type Boatid integer, sid integer, day integer a. Insert values into the reserves table. b. Add column time to the reserves table. c. Alter the column day data type to date. d. Drop the column time in the table. e. Delete the row of the table with above given conditions	
Week 2 **7**	1. a. Create a user and grant all permissions to the user. b. Insert the any three records in the employee table and use rollback. Check the result. c. Add primary key constraint and not null constraint to the employee table. d. Insert null values to the employee table and verify the result.	**CO1**
8	a. Create a user and grant all permissions to the user. b. Insert the any three records in the employee table and use rollback. Check the result. c. Add primary key constraint and not null constraint to the employee table. d. Insert null values to the employee table and verify the result.	**CO1,CO3**
9	a. Create a user and grant all permissions to the user. b. Insert the any three records in the employee table and use rollback. Check the result. c. Add primary key constraint and not null constraint to the employee table. d. Insert null values to the employee table and verify the result.	**CO1,CO3**
10	a. Create a user and grant all permissions to the user. b. Insert values in the department table and use commit.	**CO1,CO3**

	c. Add constraints like unique and not null to the department table. d. Insert repeated values and null values into the table.	
11	a. Create a user and grant all permissions to the user. b. Insert values into the table and use commit. c. Delete any three records in the department table and use rollback. d. Add constraint primary key and foreign key to the table.	**CO1,CO3**
12	a. Create a user and grant all permissions to theuser. b. Insert records in the sailor table and use commit. c. Add save point after insertion of records and verify save point. d. Add constraints not null and primary key to the sailor table.	**CO1,CO3**
Week 3 **13**	1.a. By using the group by clause, display the enames who belongs to deptno 10 along with average salary. b. Display lowest paid employee details under each department. c. Display number of employees working in each department and their department number. d. Using built in functions, display number of employees working in each department and their department name from dept table. Insert deptname to dept table and insert deptname foreach row, do the required thing specified above. e. List all employees which start with either B or C. f. Display only these ename of employees where the maximum salary is greater than or equal to 5000.	
14	2. a. Calculate the average salary for each different job. b. Show the average salary of each job excluding manager. c. Show the average salary for all departments employing more than three people. d. Display employees who earn more than the lowest salary in department 30 e. Show that value returned by sign (n) function. f. How many days between day of birth to current date	
15	3. a. Show that two substring as single string. b. List all employee names, salary and 15% rise in salary. c. Display lowest paid employee details under each manager d. Display the average monthly salary bill for each deptno. e. Show the average salary for all departments employing more than two people. f. By using the group by clause, display the eid who belongs to deptno 05 along with average salary.	

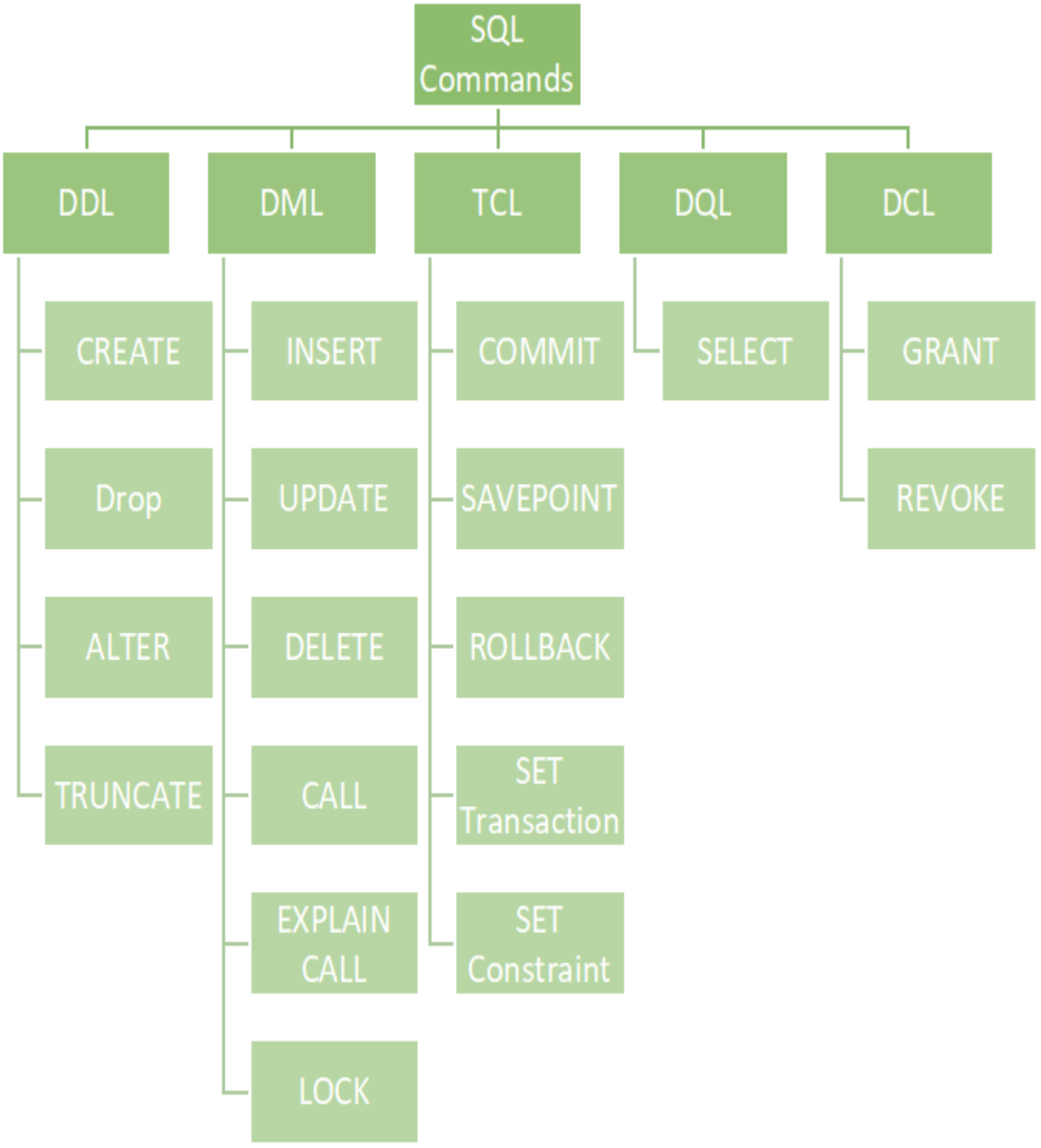
16	4. a. Count the number of employees in department20 b. Find the minimum salary earned by clerk. c. Find minimum, maximum, average salary of all employees. d. List the minimum and maximum salaries for each job type. e. List the employee names in descending order. f. List the employee id, names in ascending orderby empid.	
17	5. a. Find the sids ,names of sailors who have reserved all boats called "INTERLAKE Find the age of youngest sailor who is eligible to vote for each rating level with at least two such sailors. b. Find the sname , bid and reservation date for each reservation. c. Find the ages of sailors whose name begin and end with B and has at least 3 characters. List in alphabetic order all sailors who have reserved red boat. e. Find the age of youngest sailor for each rating level	
18	6. a. List the Vendors who have delivered products within 6 months from order date. b. Display the Vendor details who have supplied both Assembled and Subparts. c. Display the Sub parts by grouping the Vendor type (Local or NonLocal). d. Display the Vendor details in ascending order. e. Display the Sub part which costs more than any of the Assembled parts. f. Display the second maximum cost Assembled part	
Week 4 **19**	1. a. Write a PL/SQL program to swap two numbers. b. Write a PL/SQL program to find the largest of three numbers.	
20	2. a. Write a PL/SQL program to find the total and average of 6 subjects and display thegrade. b. Write a PL/SQL program to find the sum of digits in a given number.	
21	3. a. Write a PL/SQL program to display the number in reverse order. b. Write a PL / SQL program to check whether the given number is prime or not.	
22	4. a. Write a PL/SQL program to find the factorial of a given number. b. Write a PL/SQL code block to calculate the area of a circle for a value of radius varying from 3 to 7. Store the radius and the corresponding	

	values of calculated area in an empty table named areas, consisting of two columns radius and area.	
23	5. Write a PL/SQL program to accept a string and remove the vowels from the string. (When „hello" passed to the program it should display „Hll" removing e and o from the world Hello).	
24	Other Example 1- Cursor	**CO3, CO4**
25	Other Example 2 – Trigger	

SQL Introduction

Standard language for querying and manipulating data Type of SQL statements are divided into five different categories:

- Data definition language (DDL),
- Data manipulation language (DML),
- Data Control Language (DCL),
- Transaction Control Statement (TCS),
- Session Control Statements (SCS).



Data Definition Language (DDL)

Data definition statement are use to define the database structure or table.

Statement	Description
CREATE	Create new database/table.
ALTER	Modifies the structure of database/table.
DROP	Deletes a database/table.
TRUNCATE	Remove all table records including allocated table spaces
RENAME	Rename the database/table.

Data Manipulation Language (DML)

Data manipulation statements are used for managing data within table object.

Statement	**Description**
SELECT	Retrieve data from the table.
INSERT	Insert data into a table.
UPDATE	Updates existing data with new data within a table.
DELETE	Deletes the records/ rows from the table.
MERGE	MERGI (also called UPSERT) statements to INSERT new records or UPDATE existing records depending on condition matches or not.
LOCK TABLE	LOCK TABLE, statement to lock one of more tables in a specified mode. Table access denied to a other users for the duration of your table operation.

Data Control Language (DCL)

Data control statement are use to give privileges to access limited data.

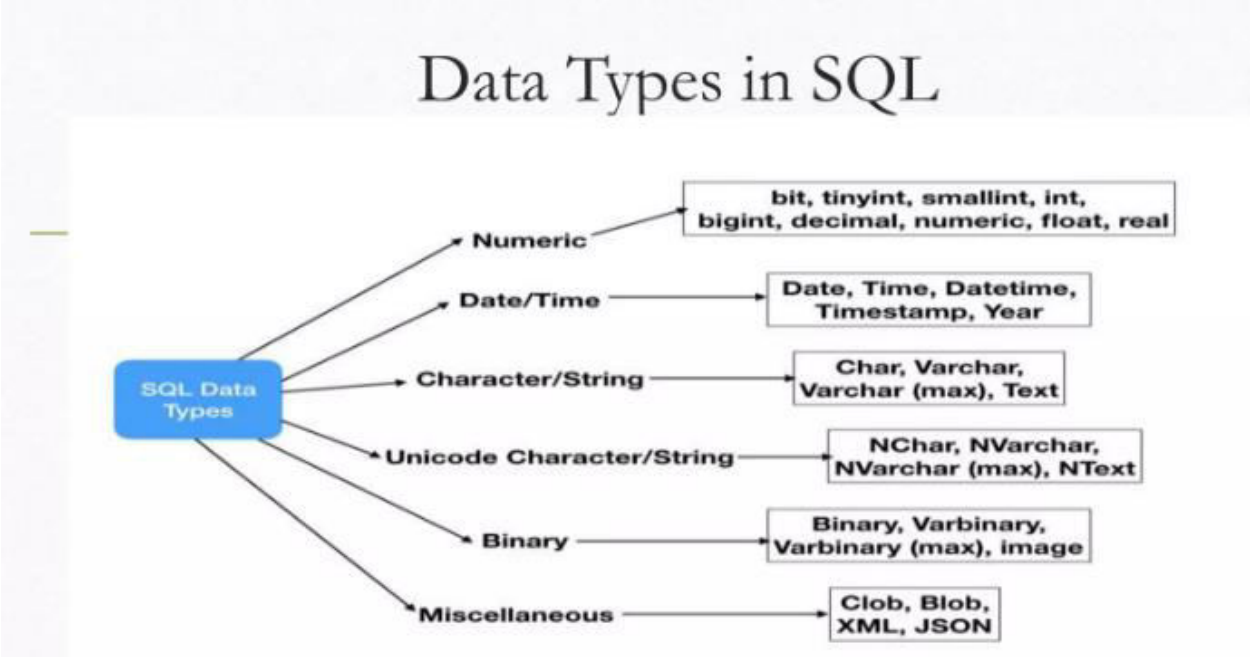
Statement	**Description**
GRANT	Gives privileges to user for accessing database data.
REVOKE	Take back for given privileges.
ANALYZE	ANALYZE statement to collect statistics information about index, cluster, table.
AUDIT	To track the occurrence of a specific SQL. Statement or all SQL statements during the user sessions.
COMMENT	Write comment to the data table.

Transaction Control Language (TCL)

Transaction control statements are used to store the changes permanently into database.

Statement	Description
COMMIT	Permanent work saves into database.
ROLLBACK	Restore database to original form since the last COMMIT.
SAVEPOINT	Create SAVEPOINT for later use ROLLBACK the new changes.
SET TRANSACTION	SET TRANSACTION command set the transaction properties

Data Types in SQL:



DDL Commands-Create Table

–Used for creating a table

```
Syntax:
CREATE TABLE table_name(
column_name1 datatype(size),
column_name2 datatype(size) ... );

Example:
SQL> CREATE TABLE users_info( no NUMBER(3,0), name VARCHAR(30), address
VARCHAR(70), contact_no VARCHAR(12) );
Table created.
```

DDL Commands-Drop Table

-Used for deleting or removing a table

Syntax

```
DROP TABLE table_name;
```

Example

```
SQL> DROP TABLE users_info;
 Table dropped.
```

DDL Commands-Alter Table

-Used to add, manage or update table structure

ALTER TABLE Statement can do the following:

- TABLE RENAME
- ADD NEW COLUMN IN TABLE
- MODIFY EXISTING COLUMN IN TABLE
- RENAME COLUMN IN TABLE
- DROP THE EXISTING COLUMN IN TABLE

SQL TABLE RENAME

You can rename the SQL table using this syntax,

Syntax

```
ALTER TABLE table_name RENAME TO new_table_name;
```

Example

```
SQL> ALTER TABLE userinfo RENAME TO user_info;
Table altered.
```

SQL ADD NEW COLUMN IN TABLE

You can add new column in table using this syntax,

Syntax

```
ALTER TABLE table_name ADD column_name datatype[(size)];
```

Example

```
SQL> ALTER TABLE user_info ADD state VARCHAR2(12);
Table altered.
```

SQL ADD MULTIPLE COLUMN IN TABLE

You can add multiple column in table at a time using this syntax,

Syntax

```
ALTER TABLE table_name ADD ( column_name1 datatype[(size)],
column_name2 datatype[(size)], ... );
```

Example

```
SQL> ALTER TABLE user_info ADD (city VARCHAR2(30), country
VARCHAR2(30) );
Table altered.
```

SQL RENAME COLUMN IN TABLE

You can rename the existing column in table using this syntax,

Syntax

```
ALTER TABLE table_name RENAME COLUMN old_column_name TO
new_column_name;
```

Example

```
SQL> ALTER TABLE user_info RENAME COLUMN no TO sno;
Table altered.
```

SQL DROP THE COLUMN IN TABLE

You can drop existing column in table using this syntax,

Syntax

```
ALTER TABLE table_name DROP COLUMN column_name;
```

Example

```
SQL> ALTER TABLE user_info DROP COLUMN country;
Table altered.
```

DML Commands-Insert

INSERT INTO statement

Using INSERT INTO statement to insert record into database.
When inserting data into table no need to specify the column names if values is table structure wise (column wise).

Syntax

```
INSERT INTO table_name VALUES (value1, value2, value3, ...);
```

Example

```
SQL> INSERT INTO users_info VALUES (1, 'Opal Kole', '63 street Ct.', '000-444-7847');
1 row created.
```

When you inserting data into table and you haven't know table structure you must specify the column name.

Syntax

```
INSERT INTO table_name [ (column_name1, column_name2, ...) ] VALUES (value1, value2, ...);
```

Example

```
SQL> INSERT INTO users_info (name, address, no, contact_no) VALUES ('Beccaa Moss', '2500 green
city.', 3, '000-444-7142');
1 row created.
```

INSERT ALL statement

Using INSERT ALL statement to insert more then one records into table.We can insert more then one record in single SQL INSERT statement.

Syntax

```
INSERT ALL INTO table_name [ (column_name1, column_name2, ...) ] VALUES (record1_value1,
record1_value2, ...) INTO table_name [ (column_name1, column_name2, ...) ] VALUES
(record2_value1, record2_value2, ...) INTO table_name [ (column_name1, column_name2, ...) ]
VALUES (record3_value1, record3_value2, ...) .... SELECT * FROM dual;
```

Example

```
SQL> INSERT ALL INTO users_info (no, name, address, contact_no) VALUES (4, 'Paul Singh', '1343
Prospect St', '000-444-7141') INTO users_info (no, name, address, contact_no) VALUES (5, 'Ken
Myer', '137 Clay Road', '000-444-7084') INTO users_info (no, name, address, contact_no) VALUES
(6, 'Jack Evans', '1365 Grove Way', '000-444-7957') INTO users_info (no, name, address,
contact_no) VALUES (7, 'Reed Koch', '1274 West Street', '000-444-4784') SELECT * FROM dual; 4
rows created.
```

SQL Multiple Row Insert into Table

You can insert multiple record by this way first you execute INSERT INTO statement with & sign with column name. If you want to add another record you just execute forward slash (/) to again execute last statement automatically and you can insert new data again.

```
SQL> INSERT INTO users_info VALUES (&no,'&name','&address',&contact_no);
 Enter value for no: 8
Enter value for name: 'Gabe Hee'
Enter value for address: '1220 Dallas Drive'
Enter value for contact_no: '000-444-4584'
old 1: INSERT INTO users_info VALUES (&no, &name, &address, &contact_no)
new 1: INSERT INTO users_info VALUES (8, 'Gabe Hee', '1220 Dallas Drive', '000-444-4584')
1 row created.
SQL> /
Enter value for no: 9
Enter value for name: 'Ben Mares'
Enter value for address: '101 Candy Road'
Enter value for contact_no: '000-444-5484'
old 1: INSERT INTO users_info VALUES (&no, &name, &address, &contact_no)
new 1: INSERT INTO users_info VALUES (9, 'Ben Mares', '101 Candy Road', '000-444-5484')
1 row created.
```

SQL Insert Data only in specified COLUMNS

You can insert data in specific columns. When you write INSERT statement you have to specify column name for inserting only that column data into table.

Syntax

```
INSERT INTO Table_Name (specific_column_name1, ...) VALUES
(value1,...);
```

Example

```
SQL> INSERT INTO users_info(no, name) VALUES (10, 'Sariya Vargas');
1 row created.
```

INSERT INTO SELECT Statement

INSERT INTO SELECT Statement is used to insert data into a table from another table.

Syntax

```
INSERT INTO new_table_name [(column_name1,column_name2,...)] SELECT
column_name1, column_name1 ... FROM another_table_name [WHERE
condition];
```

Example

```
SQL> CREATE TABLE demo_tbl( no NUMBER(3), name VARCHAR2(50) ); Table
created. SQL> INSERT INTO demo_tbl (no, name) SELECT no, name FROM
users_info; 10 rows created.
```

DML Commands-Update Command

SQL UPDATE statement to update table records with in database. You can update all table row or update data only matching condition using WHERE clause.

SQL UPDATE All Rows

Syntax

```
UPDATE table_name SET column_name1 = value1, column_name2 = value2, ..Where
Condition;
```

Example Statement

```
SQL> UPDATE demo1 SET contact_no = 444;
10 rows updated.
```

.

DML Commands-Delete Command

SQL DELETE Statement is used to delete one or more then one row removed from table.

SQL DELETE Query use following two way,

- Remove all TABLE rows
- Remove only specific TABLE row/rows

Remove only specific TABLE row

Syntax

```
DELETE FROM table_name [ WHERE condition ] [ LIMIT number ];
```

Example

```
SQL> DELETE FROM demo1 WHERE NO = 10;
1 row deleted.
```

Remove all TABLE rows

Syntax

```
DELETE FROM table_name;
```

Example

```
SQL> DELETE FROM demo1;
9 rows deleted.
```

Assignment 1

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

Name Type

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

a. Add a column commission with domain to the Employee table.
b. Insert any five records into the table.
c. Update the column details of job
d. Rename the column of Employ table using alter command.
e. Delete the employee whose empno is19.

Solution:

Create table Employee(Empno Number(5) primary key, Ename Varchar2(20),Job Varchar2(20),Mgr Number(5), Salary Number(5));

Desc Employee;

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
EMPLOYEE	EMPNO	Number	-	5	0	1	-	-	-
	ENAME	Varchar2	20	-	-	-	✓	-	-
	JOB	Varchar2	20	-	-	-	✓	-	-
	MGR	Number	-	5	0	-	✓	-	-
	SALARY	Number	-	5	0	-	✓	-	-
									1 - 5

a. Add a column commission with domain to the Employee table.

Alter Table Employee Add (Commission number(5));

Desc Employee;

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
EMPLOYEE	EMPNO	Number	-	5	0	1	-	-	-
	ENAME	Varchar2	20	-	-	-	✓	-	-
	JOB	Varchar2	20	-	-	-	✓	-	-
	MGR	Number	-	5	0	-	✓	-	-
	SALARY	Number	-	5	0	-	✓	-	-
	COMMISSION	Number	-	5	0	-	✓	-	-
									1 - 6

b. Insert any five records into the table.

Insert into Employee values(1001,'Ramesh','Accountant', 105,25000,1000);

Insert into Employee values(1002,'Aaditi','Developer', 106,35000,1000);

Insert into Employee values(1003,'Yuvaraj','Accountant', 107,30000,1000);

Insert into Employee values(1004,'Shourya','HR', 108,40000,1000);

Insert into Employee values(1005,'Anuja','Tester', 109,45000,1000);

Select * from Employee;

EMPNO	ENAME	JOB	MGR	SALARY	COMMISION
1001	Ramesh	Accountant	105	25000	1000
1002	Aaditi	Developer	106	35000	1000
1003	Yuvaraj	Accountant	107	30000	1000
1004	Shourya	HR	108	40000	1000
1005	Anuja	Tester	109	45000	1000

c. Update the column details of job

Update Employee SET Job = 'Analyst' Where Empno = 1003;

Select * from Employee;

EMPNO	ENAME	JOB	MGR	SALARY	COMMISSION
1001	Ramesh	Accountant	105	25000	1000
1002	Aaditi	Developer	106	35000	1000
1003	Yuvaraj	Analyst	107	30000	1000
1004	Shourya	HR	108	40000	1000
1005	Anuja	Tester	109	45000	1000

d. Rename the column of Employ table using alter command.

Alter table Employee Rename column Commission TO Profit;

Select * From Employee;

EMPNO	ENAME	JOB	MGR	SALARY	PROFIT
1001	Ramesh	Accountant	105	25000	1000
1002	Aaditi	Developer	106	35000	1000
1003	Yuvaraj	Analyst	107	30000	1000
1004	Shourya	HR	108	40000	1000
1005	Anuja	Tester	109	45000	1000

e. Delete the employee whose empno is19.

Delete from Employee where Empno=19;

```
Ans=> 0 row(s) deleted.
```

Assignment 2

2. Create department table with the following structure.

Name Type

Deptno Number

Deptname Varchar2(20)

location Varchar2(20)

a. Add column designation to the department table.

b. Insert values into the table.

c. List the records of emp table grouped by deptno.

d. Update the record where deptno is9.

e. Delete any column data from the table

Solution :

Create table Department(Deptno number(5) primary key, DeptName varchar2(20),Location varchar2(20));

Desc Department;

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
DEPARTMENT	DEPTNO	Number	-	5	0	1	-	-	-
	DEPTNAME	Varchar2	20	-	-	-	✓	-	-
	LOCATION	Varchar2	20	-	-	-	✓	-	-
									1 - 3

a. Add column designation to the department table.

Alter Table Department Add (Designation varchar2(20));

Desc Department;

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
DEPARTMENT	DEPTNO	Number	-	5	0	1	-	-	-
	DEPTNAME	Varchar2	20	-	-	-	✓	-	-
	LOCATION	Varchar2	20	-	-	-	✓	-	-
	DESIGNATION	Varchar2	20	-	-	-	✓	-	-
									1 - 4

b. Insert values into the table.

Insert into Department values(201,'Account','Ground Floor','CA');

Insert into Department values(202,'Sales','First Floor','Accountant');

Insert into Department values(203,'Purchase','Second Floor','Accountant');

Insert into Department values(204,'HR','Third Floor','HR');

Insert into Department values(205,'Salary','Four Floor',1005,'Cosultant');

Select * from Department;

DEPTNO	DEPTNAME	LOCATION	DESIGNATION
201	Account	Ground Floor	CA
202	Sales	First Floor	Accountant
203	Purchase	Second Floor	Accountant
204	HR	Third Floor	HR
205	Salary	Four Floor	Cosultant

c. List the records of emp table grouped by deptno.

Select Deptno,DeptName

from Department

group by Deptno,Deptname;

DEPTNO	DEPTNAME
202	Sales
204	HR
203	Purchase
201	Account
205	Salary

d. Update the record where deptno is9.
Update Department SET Designation = 'Analyst' Where Deptno = 205;

```
1 row(s) updated.
```

Select * from Department;

DEPTNO	DEPTNAME	LOCATION	DESIGNATION
201	Account	Ground Floor	CA
202	Sales	First Floor	Accountant
203	Purchase	Second Floor	Accountant
204	HR	Third Floor	HR
205	Salary	Four Floor	Analyst

e. Delete any column data from the table

Alter table Department Drop(Designation);

Table altered

Select * from Department;

DEPTNO	DEPTNAME	LOCATION
201	Account	Ground Floor
202	Sales	First Floor
203	Purchase	Second Floor
204	HR	Third Floor
205	Salary	Four Floor

Assignment 3

3. Create a table called Customer table

Name Type

Cust name Varchar2(20)

Cust street Varchar2(20)

Cust city Varchar2(20)

a. Insert records into the table.

b. Add salary column to the table.

c. Alter the table column domain.

d. Drop salary column of the customer table.

e. d the rows of customer table whose custcityis 'hyd'.

Solution: Create table Customer(CName varchar2(20),Street varchar2(20),City varchar2(20));

Desc Customer;

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
CUSTOMER	CNAME	Varchar2	20	-	-	-	✓	-	-
	STREET	Varchar2	20	-	-	-	✓	-	-
	CITY	Varchar2	20	-	-	-	✓	-	-

1 - 3

b. Add salary column to the table.

Alter Table Customer Add (Salary number(5))

```
Table altered.
```

Desc Customer;

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
CUSTOMER	CNAME	Varchar2	20	-	-	-	✓	-	-
	STREET	Varchar2	20	-	-	-	✓	-	-
	CITY	Varchar2	20	-	-	-	✓	-	-
	SALARY	Number	-	5	0	-	✓	-	-

1 - 4

a. Insert records into the table.

```
insert into Customer values('Shyam','Shastri Road','Pune',60000);

insert into Customer values('Ramesh','Tilak Road','Pune',40000);

insert into Customer values('Aaditi','Tilak Road','Pune',25000);

insert into Customer values('Anaya','J M Road','Pune',52000);
```

Select * from Customer;

CNAME	STREET	CITY	SALARY
Ram	J M Road	Pune	50000
Shyam	Shastri Road	Pune	60000
Ramesh	Tilak Road	Pune	40000
Aaditi	Tilak Road	Pune	25000
Anaya	J M Road	Pune	52000

c. Alter the table column domain.
Alter table Column domain.

ALTER table Customer Modify (Salary number(8) not null);

Table altered.-->

Alter table Customer rename column Salary To Income;

CNAME	STREET	CITY	INCOME
Ram	J M Road	Pune	50000
Shyam	Shastri Road	Pune	60000
Ramesh	Tilak Road	Pune	40000
Aaditi	Tilak Road	Pune	25000
Anaya	J M Road	Pune	52000

d. Drop salary column of the customer table.
Alter table Customer drop column Income;

CNAME	STREET	CITY
Ram	J M Road	Pune
Shyam	Shastri Road	Pune
Ramesh	Tilak Road	Pune
Aaditi	Tilak Road	Pune
Anaya	J M Road	Pune

e. d the rows of customer table whose custcityis ‘hyd’insert into Customer values('Ram','J M Road','Pune',50000);

Delete from Customer where city='hyd';

```
0 row(s) deleted.
```

Assignment 4

Create a table called branch table.
Name Type
Branch name Varchar2(20)
Branch city Varchar2(20)
asserts Number
a. Increase the size of data type for asserts to the branch.
b. Add and drop a column to the branch table.
c. Insert values to the table.
d. Update the branch namecolumn
e. Delete any two columns from the table

Create table Branch(BranchName varchar2(20),City varchar2(20),Asserts number(5));

Desc Branch;

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
BRANCH	BRANCHNAME	Varchar2	20	-	-	-	✓	-	-
	CITY	Varchar2	20	-	-	-	✓	-	-
	ASSERTS	Number	-	5	0	-	✓	-	-
									1 - 3

a. Increase the size of data type for asserts to the branch.
Alter table Branch Modify(Asserts Number(10));

Desc Branch;

b. Add and drop a column to the branch table.
Add and Drop column to table.

Alter Table Branch Add (Location varchar2(20));

```
Table altered.
```

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
BRANCH	BRANCHNAME	Varchar2	20	-	-	-	✓	-	-
	CITY	Varchar2	20	-	-	-	✓	-	-
	ASSERTS	Number	-	10	0	-	✓	-	-
	LOCATION	Varchar2	20	-	-	-	✓	-	-
									1 - 4

Desc Branch;

Alter table Branch Drop(Location);

Desc Branch;

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
BRANCH	BRANCHNAME	Varchar2	20	-	-	-	✓	-	-
	CITY	Varchar2	20	-	-	-	✓	-	-
	ASSERTS	Number	-	10	0	-	✓	-	-

1 - 3

c. Insert values to the table.

insert into Branch values('P-001','Pune',1000000);

insert into Branch values('P-002','Mumbai',1000000);

insert into Branch values('P-003','Kharadi',1000000);

insert into Branch values('P-004','Wagholi',1000000);

insert into Branch values('P-005','Hinjewadi',1000000);

select * from Branch;

BRANCHNAME	CITY	ASSERTS
P-001	Pune	1000000
P-002	Mumbai	1000000
P-003	Kharadi	1000000
P-004	Wagholi	1000000
P-005	Hinjewadi	1000000

d. Update the branch namecolumn

Update Branch Set Branchname='P-oo1@' where City='Pune';

```
1 row(s) updated.
```

Select * from Branch;

BRANCHNAME	CITY	ASSERTS
P-oo1@	Pune	1000000
P-002	Mumbai	1000000
P-003	Kharadi	1000000
P-004	Wagholi	1000000
P-005	Hinjewadi	1000000

e. Delete any two columns from the table

Delete any two columns from table;

Alter table Branch Drop(Asserts);

```
Table altered.
```

Select * from Branch;

BRANCHNAME	CITY
P-oo1@	Pune
P-002	Mumbai
P-003	Kharadi
P-004	Wagholi
P-005	Hinjewadi

Alter table Branch Drop(City);

BRANCHNAME
P-oo1@
P-002
P-003
P-004
P-005

Create a table called sailor table

Name Type

Sid Number

Sname Varchar2(20)

rating Varchar2(20)

a. Add column age to the sailor table.

b. Insert values into the sailortable.

c. Delete the row with rating>8.

d. Update the column details of sailor.

e. Insert null values into the table.

Create table Sailour(Sid Number(5) primary key,Sname varchar2(20),Rating varchar2(20));

DescSailour;

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
SAILOUR	SID	Number	-	5	0	1	-	-	-
	SNAME	Varchar2	20	-	-	-	✓	-	-
	RATING	Varchar2	20	-	-	-	✓	-	-

1 - 3

a. Add column age to the sailor table.

Alter table Sailour Add (Age Number(2));

DescSailour ;

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
SAILOUR	SID	Number	-	5	0	1	-	-	-
	SNAME	Varchar2	20	-	-	-	✓	-	-
	RATING	Varchar2	20	-	-	-	✓	-	-
	AGE	Number	-	2	0	-	✓	-	-

1 - 4

b. Insert values into the sailortable.

insert into Sailour values(1,'Ramesh','A1',28);

insert into Sailour values(2,'Dinesh','A2',25);

insert into Sailour values(3,'Amar','A3',30);

insert into Sailour values(4,'Samarth','A4',35);

insert into Sailour values(5,'Gourav','A5',32);

select * from Sailour;

SID	SNAME	RATING	AGE
1	Ramesh	A1	28
2	Dinesh	A2	25
3	Amar	A3	30
4	Samarth	A4	35
5	Gourav	A5	32

c. Delete the row with rating>8.
Delete from Sailour where Rating='8';

0 row(s) deleted

Delete from Sailour where Rating='A5';

1 row(s) deleted

Select * from Sailour;

SID	SNAME	RATING	AGE
1	Ramesh	A1	28
2	Dinesh	A2	25
3	Amar	A3	30
4	Samarth	A4	35

d. Update the column details of sailor.
Update Sailour Set Sname='Ramakant' where Rating='A1';

```
1 row(s) updated.
```
Select * from Sailour;

SID	SNAME	RATING	AGE
1	Ramakant	A1	28
2	Dinesh	A2	25
3	Amar	A3	30
4	Samarth	A4	35

e. Insert null values into the table.

Insert into Sailour values(6,'','A2',26);

SID	SNAME	RATING	AGE
1	Ramakant	A1	28
2	Dinesh	A2	25
3	Amar	A3	30
4	Samarth	A4	35
6	-	A2	26

Create a table called reserves table
Name Type
Boat id Integer
sid Integer
day Integer
a. Insert values into the reserves table.
b. Add column time to the reserves table.
c. Alter the column day data type todate.
d. Drop the column time in the table.
e. Delete the row of the table with some condition.

Create table Reserves(Boatidint,Sidint,Dayint);

Desc Reserves;

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
RESERVES	BOATID	Number	-	-	0	-	✓	-	-
	SID	Number	-	-	0	-	✓	-	-
	DAY	Number	-	-	0	-	✓	-	-

1 - 3

b. Add column time to the reserves table.
Alter table Reserves Add (Time int);

Desc Reserves;

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
RESERVES	BOATID	Number	-	-	0	-	✓	-	-
	SID	Number	-	-	0	-	✓	-	-
	DAY	Number	-	-	0	-	✓	-	-
	TIME	Number	-	-	0	-	✓	-	-

1 - 4

c. Alter the column day data type todate.

Alter table Reserves Modify (Day date);

Desc Reserves;

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
RESERVES	BOATID	Number	-	-	0	-	✓	-	-
	SID	Number	-	-	0	-	✓	-	-
	DAY	Date	7	-	-	-	✓	-	-
	TIME	Number	-	-	0	-	✓	-	-

1 - 4

d. Drop the column time in the table.
Alter table Reserves Drop(Time);

```
Table altered.
```

BOATID	SID	DAY
1	1	31-MAR-23
2	2	12-MAR-23
3	3	15-APR-23
4	4	18-MAY-23
5	5	20-MAY-23

a. Insert values into the reserves table.
insert into Reserves values(1,1,'31-Mar-2023',40);

insert into Reserves values(2,2,'12-Mar-2023',25);

insert into Reserves values(3,3,'15-Apr-2023',30);

insert into Reserves values(4,4,'18-May-2023',35);

insert into Reserves values(5,5,'20-May-2023',32);

Select * from Reserves;

BOATID	SID	DAY	TIME
1	1	31-MAR-23	40
2	2	12-MAR-23	25
3	3	15-APR-23	30
4	4	18-MAY-23	35
5	5	20-MAY-23	32

e. Delete the row of the table with some condition

Delete from Reserves where Boatid=1;

WEEK -2

QUERIES USING DDL AND DML

SQL Constraints

SQL constraints are used to specify rules for data in a table.

SQL Create Constraints

Constraints can be specified when the table is created with the CREATE TABLE statement, or after the table is created with the ALTER TABLE statement.

Syntax

```
CREATE TABLE table_name (
   column1 datatype constraint,
   column2 datatype constraint,
   column3 datatype constraint,
   ....
);
```

SQL Constraints

SQL constraints are used to specify rules for the data in a table.

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 - Prevents actions that would destroy links between tables
- CHECK - Ensures that the values in a column satisfies a specific condition
- DEFAULT - Sets a default value for a column if no value is specified

SQL NOT NULL Constraint

By default, a column can hold NULL values.

The NOT NULL constraint enforces a column to NOT accept NULL values.

This enforces a field to always contain a value, which means that you cannot insert a new record, or update a record without adding a value to this field.

SQL NOT NULL on CREATE TABLE

The following SQL ensures that the "ID", "LastName", and "FirstName" columns will NOT accept NULL values when the "Persons" table is created:

Example:

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255) NOT NULL,
    Age int
);
```

SQL NOT NULL on ALTER TABLE

To create a NOT NULL constraint on the "Age" column when the "Persons" table is already created, use the following SQL:

```
ALTER TABLE Persons
MODIFY Age int NOT NULL;
```

SQL UNIQUE Constraint

The UNIQUE constraint ensures that all values in a column are different.

Both the UNIQUE and PRIMARY KEY constraints provide a guarantee for uniqueness for a column or set of columns.

A PRIMARY KEY constraint automatically has a UNIQUE constraint.

However, you can have many UNIQUE constraints per table, but only one PRIMARY KEY constraint per table.

SQL UNIQUE Constraint on CREATE TABLE

The following SQL creates a UNIQUE constraint on the "ID" column when the "Persons" table is created:

```
CREATE TABLE Persons (
    ID int NOT NULL UNIQUE,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int
);
```

To name a UNIQUE constraint, and to define a UNIQUE constraint on multiple columns, use the following SQL syntax:

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    CONSTRAINT UC_Person UNIQUE (ID,LastName)
);
```

SQL UNIQUE Constraint on ALTER TABLE

To create a UNIQUE constraint on the "ID" column when the table is already created, use the following SQL:

```
ALTER TABLE Persons
ADD UNIQUE (ID);
```

To name a UNIQUE constraint, and to define a UNIQUE constraint on multiple columns, use the following SQL syntax:

```
ALTER TABLE Persons
ADD CONSTRAINT UC_Person UNIQUE (ID,LastName);
```

DROP a UNIQUE Constraint

To drop a UNIQUE constraint, use the following SQL:

```
ALTER TABLE Persons
DROP CONSTRAINT UC_Person;
```

SQL PRIMARY KEY Constraint

The PRIMARY KEY constraint uniquely identifies each record in a table.

Primary keys must contain UNIQUE values, and cannot contain NULL values.

A table can have only ONE primary key; and in the table, this primary key can consist of single or multiple columns (fields).

SQL PRIMARY KEY on CREATE TABLE

The following SQL creates a PRIMARY KEY on the "ID" column when the "Persons" table is created:

```
CREATE TABLE Persons (
    ID int NOT NULL PRIMARY KEY,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int
);
```

To allow naming of a PRIMARY KEY constraint, and for defining a PRIMARY KEY constraint on multiple columns, use the following SQL syntax:

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    CONSTRAINT PK_Person PRIMARY KEY (ID,LastName)
);
```

SQL PRIMARY KEY on ALTER TABLE

To create a PRIMARY KEY constraint on the "ID" column when the table is already created, use the following SQL:

```
ALTER TABLE Persons
ADD PRIMARY KEY (ID);
```

To allow naming of a PRIMARY KEY constraint, and for defining a PRIMARY KEY constraint on multiple columns, use the following SQL syntax:

```
ALTER TABLE Persons
ADD CONSTRAINT PK_Person PRIMARY KEY (ID,LastName);
```

DROP a PRIMARY KEY Constraint

To drop a PRIMARY KEY constraint, use the following SQL:

```
ALTER TABLE Persons
DROP CONSTRAINT PK_Person;
```

SQL FOREIGN KEY Constraint

The FOREIGN KEY constraint is used to prevent actions that would destroy links between tables.

A FOREIGN KEY is a field (or collection of fields) in one table, that refers to the PRIMARY KEY in another table.

The table with the foreign key is called the child table, and the table with the primary key is called the referenced or parent table.

Look at the following two tables:

Persons Table

PersonID	LastName	FirstName
1	Hansen	Ola
2	Svendson	Tove
3	Pettersen	Kari

Orders Table

OrderID	OrderNumber	PersonID
1	77895	3
2	44678	3
3	22456	2
4	24562	1

Notice that the "PersonID" column in the "Orders" table points to the "PersonID" column in the "Persons" table.

The "PersonID" column in the "Persons" table is the PRIMARY KEY in the "Persons" table.

The "PersonID" column in the "Orders" table is a FOREIGN KEY in the "Orders" table.

The FOREIGN KEY constraint prevents invalid data from being inserted into the foreign key column, because it has to be one of the values contained in the parent table.

SQL FOREIGN KEY on CREATE TABLE

The following SQL creates a FOREIGN KEY on the "PersonID" column when the "Orders" table is created:

```
CREATE TABLE Orders (
    OrderID int NOT NULL PRIMARY KEY,
    OrderNumber int NOT NULL,
    PersonID int FOREIGN KEY REFERENCES Persons(PersonID)
);
```

To allow naming of a FOREIGN KEY constraint, and for defining a FOREIGN KEY constraint on multiple columns, use the following SQL syntax:

```
CREATE TABLE Orders (
   OrderID int NOT NULL,
   OrderNumber int NOT NULL,
   PersonID int,
   PRIMARY KEY (OrderID),
   CONSTRAINT FK_PersonOrder FOREIGN KEY (PersonID)
   REFERENCES Persons(PersonID)
);
```

SQL FOREIGN KEY on ALTER TABLE

To create a FOREIGN KEY constraint on the "PersonID" column when the "Orders" table is already created, use the following SQL:

```
ALTER TABLE Orders
ADD CONSTRAINT FK_PersonOrder
FOREIGN KEY (PersonID) REFERENCES Persons(PersonID);
```

DROP a FOREIGN KEY Constraint

To drop a FOREIGN KEY constraint, use the following SQL:

```
ALTER TABLE Orders
DROP CONSTRAINT FK_PersonOrder;
```

SQL CHECK Constraint

The CHECK constraint is used to limit the value range that can be placed in a column.

If you define a CHECK constraint on a column it will allow only certain values for this column.

If you define a CHECK constraint on a table it can limit the values in certain columns based on values in other columns in the row.

SQL CHECK on CREATE TABLE

The following SQL creates a CHECK constraint on the "Age" column when the "Persons" table is created. The CHECK constraint ensures that the age of a person must be 18, or older:

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int CHECK (Age>=18)
);
```

To allow naming of a CHECK constraint, and for defining a CHECK constraint on multiple columns, use the following SQL syntax:

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    City varchar(255),
    CONSTRAINT CHK_Person CHECK (Age>=18 AND City='Sandnes')
);
```

SQL CHECK on ALTER TABLE

To create a CHECK constraint on the "Age" column when the table is already created, use the following SQL:

```
ALTER TABLE Persons
ADD CHECK (Age>=18);
```

To allow naming of a CHECK constraint, and for defining a CHECK constraint on multiple columns, use the following SQL syntax:

```
ALTER TABLE Persons
ADD CONSTRAINT CHK_PersonAge CHECK (Age>=18 AND City='Sandnes');
```

DROP a CHECK Constraint

To drop a CHECK constraint, use the following SQL:

```
ALTER TABLE Persons
DROP CONSTRAINT CHK_PersonAge;
```

SQL DEFAULT Constraint

The DEFAULT constraint is used to set a default value for a column.

The default value will be added to all new records, if no other value is specified.

SQL DEFAULT on CREATE TABLE

The following SQL sets a DEFAULT value for the "City" column when the "Persons" table is created:

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    City varchar(255) DEFAULT 'Sandnes'
);
```

SQL DEFAULT on ALTER TABLE

To create a DEFAULT constraint on the "City" column when the table is already created, use the following SQL:

```
ALTER TABLE Persons
MODIFY City DEFAULT 'Sandnes';
```

DROP a DEFAULT Constraint

To drop a DEFAULT constraint, use the following SQL:

```
ALTER TABLE Persons
ALTER COLUMN City DROP DEFAULT;
```

Assignment -7

1. **a. Create a user and grant all permissions to the user.**
   b. Insert the any three records in the employee table and use rollback. Check the result.
   c. Add primary key constraint and not null constraint to the employee table.
   d. Insert null values to the employee table and verify the result.

SOLUTION:

a) Create a user and grant all permissions to the user.

CONNECT <USER-NAME>/<PASSWORD>@<DATABASE NAME>;

Example:

```
SQL> connect
Enter user-name: system
Enter password:
Connected.
```

Here username is system and password is system.

--Create user query

CREATE USER <USER NAME> IDENTIFIED BY <PASSWORD>;

Example:

Create user anita identified by anita123;

Here anita is username and anita123 is password.

```
SQL> connect
Enter user-name: system
Enter password:
Connected.
SQL> create user anita identified by anita123;

User created.
```

Now open new SQL command Line and try to login by using new username and password

```
SQL> connect;
Enter user-name: anita
Enter password:
ERROR:
ORA-01045: user ANITA lacks CREATE SESSION privilege; logon denied
```

It will give error user ANITA lacks CREATE SESSION privilege; logon denied to solve this error user anita grant connect privilege using following command

--Provide roles

Syntax:

GRANT CONNECT,RESOURCE,DBA TO <USER NAME>;

```
SQL> grant connect,resource,dba to anita;

Grant succeeded.
```

Now open new SQL command Line and try to login by using new username and password

```
SQL> connect ;
Enter user-name: anita
Enter password:
Connected.
```

Now user can able to log in.

--Assigning privileges

From system user log in provide grant all privileges to user

GRANT ALL PRIVILEGES TO <USER NAME>;

```
SQL> connect
Enter user-name: system
Enter password:
Connected.
SQL> create user anita identified by anita123;

User created.

SQL> grant connect,resource,dba to anita;

Grant succeeded.

SQL> grant all privileges to anita;

Grant succeeded.
```

--Provide access to tables.

Syntax: GRANT SELECT, UPDATE, INSERT, DELETE ON <TABLE NAME> TO <USER NAME>;

Before giving access to table by using following command we can check list of tables in system user.

Select * from tab;

```
SQL> select * from tab;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
AQ$DEF$_AQCALL                 VIEW
AQ$DEF$_AQERROR                VIEW
AQ$_DEF$_AQCALL_F              VIEW
AQ$_DEF$_AQERROR_F             VIEW
AQ$_INTERNET_AGENTS            TABLE
AQ$_INTERNET_AGENT_PRIVS       TABLE
AQ$_QUEUES                     TABLE
AQ$_QUEUE_TABLES               TABLE
AQ$_SCHEDULES                  TABLE
CATALOG                        SYNONYM
COL                            SYNONYM

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
DEF$_AQCALL                    TABLE
DEF$_AQERROR                   TABLE
DEF$_CALLDEST                  TABLE
DEF$_DEFAULTDEST               TABLE
DEF$_DESTINATION               TABLE
```

It will show list of tables as shown above.

Now if we want to provide access to employee table to user anita type following command

GRANT SELECT, UPDATE, INSERT, DELETE ON <TABLE NAME> TO <USER NAME>;

Grant select, update, insert ,delete on employee to anita;

```
SQL> Grant select,update,insert ,delete on employee to anita;

Grant succeeded.
```

Now from user anita login try to access the employee table.

```
SQL> connect ;
Enter user-name: anita
Enter password:
Connected.
SQL> clear;
SQL>
SQL> select * from employee;
select * from employee
              *
ERROR at line 1:
ORA-00942: table or view does not exist
```

It will give error table or view does not exist to solve this error type command

Select * from system.employee;

```
SQL> connect ;
Enter user-name: anita
Enter password:
Connected.
SQL> clear;
SQL>
SQL> select * from employee;
select * from employee
              *
ERROR at line 1:
ORA-00942: table or view does not exist


SQL> Select  * from system.employee;

no rows selected

SQL>
```

In this case employee table does not have any data that's why it is showing no rows selected.

Insert the any three records in the employee table and use rollback. Check the result.

```
SQL> insert into System.Employee values(118,'Avantika',50000,'Devloper');
1 row created.
SQL> insert into System.Employee values(119,'Aadarsh',45000,'Tester');
1 row created.
SQL> insert into System.Employee values(120,'Suman',42000,'Accountant');
1 row created.
SQL> Select * from System.Employee;
   DEPTNO ENAME                SAL JOB
--------- ------------------- ---------- --------------------
       110 Aasha                40000 Developer
       110 Aaditya              25000 Tester
       111 Aakash               30000 HR
       112 Aajay                25000 Accountant
       113 Aangha               30000 Manager
       114 Avantika             20000 HR
       115 Sarika               45000 HR
       116 Raman                55000 Tester
       117 Rakhi                25000 Devloper
       118 Avantika             50000 Devloper
       119 Aadarsh              45000 Tester
       120 Suman                42000 Accountant
12 rows selected.
SQL> Rollback;
Rollback complete.
```

SQL> Select * from System.Employee;

DEPTNO	ENAME	SAL	JOB
110	Aasha	40000	Developer
110	Aaditya	25000	Tester
111	Aakash	30000	HR
112	Aajay	25000	Accountant
113	Aangha	30000	Manager
114	Avantika	20000	HR
115	Sarika	45000	HR
116	Raman	55000	Tester
117	Rakhi	25000	Devloper

9 rows selected.

c. Add primary key constraint and not null constraint to the employeetable.

Alter table Employee modify (Deptno number(20) primary key, ename varchar2(20) not null);

Table altered.

d) Insert null values to the employee table and verify theresult.

insert into System.Employee values(121,'Sakshi',25000,Null);

1 row created.

Select * from System.Employee;

DEPTNO	ENAME	SAL	JOB
110	Aasha	40000	Developer
110	Aaditya	25000	Tester
111	Aakash	30000	HR
112	Aajay	25000	Accountant
113	Aangha	30000	Manager
114	Avantika	20000	HR
115	Sarika	45000	HR
116	Raman	55000	Tester
117	Rakhi	25000	Devloper
121	Sakshi	25000	

10 rows selected.

Assignment - 8

a. Create a user and grant all permissions to the user.
b. Insert values in the department table and use commit.
c. Add constraints like unique and not null to the department table.
d. Insert repeated values and null values into the table

---

a. Create a user and grant all permissions to the user.

```
SQL> connect;
Enter user-name: System
Enter password:mca
Connected.
SQL> Create user Student identified by Student123;
User created.
SQL> grant connect,resource,dba to Student;
Grant succeeded.
SQL> grant all privileges to Student;

Grant succeeded.
SQL> connect;
Enter user-name: Student
Enter password:Student123
Connected.
SQL> Grant connect,resource,dba to Student;
Grant succeeded.
```

b. Insert values in the department table and use commit.

```
 select * from System.Department;
   DEPTNO DEPTNAME           LOCATION
---------- -------------------- --------------------
       201 Account              Ground Floor
       202 Sales               First Floor
       203 Purchase             Second Floor
       204 HR                  Third Floor
       205 Salary               Four Floor

SQL> insert into system.department values(206,'IT','Fifth Floor');
1 row created.
SQL> insert into system.department values(207,'TAX','Sixth Floor');
1 row created.
SQL> Select * from System.Department;

   DEPTNO DEPTNAME           LOCATION
---------- -------------------- --------------------
       201 Account              Ground Floor
       202 Sales               First Floor
       203 Purchase             Second Floor
       204 HR                  Third Floor
       205 Salary               Four Floor
       206 IT                  Fifth Floor
       207 TAX                  Sixth Floor
7 rows selected.
SQL > commit;
```

Commit complete.

c. Add constraints like unique and not null to the department table.

Alter table system.department modify(deptno number unique);
Table altered.

Alter table system.department modify(location varchar2(30) not null);

Table altered.

e. Insert repeated values and null values into the table.

insert into system.department values(208,'TAX','Sixth Floor');

1 row created.

insert into system.department values(209,NULL,'Sixth Floor');

1 row created.

Assignment - 9

a. create a user and grant all permissions to the user.
b. Insert values in the table and use commit.
c. Delete Any three records in the department table and use rollback.
d.Add constraints primary key and foreign key to the table.

---

a. create a user and grant all permissions to the user.

```
SQL> connect;
Enter user-name: System
Enter password:mca
Connected.
SQL> Create user MCAStudent identified by Mcastudent123;
User created.
SQL> grant connect,resource,dba to MCAStudent;
Grant succeeded.
SQL> grant all privileges to MCAStudent;
Grant succeeded.
SQL> connect;
Enter user-name: MCAStudent
Enter password:Mcastudent123
Connected.
SQL> Grant connect,resource,dba to MCAStudent;
Grant succeeded.
```

b. Insert values in the table and use commit.

```
SQL> select * from System.Department;
   DEPTNO DEPTNAME           LOCATION
---------- -------------------- ------------------------------
      201 Account            Ground Floor
      202 Sales              First Floor
      203 Purchase           Second Floor
      204 HR                 Third Floor
      205 Salary             Four Floor
      206 IT                 Fifth Floor
      207 TAX                Sixth Floor
      208 TAX                Sixth Floor
      209                    Sixth Floor
9 rows selected.
SQL> insert into System.Department values(210,'Computer','Seventh Floor');
1 row created.
SQL> commit;
Commit complete.
```

```
SQL> select * from System.Department;

   DEPTNO DEPTNAME          LOCATION
--------- -------------------- ------------------------------
      201 Account           Ground Floor
      202 Sales              First Floor
      203 Purchase          Second Floor
      204 HR                Third Floor
      205 Salary             Four Floor
      206 IT                Fifth Floor
      207 TAX               Sixth Floor
      208 TAX               Sixth Floor
      209                   Sixth Floor
      210 Computer           Seventh Floor

10 rows selected.
```

c. Delete Any three records in the department table and use rollback.

```
SQL> Delete from System.Department
  2  Where  Deptno=209;

1 row deleted.

SQL> select * from System.Department;

   DEPTNO DEPTNAME          LOCATION
--------- -------------------- ------------------------------
      201 Account           Ground Floor
      202 Sales             First Floor
      203 Purchase           Second Floor
      204 HR                Third Floor
      205 Salary             Four Floor
      206 IT                Fifth Floor
      207 TAX               Sixth Floor
      208 TAX               Sixth Floor
      210 Computer           Seventh Floor

9 rows selected.

SQL> Rollback;

Rollback complete.
```

```
SQL> select * from System.Department;

   DEPTNO DEPTNAME          LOCATION
--------- ------------------- ------------------------------
      201 Account            Ground Floor
      202 Sales              First Floor
      203 Purchase           Second Floor
      204 HR                 Third Floor
      205 Salary             Four Floor
      206 IT                 Fifth Floor
      207 TAX                Sixth Floor
      208 TAX                Sixth Floor
      209                    Sixth Floor
      210 Computer           Seventh Floor

10 rows selected.
```

d. Add constraints primary key and foreign key to the table.

```
Alter Table System.Department ADD Primary Key (Deptno);
  Table Altered.
```

……

Assignment - 10

a. create a user and grant all permissions to the user.
b.Insert values in the sailor table and use commit.
c. Add savepoint after insertion of records and verify savepoint.
d.Add constraints not null and primary key to the sailor table.

……

a. create a user and grant all permissions to the user.

```
SQL> connect;
Enter user-name: System
Enter password:mca
```

```
Connected.
SQL> Create user MCAStudent identified by Mcastudent123;
User created.
SQL> grant connect,resource,dba to MCAStudent;
Grant succeeded.
SQL> grant all privileges to MCAStudent;
Grant succeeded.
SQL> connect;
Enter user-name: MCAStudent
Enter password:Mcastudent123
Connected.
SQL> Grant connect,resource,dba to MCAStudent;
Grant succeeded.
```

b. Insert values in the sailor table and use commit.

```
SQL> select * from System.Sailor;

       SID SNAME                AGE RATING
---------- -------------------- ---------- --------------------
       101 Rakhi                25 45
       102 Aniket               25 55
       103 Babita               25 65
       104 Sameer               25 75
       105 Parag                25 40

SQL> insert into System.Sailor values(106,'Raman',30,50);

1 row created.

SQL> commit;

Commit complete.
```

```
SQL> select * from System.Sailor;

       SID SNAME                AGE RATING
---------- -------------------- ---------- --------------------
       101 Rakhi                25 45
       102 Aniket               25 55
       103 Babita               25 65
       104 Sameer               25 75
       105 Parag                25 40
       106 Raman                30 50
```

6 rows selected.

c. Add savepoint after insertion of records and verify savepoint.

```
SQL>  savepoint week2;

Savepoint created.

SQL> rollback to week2;

Rollback complete.
```

d. Add constraints not null and primary key to the sailor table.

```
ALTER TABLE System.Sailor
 modify sid not null;

Table altered.

ALTER TABLE System.Sailor
ADD PRIMARY KEY (sname);

Table altered.
```

Assignment -11

a. create a user and grant all permissions to the user.
b.Use revoke command to remove user permissions.
c. change password of the user created.
d.add constraint foreign key and not null.

---

a. create a user and grant all permissions to the user.

```
SQL> connect;
Enter user-name: System
Enter password:mca
```

Connected.
SQL> Create user MCAStudent identified by Mcastudent123;
User created.
SQL> grant connect,resource,dba to MCAStudent;
Grant succeeded.
SQL> grant all privileges to MCAStudent;
Grant succeeded.
SQL> connect;
Enter user-name: MCAStudent
Enter password:Mcastudent123
Connected.
SQL> Grant connect,resource,dba to MCAStudent;
Grant succeeded.

b.Use revoke command to remove user permissions

Revoke all privileges on system.Employee from MCAStudent;
Revoke succeeded.

c. Change password of the user created.

SQL> connect
Enter user-name: MCAStudent
Enter password:MCAStudent123
Connected.
SQL> alter user MCAStudent identified by MCAStudent1234;
User altered.
SQL> connect
Enter user-name: MCAStudent
Enter password:MCAStudent1234
Connected.

d. Add constraint foreign key and not null.

SQL> ALTER TABLE Employee
modify EName varchar2(20) NOTNULL;

1 Table Altered.

SQL> ALTER TABLE Branches
ADD FOREIGN KEY (location) REFERENCES department(location);

1 Table Altered.

Assignment – 12

a. create a user and grant all permissions to the user.
b.Update The Table reserves and use savepoint and rollback.
c. Add Constraint primary key,foreign key and not null to the reserves table.
d.delete constraint notnull to the table column

SOLUTION:

a) create a user and grant all permissions to the user.

SQL > create user MCAdb identified by MCA123;

User created.

SQL> grant connect,resource,dba to MCAdb;

Grant succeeded.

SQL> grant all privileges to MCAdb;

Grant succeeded.

SQL> connect;
Enter user-name: MCAdb
Enter password:MCA123
Connected
SQL> grant connect,resource,dba to MCAdb;

Grant succeeded.

b.Update the Table reserves and use savepoint and rollback.

SQL> create table reserve(boatid number(5),sid number(5),day number(10));

Table created.

SQL> insert into reserve values(1,5,10);

1 row created.

SQL> insert into reserve values(2,6,11);

1 row created.

SQL> insert into reserve values(3,8,12);

1 row created.

SQL> select * from System.Reserve;

BOATID	SID	DAY
1	5	10
2	6	11
3	8	12

SQL> update Reserve set day=15 where boatid=2;

1 row updated.

SQL>savepoint assignment6;

Savepoint created.
SQL> select * from System.Reserve;

BOATID	SID	DAY
1	5	10
2	6	15
3	8	12

SQL> update Reserve set day=25 where boatid=2;

1 row updated.

SQL> rollback;

Rollback complete.

c. Add Constraint primary key,foreign key and not null to the reserves table.

SQL> alter table system.reserve add primary key(boatid);

Table altered.

1 Table Altered.

SQL> ALTER TABLE Reserve
ADD FOREIGN KEY (sname) REFERENCES boat(sname);

1 Table Altered.

SQL> ALTER TABLE reserve
modify sid number NOTNULL;
1 Table Altered.

d.delete constraint notnull to the table column

SQL> ALTER TABLE
reserves modify (sid int
null);
1 Table Altered.

Week 3

SQL Aggregate Functions

An aggregate function is a function that performs a calculation on a set of values, and returns a single value.

Aggregate functions are often used with the GROUP BY clause of the SELECT statement. The GROUP BY clause splits the result-set into groups of values and the aggregate function can be used to return a single value for each group.

The most commonly used SQL aggregate functions are:

- MIN() - returns the smallest value within the selected column
- MAX() - returns the largest value within the selected column
- COUNT() - returns the number of rows in a set
- SUM() - returns the total sum of a numerical column
- AVG() - returns the average value of a numerical column

Aggregate functions ignore null values (except for COUNT()).

Analyzing Amazon Customer Transaction Data

To bring these concepts to life, we'll be working with the `product_spend` dataset which contains data on what products Amazon customers are buying, and how much they are buying them for.

Table: `product_spend` Sample Data

Category	product	user_id	spend	transaction_date
Appliance	washing machine	123	219.80	03/02/2022 11:00:00
electronics	vacuum	178	152.00	04/05/2022 10:00:00
electronics	wireless headset	156	249.90	07/08/2022 10:00:00
electronics	vacuum	145	189.00	07/15/2022 10:00:00
electronics	computer mouse	195	45.00	07/01/2022 11:00:00
Appliance	refrigerator	165	246.00	12/26/2021 12:00:00
Appliance	refrigerator	123	299.99	03/02/2022 11:00:00
...	...	...	...	...

Let's dive into some practical examples of using aggregate functions to analyze this Amazon data.

Counting Number of Orders with SQL COUNT()

The `COUNT()` function enables you to count the total number of rows in a table. We can use the `COUNT()` function on the `user_id` column as follows:

Here's an example `COUNT` query:

```
SELECT COUNT(user_id)
FROM product_spend;
```

Results

Count
15

Frequently, one of the first things we do after `SELECT *` 'ing the data to explore it is then count how big the dataset is. We do this with the following query:

```
SELECT COUNT(*)
FROM table_name;
```

Calculating Total Sales with SQL SUM()

The `SUM()` function calculates the sum of numeric values in a column. To calculate the total amount of money spent on Amazon products, we can use the `SUM()` function to sum the `spend` column.

```
SELECT SUM(spend)
FROM product_spend;
```

Results

Sum
2142.76

Finding the Average Price with SQL AVG()

The `AVG()` function computes the average value of numeric data in a column. To find the average price of an Amazon order, we can use the `AVG()` function on the `spend` column.

```
SELECT AVG(spend)
FROM product_spend;
```

Results

avg

avg
142.8506666666666667

Finding the Minimum with SQL MIN()

The `MIN()` function identifies the smallest value in a column. To find the lowest priced Amazon product order, we can use the `MIN()` function on the `spend` column.

```
SELECT MIN(spend)
FROM product_spend;
```

Results

Min
7.99

Discovering the Maximum Price with SQL MAX()

The `MAX()` function retrieves the largest value in a column. To find the highest priced Amazon product order, we can use the `MAX()` function on the `spend` column.

```
SELECT MAX(spend)
FROM product_spend;
```

Results

Max
299.99

Assignment -13

Create table Employee (Deptno int ,Ename varchar(20) ,Sal int,Deptname varchar(20));

insert into Employee values(110,'Aasha',40000,'IT');

insert into Employee values(110,'Aaditya',25000,'IT');

insert into Employee values(111,'Aakash',30000,'HR');

insert into Employee values(112,'Aajay',25000,'Account');

insert into Employee values(113,'Aangha',30000,'Sales');

select * from Employee;

DEPTNO	ENAME	SAL	DEPTNAME
110	Aasha	40000	IT
110	Aaditya	25000	IT
111	Aakash	30000	HR
112	Aajay	25000	Account
113	Aangha	30000	Sales

Q.1)Display the enames who belogs to department no. 110 along with average salary.

(By using Group by clause)

SELECT ENAME, DEPTNO, AVG(SAL) FROM Employee WHERE DEPTNO=110
GROUP BY ENAME,DEPTNO;

ENAME	DEPTNO	AVG(SAL)
Aasha	110	40000
Aaditya	110	25000

Q.2) Display lowest paid employee details under each department.

SELECT DEPTNAME,MIN(SAL) FROM Employee GROUP BY DEPTNAME;

DEPTNAME	MIN(SAL)
IT	25000
HR	30000
Account	25000
Sales	30000

Q.3 Display number of Employees working in each department with their department number.

SELECT DEPTNO, DEPTNAME, COUNT(*) FROM EMPLOYEE GROUP BY DEPTNO,DEPTNAME;

DEPTNO	DEPTNAME	COUNT(*)
113	Sales	1
110	IT	2
111	HR	1
112	Account	1

SELECT DEPTNO, COUNT(*) FROM EMPLOYEE GROUP BY DEPTNO;

DEPTNO	COUNT(*)
113	1
112	1
110	2
111	1

Q.4) Display Number of Employees working in each Department.

SQL>SELECT DEPTNO,DEPTNAME,COUNT(*) FROM EMPLOYEE GROUP BY DEPTNO,DEPTNAME;

DEPTNO	DEPTNAME	COUNT(*)
113	Sales	1
110	IT	2
111	HR	1

112	Account	1

E) List all employees starts with either B or C.

SQL>SELECT ENAME FROM EMPLOYEE WHERE ENAME LIKE 'B%' OR ENAME LIKE 'C%' ORDER BY ENAME;

INSERT INTO EMPLOYEE VALUES (114,'BALU',20000, 'CSE');

ENAME
BALU

F) Display only these employee names having maximum salary grater than or equal to 5000 .

SQL>SELECT ENAME FROM EMPLOYEE WHERE SAL>=5000;

ENAME
Aasha
Aaditya
Aakash
Aajay
Aangha
BALU

Assignment -14

Create table Employee (Deptno int ,Ename varchar(20) ,Sal int,Job varchar(20));

insert into Employee values(110,'Aasha',40000,'Developer');

insert into Employee values(110,'Aaditya',25000,'Tester');

insert into Employee values(111,'Aakash',30000,'HR');

insert into Employee values(112,'Aajay',25000,'Accountant');

insert into Employee values(113,'Aangha',30000,'Manager');

select * from Employee;

DEPTNO	ENAME	SAL	JOB
110	Aasha	40000	Developer
110	Aaditya	25000	Tester
111	Aakash	30000	HR
112	Aajay	25000	Accountant
113	Aangha	30000	Manager

a. Calculate average salary for each different job.

Select Job,Avg(Sal) from Employee group by Job;

JOB	AVG(SAL)
Manager	30000
Tester	25000
HR	30000
Developer	40000
Accountant	25000

b. Show average salary of each job excluding manager.
Select Job, Avg(Sal) from Employee where job not in ('Manager') group by Job;

JOB	AVG(SAL)
Tester	25000
HR	30000
Developer	40000
Accountant	25000

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

Select Deptno ,Avg(Sal),Count(*) from Employee
Group by Deptno having count(*)>=2;

DEPTNO	AVG(SAL)	COUNT(*)
110	32500	2

d. Display employees who earn more than the lowest salary in department 30.

Select Ename,Sal from Employee where Deptno =30
And Sal>(Select Min(Sal) from Employee where Deptno=30);

no data found

Select Ename,Sal from Employee where Deptno =110
And Sal>(Select Min(Sal) from Employee where Deptno=110);

ENAME	SAL
Aasha	40000

e) Show that value returned by sign (n) function.
Select SIGN(-15),SIGN(0),SIGN(15) FROM DUAL;

SIGN(-15)	SIGN(0)	SIGN(15)
-1	0	1

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

SQL>DECLARE @START DATETIME
DECLARE @END DATETIME
SET @START='2000-10-04'
SET @END='2021-12-19'
SELECT DATEDIFF(D,@START,@END);
SELECT DATEDIFF(D,2000-10-04,2021-12-20);

Assignment -15

Create table Employee (Deptno int ,Ename varchar(20) ,Sal int,Job varchar(20));

insert into Employee values(110,'Aasha',40000,'Developer');

insert into Employee values(110,'Aaditya',25000,'Tester');

insert into Employee values(111,'Aakash',30000,'HR');

insert into Employee values(112,'Aajay',25000,'Accountant');

insert into Employee values(113,'Aangha',30000,'Manager');

select * from Employee;

DEPTNO	ENAME	SAL	JOB
110	Aasha	40000	Developer
110	Aaditya	25000	Tester
111	Aakash	30000	HR
112	Aajay	25000	Accountant
113	Aangha	30000	Manager

a. **Show that two substring as single string.**

SELECT CONCAT(CONCAT(ENAME,' '),JOB) AS ENAME FROM EMPLOYEE ORDER BY ENAME;

ENAME
Aaditya Tester
Aajay Accountant
Aakash HR
Aangha Manager
Aasha Developer

b. **List all employee names,salary and 15% rise in a salary.**

SELECT ENAME,TO_CHAR(1.15*SAL) AS "SAL" FROM EMPLOYEE;

ENAME	SAL
Aasha	46000
Aaditya	28750

Aakash	34500
Aajay	28750
Aangha	34500

SELECT ENAME,SAL,TO_CHAR(1.15*SAL) AS "UPDATED SALARY" FROM EMPLOYEE;

ENAME	SAL	UPDATED SALARY
Aasha	40000	46000
Aaditya	25000	28750
Aakash	30000	34500
Aajay	25000	28750
Aangha	30000	34500

c. Display lowest paid employee details under each department.

Select Deptno,Min(sal) From Employee
where Deptno is not null Group by Deptno
Order by min(Sal) Desc;

DEPTNO	MIN(SAL)
113	30000
111	30000
110	25000
112	25000

d. Display the average monthly salary bill for each department no.

Select Deptno ,Avg(Sal) As AVGSalary from Employee Group by Deptno;

DEPTNO	AVGSALARY
113	30000
112	25000
110	32500

111	30000

e. **Show the average salary for all departments employing more than or equal to two people.**
Select Deptno,Avg(Sal),Count(Ename)from Employee Group by Deptno having Count(Ename)>=2;

DEPTNO	AVG(SAL)	COUNT(ENAME)
110	32500	2

f. **By using the group by clauses ,display the Eid who belongs to deptno 05 along with average salary.**

Select Deptno,Avg(Sal) from Employee where Deptno=12 Group By Deptno;

no data found

Select Deptno,Avg(Sal) from Employee where Deptno=110 Group By Deptno;

DEPTNO	AVG(SAL)
110	32500

Assignment -16

Create table Employee (Deptno int ,Ename varchar(20) ,Sal int,Job varchar(20));

insert into Employee values(110,'Aasha',40000,'Developer');

insert into Employee values(110,'Aaditya',25000,'Tester');

insert into Employee values(111,'Aakash',30000,'HR');

insert into Employee values(112,'Aajay',25000,'Accountant');

insert into Employee values(113,'Aangha',30000,'Manager');

select * from Employee;

DEPTNO	ENAME	SAL	JOB
110	Aasha	40000	Developer
110	Aaditya	25000	Tester
111	Aakash	30000	HR
112	Aajay	25000	Accountant
113	Aangha	30000	Manager

a)Count the number of Employees in department 20.

Select count(Ename)From Employee where Deptno=20;

COUNT(ENAME)
0

Select count(Ename)From Employee where Deptno=110;

COUNT(ENAME)
2

b)Find the minimum salary earned by clerk.

Select Min(sal)from Employee Where Job='Clerk';

MIN(SAL)
-

Select Min(sal)from Employee Where Job='Developer';

MIN(SAL)
40000

c)Find minimum, maximum,average salary of all employees.

Select Ename,Sum(Sal),Avg(Sal),Max(Sal),Min(Sal) from Employee Group by Ename;

ENAME	SUM(SAL)	AVG(SAL)	MAX(SAL)	MIN(SAL)
Aakash	30000	30000	30000	30000
Aajay	25000	25000	25000	25000
Aasha	40000	40000	40000	40000
Aangha	30000	30000	30000	30000
Aaditya	25000	25000	25000	25000

d)List the minimum and maximum salaries for each job type.

insert into Employee values(114,'Avantika',20000,'HR');

Select Job,Min(Sal),Max(Sal) from Employee group by Job;

JOB	MIN(SAL)	MAX(SAL)
Manager	30000	30000
Tester	25000	25000
HR	20000	30000
Developer	40000	40000
Accountant	25000	25000

e) List the employee names in descending order.
SELECT * FROM EMPLOYEE ORDER BY ENAME DESC;

DEPTNO	ENAME	SAL	JOB
114	Avantika	20000	HR
110	Aasha	40000	Developer
113	Aangha	30000	Manager
111	Aakash	30000	HR
112	Aajay	25000	Accountant
110	Aaditya	25000	Tester

SELECT * FROM EMPLOYEE ORDER BY JOB DESC;

DEPTNO	ENAME	SAL	JOB
110	Aaditya	25000	Tester
113	Aangha	30000	Manager
114	Avantika	20000	HR
111	Aakash	30000	HR
110	Aasha	40000	Developer
112	Aajay	25000	Accountant

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

SELECT * FROM EMPLOYEE ORDER BY ENAME ASC;

DEPTNO	ENAME	SAL	JOB
110	Aaditya	25000	Tester
112	Aajay	25000	Accountant
111	Aakash	30000	HR
113	Aangha	30000	Manager
110	Aasha	40000	Developer
114	Avantika	20000	HR

SELECT * FROM EMPLOYEE ORDER BY JOB ASC;

DEPTNO	ENAME	SAL	JOB
110	Aaditya	25000	Tester
113	Aangha	30000	Manager
114	Avantika	20000	HR
111	Aakash	30000	HR
110	Aasha	40000	Developer
112	Aajay	25000	Accountant

Assignment -17

create table sailor(sid number primary key,sname varchar2(20),age number ,rating varchar2(20));

insert into sailor values(101,'Rakhi',25,45)
insert into sailor values(102,'Aniket',25,55)
insert into sailor values(103,'Babita',25,65)
insert into sailor values(104,'Sameer',25,75)
insert into sailor values(105,'Parag',25,40)

Select * from Sailor;

SID	SNAME	AGE	RATING
101	Rakhi	25	45
102	Aniket	25	55
103	Babita	25	65
104	Sameer	25	75
105	Parag	25	40

create table ReservesB(boatid number primary key ,Rdate date ,sid number references Sailor(sid));

insert into ReservesB values(201,'31-Mar-2023',101);
insert into ReservesB values(202,'3-Mar-2023',102);
insert into ReservesB values(203,'10-Mar-2023',103);
insert into ReservesB values(204,'15-Mar-2023',104);
insert into ReservesB values(205,'20-Mar-2023',105);

Select * from ReservesB ;

BOATID	RDATE	SID
201	31-MAR-23	101
202	03-MAR-23	102
203	10-MAR-23	103
204	15-MAR-23	104
205	20-MAR-23	105

SELECT * FROM SAILORS WHERE SID IN (SELECT SID FROM RESERVESB INNER JOIN BOATS ON RESERVES.BID=BOATS.BID WHERE BOATS.BNMAE='INTERLAKE');
SID S

B.FIND THE AGE OF YOUNGEST SAILOR WHO IS ELIGIBLE TO VOTE FOR EACH RATING LEVEL WITH AT LEAST TWO SUCH SAILORS.

SQL> SELECT S.RATING, MIN(S.AGE) AS MINAGE FROM SAILORS S WHERE S.AGE>18 GROUP BY S.RATING HAVING COUNT(*) > 1

C. FIND THE SNAME , BID AND RESERVATION DATE FOR EACH RESERVATION.

SQL> SELECT S.SNAME, R.BID, R.DAY FROM SAILORS S, RESERVES R WHERE S.SID = R.SID

D.FIND THE AGES OF SAILORS WHOSE NAME BEGIN AND END WITH B AND HAS AT LEAST 3 CHARACTERS.

SQL> SELECT S.SID, S.AGE FROM SAILORS S WHERE S.SNAME LIKE 'B_%B';

E.LIST IN ALPHABETIC ORDER ALL SAILORS WHO HAVE RESERVED RED BOAT.

SQL>SELECT S.NAME, S.AGE FROM SAILORS S, RESERVES R, BOATS B WHERE S.ID = R.SID AND R.BID = B.ID AND B.COLOR = 'RED' ORDER BY S.NAME;

F.FIND THE AGE OF YOUNGEST SAILOR FOR EACH RATING LEVEL.

SQL> SELECT S.RATING, MIN (S.AGE) FROM SAILORS S GROUP BY S.RATING

Assignment 18:

6. a. List the Vendors who have delivered products within 6 months from order date.

b. Display the Vendor details who have supplied both Assembled and Subparts.

c. Display the Sub parts by grouping the Vendor type (Local or NonLocal).

d. Display the Vendor details in ascending order.

e. Display the Sub part which costs more than any of the Assembled parts.

f. Display the second maximum cost Assembled part

```
SQL> create table orderinfo (orderid int primary key,product varchar(25),orderdate date);

Table created.

SQL> desc orderinfo;
 Name                                      Null?    Type
 ----------------------------------------------------------------------- -------- ------------------------------------------------
 ORDERID                                   NOT NULL NUMBER
 PRODUCT                                            VARCHAR2(20)
 ORDERDATE

SQL> create table vendor(vid int primary key,vname varchar(25),deliverydate date,orderid int
references orderinfo(orderid),vtype varchar(25));

Table created.

SQL> desc vendor;
 Name                                      Null?    Type
 ----------------------------------------------------------------------- -------- ------------------------------------------------
 VID                                       NOT NULL NUMBER
 VNAME                                              VARCHAR2(20)
 DELIEVERYDATE                                      DATE
 ORDERID                                            NUMBER
 VTYPE                                              VARCHAR2(25)
```

SQL> create table product (pid int primary key,ptype varchar(25),pcost int,vid int references vendor(vid),pname varchar(25));

Table created.

SQL> desc product;

Name	Null?	Type
PID	NOT NULL	NUMBER(38)
PTYPE		VARCHAR2(25)
PCOST		NUMBER(38)
VID		NUMBER(38)
PNAME		VARCHAR2(25)

SQL> create table subpart (subpartname varchar(25),subpartcost int,pid int references product(pid));

Table created.

SQL> desc subpart;

Name	Null?	Type
SUBPARTNAME		VARCHAR2(25)
SUBPARTCOST		NUMBER(38)
PID		NUMBER(38)

SQL> select * from orderinfo;

ORDERID	PRODUCT	ORDERDATE
3	computer	04-APR-24
4	car	07-SEP-23
5	mobile	28-MAY-22

SQL> select * from vendor;

VID	VNAME	DELIEVERY	ORDERID	VTYPE
11	john	10-JUN-24	3	local
12	david	16-NOV-23	4	nonlocal
13	mike	14-JAN-23	5	local

SQL> select * from product;

PID	PTYPE	PCOST	VID	PNAME
111	subpart	1000	11	computer
112	assembled	700	12	car
113	assembled	500	13	mobile

SQL> select * from subpart;

SUBPARTNAME	SUBPARTCOST	PID
mouse	300	111
cpu	800	111
keyboard	200	111

a. List the Vendors who have delivered products within 6 months from order date.

SQL> select vid,vname,delieverydate,vtype,vendor.orderid from vendor,orderinfo where vendor.orderid=orderinfo.orderid and months_between(orderdate,delieverydate)<=6;

VID VNAME DELIEVERY VTYPE ORDERID

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

11 john 10-JUN-24 local 3

12 david 16-NOV-23 nonlocal 4

13 mike 14-JAN-23 local 5

b.. Display the Vendor details who have supplied both Assembled and Subparts.

select vendor.vid,vname,delieverydate,vtypefrom vendor,product where vendor.vid=product.vid and ptype in ('assembled','subparts') ;

select vendor.vid,vname,delieverydate,vtype from vendor,product where vendor.vid=product.vid and ptype in ('assembled','subparts')

SQL> /

VID VNAME DELIEVERY VTYPE

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

12 david 16-NOV-23 nonlocal

13 mike 14-JAN-23 local

c.Display the Sub parts by grouping the Vendor type (Local or NonLocal).

select vtype,subpartname from vendor join product on vendor.vid=product.vid join subpart on product.pid=subpart.pid group by vtype, subpartname;

SQL> select vtype,subpartname from vendor join product on vendor.vid=product.vid join subpart on product.pid=subpart.pid
2 group by vtype,subpartname;

VTYPE SUBPARTNAME

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

local mouse

local keyboard

local cpu

d.Display the Vendor details in ascending order.

```
select * from vendor order by vname asc;
```

```
VID VNAME              DELIEVERY    ORDERID VTYPE
---------- -------------------- --------- ---------- -------------------------
        12 david              16-NOV-23          4              nonlocal
        11 john               10-JUN-24          3                 local
        13 mike               14-JAN-23          5                 local
```

e.Display the Sub part which costs more than any of the Assembled parts

```
SQL> /
select subpartname,subpartcost,pid from subpart where subpartcost >
(select max(pcost) from product where ptype='assembled');
```

```
SUBPARTNAME            SUBPARTCOST       PID
------------------------ ----------- ----------
cpu                                800                  111.
```

f.Display the second maximum cost Assembled part

```
select pid,ptype,pcost from product where pcost = (
select max(pcost) from product where pcost<(
select max(pcost) from product where ptype='assembled'));
select max(pcost) from product where ptype='assembled'));
```

```
     PID PTYPE                    PCOST
---------- ------------------------ ----------
     113 assembled                   500
```

Week-4

INTRODUCTION TO PL/ SQL :

PL/SQL is a block structured language that enables developers to combine the power of SQL with procedural statements .All the statements of a block are passed to oracle engine all at once which increases processing speed and decreases the traffic.

Basics of PL/SQL

- •PL/SQL stands for Procedural Language extensions to the Structured Query Language (SQL).
- PL/SQL is a combination of SQL along with the procedural features of programming languages.
- Oracle uses a PL/SQL engine to processes the PL/SQL statements.
- PL/SQL includes procedural language elements like conditions and loops. It allows declaration of constants and variables, procedures and functions, types and variable of those types and triggers.

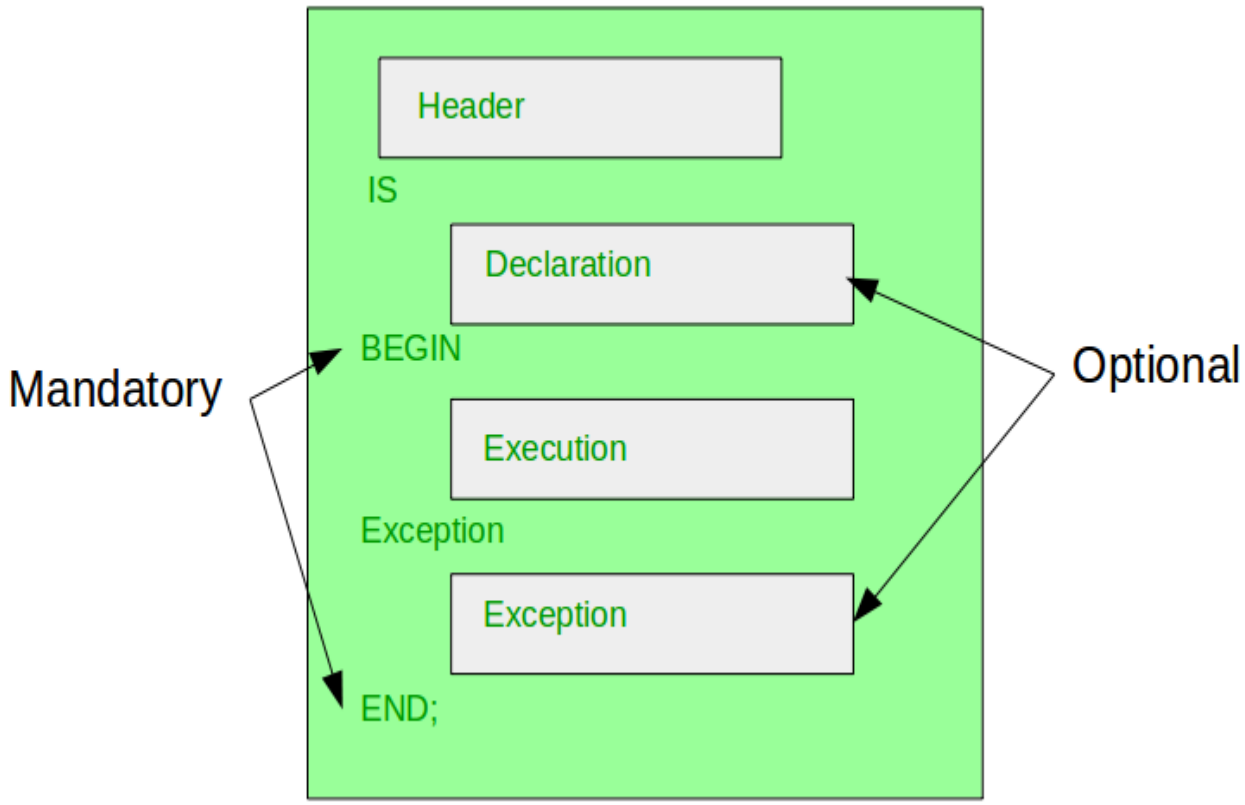
Disadvantages of SQL:

- SQL doesn't provide the programmers with a technique of condition checking, looping and branching.
- SQL statements are passed to Oracle engine one at a time which increases traffic and decreases speed.
- SQL has no facility of error checking during manipulation of data.

Features of PL/SQL:

1. PL/SQL is basically a procedural language, which provides the functionality of decision making, iteration and many more features of procedural programming languages.
2. PL/SQL can execute a number of queries in one block using single command.
3. One can create a PL/SQL unit such as procedures, functions, packages, triggers, and types, which are stored in the database for reuse by applications.
4. PL/SQL provides a feature to handle the exception which occurs in PL/SQL block known as exception handling block.
5. Applications written in PL/SQL are portable to computer hardware or operating system where Oracle is operational.
6. PL/SQL Offers extensive error checking.

Block Definition Structure:
In PL/SQL, All statements are classified into units that is called Blocks. PL/SQL blocks can include variables, SQL statements, loops, constants, conditional statements and exception handling. Blocks can also build a function or a procedure or a package.



The Declaration section: Code block start with a declaration section, in which memory variables, constants, cursors and other oracle objects can be declared and if required initialized.
The Begin section: Consist of set of SQL and PL/SQL statements, which describe processes that have to be applied to table data. Actual data manipulation, retrieval, looping and branching constructs are specified in this section.
The Exception section: This section deals with handling errors that arise during execution data manipulation statements, which make up PL/SQL code block. Errors can arise due to syntax, logic and/or validation rule.
The End section: This marks the end of a PL/SQL block.

1. **Example to show how to declare variables in PL/SQL :**

```
SQL> SET SERVEROUTPUT ON;

 SQL> DECLARE

    var1 INTEGER;

    var2 REAL;

    var3 varchar2(20) ;

BEGIN

    null;

END;

/
```

1. Output:

```
PL/SQL procedure successfully completed.
```

1. **Explanation*:***
   - **SET SERVEROUTPUT ON**: It is used to display the buffer used by the dbms_output.
   - **var1 INTEGER :** It is the declaration of variable, named ***var1*** which is of integer type. There are many other data types that can be used like float, int, real, smallint, long etc. It also supports variables used in SQL as well like NUMBER(prec, scale), varchar, varchar2 etc.
   - **PL/SQL procedure successfully completed.:** It is displayed when the code is compiled and executed successfully.
   - **Slash (/) after END;:** The slash (/) tells the SQL*Plus to execute the block.
   - **Assignment operator (:=)** : It is used to assign a value to a variable.
   - **Displaying Output**: The outputs are displayed by using DBMS_OUTPUT which is a built-in package that enables the user to display output, debugging information, and send messages from PL/SQL blocks, subprograms, packages, and triggers. Let us see an example to see how to display a message using PL/SQL :

2) A program for maximum number between two.

```
DECLARE
 a NUMBER;
 b NUMBER;
 BEGIN
 a:=&a;
 b:=&b;
 if(a>b) then
 dbms_output.Put_line('Max number is '||a);
 else
 dbms_output.Put_line('Max number is '||b);
 end if;
 END;
 /
Enter value for a: 11
old   5: a:=&a;
new   5: a:=11;
Enter value for b: 22
old   6: b:=&b;
new   6: b:=22;
Max number is 22
PL/SQL procedure successfully completed.
```

3) Square of a number.

```
SQL> set serveroutput on;
SQL>  declare
  2   x integer;
  3   begin
  4   x:=&x;
  5   dbms_output.put_line('square of number: '||(x*x));
  6   end;
  7   /
Enter value for x: 10
old   4:  x:=&x;
new   4:  x:=10;
square of number: 100

PL/SQL procedure successfully completed.
```

Assignment - 19

Q.1 a) Write a pl/sql program to swap two numbers.

```
declare
 -- declare variable num1, num2
 -- and temp of datatype number
 num1 number;
 num2 number;
 temp number;
 begin
 num1:=1000;
 num2:=2000;
 -- print result before swapping
 dbms_output.put_line('before');
 dbms_output.put_line('num1 = '|| num1 ||' num2 = '|| num2);
 --swapping of numbers num1 and num2
 temp := num1;
 num1 := num2;
 num2 := temp;
 -- print result after swapping
 dbms_output.put_line('after');
 dbms_output.put_line('num1 = '|| num1 ||' num2 = '|| num2);
 end; before
num1 = 1000 num2 = 2000
after
num1 = 2000 num2 = 1000
PL/SQL procedure successfully completed.
```

Q.1 b) Write a pl/sql program to find the largest of three numbers.

```
DECLARE
  NUMBER := 50;
  b NUMBER := 60;
  c NUMBER := 20;
   BEGIN
   IF a > b AND a > c THEN
   dbms_output.Put_line('Greatest number is '||a);
  ELSIF b > a AND b > c THEN
   dbms_output.Put_line('Greatest number is ' ||b);
 ELSE
  dbms_output.Put_line('Greatest number is '  ||c);
 END IF;
 END;
  /
Greatest number is 60
PL/SQL procedure successfully completed.
```

Assignment - 20

Q.2 a) Write a pl/sql program to find total and average of 3 subjects and display the grades.

```
declare
 s1 number(10);
 S2 number(10);
 s3 number(10);
 total number(10);
 per number(10);
  begin
 s1:=70;
 s2:=70;
s3:=75;
 total:=(s1+s2+s3);
 per:=(total/300)*100;
  if s1<40 or s2<40 or s3<40  then
 dbms_output.put_line('FAIL');
 end if;
 if 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;
   /
GRADE B
PERCENTAGE IS 72

PL/SQL procedure successfully completed.
```

Q.2 b) Write a pl/sql program to find sum of digits in a given numbers.

```
SQL> set serveroutput on
declare
 n integer;
s integer:=0;
 r integer:=0;
 begin
 n:=&n;
 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;
  /
Enter value for n: 123
old   6: n:=&n;
new   6: n:=123;
sum of digits of given number is6

PL/SQL procedure successfully completed.
```

Assignment - 21

Q.3) a Write a PL/SQL program to display the number in reverse order.

```
SQL> set serveroutput on
SQL> declare
num1
number(5); num2
number(5); rev
number(5); begin
num1:=&num1;
rev:=0;
while num1>0
 loop
 num2:=num1 mod 10;
 rev:=num2+(rev*10);
 num1:=floor(num1/10);
 end loop;
 dbms_output.put_line('Reverse number is:   '||rev);
end;
 /
Enter value for num1: 15
old   6: num1:=&num1;
new   6: num1:=15;
Reverse number is: 51
PL/SQL procedure successfully completed.
```

Q.3-b) Write a PL/SQL program to check whether the given number is prime or not.

```
SQL> declare
  num
  number; i
  number:=1; c
  number:=0;
  begin
  num:=&num;
  for i in 1..num
  loop
 if((mod(num,i))=0)
 then
 c:=c+1;
  end if;
 end
 loop;
 if(c>2)
 then
 dbms_output.put_line(num||' Given no. is not a prime');
 else
 dbms_output.put_line(num||' Given no.is prime'); end if;
 end;
  /
Enter value for num: 17
old   7: num:=&num;
new   7: num:=17;
17  Given no. is prime
PL/SQL procedure successfully completed.
```

Q.4 a) Write a PL/SQL program to find the 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;
 /
Enter value for n: 5
old   3: n number(4):=&n;
new   3: n number(4):=5;
the factorial of 5is:120
PL/SQL procedure successfully completed.
```

Q.4 b) Write a PL/SQL program to calculate the area of a circle for a value of radius varying from 3 to 7. Store the radius and the corresponding values of calculated area in table areas. Consisting of two columns radius and area.

```
SQL> create table areas ( r number(2), area number (14,2));

Table created.

SQL> declare
 r number(5);
 area number(14,2);
 pi constant number (4,2):=3.14;
  begin
 r:=3;
  while r<=7
  loop
  area:=pi*power(r,2);
  insert into areas values(r,area );
  r:=r+1;
  end loop;
 end;
  /
PL/SQL procedure successfully completed.
SQL> select * from areas;
```

R	AREA
3	28.26
4	50.24
5	78.5
6	113.04
7	153.86

Assignment - 23

Q.5 a) Write a PL/SQL program to accept a string and remove the vowels from the string. (When 'hello' passed to the program it should display 'Hll' removing e and o from the world Hello).

```
SQL> set serveroutput on
SQL> declare
 vstring varchar2(20):='&vstring';
 vnewstring varchar2(100);
 begin
 vnewstring := regexp_replace(vstring, '[aeiouAEIOU]','');
 dbms_output.put_line('The new string is: ' || vnewstring);
  end;
  /
Enter value for vstring: hello
old   2: vstring varchar2(20):='&vstring';
new   2: vstring varchar2(20):='hello';
The new string is: hll
PL/SQL procedure successfully completed.
```

Assignment 24

CONCEPT OF CURSORS

we will discuss the cursors in PL/SQL. Oracle creates a memory area, known as the context area, for processing an SQL statement, which contains all the information needed for processing the statement; for example, the number of rows processed, etc.

A **cursor** is a pointer to this context area. PL/SQL controls the context area through a cursor. A cursor holds the rows (one or more) returned by a SQL statement. The set of rows the cursor holds is referred to as the **active set**.

You can name a cursor so that it could be referred to in a program to fetch and process the rows returned by the SQL statement, one at a time. There are two types of cursors −

- Implicit cursors
- Explicit cursors

Implicit Cursors

Implicit cursors are automatically created by Oracle whenever an SQL statement is executed, when there is no explicit cursor for the statement. Programmers cannot control the implicit cursors and the information in it.

Whenever a DML statement (INSERT, UPDATE and DELETE) is issued, an implicit cursor is associated with this statement. For INSERT operations, the cursor holds the data that needs to be inserted. For UPDATE and DELETE operations, the cursor identifies the rows that would be affected.

In PL/SQL, you can refer to the most recent implicit cursor as the **SQL cursor**, which always has attributes such as **%FOUND, %ISOPEN, %NOTFOUND**, and **%ROWCOUNT**. The SQL cursor has additional attributes, **%BULK_ROWCOUNT** and **%BULK_EXCEPTIONS**, designed for use with the **FORALL** statement. The following table provides the description of the most used attributes −

S.No	Attribute & Description
1	**%FOUND** Returns TRUE if an INSERT, UPDATE, or DELETE statement affected one or more rows or a SELECT INTO statement returned one or more rows. Otherwise, it returns FALSE.
2	**%NOTFOUND** The logical opposite of %FOUND. It returns TRUE if an INSERT, UPDATE, or DELETE statement affected no rows, or a SELECT INTO statement returned no rows. Otherwise, it returns FALSE.

3	**%ISOPEN** Always returns FALSE for implicit cursors, because Oracle closes the SQL cursor automatically after executing its associated SQL statement.
4	**%ROWCOUNT** Returns the number of rows affected by an INSERT, UPDATE, or DELETE statement, or returned by a SELECT INTO statement.

Any SQL cursor attribute will be accessed as **sql%attribute_name** as shown below in the example.

Example

We will be using the CUSTOMERS table we had created and used in the previous chapters.

Select * from customers;

```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
+----+----------+-----+-----------+----------+
```

The following program will update the table and increase the salary of each customer by 500 and use the **SQL%ROWCOUNT** attribute to determine the number of rows affected −

```
DECLARE
  total_rows number(2);
BEGIN
  UPDATE customers
  SET salary = salary + 500;
  IF sql%notfound THEN
    dbms_output.put_line('no customers selected');
  ELSIF sql%found THEN
    total_rows := sql%rowcount;
    dbms_output.put_line( total_rows || ' customers selected ');
  END IF;
END;
```

```
/
```

When the above code is executed at the SQL prompt, it produces the following result −

6 customers selected

PL/SQL procedure successfully completed.

If you check the records in customers table, you will find that the rows have been updated −

Select * from customers;

```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2500.00 |
|  2 | Khilan   |  25 | Delhi     |  2000.00 |
|  3 | kaushik  |  23 | Kota      |  2500.00 |
|  4 | Chaitali |  25 | Mumbai    |  7000.00 |
|  5 | Hardik   |  27 | Bhopal    |  9000.00 |
|  6 | Komal    |  22 | MP        |  5000.00 |
+----+----------+-----+-----------+----------+
```

Explicit Cursors

Explicit cursors are programmer-defined cursors for gaining more control over the **context area**. An explicit cursor should be defined in the declaration section of the PL/SQL Block. It is created on a SELECT Statement which returns more than one row.

The syntax for creating an explicit cursor is −

CURSOR cursor_name IS select_statement;

Working with an explicit cursor includes the following steps −

- Declaring the cursor for initializing the memory
- Opening the cursor for allocating the memory
- Fetching the cursor for retrieving the data
- Closing the cursor to release the allocated memory

Declaring the Cursor

Declaring the cursor defines the cursor with a name and the associated SELECT statement. For example −

```
CURSOR c_customers IS
  SELECT id, name, address FROM customers;
```

Opening the Cursor

Opening the cursor allocates the memory for the cursor and makes it ready for fetching the rows returned by the SQL statement into it. For example, we will open the above defined cursor as follows −

```
OPEN c_customers;
```

Fetching the Cursor

Fetching the cursor involves accessing one row at a time. For example, we will fetch rows from the above-opened cursor as follows −

```
FETCH c_customers INTO c_id, c_name, c_addr;
```

Closing the Cursor

Closing the cursor means releasing the allocated memory. For example, we will close the above-opened cursor as follows −

```
CLOSE c_customers;
```

Example

Following is a complete example to illustrate the concepts of explicit cursors ;

```
DECLARE
  c_id customers.id%type;
  c_name customers.name%type;
  c_addr customers.address%type;
  CURSOR c_customers is
    SELECT id, name, address FROM customers;
BEGIN
  OPEN c_customers;
  LOOP
  FETCH c_customers into c_id, c_name, c_addr;
    EXIT WHEN c_customers%notfound;
    dbms_output.put_line(c_id || ' ' || c_name || ' ' || c_addr);
  END LOOP;
  CLOSE c_customers;
END;
/
```

When the above code is executed at the SQL prompt, it produces the following result −

1 Ramesh Ahmedabad

2 Khilan Delhi

3 kaushik Kota

4 Chaitali Mumbai

5 Hardik Bhopal

6 Komal MP

PL/SQL procedure successfully completed.

Assignment 25

we will discuss Triggers in PL/SQL. Triggers are stored programs, which are automatically executed or fired when some events occur. Triggers are, in fact, written to be executed in response to any of the following events −

- A **database manipulation (DML)** statement (DELETE, INSERT, or UPDATE)
- A **database definition (DDL)** statement (CREATE, ALTER, or DROP).
- A **database operation** (SERVERERROR, LOGON, LOGOFF, STARTUP, or SHUTDOWN).

Triggers can be defined on the table, view, schema, or database with which the event is associated.

Benefits of Triggers

Triggers can be written for the following purposes −

- Generating some derived column values automatically
- Enforcing referential integrity
- Event logging and storing information on table access
- Auditing
- Synchronous replication of tables
- Imposing security authorizations
- Preventing invalid transactions

Creating Triggers

The syntax for creating a trigger is −

```
CREATE [OR REPLACE ] TRIGGER trigger_name
{BEFORE | AFTER | INSTEAD OF }
{INSERT [OR] | UPDATE [OR] | DELETE}
[OF col_name]
ON table_name
[REFERENCING OLD AS o NEW AS n]
[FOR EACH ROW]
```

```
WHEN (condition)
DECLARE
  Declaration-statements
BEGIN
  Executable-statements
EXCEPTION
  Exception-handling-statements
END;
```

Where,

- CREATE [OR REPLACE] TRIGGER trigger_name − Creates or replaces an existing trigger with the *trigger_name*.
- {BEFORE | AFTER | INSTEAD OF} − This specifies when the trigger will be executed. The INSTEAD OF clause is used for creating trigger on a view.
- {INSERT [OR] | UPDATE [OR] | DELETE} − This specifies the DML operation.
- [OF col_name] − This specifies the column name that will be updated.
- [ON table_name] − This specifies the name of the table associated with the trigger.
- [REFERENCING OLD AS o NEW AS n] − This allows you to refer new and old values for various DML statements, such as INSERT, UPDATE, and DELETE.
- [FOR EACH ROW] − This specifies a row-level trigger, i.e., the trigger will be executed for each row being affected. Otherwise the trigger will execute just once when the SQL statement is executed, which is called a table level trigger.
- WHEN (condition) − This provides a condition for rows for which the trigger would fire. This clause is valid only for row-level triggers.

Example

To start with, we will be using the CUSTOMERS table we had created and used in the previous chapters −

Select * from customers;

```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
+----+----------+-----+-----------+----------+
```

The following program creates a **row-level** trigger for the customers table that would fire for INSERT or UPDATE or DELETE operations performed on the CUSTOMERS table. This trigger will display the salary difference between the old values and new values −

```
CREATE OR REPLACE TRIGGER display_salary_changes
BEFORE DELETE OR INSERT OR UPDATE ON customers
FOR EACH ROW
WHEN (NEW.ID > 0)
DECLARE
  sal_diff number;
BEGIN
  sal_diff := :NEW.salary  - :OLD.salary;
  dbms_output.put_line('Old salary: ' || :OLD.salary);
  dbms_output.put_line('New salary: ' || :NEW.salary);
  dbms_output.put_line('Salary difference: ' || sal_diff);
END;
/
```

When the above code is executed at the SQL prompt, it produces the following result −

Trigger created.