

# **Diploma Engineering**

## **Laboratory Manual**

### **(Relational Database Management Systems) (4330702)**

[Computer Engineering Semester - III]

Enrolment No	
Name	
Branch	
Academic Term	
Institute	



**Directorate Of Technical Education  
Gandhinagar - Gujarat**

### **DTE's Vision**

- To provide globally competitive technical education.
- To provide globally competitive technical education;
- Remove geographical imbalances and inconsistencies;
- Develop student friendly resources with a special focus on girls' education and support to weaker sections;
- Develop programs relevant to industry and create a vibrant pool of technical professionals.

### **Institute's Vision**

### **Institute's Mission**

### **Department's Vision**

### **Department's Mission**

# Certificate

This is to certify that Mr./Ms .....  
Enrollment No. .... of 3<sup>rd</sup> Semester of *Diploma in Computer Engineering* of ..... (GTU Code) has  
satisfactorily completed the term work in course **Relational Database Management Systems** for the academic year: ..... Term: Odd  
prescribed in the GTU curriculum.

Place: .....

Date: .....

**Signature of Course Faculty**

**Head of the Department**

## Preface

The primary aim of any laboratory/Practical/field work is enhancement of required skills as well as creative ability amongst students to solve real time problems by developing relevant competencies in psychomotor domain. Keeping in view, GTU has designed competency focused outcome-based curriculum -2021 (COGC-2021) for Diploma engineering programmes. In this more time is allotted to practical work than theory. It shows importance of enhancement of skills amongst students and it pays attention to utilize every second of time allotted for practical amongst Students, Instructors and Lecturers to achieve relevant outcomes by performing rather than writing practice in study type. It is essential for effective implementation of competency focused outcome- based green curriculum-2021. Every practical has been keenly designed to serve as a tool to develop & enhance relevant industry needed competency in each and every student. These psychomotor skills are very difficult to develop through traditional chalk and board content delivery method in the classroom. Accordingly, this lab manual has been designed to focus on the industry defined relevant outcomes, rather than old practice of conducting practical to prove concept and theory.

By using this lab manual, students can read procedure one day in advance to actual performance day of practical experiment which generates interest and also, they can have idea of judgement of magnitude prior to performance. This in turn enhances predetermined outcomes amongst students. Each and every Experiment /Practical in this manual begins by competency, industry relevant skills, course outcomes as well as practical outcomes which serve as a key role for doing the practical. The students will also have a clear idea of safety and necessary precautions to be taken while performing experiment.

This manual also provides guidelines to lecturers to facilitate student-centred lab activities for each practical/experiment by arranging and managing necessary resources in order that the students follow the procedures with required safety and necessary precautions to achieve outcomes. It also gives an idea that how students will be assessed by providing Rubrics.

RDBMS is the cornerstone of modern data management, providing a structured and organized approach to storing, retrieving, and manipulating data. By leveraging the power of tables, relationships, and queries, RDBMS allows us to harness the potential of data, enabling businesses to make informed decisions, optimize processes, and gain valuable insights. In this subject, we delve into the fundamental concepts, principles, and practices of RDBMS, exploring its architecture, data modelling techniques, query optimization, and transaction management.

Although we try our level best to design this lab manual, but always there are chances of improvement. We welcome any suggestions for improvement.

### Programme Outcomes (POs):

**PO1: Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the *engineering* problems.

**PO2: Problem analysis:** Identify and analyse well-defined *engineering* problems using codified standard methods.

**PO3: Design/ development of solutions:** Design solutions for *engineering* well-defined technical problems and assist with the design of systems components or processes to meet specified needs.

**PO4: Engineering Tools, Experimentation and Testing:** Apply modern *engineering* tools and appropriate technique to conduct standard tests and measurements.

**P05: Engineering practices for society, sustainability and environment:** Apply appropriate technology in context of society, sustainability, environment and ethical practices.

**P06: Project Management:** Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.

**P07: Life-long learning:** Ability to analyse individual needs and engage in updating in the context of technological changes *in field of engineering*.

### Practical Outcome - Course Outcome matrix

**Course Outcomes (COs):**

CO1: Perform queries on datasets using SQL\*Plus

CO2: Perform joins, sub-queries and nested queries on multiple tables using SQL\*plus

CO3: Apply rules on datasets using SQL\*Plus constraints

CO4: Write PL/SQL block using concept of Cursor Management, Error Handling, Package and Triggers

CO5: Apply various Normalization techniques.

S. No.	Practical Outcome	CO1	CO2	CO3	CO4	CO5
1.	Implement SQL queries to perform various DDL Commands.	✓	-	-	-	-
2.	a. Implement SQL queries to perform various DML Commands. b. Retrieve data using SELECT command and various SQL operators.	✓		-	-	-
3.	Perform queries for TCL and DCL Commands	✓		-	-	-
4.	Implement SQL queries using Date functions like add-months, months-between, round, nextday, truncate etc	-	✓	-	-	-
5.	Implement SQL queries using Numeric functions like abs, ceil, power, mod, round, trunc, sqrt etc. and Character Functions like initcap, lower, upper, ltrim, rtrim, replace, substring, instr etc.	-	✓	-	-	-
6.	Implement SQL queries using Conversion Functions like to- char, to-date, to-number and Group functions like Avg, Min, Max, Sum, Count, Decode etc.	-	✓	-	-	-
7.	Implement SQL queries using Group by, Having and Order by Clause	-	✓	-	-	-
8.	Implement SQL queries using simple Case Operations and using Group functions and Case operations for getting summary data	-	✓	-	-	-
9.	Implement SQL queries using Set operators like Union, union all, Intersect, Minus etc.	-	✓	-	-	-
10.	Retrieve data spread across various tables or same table using various Joins.	-	✓	-	-	-
11.	Retrieve data from multiple tables using Sub-queries (Multiple, Correlated) (write minimum 3 level subquery)	-	✓	-	-	-
12.	Perform queries to Create, alter and update views	-	-	✓	-	-

<b>13.</b>	Implement Practical-1 again with Domain Integrity, Entity Integrity and Referential Integrity constraints.	-	-	✓	-	-
<b>14.</b>	Perform queries to Create synonyms, sequence and index	-	-	✓	-	-
<b>15.</b>	Implement PL/SQL programs using control structures	-	-	-	✓	-
<b>16.</b>	Implement PL/SQL programs using Cursors	-	-	-	✓	-
<b>17.</b>	Implement PL/SQL programs using exception handling.	-	-	-	✓	-
<b>18.</b>	Implement user defined procedures and functions using PL/SQL blocks	-	-	-	✓	-
<b>19.</b>	Perform various operations on packages.	-	-	-	✓	-
<b>20.</b>	Implement various triggers	-	-	-	✓	-
<b>21.</b>	Micro-Project(E-R Diagrams and Normalization)	✓	✓	✓	✓	✓

## Industry Relevant Skills

The following industry relevant skills of the competency “**Design database tables, writing optimized queries for integration and other application, creating program views, functions and stored procedure.**” are expected to be developed in the student by undertaking the practical of this laboratory manual.

## Guidelines to Course Faculty

1. Course faculty should demonstrate experiment with all necessary implementation strategies described in curriculum.
2. Course faculty should explain industrial relevance before starting of each experiment.
3. Course faculty should Involve & give opportunity to all students for hands on experience.
4. Course faculty should ensure mentioned skills are developed in the students by asking.
5. Utilise 2 hrs of lab hours effectively and ensure completion of write up with quiz also.
6. Encourage peer to peer learning by doing same experiment through fast learners.

## Instructions for Students

1. Organize the work in the group and make record of all observations.
2. Students shall develop maintenance skill as expected by industries.
3. Student shall attempt to develop related hand-on skills and build confidence.
4. Student shall develop the habits of evolving more ideas, innovations, skills etc.
5. Student shall refer technical magazines and data books.
6. Student should develop habit to submit the practical on date and time.
7. Student should well prepare while submitting write-up of exercise.



### Continuous Assessment Sheet

Enrolment No:

Term:

Name:

Sr	Practical Outcome/Title of experiment	Page	Date	Marks (25)	Sign
1.	Implement SQL queries to perform various DDL Commands.				
2.	a. Implement SQL queries to perform various DML Commands. b. Retrieve data using SELECT command and various SQL operators.				
3.	Perform queries for TCL and DCL Commands				
4.	Implement SQL queries using Date functions like add-months, months-between, round, nextday, truncate etc				
5.	Implement SQL queries using Numeric functions like abs, ceil, power, mod, round, trunc, sqrt etc. and Character Functions like initcap, lower, upper, ltrim, rtrim, replace, substring, instr etc.				
6.	Implement SQL queries using Conversion Functions like to- char, to-date, to-number and Group functions like Avg, Min, Max, Sum, Count, Decode etc.				
7.	Implement SQL queries using Group by, Having and Order by Clause				
8.	Implement SQL queries using simple Case Operations and using Group functions and Case operations for getting summary data				
9.	Implement SQL queries using Set operators like Union, union all, Intersect, Minus etc.				
10.	Retrieve data spread across various tables or same table using various Joins.				
11.	Retrieve data from multiple tables using Sub-queries (Multiple, Correlated) (write minimum 3 level subquery)				
12.	Perform queries to Create, alter and update views				
13.	Implement Domain Integrity, Entity Integrity and Referential Integrity constraints.				
14.	Perform queries to Create synonyms, sequence and index				
15.	Implement PL/SQL programs using control structures				
16.	Implement PL/SQL programs using Cursors				
17.	Implement PL/SQL programs using exception handling.				

<b>18.</b>	Implement user defined procedures and functions using PL/SQL blocks				
<b>19.</b>	Perform various operations on packages.				
<b>20.</b>	Implement various triggers				
<b>21.</b>	Micro-Project (E-R Diagrams and Normalization)				

Date: .....

**Practical No.1: Implement SQL queries to perform various DDL(Data Definition Language) Commands. (Create minimum 5 tables as per given definition with different data types and operate upon them).**

1. Create a table named "**users**" with the following columns/ attributes:  
id (integer), name (varchar2(50)), email (varchar2(100))  
password (varchar2(100))
2. Create a table named "**products**" with the following columns:  
id (integer), name (varchar2(100)), price (decimal(10,2))  
description (varchar2(500))
3. Create a table named "**orders**" with the following columns:  
id (integer), user\_id (integer), product\_id (integer)  
quantity (integer), price(number(5,2))
4. Create a table named "**Students**" with the following columns/ attributes:  
student\_id (integer), name (varchar2(50)), age (integer), gender (varchar2(6))
5. Create a table named "**Courses**" with the following columns/ attributes:  
course\_id (integer), name (varchar2(50)), credits (integer), instructor  
(varchar2(50))
6. Change name column with width of 100 characters in Users table.
7. Add column address (varchar2(100)) in students table.
8. Remove column price from Orders table.
9. Remove all data of Products table while keeping the structure intact.
10. Remove Courses table completely along with data and structure. (create it again afterwards for future reference)

**A. Objective:**

To perform various Data Definition Language (DDL) commands to manage the structure of databases, tables, and indexes within a database management system (DBMS) using SQL commands

**B. Expected Program Outcomes (POs): PO1, PO5, PO7**

**C. Expected Skills to be developed based on competency:**

Database design, SQL syntax and queries related to table creation, alteration and deletion.

**D. Expected Course Outcomes(Cos)**

Perform queries on datasets using SQL\*Plus

**E. Practical Outcome(PRo)**

Write data definition queries, compilation, debugging, executing using oracle software

**F. Expected Affective Domain Outcome (ADos)**

- Follow safety practices.
- Practice good housekeeping.
- Demonstrate working as a leader/a team member.
- Maintain tools and equipment

- Follow ethical practices.

**G. Prerequisite Theory:**

❖ **SQL DATA TYPES: -**

- CHAR (size)
- VARCHAR (size)/VARCHAR2 (size)
- DATE
- NUMBER (P, S)
- LONG
- RAW/LONGRAW

➤ **CHAR (SIZE)**

- This data type is used to store alphanumeric data of fixed length.
- The size represents length of character to be stored.
- This data type can store maximum 2000 characters.
- This is faster to access but requires more storage.

➤ **VARCHAR (size)/VARCHAR2 (size)**

- This data type is used to store variable length alphanumeric data.
- The maximum size of this data type is 4000 characters.
- This is slower to access in comparison to CHAR but is efficient for storage.

➤ **DATE**

- This data type is used to store date and time.
- The standard format is DD-MON-YY for example 21-JUN-09.
- Date time stores time in the 24-hours format.
- Range of this data type is January 1, 4712 B.C. to December 31, 4712A.D.

➤ **NUMBER (P,S)**

- The NUMBER data type is used store numbers (both Integer and floating-point numbers).
- We can store number up to 38 characters long.
- The precision (P), determines the maximum length of number, and scale (s), determines the decimal point sign(.) on the right of a number (P).

➤ **LONG**

- This data type is used to store variable length character strings upto 2GB.
- LONG data can be used to store binary data in ASCII format.

➤ **RAW/LONG RAW**

- The RAW/LONG RAW data type is used to store binary data, such as digitized picture or image.
- RAW data type can have a maximum length of 255 bytes.
- LONG RAW data type can contain upto 2GB.

❖ **DESCRIBE COMMAND**

- DESCRIBE or DESC (both are same) command to describe the structure of a table.
- DESCRIBE or DESC command shows the structure of table which include name of the column, data-type of column and the nullability which means, that column can contain null values or not.
- All of these features of table are described at the time of Creation of table.

**Syntax:**

SQL> DESCRIBE TABLE NAME;  
OR

SQL> DESC TABLE NAME;

❖ **SQL DDL (DATA DEFINITION LANGUAGE) COMMANDS: -**

➤ CREATE, ALTER, RENAME, TRUNCATE, DROP

➤ **CREATE**

▪ Create command is used to create a table in database.

**Syntax:**

SQL> Create table table name  
( columnname\_1 datatype(datasize), columnname\_2 datatype(datasize),  
.....  
columnname\_ndatatype(datasize) );

**Example:**

SQL> Create table client\_master (client\_no varchar2(6), name varchar2(20), city  
varchar2(15), pincode number(8), state varchar2(15));

**Show the Structure of a Table:**

**Syntax:**

SQL> DESCRIBE CLIENT\_MASTER;

NAME	NULL?	TYPE
CLIENT_NO		VARCHAR2(6)
NAME		VARCHAR2(20)
CITY		VARCHAR2(15)
PINCODE		NUMBER(8)
STATE		VARCHAR2(15)

➤ **ALTER**

▪ ALTER command is used to ALTER (Modify structure of) a table in database.

▪ ALTER command is used in five ways.

1. To add new column in table.
2. To modify data type or data size of existing column of table.
3. To drop (remove) the existing column of table.
4. To rename the name of existing table.
5. To rename the name of existing column of table.

**Syntax: (1<sup>st</sup> way- to add new column)**

SQL> Alter table tablename add (  
new\_columnname\_1 datatype(datasize), new\_columnname\_2 datatype(datasize),  
.....  
new\_columnname\_ndatatype(datasize));

**Example:**

SQL> Alter Table Client\_Master Add(Mobile\_No Number(8), Email Char(15));

**Show the Structure of a Table:**

**Syntax:**

```
sql> describe client_master;
```

name	null?	type
client_no		varchar2(6)
name		varchar2(20)
city		varchar2(15)
pincode		number(8)
state		varchar2(15)
mobile_no		number(8)
email		char(15)

**Syntax: (2<sup>nd</sup> way-modify data type or data size)**

```
SQL> Alter table tablename modify (existing_columnname_1 new_datatype
(new_datasize),.....existing_columnname_n new_datatype(new_datasize));
```

**Example:**

```
SQL> alter table client_master modify(mobile_no number(10), email varchar2(15));
```

**Show the Structure of a Table:****Syntax:**

```
SQL> DESCRIBE CLIENT_MASTER;
```

NAME	NULL?	TYPE
CLIENT_NO		VARCHAR2(6)
NAME		VARCHAR2(20)
CITY		VARCHAR2(15)
PINCODE		NUMBER(8)
STATE		VARCHAR2(15)
MOBILE_NO		NUMBER(10)
EMAIL		VARCHAR2(15)

**Syntax: (3<sup>rd</sup> way-remove the column)**

```
SQL> ALTER TABLE TABLENAME DROP COLUMN COLUMNNAME;
```

**Example:**

```
SQL> ALTER TABLE CLIENT_MASTER DROP COLUMN MOBILE_NO;
```

**Show the Structure of a Table:****Syntax:**

```
SQL> DESCRIBE CLIENT_MASTER;
```

NAME	NULL?	TYPE
CLIENT_NO		VARCHAR2(6)
NAME		VARCHAR2(20)
CITY		VARCHAR2(15)
PINCODE		NUMBER(8)
STATE		VARCHAR2(15)
EMAIL		VARCHAR2(15)

**Syntax: (4th way-rename table name)**

SQL> ALTER TABLE OLD\_TABLENAME RENAME TO NEW\_TABLENAME;

Example:

SQL> ALTER TABLE CLIENT\_MASTER RENAME TO CLIENT\_MASTER\_COPY;

Show the Structure of a Table:

Syntax:

SQL> DESCRIBE CLIENT\_MASTER\_COPY;

NAME	NULL?	TYPE
CLIENT_NO		VARCHAR2(6)
NAME		VARCHAR2(20)
CITY		VARCHAR2(15)
PINCODE		NUMBER(8)
STATE		VARCHAR2(15)
EMAIL		VARCHAR2(15)

**Syntax: (5th way-rename the column name)**

SQL> ALTER TABLE TABLENAME RENAME COLUMN OLD\_COLUMNNAME TO NEW\_COLUMNNAME;

Example:

SQL> ALTER TABLE CLIENT\_MASTER RENAME COLUMN EMAIL TO EMAIL\_ID;

Show the Structure of a Table:

Syntax:

SQL> DESCRIBE CLIENT\_MASTER;

NAME	NULL?	TYPE
CLIENT_NO		VARCHAR2(6)
NAME		VARCHAR2(20)
CITY		VARCHAR2(15)
PINCODE		NUMBER(8)
STATE		VARCHAR2(15)
EMAIL_ID		VARCHAR2(15)
RENAME		

RENAME command is used to give new name of table.

Syntax:

SQL> RENAME OLD\_TABLENAME TO NEW\_TABLENAME;

Example:

SQL> RENAME CLIENT\_MASTER\_COPY RENAME TO CLIENT\_MASTER;

TRUNCATE

TRUNCATE command is used to remove all records from a table, including all memory allocated for the records.

Syntax:

SQL> TRUNCATE TABLE TABLE\_NAME;

Example:

SQL>TRUNCATE TABLE CLIENT\_MASTER;

DROP

DROP command is used to delete all records of a table along with its structure in database.

Syntax:

```
SQL> DROP TABLE TABLE_NAME;
```

Example:

```
SQL> DROP TABLE CLIENT_MASTER;
```

**H. Resources/Equipment Required**

Computer or laptop with a database management system (DBMS) software installed, such as Oracle

**I. Safety and necessary Precautions followed**

Shutdown computer system properly once the Lab hours are finished

**J. Source code / Solutions to queries:**







**K. Practical related Quiz/Exercise.**

1. The \_\_\_\_ command is used to completely remove a table from the database.
  - A. INSERT
  - B. ALTER
  - C. DROP
  - D. CREATE
2. The \_\_\_\_ command is used to modify columns in a table.
  - A. DROP
  - B. INSERT
  - C. ALTER
  - D. CREATE
3. Pick the element which you must specify while creating a table.
  - A. Column name
  - B. Column Data type
  - C. Column size
  - D. All of the above
4. Which of the following command is used to see the structure of a table?
  - A. UPDATE
  - B. SHOW
  - C. DESCRIBE
  - D. SPOOL
5. What does NUMBER (8, 2) in oracle mean?
  - A. It means there are 8 digits in total, 6 digits before the decimal and 2 after the decimal
  - B. It means there are 10 digits in total with 8 digits before the decimal and 2 after decimal
  - C. It means there are 2 digits before the decimal and 8 after the decimal point
  - D. None of the above.

**L. References / Suggestions**

<https://www.geeksforgeeks.org/ddl-commands-syntax/>  
<https://www.w3schools.com/sql/>  
<https://www.tutorialspoint.com/sql/index.htm>

**M. Assessment-Rubrics**

Excellent (21-25 Marks)	Good (15-20 Marks)	Satisfactory (7-14 Marks)	Fair (0-6 Marks)
<ul style="list-style-type: none"> <li>Student executes the queries with correct output.</li> <li>Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>Student executes the queries having correct output with external guidance.</li> <li>Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>Student executes the queries with incorrect output.</li> <li>Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>Student is not able to execute the queries.</li> <li>Student does not submit the lab report in specified time limit.</li> </ul>

**Signature with date**

Date: .....

**Practical No.2:**

- a. Implement SQL queries to perform various DML Commands.
  - I. Insert 5 records in Products table
  - II. Insert 5 records in Orders table
  - III. Update Product Name in products table
  - IV. Delete a product from Product table
- b. Retrieve data using SELECT command and various SQL operators.
  - I. Show all products with product id <5.
  - II. Show all Product IDs having price greater than 50 and quantity <5.
  - III. Show all employees
  - IV. Show employee name, salary of employees whose salary is <8000.
  - V. Show employee name and salary whose salary between 5000 and 15000.
  - VI. Show employees who are not in department 10.
  - VII. Show employees who are not earning any commission.
  - VIII. Show employee name and total earning of all employees.
  - IX. Show employees whose name starts with 'S' character.
  - X. Show employees whose name have minimum two 'A' characters.

**Note:**

- ✓ You can use tables created in PRACTICAL1 or default tables available in HR schema to perform any DML commands.
- ✓ **Show only top 2-3 sample rows in output if output is more than 5 rows.**
- ✓ You can list available default tables in HR Schema using below Query  
 SELECT OWNER, TABLE\_NAME FROM DBA\_TABLES WHERE OWNER='HR';

**HR Table Descriptions****Table EMPLOYEES**

Name	Null?	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
FIRST_NAME		VARCHAR2(20)
LAST_NAME	NOT NULL	VARCHAR2(25)
EMAIL		NOT NULL VARCHAR2(25)
PHONE_NUMBER		VARCHAR2(20)
HIRE_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
SALARY		NUMBER(8,2)
COMMISSION_PCT		NUMBER(2,2)
MANAGER_ID		NUMBER(6)
DEPARTMENT_ID		NUMBER(4)

**Table DEPARTMENTS**

Name	Null?	Type
DEPARTMENT_ID	NOT NULL	NUMBER(4)
DEPARTMENT_NAME	NOT NULL	VARCHAR2(30)
MANAGER_ID		NUMBER(6)
LOCATION_ID		NUMBER(4)

**Table JOBS**

Name	Null?	Type
JOB_ID	NOT NULL	VARCHAR2(10)
JOB_TITLE	NOT NULL	VARCHAR2(35)
MIN_SALARY		NUMBER(6)
MAX_SALARY		NUMBER(6)

**Table JOB\_HISTORY**

Name	Null?	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
START_DATE	NOT NULL	DATE
END_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
DEPARTMENT_ID		NUMBER(4)

**Table COUNTRIES**

Name	Null?	Type
COUNTRY_ID	NOT NULL	CHAR(2)
COUNTRY_NAME		VARCHAR2(40)
REGION_ID		NUMBER

**Table LOCATIONS**

Name	Null?	Type
LOCATION_ID	NOT NULL	NUMBER(4)
STREET_ADDRESS		VARCHAR2(40)
POSTAL_CODE		VARCHAR2(12)
CITY	NOT NULL	VARCHAR2(30)
STATE_PROVINCE		VARCHAR2(25)
COUNTRY_ID		CHAR(2)

Table REGIONS

Name	Null?	Type
REGION_ID	NOT NULL	NUMBER
REGION_NAME		VARCHAR2(25)

### A. Objective:

To perform various Data Manipulation Language (DML) commands is to enable users to interact with a database and modify its contents in a structured and controlled manner.

**B. Expected Program Outcomes (POs): P01,P05,P07**

**C. Expected Skills to be developed based on competency:**

## Compilation, debugging, executing SQL queries

#### D. Expected Course Outcomes(Cos)

## Perform queries on datasets using SQL\*Plus

### E. Practical Outcome(Pro)

Write data definition queries , compilation, debugging, executing using oracle software

### E. Expected Affective domain Outcome (ADos)

- Follow safety practices.
- Practice good housekeeping.
- Demonstrate working as a leader/a team member.
- Maintain tools and equipment
- Follow ethical practices.

### F. Prerequisite Theory:

### ❖ SQL DML (DATA MANIPLUATION LANGUAGE) COMMANDS: -

➤ INSERT

► **UPDATE**

➤ DELETE

**➤ INSERT**

- Insert command is Used to insert data into a table or create a new row in table:
- We have two methods to insert data in a table

1. By value method

## 2. By address method

**Syntax: (1<sup>st</sup> way-BY value method)**

### 1<sup>st</sup> Way

SQL>INSERT INTO TABLENAME VALUES (VALUE\_1,VALUE\_2,... VALUE\_N);

OR

### 2<sup>nd</sup> Way

```
SQL> INSERT INTO TABLENAME (COLUMN_1, COLUMN_2, COLUMN_3 .....COLUMN_N)
VALUES (VALUE 1, VALUE 2, VALUE 3 ... VALUE N);
```

**Example:**

### 1<sup>st</sup> Way

```
SQL> INSERT INTO CLIENT_MASTER VALUES ('C00001', 'RAHUL', 'MUMBAI', 400054,'
MAHARASHTRA');
```

OR

## 2<sup>nd</sup> Way

```
SQL>INSERT INTO CLIENT_MASTER (CLIENT_NO, NAME,CITY, PINCODE, STATE) VALUES ('C00001', 'RAHUL', 'MUMBAI', 400054, 'MAHARASHTRA');
```

[INSERT MORE VALUE]

```
SQL>INSERT INTO CLIENT_MASTER VALUES ('C00002', 'MANISH', 'BANGALORE',560001, 'KARNATAKA');
```

To insert a new record again you have to type entire insert command, if there are lot of records this will be difficult. This will be avoided by using address method.

### Show the Records of a Table:

#### Syntax:

```
SQL> SELECT * FROM CLIENT_MASTER;
```

CLIENT_NO	NAME	CITY	PINCODE	STATE
C00001	RAHUL	MUMBAI	400054	MAHARASHTRA
C00002	MANISH	BANGALORE	560001	KARNATAKA

#### Syntax: (2<sup>nd</sup> way-BY address method)

```
SQL>INSERT INTO TABLENAME VALUES (&COLUMN_1, &COLUMN_2,...  
& COLUMN_N);
```

OR

```
SQL> INSERT INTO TABLENAME(COLUMN_1, COLUMN_2, COLUMN_3 .....COLUMN_N)  
VALUES (&COLUMN_1, &COLUMN_2, &COLUMN_3 .... &COLUMN_N);
```

This will prompt you for the values but for every insert you have to use forward slash .

#### Example:

```
SQL> INSERT INTO CLIENT_MASTER VALUES ('&CLIENT_NO', '&NAME', '&CITY',  
&PINCODE, '&STATE');
```

Enter value for client\_no: C0001

Enter value for name: Rahul

Enter value for city: Mumbai

Enter value for pincode: 400054

Enter value for state: Maharashtra

SQL>

Enter value for client\_no: C0002

Enter value for name: Manish

Enter value for city: Bangalore

Enter value for pincode: 560001

Enter value for state: Karnataka

### Show the Records of a Table:

#### Syntax:

```
SQL> SELECT * FROM CLIENT_MASTER;
```

CLIENT_NO	NAME	CITY	PINCODE	STATE
C00001	RAHUL	MUMBAI	400054	MAHARASHTRA
C00002	MANISH	BANGALORE	560001	KARNATAKA

➤ **UPDATE**

- The UPDATE command is used to change or modify data of a table
- The Update command is used to update selected rows from table or all the rows from table

**Syntax:(update selected rows)**

SQL> UPDATE TABLENAME SET COLUMNNAME\_1 = EXPRESSION\_1, COLUMNNAME\_2 = EXPRESSION\_2 WHERE CONDITION;

**Example:**

SQL>UPDATE CLIENT\_MASTER SET CITY='BOMBAY' WHERE CLIENT\_NO='C00002';

**Show the Records of a Table:**

**Syntax:**

SQL> SELECT \* FROM CLIENT\_MASTER;

CLIENT_NO	NAME	CITY	PINCODE	STATE
C00001	RAHUL	MUMBAI	400054	MAHARASHTRA
C00002	MANISH	BOMBAY	560001	KARNATAKA

**Syntax:(update all rows)**

SQL>UPDATE TABLENAME SET COLUMNNAME\_1 = EXPRESSION\_1;

**Example:**

SQL>UPDATE CLIENT\_MASTER SET NAME = 'OHM';

**Show the Records of a Table:**

**Syntax:**

SQL> SELECT \* FROM CLIENT\_MASTER;

CLIENT_NO	NAME	CITY	PINCODE	STATE
C00001	OHM	MUMBAI	400054	MAHARASHTRA
C00002	OHM	BOMBAY	560001	KARNATAKA

➤ **DELETE**

- The DELETE command is used to delete data or rows from a table
- The delete command is used to delete selected rows from table or All the rows from table

**Syntax: (delete selected rows)**

SQL> DELETE FROM TABLENAME WHERE CONDITION;

**Example:**

SQL> DELETE FROM CLIENT\_MASTER WHERE CLIENT\_NO='C00001';

**Show the Records of a Table:**

**Syntax:**

SQL> SELECT \* FROM CLIENT\_MASTER;

CLIENT_NO	NAME	CITY	PINCODE	STATE
C00002	OHM	BOMBAY	560001	KARNATAKA

**Syntax: (delete all rows)**

SQL> DELETE FROM TABLE NAME;

**Example:**

SQL> DELETE FROM CLIENT\_MASTER;

**Show the Records of a Table:**

**Syntax:**

SQL> SELECT \* FROM CLIENT\_MASTER;

SQL>No Rows selected

❖ **SELECT**



- The SELECT command is used to retrieve selected rows from one or more tables.

Syntax: (retrieve all table data)

```
SQL>SELECT * FROM TABLE_NAME;
```

Example:

```
SQL> SELECT * FROM CLIENT_MASTER;
```

Syntax: (retrieve particular columns of a table)

```
SQL>SELECT COLUMNNAME_1, COLUMNNAME_2 FROM TABLE_NAME;
```

```
SQL> SELECT NAME, CITY FROM CLIENT_MASTER;
```

Syntax: (retrieve particular column of particular row of a table)

```
SQL> SELECT COLUMNNAME_N FROM TABLE_NAME WHERE CONDITION;
```

```
SQL>SELECT NAME FROM CLIENT_MASTER WHERE CITY='BOMBAY';
```

Syntax: (retrieve particular rows of a table )

```
SQL> SELECT * FROM TABLE_NAME WHERE CONDITION;
```

```
SQL>SELECT * FROM CLIENT_MASTER WHERE CITY='BOMBAY';
```

### ❖ **SQL Operators**

#### ➤ Arithmetic Operators

- Oracle allows arithmetic operators to use while viewing records from table or while performing data manipulation operations such as Insert, Update and Delete.

- These are:

+ Addition	* Multiplication
** Exponentiation	() Enclosed operation
- Subtraction	/ Division

Example:

```
SQL>Select product_no, description, sell_price * 0.05 from product_master;
```

#### ➤ **Logical Operators**

- The AND Operator
- The OR Operator
- The NOT Operator

#### ➤ **The AND Operator**

- The AND Operator allows creating an SQL statement based on two or more conditions being met.
- It can be used in any valid SQL statement such as select, insert, or delete.

Example:

```
SQL>SELECT PRODUCT_NO, DESCRIPTION, SELL_PRICE * 0.05 FROM PRODUCT_MASTER  
WHERE SELL_PRICE BETWEEN 500 AND 1100;
```

#### ➤ **The OR Operator**

- The OR condition allows creating an SQL statement where records are returned when any one of the conditions are met
- It can be used in any valid SQL statement such as select, insert, or delete.

Example:

```
SQL>SELECT CLIENT_NO, NAME, CITY, PINCODE FROM CLIENT_MASTER  
WHERE PINCODE=4000054 OR PINCODE=4000057;
```

#### ➤ **The NOT Operator**

- The Oracle engine will process all rows in a table and display only those Records that do not satisfy the condition specified

Example:

```
SQL>SELECT * FROM CLIENT_MASTER WHERE NOT (CITY='BOMBAY' OR CITY='DELHI');
```

➤ **Comparison Operators**

- Comparison operators are used in condition to compare one expression with other.
- The comparison operators are =, <, >, >=, <=, !=, between, like, is null and in operators

➤ **Between operator**

- Between operator is used to check between two values or specific range

Example:

```
SQL>SELECT * FROM SALESMAN_MASTER WHERE SALARY BETWEEN 5000 AND 8000;
```

The above select statement will display only those rows where salary of salesman is between 5000 and 8000.

➤ **In operator:**

- The IN operator can be used to select rows that match one of the values in a list.

```
SQL> SELECT * FROM CLIENT_MASTER WHERE CLIENT_NO IN(C00001, C00003);
```

The above query will retrieve only those rows where client\_no is either in C00001 or C00003

➤ **Like operator:**

- The Like operator is used to search character pattern.
- The like operator is used with special character % and \_(underscore)

```
SQL> SELECT * FROM CLIENT_MASTER WHERE CITY LIKE 'B% ';
```

The above select statement will display only those rows where city is start with 'B' followed by any number of any characters

- % sign is used to refer number of character (it similar to \* asterisk wildcard in DOS).
- While \_(underscore) is used to refer single character.

```
SQL> SELECT * FROM CLIENT_MASTER WHERE NAME LIKE '_AHUL';
```

**G. Resources/Equipment Required**

Computer or laptop with a database management system (DBMS) software installed, such as Oracle

**H. Safety and necessary Precautions followed**

Shutdown computer system properly once the Lab hours are finished

**I. Source code / Solutions to queries:**







### K. Practical related Quiz.

- Which of the following is true about inserting news rows to a table?
  - You must list values in the default order of the columns in the table.
  - You can also list the columns in the INSERT clause.
  - You can use the INSERT statement to add rows from one table to another.
  - All of the above.
- Which of the following code will successfully delete the table LOCATIONS from the database?
  - DROP TABLE locations;
  - DELETE TABLE locations;
  - TRUNCATE TABLE locations;
  - None of the above.
- Which operator is used to compare a value to a specified list of values?
  - ANY
  - BETWEEN
  - ALL
  - IN
- What operator tests column for absence of data
  - NOT Operator
  - Exists Operator
  - IS NULL Operator
  - None of the above
- In which of the following cases a DML statement is not executed?
  - When existing rows are modified.
  - When a table is deleted.
  - When some rows are deleted.
  - All of the above

### L. References / Suggestions

<https://www.geeksforgeeks.org/ddl-commands-syntax/>

<https://www.w3schools.com/sql/>

<https://www.tutorialspoint.com/sql/index.htm>

### M. Assessment-Rubrics

Excellent (21-25 Marks)	Good (15-20 Marks)	Satisfactory (7-14 Marks)	Fair (0-6 Marks)
<ul style="list-style-type: none"> <li>Student executes the queries with correct output.</li> <li>Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>Student executes the queries having correct output with external guidance.</li> <li>Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>Student executes the queries with incorrect output.</li> <li>Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>Student is not able to execute the queries.</li> <li>Student does not submit the lab report in specified time limit.</li> </ul>

**Sign with Date**

**Date:** .....

### **Practical No.3:**

Perform SQL queries for TCL (Transaction Control Language) and DCL (Data Control Language) Commands.

Note: You can perform TCL and DCL commands tables created in Practical 1 or tables in HR Schema tables.

**1. Perform TCL Commands:**

- a. Update the salary of the employee with the employee\_id 100 to 5000 and permanently save record.
- b. Delete all employees from the employees table whose job\_id is 'SA\_REP'. View updated data and then bring back the old data using TCL command.

**2. Perform DCL Commands:**

- a. Create a user account with your enrolment number.
- b. Grant select, update of all HR tables to your new user account as in 2(a).
- c. Copy all HR tables to your user account using create table as select query.
- d. Update salary of HR user's employee table employee John to 5000 while staying login in your user account created in 2(a).
- e. Revoke the update privilege on all HR tables given in 2(b).
- f. Try to change salary of HR user's employee table employee John to 6000 while staying login in your user account created in 2(a) and write the output you get.

**A. Objective:**

- TCL commands are used to manage transactions in SQL databases, to commit a transaction and permanently save the changes made during the transaction and to roll back a transaction and undo any changes made during the transaction
- DCL commands are used to manage user permissions and access to the database, to grant or revoke privileges to users or roles for specific database objects, to create or drop user accounts and manage user authentication

**B. Expected Program Outcomes (POs): PO1, PO5, PO7**

**C. Expected Skills to be developed based on competency:**

Compilation, debugging, executing SQL queries

**D. Expected Course Outcomes(Cos)**

Perform queries on datasets using SQL\*Plus

**E. Practical Outcome(Pro)**

Write data definition queries, compilation, debugging, executing using SQL

**F. Expected Affective Domain Outcome (ADos)**

- Follow safety practices.
- Practice good housekeeping.
- Demonstrate working as a leader/a team member.
- Maintain tools and equipment
- Follow ethical practices.

**G. Prerequisite Theory:**

**TCL Commands in SQL- Transaction Control Language Examples:**

Transaction Control Language can be defined as the portion of a database language used for maintaining consistency of the database and managing transactions in database. A set of SQL statements that are co-related logically and executed on the data stored in the table is known as

transaction. In this tutorial, you will learn different TCL Commands in SQL with examples and difference between them.

## 1. Commit Command

## 2. Rollback Command

## 3. Savepoint Command

**Commit Commands:**

The main use of Commit command is to make the changes done by the transaction permanent in the database.

**Syntax:** COMMIT;

**Example:**

```
UPDATE STUDENT SET STUDENT_NAME = 'Maria' WHERE STUDENT_NAME = 'Meena';
COMMIT;
```

By using the above set of instructions, you can update the wrong student name by the correct one and save it permanently in the database. The update transaction gets completed when commit is used. If commit is not used, then there will be lock on 'Meena' record till the rollback or commit is issued.

Now have a look at the below table where 'Meena' is updated and there is a lock on her record. The updated value is permanently saved in the database after the use of commit and lock is released.

Student ID	Student Name
78	Jane
79	Meena

After Update :

Student ID	Student Name
78	Jane
79	Maria

After Commit :

Student ID	Student Name
78	Jane
79	Maria

**Rollback:**

Using this command, the database can be restored to the last committed state. Additionally, it is also used with savepoint command for jumping to a savepoint in a transaction.

**Syntax:**

ROLLBACK;

**OR**

ROLLBACK TO SAVEPOINT\_NAME;

**Example:**

```
UPDATE STUDENT SET STUDENT_NAME = 'Manish' WHERE STUDENT_NAME = 'Meena';
```

**ROLLBACK;**

This command is used when the user realizes that he/she has updated the wrong information and wants to undo this update. The users can issue ROLLBACK command and then undo the update. Have a look at the below tables to know better about the implementation of this command.

Student ID	Student Name
78	Jane



79	Meena
----	-------

After Update :

Student ID	Student Name
78	Jane
79	Manish

After Rollback:

Student ID	Student Name
78	Jane
79	Meena

### Savepoint:

The main use of the Savepoint command is to save a transaction temporarily. This way users can rollback to the point whenever it is needed.

**Syntax:** *SAVEPOINT SAVEPOINT\_NAME;*

**Example:**

SAVEPOINT SAVEPOINT\_NAME;

Following is the table of a school class

ID	Name
98	Anita
99	Maria
100	Katilyn

**Use some SQL queries on the above table and then watch the results**

INSERT into CLASS VALUES (101, 'Rahul');

Commit;

UPDATE CLASS SET NAME= 'Tyler' where id= 101;

SAVEPOINT A;

INSERT INTO CLASS VALUES (102, 'Zack');

Savepoint B;

INSERT INTO CLASS VALUES (103, 'Bruno')

Savepoint C;

SELECT \* FROM CLASS;

The result will look like

ID	Name
98	Anita
99	Maria
100	Katilyn
101	Tyler
102	Zack
103	Bruno

**Now rollback to savepoint B**

ROLLBACK TO B;

SELECT \* FROM CLASS;

ID	Name
98	Anita

99	Maria
100	Katilyn
101	Tyler
102	Zack

**Now rollback to savepoint A**

ROLLBACK TO A;

SELECT \* FROM CLASS;

ID	Name
98	Anita
99	Maria
100	Katilyn
101	Tyler

### **DCL Commands**

#### **Create user –**

If a login ID does not exist, it has to be created manually. The following steps indicate how a DBA login ID is created and appropriate privileges given to the DBA login ID with a special focus on creating table space.

#### **Creating a User:-**

**SYNTAX:** - CREATE USER USERNAME IDENTIFIED BY PASSWORD;

#### **Example:-**

```
Run SQL Command Line
SQL>create user abc identified by abc123;

User Created.
```

When we create a user in SQL, it is not even allowed to login and create a session until and unless proper permissions/privileges are granted to the user.

Following command can be used to grant the session creating privileges.

#### **SYNTAX:** -

GRANT CREATE SESSION, connect, resource TO username;

#### **Unlocking a user account: -**

**SYNTAX:** - ALTER USER account IDENTIFIED BY password ACCOUNT UNLOCK;

### **DCL Commands**

Oracle provides two commands - GRANT and REVOKE - to control the access of various database objects.

#### **GRANT COMMAND**

Grants a privilege to a user. It means that giving authority to other user by administrator. If you are an administrator, then only you have the authority for granting the privilege to other users. User can grant privilege only if user has been granted that privilege

**Syntax:** GRANT < OBJECT PRIVILEGES > ON <OBJECTNAME> TO <USERNAME> [WITH GRANT OPTION];

#### **OBJECT PRIVILEGES**

Each object privilege that is granted authorizes the grantee to perform some operation on the object. A user can grant all the privileges or grant only specific object privileges.

The list of object privileges is as follows:

ALTER: Allows the grantee to change the table definition with the ALTER TABLE command

DELETE: Allows the grantee to remove the records from the table with the DELETE command

INDEX: Allows the grantee to create an index on the table with the CREATE INDEX command

INSERT: Allows the grantee to add records to the table with the INSERT command

SELECT: Allows the grantee to query the table with the SELECT command

UPDATE: Allows the grantee to modify the records in the tables with the UPDATE command

**Example:**

Give the user rahul permission to only view and modify records in the table client\_master.

Run SQL Command Line

```
SQL>GRANT SELECT, UPDATE ON client_master TO rahul;
```

Grant succeeded.

**REVOKE COMMAND**

The REVOKE statement is used to deny the grant given on an object. Revokes a privilege from a user. It is used in getting back authority from user.

**Syntax:** REVOKE < OBJECT PRIVILEGES > ON <OBJECT NAME> FROM <USERNAME>;

**Example:**

All privileges on the table salesman\_master have been granted to rahul. Take back the Delete privilege on the table.

Run SQL Command Line

```
SQL>REVOKE DELETE ON salesman_master FROM Rahul;
```

Revoke succeeded.

**H. Resources/Equipment Required**

Computer or laptop with a database management system (DBMS) software installed, such as Oracle

**I. Safety and necessary Precautions followed**

Shutdown computer system properly once the Lab hours are finished

**J. Source code / Solutions to queries:**





### K. Practical related Quiz.

1. Command that comes under DCL is/are -  
 A. GRANT      B. REVOKE      C. Both A. and B.      D. None of the above
2. Following the completion of a transaction, it must be executed to save all the operations performed in the transaction. Here we are talking about which command?  
 A. REVOKE      B. COMMIT      C. ROLLBACK      D. SAVE
3. Difference between GRANT & REVOKE command is/are?  
 A. The GRANT command can be used to grant a user access to databases and tables whereas The REVOKE command can be used to revoke all access privileges already assigned to the user.  
 B. The REVOKE command can be used to grant a user access to databases and tables whereas The GRANT command can be used to revoke all access privileges already assigned to the user.  
 C. A transaction can be rolled back to its last saved state.  
 D. None of the above
4. Which of the following is TRUE about TCL?  
 A. Transactions can be saved to the database and rolled back with the help of TCL commands in SQL.  
 B. There will be certain privileges that each user has; consequently, the data can be accessed by them using TCL.  
 C. Our data is stored in a table that is described by the schema, thus TCL commands deal with the schema.  
 D. SQL TCL commands can be used to perform any kind of retrieval or manipulation of the data present in SQL tables.
5. \_\_\_\_\_ commands in SQL allow controlling access to data within database.  
 A. Database      B. Data      C. Data control      D. All of the Mentioned

### L. References / Suggestions

<https://www.geeksforgeeks.org/ddl-commands-syntax/>

<https://www.w3schools.com/sql/>

<https://www.tutorialspoint.com/sql/index.htm>

### M. Assessment-Rubrics

Excellent (21-25 Marks)	Good (15-20 Marks)	Satisfactory (7-14 Marks)	Fair (0-6 Marks)
<ul style="list-style-type: none"> <li>Student executes the queries with correct output.</li> <li>Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>Student executes the queries having correct output with external guidance.</li> <li>Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>Student executes the queries with incorrect output.</li> <li>Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>Student is not able to execute the queries.</li> <li>Student does not submit the lab report in specified time limit.</li> </ul>

**Sign with Date**

Date: .....

**Practical No. 4:**

Implement SQL queries using Date functions like add-months, months-between, round, nextday, truncate etc.

Write down the queries using date function: -

1. Find a date after adding 4 months to '01-06-19'.
2. Find total months between '01-01-19' to '01-05-19'.
3. Find a last date of '01-02-19'.
4. Find out date of next Tuesday after '18-06-19'.
5. Find the nearest date for '17-06-19 12:35pm', '17-06-19 12:35am' and '17-06-19 01:30pm'.
6. Find the lower nearest date for '17-06-19 12:35pm', '17-06-19 12:35am' and '17-06-19 01:30pm'.
7. Find out the date of next Sunday after today.
8. An employee retires on the last day of the month after completing 30 years of service. Using HR employee table, list employee name and his/her retirement date.

**A. Objective:**

SQL provides various date functions to perform common operations on date and time values, such as extracting date and time components, calculating differences between dates, and formatting dates in different ways

**B. Expected Program Outcomes (POs): PO1,PO2,PO3,PO4,PO5,PO7****C. Expected Skills to be developed based on competency:**

Compilation, debugging, executing SQL queries

**D. Expected Course Outcomes(Cos)**

Perform joins, sub-queries and nested queries on multiple tables using SQL\*plus

**E. Practical Outcome(Pro)**

Write data definition queries , compilation, debugging, executing using SQL

**F. Expected Affective domain Outcome (ADos)**

- Follow safety practices.
- Practice good housekeeping.
- Demonstrate working as a leader/a team member.
- Maintain tools and equipment
- Follow ethical practices.

**G. Prerequisite Theory:****❖ SQL DATE FUNCTIONS:-**

- |                  |            |         |
|------------------|------------|---------|
| ➤ ADD_MONTHS     | ➤ LAST_DAY | ➤ ROUND |
| ➤ MONTHS_BETWEEN | ➤ NEXT_DAY | ➤ TRUNC |

NOTE:- CONSIDER '31-DEC-2012' AS A SYSDATE.

➤ ADD\_MONTHS

- ADD\_MONTHS function returns new date after adding the number of months specified in the function.

**Syntax:**

```
SQL> ADD_MONTHS (D, N);
```

**Example:**

```
SQL> SELECT ADD_MONTHS (SYSDATE,3) "ADD MONTHS" FROM DUAL;
```

```
SQL> ADD MONTHS
```

```
31-MAR-13
```

- IF n IS NEGATIVE, Then n MONTH will be subtracted.

```
SQL> SELECT ADD_MONTHS (SYSDATE,-3) "ADD MONTHS" FROM DUAL;
```

```
SQL> ADD MONTHS
```

```
30-SEP-12
```

➤ **MONTHS\_BETWEEN**

- MONTHS\_BETWEEN function returns number of months between date1 and date2.

**Syntax:**

```
SQL> MONTHS_BETWEEN (DATE1, DATE2);
```

**Example:**

```
SQL> SELECT MONTHS_BETWEEN ('31-MAR-13','31-DEC-12') "MONTHS BETWEEN"  
FROM DUAL;
```

```
SQL> MONTHS BETWEEN
```

```
3
```

➤ **LAST\_DAY**

- LAST\_DAY function returns the last date of the month specified with the function.

**Syntax:**

```
SQL> LAST_DAY (DATE);
```

**Example:**

```
SQL> SELECT LAST_DAY ('1-MAR-13') "LASTDAY" FROM DUAL;
```

```
SQL> LASTDAY
```

```
31-MAR-13
```

➤ **NEXT\_DAY**

- NEXT\_DAY function returns the date of the next named weekday specified by day relative to date.

**Syntax:**

```
SQL> NEXT_DAY (DATE, DAY);
```

**Example:**

```
SQL> SELECT NEXT_DAY ('3-JUNE-19','SUNDAY') "NEXTDAY" FROM DUAL;
```

```
SQL> NEXTDAY
```

```
09-JUNE-19
```

➤ **ROUND**



- ROUND function returns the rounded date according to format.
- A format can be any valid format.
- If format is omitted, date is rounded to the next day if time is 12.00 pm or later otherwise return today's date.

**Syntax:**

```
SQL>ROUND (DATE, FORMAT);
```

**Example:**

```
SQL>SELECT ROUND(TO_DATE('31-DEC-12 03:30:45 PM','DD-MM-YY HH:MI:SS PM'))  
FROM DUAL;
```

```
SQL>ROUND(TO_
```

```
-----
```

```
01-JAN-13
```

```
SQL> SELECT ROUND(TO_DATE('31-DEC-12 12:00:00 PM','DD-MM-YY HH:MI:SS PM'))  
FROM DUAL;
```

```
SQL>ROUND(TO_
```

```
-----
```

```
01-JAN-13
```

```
SQL> SELECT ROUND(TO_DATE('31-DEC-12 03:30:45 AM','DD-MM-YY HH:MI:SS PM'))  
FROM DUAL;
```

```
SQL>ROUND(TO_
```

```
-----
```

```
31-DEC-12
```

➤ **TRUNC**

- TRUNC function returns the truncated date according to format.
- A format can be any valid format.
- If format is omitted, date is truncated to 12.00 am.

**Syntax:**

```
SQL>TRUNC (DATE, FORMAT);
```

**Example:**

```
SQL>SELECT TRUNC(TO_DATE('31-DEC-12 03:30:45 PM','DD-MM-YY HH:MI:SS PM'))  
FROM DUAL;
```

```
SQL>TRUNC(TO_
```

```
-----
```

```
31-DEC-12
```

```
SQL> SELECT TRUNC(TO_DATE('31-DEC-12 12:00:00 PM','DD-MM-YY HH:MI:SS PM'))  
FROM DUAL;
```

```
SQL>TRUNC(TO_
```

```
-----
```

```
31-DEC-12
```

```
SQL> SELECT TRUNC(TO_DATE('31-DEC-12 03:30:45 AM','DD-MM-YY HH:MI:SS PM'))  
FROM DUAL;
```

```
SQL>TRUNC(TO_
```

-----

31-DEC-12

**H. Resources/Equipment Required**

Computer or laptop with a database management system (DBMS) software installed, such as Oracle

**I. Safety and necessary Precautions followed**

Shutdown computer system properly once the Lab hours are finished

**J. Source code / Solutions to queries:**



### K. Practical related Quiz

- Result of the below two queries will be:  
`SELECT DATEDIFF(week,'2005-12-31 23:59:59.9999999','2006-01-01 00:00:00.0000000') from dual;`  
`SELECT DATEDIFF(hour,'2005-12-31 23:59:59.9999999','2006-01-01 00:00:00.0000000') from dual;`  
 A 24, 1 B 1, 24 C 1, 1D None of the above
- Result of the below query will be:  
`SET DATEFORMAT mdy;`  
`SELECT ISDATE('12/31/208');`  
 A 0 B 1 C 2 D 3.
- When you convert to date and time data types, SQL rejects all values it cannot recognize as dates or times.  
 A True B False
- If result of below query :  
`select CONVERT(VARCHAR(19),GETDATE())`  
 is: Jul 29 2014 9:46AM  
 what will be the result of below query:  
`select CONVERT(VARCHAR(10),GETDATE(),10)`  
 A Jul 29 2014 B 07-29-14 9:46AM  
 C 07-29-14 D Jul 29 201
- Which of the following function checks whether the expression is a valid date or not?  
 A ISDATE B ISDAY C ISVALID D ISYEAR

### L. References / Suggestions

<https://www.geeksforgeeks.org/ddl-commands-syntax/>  
<https://www.w3schools.com/sql/>  
<https://www.tutorialspoint.com/sql/index.htm>

### M. Assessment-Rubrics

Excellent (21-25 Marks)	Good (15-20 Marks)	Satisfactory (7-14 Marks)	Fair (0-6 Marks)
<ul style="list-style-type: none"> <li>Student executes the queries with correct output.</li> <li>Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>Student executes the queries having correct output with external guidance.</li> <li>Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>Student executes the queries with incorrect output.</li> <li>Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>Student is not able to execute the queries.</li> <li>Student does not submit the lab report in specified time limit.</li> </ul>

Sign with Date

**Date:** .....

### **Practical No .5:**

Implement SQL queries using Numeric functions like abs, ceil, power, mod, round, trunc, sqrt etc. and Character Functions like initcap, lower, upper, ltrim, rtrim, replace, substring, instr etc.

#### **Write down the queries using numeric function:-**

1. Find the absolute value of -15.
2. Find the square root of 81.
3. Find the value for 3 raised to power 4.
4. Find the remainder for 16 divided by 5.
5. Find the largest integer value which is greater than or equal to -27.2.
6. Find the smallest integer value which is less than or equal to -24.2.
7. Find the value for 182.284 which is rounded to -2.
8. Find the value for 182.284 which is rounded to 1.
9. Find the value for 182.284 which is truncated to -2.
10. Find the value for 182.284 which is truncated to 1.
11. Find the value for e which is raised to power 4.

#### **Write down the queries using character function:-**

Note : Consider Employees table from HR schema to solve queries.

1. Count number of characters in a string "Computer Engineering".
2. Count number of characters in each employee's name.
3. Convert string "COMPUTER" in lowercase.
4. Display first name of all employees in lowercase.
5. Display last name of all employees in uppercase.
6. Convert string "oracle10g" in uppercase.
7. Display first letter of each word in a string "character function" in uppercase.
8. Extract 11 characters from the string "Computer Engineering", starting from 12<sup>th</sup> character.
9. Extract first 3 characters from employee's first name.
10. Display last name of all employees' right justified with total length of 20 characters, fill up the blank characters with "#".
11. Display last name of all employees left justified with total length 15 characters, fill up the blank characters with "\*".
12. Remove characters "gbrsea" from string "greatest" from left side.
13. Change character "ornt" by "xynt" from the string "government".
14. Change the word "govern" by "suppli" in a string "government".
15. Display ascii values of 's', 'A' and 'a'.
16. Find the first occurrence of "base" in string 'Database'.

**A. Objective:**

Numeric functions are used to perform mathematical operations, while character functions are used to manipulate and perform operations on character data

**B. Expected Program Outcomes (POs): PO1,PO2,PO3,PO4,PO5,PO7**

**C. Expected Skills to be developed based on competency:**

Compilation, debugging, executing SQL queries

**D. Expected Course Outcomes(Cos)**

Perform joins, sub-queries and nested queries on multiple tables using SQL\*plus.

**E. Practical Outcome(Pro)**

Write data definition queries , compilation, debugging, executing using SQL

**F. Expected Affective domain Outcome (ADos)**

- Follow safety practices.
- Practice good housekeeping.
- Demonstrate working as a leader/a team member.
- Maintain tools and equipment
- Follow ethical practices.

**G. Prerequisite Theory:**

❖ **SQL NUMERIC FUNCTIONS:-**

- 1) ABS    2) CEIL        3) EXP        4) FLOOR    5) POWER    6) MOD  
7) ROUND        8) TRUNC    9) SQRT

➤ **ABS()**

- **ABS()** function returns absolute value.

**Syntax:** ABS (N);

**Example:**

```
SQL> SELECT ABS (3), ABS(-3) "ABSOLUTEVALUE" FROM DUAL;
          3          3
ABSOLUTEVALUE ABSOLUTEVALUE
-----
3              3
```

➤ **CEIL()**

- **CEIL()** function returns the smallest integer that is greater than or equal to a specific value.

**Syntax:** CEIL (N);

**Example:**

```
SQL> SELECT CEIL(-25.4)"CEIL1",CEIL(25.4)"CEIL2" FROM DUAL;
          CEIL1    CEIL2
          -----
          -25      26
```

➤ **EXP()**

- **EXP()** function returns **e** raised to the **nth** power, where e=2.71828183.

**Syntax:** EXP (N);

**Example:**

```
SQL> SELECT EXP ('1') "EXP" FROM DUAL;
SQL> EXP
```

2.71828183

➤ **FLOOR()**

- **FLOOR()** function returns the greatest integer that is less than or equal to a specific value.

**Syntax:** FLOOR (N);

**Example:**

SQL>SELECT FLOOR ('25.4') "FLOOR" ,FLOOR ('25.5') "FLOOR",FLOOR ('25.5') "FLOOR"  
FROM DUAL;

SQL>FLOOR   FLOOR   FLOOR  
          25       25       25

SQL> SELECT FLOOR ('-25.4') "FLOOR" ,FLOOR ('-25.5') "FLOOR",FLOOR ('-25.5')  
"FLOOR" FROM DUAL;

SQL>FLOOR   FLOOR   FLOOR  
          -26       -26       -26

➤ **POWER()**

- **POWER()** function returns the value raised to a given positive exponent.

**Syntax:** POWER(VALUE,EXPONENT);

**Example:**

SQL> SELECT POWER (3,2) "POWER" FROM DUAL;

SQL>POWER  
          9

➤ **MOD()**

- **MOD()** function divides a value by a divisor and returns the remainder.

**Syntax:** MOD(VALUE,DIVISER);

**Example:**

SQL> SELECT MOD (9,2) "REMAINDER" FROM DUAL;

SQL>REMAINDER  
          1

➤ **ROUND()**

- **ROUND()** function rounds a number to given number of digits of precision.

**Syntax:** ROUND(VALUE,PRECISION);

**Example:**

SQL>SELECT ROUND (255.555,1) "ROUND",ROUND (255.555,2) "ROUND",ROUND  
(255.555,3) "ROUND" FROM DUAL;

SQL>ROUND ROUND ROUND  
          255.6   255.56   255.555

➤ **TRUNC()**

- **TRUNC()** function truncates digits of precision from a number.

**Syntax:** TRUNC(VALUE,PRECISION);

**Example:**

SQL>SELECT TRUNC (255.555,1) "TRUNC",TRUNC (255.555,2) "TRUNC",TRUNC  
(255.555,3) "TRUNC" FROM DUAL;

```
SQL>TRUNC TRUNC TRUNC  
      255.5   255.55  255.555
```

➤ **SQRT()**

- **SQRT()** function returns the square root of a given number.

**Syntax:** SQRT(N);

**Example:**

```
SQL> SELECT SQRT (9) "SQRT" FROM DUAL;
```

```
SQL> SQRT  
      3
```

❖ **SQL CHARACTER FUNCTIONS:-**

- |            |              |            |              |
|------------|--------------|------------|--------------|
| 1) INITCAP | 2) LOWER     | 3) UPPER   | 4) LTRIM     |
| 5) RTRIM   | 6) TRANSLATE | 7) REPLACE | 8) SUBSTRING |
| 9) LENGTH  | 10) LPAD     | 12) RPAD   | 13) ASCII    |

➤ **INITCAP()**

- **INITCAP()** function converts any string or (column of table) with the first character of each word in UPPER CASE.

**Syntax:** INITCAP(STRING);

**Example:**

```
SQL> select initcap('database management system') "1stcharacterinuppercase" from  
dual;
```

```
SQL> 1stcharacterinuppercase  
      Database Management System
```

➤ **LOWER()**

- **LOWER()** function converts any string or (column of table) into lowercase.

**Syntax:** lower(string);

**Example:**

```
SQL> select lower ('DATABASE') "lower" from dual;
```

```
SQL> lower  
      database
```

➤ **UPPER()**

- **UPPER()** function converts any string or (column of table) into uppercase.  
converts any string or (column of table) into uppercase.

**Syntax:** upper(string);

**Example:**

```
SQL> select upper ('database') "upper" from dual;
```

```
SQL> upper  
      DATABASE
```

➤ **LTRIM()**

- **LTRIM()** function returns a string with all characters contained in the set are removed from the left end of the string.

**Syntax:**

```
SQL> ltrim(mainstring, set);
```



**Example:**

```
SQL> select ltrim('000123', '0') "ltrim1", ltrim('123123Tech', '123') "ltrim2" from dual;
```

```
SQL> ltrim1 ltrim2
      123    Tech
```

➤ **RTRIM()**

- RTRIM() function returns a string with all characters contained in the set are removed from the right end of the string.

**Syntax:**

```
SQL> rtrim(mainstring, set);
```

**Example:**

```
SQL> select rtrim('123000', '0') "rtrim1", rtrim('Tech123123', '123') "rtrim2" from dual;
```

```
SQL> rtrim1 rtrim2
      123    Tech
```

➤ **TRANSLATE()**

- **TRANSLATE ( )** function Replace a sequence of character in a string with another set of characters.

**Syntax:** TRANSLATE(string1, string to replace, replacement string);

**Example:**

```
SQL> SELECT TRANSLATE ('COMPUTER', 'COMPU', 'HEA') "TRANSLATESTSTRING" FROM DUAL;
```

```
SQL> TRANSLATESTSTRING
      HEATER
```

➤ **REPLACE()**

- REPLACE ( ) function Replace a sequence of characters in a string with another set of characters.

**Syntax:** REPLACE(string1, character to be replaced, characters);

**Example:**

```
SQL> SELECT REPLACE ('COMPUTER', 'UE', 'IL') "REPLACE" , REPLACE ('COMPUTER', 'UT', 'IL') "REPLACE" FROM DUAL;
```

```
SQL> REPLACE REPLACE
```

```
-----
```

```
COMPUTER COMPILER
```

➤ **SUBSTRING()**

- SUBSTRING( ) functions allows you to extract a substring from a string.

**Syntax:** SUBSTR (string, start position, length);

**Example:**

```
SQL> SELECT SUBSTR ('COMPUTER', 1, 4) "SUBSTRING" , SUBSTR('COMPUTER', 4, 5) "SUBSTRING" FROM DUAL;
```

```
SQL> SUBSTRING SUBSTRING
```

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

COMP PUTER

➤ **Instr()**

- SQL INSTR() function detects the first occurrence of a string or a character in the other string. Thus, the INSTR() function witnesses the position of the initial/first occurrence of any string/sub-string in another string data value.

**Syntax:** INSTR(string1, string2);

**Example:**

SQL>SELECT INSTR('JYPython', 'P');

SQL> instr

-----

3

➤ **LENGTH()**

- **LENGTH()** function returns the length of string(count number of characters).

**Syntax:**

SQL>length(string);

**Example:**

SQL> select length ('DATABASE') "length" from dual;

SQL>length

8

➤ **LPAD()**

- **LPAD()** function returns string which is left padded with string2 up to length n.

**Syntax:** lpad(string , n ,string2);

**Example:**

SQL>select lpad('database',15,'3330703') "lpad" from dual;

SQL> lpad

-----

3330703database

➤ **RPAD()**

- **RPAD()** function returns string which is right padded with string2 up to length n.

**Syntax:** rpad (string , n ,string2);

**Example:**

SQL>select rpad('database',15,'3330703') "rpad" from dual;

SQL>rpad

-----

database3330703

➤ **ASCII()**

- **ASCII()** function returns the ascii code of a charcter.

**Syntax:** ascii (charcter);

**Example:**

SQL>select ascii('d')"ascii",ascii('D')"ascii" from dual;

SQL>ascii   ascii

-----

100 68

**H. Resources/Equipment Required**

Computer or laptop with a database management system (DBMS) software installed, such as Oracle

**I. Safety and necessary Precautions followed**

Shutdown computer system properly once the Lab hours are finished

**J. Source code/ Solutions to queries:**







### K. Practical related Quiz.

- What is true about Character functions?
  - They return only character values
  - They accept NUMBER values
  - They accept character arguments and can return both character and number values
  - They accept values of all data type
- What will be the outcome of the following query?  
 SELECT lower(upper(initcap('Hello World')) FROM dual;
  - Hello World
  - HELLO world
  - hello World
  - hello world
- In what scenario would you want to use the UPPER function in SQL?
  - All text is numeric
  - The text values might be mixed-case
  - You need to convert the data to date format
  - All text is unreadable or mis-aligned
- Which of the following can be used to measure the length of an input value?
  - QUOTENAME
  - LENGTH
  - DATALLENGTH
  - LEN
- Which of the following is used to replace a string with a different one, at a specified position?
  - FORMAT
  - STUFF
  - REPLICATED
  - REPLACE

### L. References / Suggestions

<https://www.geeksforgeeks.org/ddl-commands-syntax/>  
<https://www.w3schools.com/sql/>  
<https://www.tutorialspoint.com/sql/index.htm>

### M. Assessment-Rubrics

Excellent (21-25 Marks)	Good (15-20 Marks)	Satisfactory (7-14 Marks)	Fair (0-6 Marks)
<ul style="list-style-type: none"> <li>Student executes the queries with correct output.</li> <li>Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>Student executes the queries having correct output with external guidance.</li> <li>Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>Student executes the queries with incorrect output.</li> <li>Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>Student is not able to execute the queries.</li> <li>Student does not submit the lab report in specified time limit.</li> </ul>

Sign with Date

**Date:** .....

### **Practical No.6:**

Implement SQL queries using Conversion Functions like to- char, to-date, to-number and Group functions like Avg, Min, Max, Sum, Count, Decode etc.

**Note: Consider HR.EMPLOYEES table to solve all queries.**

**Write down the queries using conversion and miscellaneous function: -**

1. Convert '+01234.78' to number.
2. Convert a value 123789 to character using '9,99,999' format.
3. List out hire date in form of 'Saturday,14<sup>th</sup> feb,2009' for employee. (consider HR.EMPLOYEES table)
4. List out hire date in form of '23-march-1985' for all employee. (consider employee table)
5. If input value is 'MAX' then display message 'this is maximum', if input value is 'MIN' then display message 'this is minimum' otherwise display message 'this is equal'.

**Write down the queries using GROUP function:-**

**Consider HR.EMPLOYEES table to solve all queries.**

1. Find total employees, maximum, minimum and average salary.
2. Find department number, total employees in that department and total salary to be paid to department number 20.
3. Find minimum, maximum and average salary for job type 'SALESMAN'.

#### **A. Objective:**

Implementing SQL conversion functions in SQL queries is to convert data from one data type to another data type and manipulate data that is stored in different formats and group functions is to perform calculations on a set of values that are grouped together based on a specific column or columns. These group functions are also known as aggregate functions because they aggregate multiple rows into a single result.

#### **B. Expected Program Outcomes (POs): PO1,PO2,PO3,PO4,PO5,PO7**

#### **C. Expected Skills to be developed based on competency:**

Compilation, debugging, executing SQL queries

#### **D. Expected Course Outcomes(Cos)**

Perform joins, sub-queries and nested queries on multiple tables using SQL\*plus.

#### **E. Practical Outcome(PRo)**

Write data definition queries , compilation, debugging, executing using SQL

#### **F. Expected Affective Domain Outcome (ADos)**

- Follow safety practices.
- Practice good housekeeping.
- Demonstrate working as a leader/a team member.
- Maintain tools and equipment
- Follow ethical practices.

#### **G. Prerequisite Theory:**



❖ **SQL CONVERSION FUNCTIONS: -**

- TO\_NUMBER
- TO\_CHAR
- TO\_DATE

➤ **TO\_NUMBER (CONVERTING CHARACTER TO NUMBER)**

- **TO\_NUMBER ( )** function converts a value of a character data type to number data type.
- Returns equivalent numeric value to string.
- String must consist of 0-9, decimal point and '+' or '-' sign.

**Syntax:**

SQL>TO\_NUMBER(STRING);

**Example:**

```
SQL>SELECT    TO_NUMBER('-123.3')    "TO_NUMBER",    TO_NUMBER('+123.3')
"TO_NUMBER", TO_NUMBER('123') "TO_NUMBER" FROM DUAL;
SQL> TO_NUMBER TO_NUMBER TO_NUMBER
```

```
-----
-123.3    123.3    123
```

➤ **TO\_CHAR (CONVERTING DATE TO CHARACTER )**

- **TO\_CHAR ( )** function converts a value of DATE data type to CHAR value.
- It can also extract a part of the date, month or the year from the date value and use it for sorting or grouping of data according to the date, month or year.

**Syntax:**

SQL>TO\_CHAR(DATE,FORMAT);

**Example:**

```
SQL>SELECT    TO_CHAR    (SYSDATE,'YYYY')    "EXTRACTYEAR",    TO_CHAR
(SYSDATE,'MONTH') "EXTRACTMONTH" FROM DUAL;
SQL>EXTRACTYEAR EXTRACTMONTH
```

```
2019        JUNE
```

➤ **TO\_CHAR (CONVERTING NUMBER TO CHARACTER )**

- **TO\_CHAR ( )** function converts a value of NUMBER data type to CHAR value using optional format.
- A format consist '0','9' and ','.

**Syntax:**

SQL>TO\_CHAR (N, FORMAT);

**Example:**

```
SQL>SELECT TO_CHAR(123456,'09,0999'),TO_CHAR(123456,'99,09999') FROM DUAL;
SQL> TO_CHAR(    TO_CHAR(1
```

```
-----
12,3456    1,23456
```

➤ **TO\_DATE (CONVERTING CHARACTER TO DATE )**

- **TO\_CHAR ( )** function converts avalue of CHAR datatype to DATE data type value.

**Syntax:**

```
SQL>TO_DATE(CHAR, FORMAT);
```

**Example:**

```
SQL>SELECT TO_DATE ('06/06/19','DD/MM/YYYY') FROM DUAL;
```

```
SQL> TO_DATE('
```

```
-----
```

```
06-JUN-19
```

➤ **USEFUL DATE FORMATS FOR TO\_CHAR AND TO\_DATE FUNCTIONS:**

FORMAT	SPECIFIES	FORMAT	SPECIFIES
<b>MM</b>	Number of months (1-12)	<b>YYY</b>	Last three digit of year: 986
<b>MON</b>	3-Letter month: JAN	<b>YY</b>	Last two digits of year: 86
<b>MONTH</b>	Full month: JUNE	<b>Y</b>	Last one digit of year: 6
<b>DDD</b>	No. of day of year	<b>YEAR</b>	Year spelled out
<b>DD</b>	No. of day of month	<b>WW</b>	No. of week in year
<b>D</b>	No. of day of week	<b>W</b>	No. of week in month
<b>DY</b>	3-letter day: MON	<b>HH</b>	Hour
<b>DAY</b>	Full day: SUNDAY	<b>MI</b>	Minute
<b>YYYY</b>	Four digit year: 1986	<b>SS</b>	Second

➤ **USEFUL FORMATS THAT CAN BE USED ONLY WITH TO\_CHAR:**

FORMAT	SPECIFIES
<b>TH</b>	Produces 6 <sup>TH</sup> , 6 <sup>Th</sup> or 6 <sup>th</sup> based on DDTH, DdTH, ddTH
<b>SP</b>	Forces number to spell out, like -five
<b>SPTH/THSP</b>	Forces number to spell out, like -fifth

❖ **SQL MISCELLANEOUS FUNCTIONS:-**➤ **DECODE**

- **DECODE** function has the functionality of an IF-THEN-ELSE statement.

**Syntax:**

```
SQL>DECODE(EXP, EXP1 THEN RESULT1,EXP2 THEN RESULT2,DEFAULT);
```

**Example:**

```
SQL>SELECT      DECODE      (1,1,1000,2,2000,3000)      "DECODE",      DECODE
(2,1,1000,2,2000,3000) "DECODE", DECODE (3,1,1000,2,2000,3000)"DECODE" FROM
DUAL;
```

```
SQL> DECODE      DECODE      DECODE
-----      -----      -----
1000          2000          3000
```

❖ **SQL GROUP(AGGREGATE)FUNCTIONS:-**

- AVG
- MIN
- COUNT
- SUM
- MAX

NOTE: consider following COURSE table for group function example.

TABLE NAME: COURSE

C_ID	C_NAME	C_FEE	C_SEATS
C01	COMPUTER	10000	20
C02	IT	8000	15
C03	MECHANICAL	5000	30
C04	PRINTING	5000	15
C05	CIVIL	10000	30
C06	ELECTRICAL	NULL	NULL

➤ **AVG**

- **AVG ( )** function returns average value for a given column.
- AVG is allowed only on numeric data types.

**Syntax:**

SQL>**AVG ([DISTINCT | ALL]columnname)**

- The DISTINCT qualifier eliminates duplicates. The ALL qualifier retains duplicates.

**Example:**

SQL>SELECT AVG (C\_FEE)"avg",AVG (DISTINCT C\_FEE)"avg",  
AVG(C\_SEATS)"avg",AVG(DISTINCT C\_SEATS)"avg"FROM course;

```
SQL> avg      avg      avg      avg
-----
7600      7666.66667      22      21.66666667
```

➤ **SUM**

- **SUM ( )** function returns total sum for a given column.
- SUM is allowed only on numeric data types.

**Syntax:**

SQL>**SUM ([DISTINCT | ALL] columnname)**

- The DISTINCT qualifier eliminates duplicates. The ALL qualifier retains duplicates.

**Example:**

SQL>SELECT SUM (C\_FEE) "sum", SUM (DISTINCT C\_FEE)"sum", SUM(C\_SEATS)"sum",  
SUM(DISTINCT C\_SEATS)"sum" FROM course;

```
SQL>sum      sum      sum      sum
-----
38000      23000      110      65
```

➤ **MIN**

- **MIN( )** function returns minimum value of a given column.
- MIN is allowed only on numeric data types.

**Syntax: MIN(COLUMNNAME);**

**Example:**

```
SQL>SELECT MIN (C_FEE) "MIN", MIN (C_SEATS) "MIN" FROM COURSE;
SQL> MIN      MIN
```

```
-----
5000      15
```

➤ **MAX**

- **MAX()** function returns maximum value of a given column.
- MAX is allowed only on numeric data types.

**Syntax: MAX (COLUMNNAME);**

**Example:**

```
SQL> SELECT MAX (C_FEE) "MAX", MAX (C_SEATS) "MAX" FROM COURSE;
SQL>MAX      MAX
```

```
-----
10000      30
```

➤ **COUNT (\*)**

- **COUNT (\*)** function returns total no of rows in a table including duplicates and having null values.
- COUNT is allowed only on numeric data types.

**Syntax: COUNT(\*);**

**Example:**

```
SQL>SELECT COUNT(*) "NO. OF ROWS "FROM COURSE;
SQL> NO. OF ROWS
```

```
-----
6
```

➤ **COUNT**

- **COUNT** function returns total no of rows in a table where column does not contain null values.
- COUNT is allowed only on numeric data types.

**Syntax: COUNT ([DISTINCT| ALL] COLUMNNAME);**

**Example:**

```
SQL> SELECT COUNT (*) "NO. OF ROWS ", COUNT (C_FEE) "NO.OF ROWS",
COUNT(DISTINCT C_FEE) "NO.OF ROWS", COUNT (C_SEATS) "NO.OF ROWS", COUNT
(DISTINCT C_SEATS) "NO.OF ROWS"FROM COURSE;
SQL>NO. OF ROWS NO.OF ROWS NO.OF ROWS NO.OF ROWS NO.OF ROWS
```

```
-----
6      5      3      5      3
```

**H. Resources/Equipment Required**

Computer or laptop with a database management system (DBMS) software installed, such as Oracle

**I. Safety and necessary Precautions followed**

Shutdown computer system properly once the Lab hours are finished

**J. Source code/ Solutions to queries:**





**K. Practical related Quiz.**

1. The \_\_\_\_ aggregation operation adds up all the values of the attribute  
A. add                      B. avg                      C. max                      D. sum
2. The \_\_\_\_ aggregation function is used to retrieve minimum value  
A. add                      B. minimum                      C. min                      D. sum
3. Which of the following statements is true regarding the COUNT function?  
A. COUNT (\*) counts duplicate values and NULL values in columns of any data type.  
B. COUNT function cannot work with DATE datatypes.  
C. COUNT (DISTINCT (job\_id)) returns the number of rows excluding rows containing duplicates and including NULL values in the job\_id column.  
D. A SELECT statement using the COUNT function with a DISTINCT keyword cannot have a WHERE clause.
4. Which of the below queries will format a value 1680 as \$16,80.00?  
A. SELECT TO\_CHAR(1680.00,'\$99G99D99') FROM dual;  
B. SELECT TO\_CHAR(1680.00,'\$9,999V99') FROM dual;  
C. SELECT TO\_CHAR(1680.00,'\$9,999D99') FROM dual;  
D. SELECT TO\_CHAR(1680.00,'\$99G999D99') FROM dual;
5. Which of the following examples correctly uses the TO\_NUMBER function to convert a string to a number?  
A TO\_NUMBER('5')                      B TO\_NUMBER('five')  
C TO\_NUMBER(5)                      D TO\_NUM('5.5')

**L. References / Suggestions**

<https://www.geeksforgeeks.org/ddl-commands-syntax/>  
<https://www.w3schools.com/sql/>  
<https://www.tutorialspoint.com/sql/index.htm>

**M. Assessment-Rubrics**

Excellent (21-25 Marks)	Good (15-20 Marks)	Satisfactory (7-14 Marks)	Fair (0-6 Marks)
<ul style="list-style-type: none"> <li>Student executes the queries with correct output.</li> <li>Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>Student executes the queries having correct output with external guidance.</li> <li>Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>Student executes the queries with incorrect output.</li> <li>Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>Student is not able to execute the queries.</li> <li>Student does not submit the lab report in specified time limit.</li> </ul>

**Sign with Date**



**Date:** .....

**Practical No.7:**

Implement SQL queries using Group by, Having and Order by Clause.

Write down the queries using GROUP BY, ORDER BY AND HAVING CLAUSE: -

Consider HR Schema for respective tables to solve all queries.

1. Write a query to find the total salary paid to employees in each department. Display the department name and total salary in descending order.
2. Write a query to find the number of employees in each job title. Display the job title and the number of employees in ascending order.
3. Write a query to find the average salary of employees in each department with more than 5 employees. Display the department name and the average salary in descending order.
4. Write a query to find the department with the highest average salary. Display the department id and the average salary.
5. Write a query to find the total salary of each department id in the HR schema
6. Write a query to find the average salary of employees in each job title, where the average salary is greater than \$5,000.
7. Write a query to find the number of employees in each department who have a commission greater than 0
8. Write a query to find the number of employees hired in each year, sorted by the year in ascending order.

**A. Objective:**

Implementing SQL queries using GROUP BY, HAVING, and ORDER BY clauses is to retrieve and analyse data from a database by grouping and aggregating similar data together, filtering the results based on specific conditions, and sorting the data in a specified order.

**B. Expected Program Outcomes (POs): PO1,PO2,PO3,PO4,PO5,PO7**

**C. Expected Skills to be developed based on competency:**

Develop skills of Data analysis and Problem solving by Compilation, debugging, executing SQL queries

**D. Expected Course Outcomes(Cos)**

Perform joins, sub-queries and nested queries on multiple tables using SQL\*plus.

**E. Practical Outcome(PRo)**

Write data definition queries , compilation, debugging, executing using SQL

**F. Expected Affective domain Outcome (ADos)**

- Follow safety practices.
- Practice good housekeeping.
- Demonstrate working as a leader/a team member.
- Maintain tools and equipment
- Follow ethical practices.

**G. Prerequisite Theory:**

➤ **GROUP BY**

- GROUP BY clause is used in a SELECT statement to group rows into a set of summary rows by values of columns or expressions. The GROUP BY clause returns one row per group.
- The GROUP BY clause is often used with AGGREGATE FUNCTION such as AVG ( ),COUNT ( ),MAX ( ),MIN ( ) and SUM ( ). In this case, the aggregate function returns the summary information per group.

**Syntax:**

```
SQL> SELECT COLUMN_1, COLUMN_2, COLUMN_3 .....COLUMN_N, AGGREGATE
FUNCTION(COLUMN) FROM TABLENAME GROUP BY COLUMN_1, COLUMN_2,
COLUMN_3 .....COLUMN_N;
```

**Example:**

**SQL> select job,sum(sal) from emp group by job;**

JOB	SUM(SAL)
ANALYST	6000
CLERK	4150
SALESMAN	5600
MANAGER	8275
PRESIDENT	5000

(Above query calculate the total salary paid to each job type from employee table)

➤ **HAVING**

- HAVING clause is used in combination with the GROUP BY clause to restrict the groups of returned rows to only those who's the condition is TRUE.

**Syntax:**

```
SQL> SELECT COLUMN_1, COLUMN_2, COLUMN_3 .....COLUMN_N, AGGREGATE
FUNCTION(COLUMN) FROM TABLENAME GROUP BY COLUMN_1, COLUMN_2,
COLUMN_3 .....COLUMN_N HAVING HAVING_CONDITION;
```

**Example:**

**SQL> select job,sum(sal) from emp group by job having job='SALESMAN';**

**SQL>**

JOB	SUM(SAL)
SALESMAN	5600

(Above query calculate the total salary paid to all SALESMEN from employee table)

➤ **ORDER BY**

- The Oracle ORDER BY clause is used to sort the records in your result set. The ORDER BY clause can only be used in SELECT STATEMENT.

**Syntax:**

```
SQL> SELECT COLUMN_1, COLUMN_2, COLUMN_3 .....COLUMN_N OR * FROM
TABLENAME ORDER BY COLUMN ASC OR DESC;
```

**Example:**

**SQL>select \* from emp order by ename asc;**

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7876	ADAMS	CLERK	7788	23-MAY-87	1100	-	20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850	-	30
7782	CLARK	MANAGER	7839	09-JUN-81	2450	-	10
7902	FORD	ANALYST	7566	03-DEC-81	3000	-	20

#### **SQL> Resources/Equipment Required**

Computer or laptop with a database management system (DBMS) software installed, such as Oracle

#### **H. Safety and necessary Precautions followed**

Shutdown computer system properly once the Lab hours are finished

#### **I. Source code /Solutions to queries:**







### **J. Practical related Quiz.**

1. What is the purpose of the GROUP BY clause in SQL?
  - A It allows you to group the results of a query by one or more columns.
  - B It allows you to filter the results of a query based on a condition.
  - C It allows you to join two or more tables together.
  - D It allows you to perform calculations on the results of a query.
2. Which of the following SQL queries uses the HAVING clause correctly?
  - A. select name, count(\*) from orders where count(\*) > 10 group by name;
  - B. select name, count(\*) from orders group by name where count(\*) > 10;
  - C. select name, count(\*) from orders where count(\*) > 10 having name;
  - D. select name, count(\*) from orders group by name having count(\*) > 10;
3. What is the purpose of the HAVING clause in SQL?
  - A. It allows you to group the results of a query by one or more columns.
  - B. It allows you to join two or more tables together.
  - C. It allows you to filter the results of a query based on a condition.
  - D. It allows you to perform calculations on the results of a query.
4. Which of the following SQL queries uses the ORDER BY clause correctly?
  - A. select name, count(\*) from orders group by name order by count(\*) desc;
  - B. select name, count(\*) from orders where count(\*) > 10 group by order by count(\*);
  - C. select name, count(\*) from orders group by name having count(\*) > 10 order by count(\*) desc;
  - D. select name, count(\*) from orders having name order by count(\*) desc;
5. What is the purpose of the ORDER BY clause in SQL?
  - A. It allows you to filter the results of a query based on a condition.
  - B. It allows you to group the results of a query by one or more columns.
  - C. It allows you to perform calculations on the results of a query.
  - D. It allows you to sort the results of a query by one or more columns.

### **K. References / Suggestions**

<https://www.geeksforgeeks.org/ddl-commands-syntax/>

<https://www.w3schools.com/sql/>

<https://www.tutorialspoint.com/sql/index.htm>

**L. Assessment-Rubrics**

Excellent (21-25 Marks)	Good (15-20 Marks)	Satisfactory (7-14 Marks)	Fair (0-6 Marks)
<ul style="list-style-type: none"> <li>• Student executes the queries with correct output.</li> <li>• Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>• Student executes the queries having correct output with external guidance.</li> <li>• Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>• Student executes the queries with incorrect output.</li> <li>• Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>• Student is not able to execute the queries.</li> <li>• Student does not submit the lab report in specified time limit.</li> </ul>

**Sign with Date**



Date: .....

**Practical No.8:**

Implement SQL queries using simple Case Operations and using Group functions and Case operations for getting summary data.

1. Consider following table students.

Name	Gender	Category	Salary
A	Male	Open	5000
B	Female	Open	6000
C	Male	SC	3000
D	Male	SC	4000
E	Female	ST	4500

Write a query to get Gender wise, category wise count of students. Expected output is,

	Open	SC	ST
Male	1	1	0
Female	1	1	1

2. Write a query to get Category wise, Gender wise total Salary. Expected output is,

	Male	Female
Open	5000	6000
SC	7000	0
ST	0	4500

**A. Objective:**

Write and execute SQL queries using simple Case Operations and using Group functions and Case operations for getting summary data.

**B. Expected Program Outcomes (POs): PO1, PO2, PO3, PO4, PO5, PO7****C. Expected Skills to be developed based on competency:**

Compilation, debugging, executing SQL queries

**D. Expected Course Outcomes(Cos)**

Perform joins, sub-queries and nested queries on multiple tables using SQL\*plus.

**E. Practical Outcome(Pro)**

Write data definition queries, compilation, debugging, executing using oracle software

**F. Expected Affective domain Outcome (ADos)**

- Follow safety practices.
- Practice good housekeeping.
- Demonstrate working as a leader/a team member.
- Maintain tools and equipment
- Follow ethical practices.

**G. Prerequisite Theory:**

- The case statement in SQL returns a value on a specified condition. We can use a Case statement in select queries along with Where, Order By, and Group By clause. It can be used in the Insert statement as well. In this article, we would explore the CASE statement and its various use cases.

- **Simple CASE expression**

simple CASE expression matches an expression to a list of simple expressions to determine the result.

The following illustrates the syntax of the simple CASE expression:

```
CASE e
  WHEN e1 THEN
    r1
  WHEN e2 THEN
    r2
  WHEN en THEN
    rn
  [ ELSE r_else ]
END
```

In this syntax, Oracle compares the input expression (e) to each comparison expression e1, e2, ..., en.

The following query uses the CASE expression to calculate the discount for each product category i.e., CPU 5%, video card 10%, and other product categories 8%

```
SELECT product_name,list_price,
CASE category_id
  WHEN 1
  THEN ROUND(list_price * 0.05,2) -- CPU
  WHEN 2
  THEN ROUND(List_price * 0.1,2) -- Video Card
  ELSE ROUND(list_price * 0.08,2) -- other categories
END discount
FROM products ORDER BY product_name;
```

Example:

Suppose we have two salary groups, above 2000 and below 2000. We want to see in which group which employee falls then we can write,  
select ename,deptno, sal,case when sal<=2000 then 1 else 0 end below2000, case when sal>2000 then 1 else 0 end above2000 from emp;

Also when we want to get summary of same data, then we can write,  
select deptno, sum(case when sal<=2000 then 1 else 0 end) below2000, sum(case when sal>2000 then 1 else 0 end) above2000 from emp group by deptno;

**H. Resources/Equipment Required**

Computer or laptop with a database management system (DBMS) software installed, such as Oracle

**I. Safety and necessary Precautions followed**

Shutdown computer system properly once the Lab hours are finished

**J. Source code / Solutions to queries:**



### K. Practical related Quiz.

1. What is the purpose of using simple case operations in SQL?
  - A. To perform calculations on individual columns.
  - B. To transform data in a column based on a specified condition.
  - C. To group rows of data together based on a specified condition.
  - D. To join two or more tables together.
2. Which of the following SQL statements uses a simple case operation correctly?
  - A. select count(\*) from orders where case when amount > 100 then 1 else 0 end;
  - B. select sum(case when status = 'paid' then amount else 0) from orders;
  - C. select name, case when age > 18 then 'adult' else 'minor' end as age\_group from users;
  - D. select avg(case when score < 50 then null else score end) from grades;
3. What is the purpose of using group functions in SQL?
  - A. To perform calculations on individual columns.
  - B. To transform data in a column based on a specified condition.
  - C. To group rows of data together based on a specified condition.
  - D. To join two or more tables together.
4. Which of the following SQL statements uses a group function correctly?
  - A. select name, max(score) from students;
  - B. select count(\*) from orders where amount > 100;
  - C. select sum(amount) from orders where status = 'paid';
  - D. select avg(case when score < 50 then null else score end) from grades;

### L. References / Suggestions

<https://www.geeksforgeeks.org/ddl-commands-syntax/>

<https://www.w3schools.com/sql/>

<https://www.tutorialspoint.com/sql/index.htm>

### M. Assessment-Rubrics

Excellent (21-25 Marks)	Good (15-20 Marks)	Satisfactory (7-14 Marks)	Fair (0-6 Marks)
<ul style="list-style-type: none"> <li>Student executes the queries with correct output.</li> <li>Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>Student executes the queries having correct output with external guidance.</li> <li>Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>Student executes the queries with incorrect output.</li> <li>Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>Student is not able to execute the queries.</li> <li>Student does not submit the lab report in specified time limit.</li> </ul>

**Sign with Date**

**Date:** .....

### **Practical No.9:**

Implement SQL queries using Set operators like Union, union all, Intersect, Minus etc..

Write down the queries using SET OPERATORS: -

1. Find the employees who are either managers or sales representatives:
2. Find the employees who are both managers and sales representatives
3. Find the employees who are sales representatives but not managers
4. Find the names of departments that have either a manager or an administrative assistant
5. Find the names of departments that have both a manager and an administrative assistant
6. Find the employees who have either a manager or a subordinate with the last name 'King'

#### **A. Objective:**

Write and execute SQL queries on various Set Operators

#### **B. Expected Program Outcomes (POs): PO1, PO2, PO3, PO4, PO5, PO7**

#### **C. Expected Skills to be developed based on competency:**

Compilation, debugging, executing SQL queries

#### **D. Expected Course Outcomes(Cos)**

Perform joins, sub-queries and nested queries on multiple tables using SQL\*plus

#### **E. Practical Outcome(Pro)**

Write data definition queries, compilation, debugging, executing using oracle software

#### **F. Expected Affective domain Outcome (ADos)**

- Follow safety practices.
- Practice good housekeeping.
- Demonstrate working as a leader/a team member.
- Maintain tools and equipment
- Follow ethical practices.

#### **G. Prerequisite Theory:**

- ORACLE installation

### **❖ SQL SET OPERATORS:-**

➤ UNION

➤ INTERSECT

➤ UNION ALL

➤ MINUS

#### **UNION**

- The UNION set operator returns all distinct rows selected by either query, without ignoring the NULL values. That means any duplicate rows will be removed. (Ascending by default)

#### **Syntax:**

**SQL>SELECT COLUMNNAME FROM TABLENAME1**

**UNION**

**SELECT COLUMNNAME FROM TABLENAME2;**

#### **UNION ALL**

- The UNION ALL set operator returns all rows selected by either query, without ignoring the NULL values. That means any duplicates will remain in the final result set.

**Syntax:**

```
SQL>SELECT COLUMNNAME FROM TABLENAME
      UNION ALL
      SELECT COLUMNNAME FROM TABLENAME;
```

**INTERSECT**

- The INTERSECT set operator returns all distinct rows selected by both queries. That means only those rows common to both queries will be present in the final result set.

**Syntax:**

```
SQL>SELECT COLUMNNAME FROM TABLENAME
      INTERSECT
      SELECT COLUMNNAME FROM TABLENAME;
```

**MINUS**

- The MINUS set operator returns all distinct rows selected by the first query but not the second.

**Syntax:**

```
SQL>SELECT COLUMNNAME FROM TABLENAME
      MINUS
      SELECT COLUMNNAME FROM TABLENAME;
```

**Example: Suppose we have two different tables Fdetail3 and Fdetail5 mentioning 3<sup>rd</sup> and 5<sup>th</sup> semester faculty details as following.**

Fdetail3	
Faculty Name	Department
A	Computer
B	Mechanical
C	Computer
D	Mechanical

Fdetail5	
Faculty Name	Department
A	Computer
Q	Electrical
R	Printing
C	Computer

**Now we want to get list of all 3<sup>rd</sup> and 5<sup>th</sup> sem faculties without duplicates then,**

**SQL>select fname from fdetail3 union select fname from fdetail5;**

**Output Gives: A,B,C,D,Q,R,C**

**Example: In above mentioned example, suppose we want to keep duplicate values as it is while collecting all faculty details then,**

**SQL> select fname from fdetail3 union all select fname from fdetail5;**

**Output Gives: A,B,C,D,A,Q,R,C**

**Example: Suppose we want only those faculties which teach in both 3<sup>rd</sup> and 5<sup>th</sup> sem then,**

**SQL> select fname from fdetail3 Intersect select fname from fdetail5;**

**Output Gives: A**

**Example:**

**SQL> Suppose we want to get only those faculty which teach in sem 3 not in sem 5 then,**

**select fname from fdetail3 Minus select fname from fdetail5;**

**Output Gives: B,D**

**H. Resources/Equipment Required**

Computer or laptop with a database management system (DBMS) software installed, such as Oracle

**I. Safety and necessary Precautions followed**

Shutdown computer system properly once the Lab hours are finished

**J. Source code/Solutions to queries:**



**K. Practical related Quiz.**

1. What is the purpose of using set operators in SQL?
  - A. To perform calculations on individual columns.
  - B. To transform data in a column based on a specified condition.
  - C. To combine the results of two or more queries into a single result set.
  - D. To group rows of data together based on a specified condition.
2. Which of the following set operators removes duplicates from the result set?
  - A. UNION      B. UNION ALLC. INTERSECT      D. MINUS
3. Which of the following set operators retains duplicates in the result set?
  - A. UNION      B. UNION ALLC. INTERSECT      D. MINUS
- E. Which of the following SQL statements correctly uses the INTERSECT operator?
  - A. select \* from table1 intersect select \* from table2;
  - B. select \* from table1 union all select \* from table2;
  - C. select \* from table1 minus select \* from table2;
  - D. select \* from table1 union select \* from table2;
- F. What is the purpose of using the MINUS operator in SQL?
  - A. To find all distinct values in one query that are not in another query.
  - B. To combine the results of two or more queries into a single result set.
  - C. To find the intersection of two queries.
  - D. To perform calculations on individual columns.

**L. References / Suggestions**

<https://www.geeksforgeeks.org/ddl-commands-syntax/>

<https://www.w3schools.com/sql/>

<https://www.tutorialspoint.com/sql/index.htm>

**M. Assessment-Rubrics**

Excellent (21-25 Marks)	Good (15-20 Marks)	Satisfactory (7-14 Marks)	Fair (0-6 Marks)
<ul style="list-style-type: none"> <li>Student executes the queries with correct output.</li> <li>Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>Student executes the queries having correct output with external guidance.</li> <li>Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>Student executes the queries with incorrect output.</li> <li>Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>Student is not able to execute the queries.</li> <li>Student does not submit the lab report in specified time limit.</li> </ul>

**Sign with Date**

**Date:** .....

### **Practical No.10:**

Retrieve data spread across various tables or same table using various Joins.

1. Write a query to retrieve the employee ID, first name, last name, department name, and job title of all employees who work in the Sales department.
2. Write a query to retrieve the employee ID, first name, last name, salary, and commission percentage of all employees who have a commission percentage greater than 10%
3. Write a query to retrieve the employee ID, first name, last name, department name, and manager name of all employees who work in the same department as their manager
4. Write a query to retrieve the employee ID, first name, last name, department name, and job title of all employees who have a manager whose last name is King.
5. Write a query to retrieve the employee ID, first name, last name, department name, job title, and salary of all employees who work in the same department as employee ID 176.

#### **A. Objective:**

Write and execute SQL queries on various Joins.

#### **B. Expected Program Outcomes (POs): PO1, PO2, PO3, PO4, PO5, PO7**

#### **C. Expected Skills to be developed based on competency:**

Compilation, debugging, executing SQL queries

#### **D. Expected Course Outcomes(Cos)**

Perform joins, sub-queries and nested queries on multiple tables using SQL\*plus.

#### **G. Practical Outcome(Pro)**

Write data definition queries, compilation, debugging, executing using oracle software

#### **H. Expected Affective domain Outcome (ADos)**

- Follow safety practices.
- Practice good housekeeping.
- Demonstrate working as a leader/a team member.
- Maintain tools and equipment
- Follow ethical practices.

#### **I. Prerequisite Theory:**

- Join operation is used to combine rows from two or more tables based on a related column between them. It allows you to retrieve data from multiple tables in a single query by specifying the relationship between them. Here's a brief overview of the theory behind performing queries with joins in a DBMS:

Types of Joins:

- Inner Join: Returns only the matching rows from both tables based on the join condition.

- Left Join: Returns all rows from the left (or first) table and the matching rows from the right (or second) table. If there is no match, NULL values are included for the columns from the right table.
- Right Join: Returns all rows from the right table and the matching rows from the left table. If there is no match, NULL values are included for the columns from the left table.
- Full Join: Returns all rows from both tables. If there is no match, NULL values are included for the columns from the respective table.
- Cross Join: Returns the Cartesian product of the two tables, resulting in all possible combinations of rows.
- Join Conditions:
- Equality Join: The most common type of join, where rows are matched based on the equality of values in the specified columns.
- Non-Equality Join: Rows are matched based on conditions other than equality, such as greater than, less than, etc.
- Self-Join: Joining a table with itself based on a relationship between different columns within the same table.
- Syntax: The syntax for performing joins in a DBMS varies slightly depending on the specific database system, but the general structure is as follows:

SELECT columns FROM table1 JOIN table2 ON join\_condition;

**J. Resources/Equipment Required**

Computer or laptop with a database management system (DBMS) software installed, such as Oracle

**K. Safety and necessary Precautions followed**

Shutdown computer system properly once the Lab hours are finished

**L. Source code / Solutions to queries:**





**M. Practical related Quiz.**

1. What is the purpose of using joins in SQL?
  - A. To perform calculations on individual columns.
  - B. To transform data in a column based on a specified condition.
  - C. To combine data from two or more tables into a single result set.
  - D. To group rows of data together based on a specified condition.
2. Which of the following types of join returns only the rows that have matching values in both tables?
  - A. INNER JOIN
  - B. LEFT JOIN
  - C. RIGHT JOIN
  - D. FULL OUTER JOIN
3. Which of the following types of join returns all the rows from the left table and the matching rows from the right table, and fills in missing values with NULL?
  - A. INNER JOIN
  - B. LEFT JOIN
  - C. RIGHT JOIN
  - D. FULL OUTER JOIN
4. Which of the following types of join returns all the rows from the right table and the matching rows from the left table, and fills in missing values with NULL?
  - A. INNER JOIN
  - B. LEFT JOIN
  - C. RIGHT JOIN
  - D. FULL OUTER JOIN
5. What is the purpose of using aliases in SQL joins?
  - A. To rename tables or columns for clarity and readability.
  - B. To limit the number of rows returned in the result set.
  - C. To perform calculations on individual columns.
  - D. To group rows of data together based on a specified condition.

**N. References / Suggestions**

<https://www.geeksforgeeks.org/ddl-commands-syntax/>

<https://www.w3schools.com/sql/>

<https://www.tutorialspoint.com/sql/index.htm>

**O. Assessment-Rubrics**

Excellent (21-25 Marks)	Good (15-20 Marks)	Satisfactory (7-14 Marks)	Fair (0-6 Marks)
<ul style="list-style-type: none"> <li>Student executes the queries with correct output.</li> <li>Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>Student executes the queries having correct output with external guidance.</li> <li>Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>Student executes the queries with incorrect output.</li> <li>Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>Student is not able to execute the queries.</li> <li>Student does not submit the lab report in specified time limit.</li> </ul>

**Sign with Date**

**Date:** .....

### **Practical No.11:**

Retrieve data from multiple tables using Sub-queries (Multiple, Correlated)

1. Retrieve employees from the "employees" table who have a salary greater than the average salary of all employees.
2. Retrieve departments from the "departments" table along with the count of employees in each department.
3. Retrieve job titles from the "jobs" table of employees whose department locations are in Canada.

#### **A. Objective:**

- To retrieve data from multiple tables using sub-queries is to extract data from multiple tables based on some conditions or criteria. Sub-queries are used to break down complex queries into smaller, more manageable pieces, and help to retrieve data that is not easily obtainable from a single table.
- Multiple level subqueries are a powerful tool in SQL that allows us to extract data from nested subqueries. This means that the result of one subquery can be used as the input for another subquery, allowing for complex data extraction and manipulation.

#### **B. Expected Program Outcomes (POs): PO1,PO2,PO3,PO4,PO5,PO7**

#### **C. Expected Skills to be developed based on competency:**

Compilation, debugging, executing SQL queries

#### **D. Expected Course Outcomes(Cos)**

Perform joins, sub-queries and nested queries on multiple tables using SQL\*plus.

#### **E. Practical Outcome(Pro)**

Write data definition queries , compilation, debugging, executing using oracle software

#### **F. Expected Affective domain Outcome (ADos)**

- Follow safety practices.
- Practice good housekeeping.
- Demonstrate working as a leader/a team member.
- Maintain tools and equipment
- Follow ethical practices.

#### **G. Prerequisite Theory:**

- A sub-query is a query that is nested inside another query, and it can be used to retrieve data that will be used in the outer query.
- For example, if we want to get all employees data from Sales department then we can write,  

```
Select * from emp where deptno = (select deptno from dept where deptname='SALES');
```

If the inner query returns more than one row, then we will get error of "inner query returns multiple rows" error in oracle. To resolve it we can use IN, ANY,ALL etc operators.

For example,

```
Select * from emp where deptno in (select deptno from dept where deptname='SALES' or deptname='MARKETING');
```

- Here's an example of how to use a co-related sub-query to retrieve data from multiple tables in Oracle:

Suppose we have two tables, "Customers" and "Orders". The "Customers" table contains customer information such as customer ID, name, and address, while the "Orders" table contains information about the orders placed by each customer, such as order ID, date, and total cost.

- To retrieve the name and total number of orders for each customer, we can use a sub-query as follows:

```
SELECT c.name,  
       (SELECT COUNT(*) FROM Orders o WHERE o.customer_id = c.customer_id) as  
       num_orders FROM Customers c;
```

In this query, the sub-query is the SELECT statement inside the parentheses. It counts the number of orders for each customer by selecting all orders from the "Orders" table where the customer ID matches the customer ID in the outer query. The outer query selects the customer name from the "Customers" table and displays the number of orders returned by the sub-query as "num\_orders".

#### **H. Resources/Equipment Required**

Computer or laptop with a database management system (DBMS) software installed, such as Oracle

#### **I. Safety and necessary Precautions followed**

Shutdown computer system properly once the Lab hours are finished

#### **J. Source code / Solutions to queries:**







### K. Practical related Quiz.

1. What is a subquery in SQL?
  - A. A query that performs a calculation on individual columns.
  - B. A query that retrieves data from multiple tables.
  - C. A query that is nested within another query.
  - D. A query that groups rows of data together based on a specified condition.
2. What is a correlated subquery in SQL?
  - A. A subquery that is not dependent on the outer query.
  - B. A subquery that retrieves data from multiple tables.
  - C. A subquery that is nested within another query.
  - D. A subquery that is dependent on the outer query.
3. Which of the following is an example of a multiple-level subquery in SQL?
  - A. select \* from table1 where column1 in (select column2 from table2 where column3 = 'value');
  - B. select \* from table1 where column1 = (select max(column2) from table2);
  - C. select \* from table1 where column1 in (select column2 from table2 where column3 in (select column4 from table3));
  - D. select \* from table1 where column1 = (select avg(column2) from table2);
4. What is the purpose of using a subquery in SQL?
  - A. To perform calculations on individual columns.
  - B. To transform data in a column based on a specified condition.
  - C. To retrieve data from multiple tables.
  - D. To group rows of data together based on a specified condition.
5. What is the maximum number of levels a subquery can have in SQL?
  - A. 1      B. 2      C. 3      D. 255

### L. References / Suggestions

<https://www.geeksforgeeks.org/ddl-commands-syntax/>  
<https://www.w3schools.com/sql/>  
<https://www.tutorialspoint.com/sql/index.htm>

### M. Assessment-Rubrics

Excellent (21-25 Marks)	Good (15-20 Marks)	Satisfactory (7-14 Marks)	Fair (0-6 Marks)
<ul style="list-style-type: none"> <li>Student executes the queries with correct output.</li> <li>Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>Student executes the queries having correct output with external guidance.</li> <li>Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>Student executes the queries with incorrect output.</li> <li>Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>Student is not able to execute the queries.</li> <li>Student does not submit the lab report in specified time limit.</li> </ul>

**Sign with Date**

**Date:** .....

### **Practical No.12:**

Perform queries to Create, alter and update views

1. Create a view called "employee\_info" that shows the employee's first name, last name, job title, department name, and salary
2. Add a new column called "hire\_date" to the "employee\_info" view
3. Update the "employee\_info" view to only show employees who have a salary greater than \$50,000
4. Create a view named "my\_view" that shows only the employees who belong to department 20
5. Alter the "my\_view" view to show only the employees who belong to department 30
6. Drop the "my\_view" view from the database

#### **A. Objective:**

- To perform queries to create, alter, and update views is to allow for efficient data manipulation and retrieval in a database management system. Views are essentially virtual tables that are based on the results of a query, and they provide a simplified, customized representation of the data stored in the underlying tables.
- Creating views can be useful for simplifying complex queries, reducing the amount of code needed to retrieve certain information, and providing controlled access to specific data

#### **B. Expected Program Outcomes (POs): PO1,PO2,PO3,PO5,PO6,PO7**

#### **C. Expected Skills to be developed based on competency:**

Compilation, debugging, executing SQL queries

#### **D. Expected Course Outcomes(Cos)**

Apply rules on datasets using SQL\*Plus constraints

#### **E. Practical Outcome(Pro)**

Write View queries, compilation, debugging, executing using oracle software

#### **F. Expected Affective Domain Outcome (ADos)**

- Follow safety practices.
- Practice good housekeeping.
- Demonstrate working as a leader/a team member.
- Maintain tools and equipment
- Follow ethical practices.

#### **G. Prerequisite Theory:**

- A view is a virtual table that is derived from one or more base tables. Views can be used to simplify complex queries, to restrict access to sensitive data, or to present data in a particular format
1. Creating a View: To create a view, you need to define a query that specifies the data you want to include in the view. The query can involve one or more tables and can include various filtering conditions, aggregations, and joins. Here's the general syntax for creating a view:

- Syntax:

CREATE VIEW view\_name AS SELECT\_Query...

For example, create view emp\_dept\_20 as select \* from emp where deptno=20;

2. Altering a View: To alter a view, you can modify its definition by changing the underlying query. The alteration can include adding or removing columns, modifying the filtering conditions, or changing the join conditions. However, some DBMS may have limitations on altering views, such as not allowing modifications to the underlying query structure. The specific syntax for altering a view depends on the DBMS you are using.

For oracle, you can write, Create or Replace view view\_name as select Query....;

For example,

Create or replace view emp\_dept\_20 as select ename, salary from emp where deptno=20;

3. Removing the view: To remove the view, you can use drop command.

Drop view viewname;

For example, Drop view emp\_dept\_20;

#### **H. Resources/Equipment Required**

Computer or laptop with a database management system (DBMS) software installed, such as Oracle

#### **I. Safety and necessary Precautions followed**

Shutdown computer system properly once the Lab hours are finished

#### **J. Source code / Solutions to queries:**





### K. Practical related Quiz.

1. what is a view in sql?
  - A. a temporary table that is created in memory.
  - B. a virtual table that is based on the result set of a select statement.
  - C. a table that is created and managed by the dbms.
  - D. a table that is used to store metadata about other tables in the database.
2. how do you create a view in sql?
  - A. create view view\_name as select column\_name(s) from table\_name where condition;
  - B. create table view\_name as select column\_name(s) from table\_name where condition;
  - C. create view view\_name using table\_name where condition;
  - D. create table view\_name using select column\_name(s) from table\_name where condition;
3. how do you update a view in sql?
  - A. update view\_name set column\_name = value where condition;
  - B. update table\_name set column\_name = value where condition;
  - C. alter view view\_name as select column\_name(s) from table\_name where condition;
  - D. create or replace view view\_name as select\_query.
4. how do you alter a view in sql?
  - A. alter view view\_name as select column\_name(s) from table\_name where condition;
  - B. alter table view\_name as select column\_name(s) from table\_name where condition;
  - C. update view\_name set column\_name = value where condition;
  - D. you cannot alter a view in sql.
5. what is the benefit of using a view in sql?
  - A. views can improve query performance by reducing the amount of data retrieved.
  - B. views can be used to store data permanently.
  - C. views can be used to update data in multiple tables at once.
  - D. views can be used to create temporary tables that are stored in memory.

### L. References / Suggestions

<https://www.geeksforgeeks.org/ddl-commands-syntax/>  
<https://www.w3schools.com/sql/>  
<https://www.tutorialspoint.com/sql/index.htm>



### M. Assessment-Rubrics

Excellent (21-25 Marks)	Good (15-20 Marks)	Satisfactory (7-14 Marks)	Fair (0-6 Marks)
<ul style="list-style-type: none"> <li>• Student executes the queries with correct output.</li> <li>• Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>• Student executes the queries having correct output with external guidance.</li> <li>• Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>• Student executes the queries with incorrect output.</li> <li>• Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>• Student is not able to execute the queries.</li> <li>• Student does not submit the lab report in specified time limit.</li> </ul>

**Sign with Date**

**Date:** .....

### **Practical No.13**

Implement with Domain Integrity, Entity Integrity and Referential Integrity constraints.

1. Create a table named "**users**" with the following columns/ attributes:  
id (integer, primary key), name (varchar2(50)),email (varchar2(100))  
password (varchar2(100))
2. Create a table named "**products**" with the following columns:  
id (integer, primary key), name (varchar2(100)), price (decimal(10,2) check  
price>100), description (text)
3. Create a table named "**orders**" with the following columns:  
id (integer, primary key),user\_id (integer, foreign key references users(id)),  
product\_id (integer, foreign key references products(id))  
quantity (integer) not null

#### **A. Objective:**

- To perform queries to domain integrity, entity integrity, and referential integrity constraints is to ensure that data in a database is consistent, accurate, and reliable.
- To ensure the quality and accuracy of data stored in a database

#### **B. Expected Program Outcomes (POs): PO1,PO2,PO3,PO5,PO6,PO7**

#### **C. Expected Skills to be developed based on competency:**

Compilation, debugging, executing SQL queries

#### **D. Expected Course Outcomes(Cos)**

Apply rules on datasets using SQL\*Plus constraints.

#### **E. Practical Outcome(Pro)**

Write data definition queries using integrity constraints, compilation, debugging, executing using oracle software

#### **F. Expected Affective domain Outcome (ADos)**

- Follow safety practices.
- Practice good housekeeping.
- Demonstrate working as a leader/a team member.
- Maintain tools and equipment
- Follow ethical practices.

#### **G. Prerequisite Theory:**

- Domain Integrity, Entity Integrity, and Referential Integrity are fundamental concepts in database management systems (DBMS) like Oracle. These constraints ensure that the data stored in the database is accurate, consistent, and valid.
  1. Domain Integrity: Domain Integrity ensures that the values entered in a column or attribute of a table conform to a specified data type or domain. For example, if a column is defined as storing only positive integers, any attempt to enter a negative number or a non-integer value will be rejected.
  2. Entity Integrity: Entity Integrity ensures that each row or record in a table has a unique identifier, typically a primary key. This constraint guarantees that there

are no duplicate records in the table, ensuring that each record can be identified uniquely.

3. **Referential Integrity:** Referential Integrity ensures that the relationships between tables are maintained by enforcing the consistency of foreign key values. For example, if a table has a foreign key constraint that references a primary key in another table, the referential integrity constraint ensures that any changes to the referenced primary key value are reflected in the foreign key value in the referencing table.

- In Oracle, these constraints can be defined when creating a table or added later using the ALTER TABLE statement. We can use “constraint constraint\_name” ahead of the all constraints to give our own constraint name, otherwise oracle will give system defined constraint name.

For example,

- To add a domain integrity constraint to a table, the following syntax can be used:
- Create table tablename(columnname1 datatype1 constraint constraint\_name NotNull/Check condition, Columnname2 datatype2 NotNull/Check condition, .....ColumnnameN datatypeN NotNull/Check condition)

```
create table xyz (a number(10) constraint xyz_a_nn not null,b number(5));
```

- Alter table table\_name add constraint constraint\_name check (column\_name <value>);  
Alter table xyz add constraint xyz\_b\_ck check (b>50);
- To add a primary key constraint to a table, the following syntax can be used:  
Create table tablename(columnname1 datatype1 primary key, Columnname2 datatype2, .....ColumnnameN datatypeN)

```
alter table table_name
add constraint constraint_name
primary key (column_name)
```

- To add a foreign key constraint to a table, the following syntax can be used:

```
Create table tablename1(columnname1 datatype1 references table2(columnx),
Columnname2 datatype2, .....ColumnnameN datatypeN);
```

```
Alter table table_name add constraint constraint_name foreign key
(column_name) references referenced_table (referenced_column)
```

## H. Resources/Equipment Required

Computer or laptop with a database management system (DBMS) software installed, such as Oracle

**I. Safety and necessary Precautions followed**

Shutdown computer system properly once the Lab hours are finished

**J. Source code / Solutions to queries:**



**K. Practical related Quiz.**

1. What is domain integrity in SQL?
  - A. A set of rules that ensure that each row in a table is unique.
  - B. A set of rules that ensure that each column in a table has a valid data type and range of values.
  - C. A set of rules that ensure that each table in a database is linked to other tables in a consistent manner.
  - D. A set of rules that ensure that each table in a database has a primary key.
2. What is entity integrity in SQL?
  - A. A set of rules that ensure that each row in a table is unique.
  - B. A set of rules that ensure that each column in a table has a valid data type and range of values.
  - C. A set of rules that ensure that each table in a database is linked to other tables in a consistent manner.
  - D. A set of rules that ensure that each table in a database has a primary key.
3. What is referential integrity in SQL?
  - A. A set of rules that ensure that each row in a table is unique.
  - B. A set of rules that ensure that each column in a table has a valid data type and range of values.
  - C. A set of rules that ensure that each table in a database is linked to other tables in a consistent manner.
  - D. A set of rules that ensure that each table in a database has a primary key.
4. Which constraint is used to ensure that a column in a table is linked to a primary key column in another table?
  - A. not null constraint
  - B. check constraint
  - C. unique constraint
  - D. foreign key constraint
5. How do you create a FOREIGN KEY constraint in SQL?
  - A. alter table table\_name add foreign key (column\_name) references referenced\_table\_name (referenced\_column\_name);
  - B. create table table\_name (column\_name data\_type constraint fk\_constraint\_name foreign key (column\_name) references referenced\_table\_name (referenced\_column\_name));
  - C. create constraint fk\_constraint\_name foreign key (column\_name) references referenced\_table\_name (referenced\_column\_name);
  - D. you cannot create a foreign key constraint in sql.

**L. References / Suggestions**

<https://www.geeksforgeeks.org/ddl-commands-syntax/>  
<https://www.w3schools.com/sql/>  
<https://www.tutorialspoint.com/sql/index.htm>

### M. Assessment-Rubrics

Excellent (21-25 Marks)	Good (15-20 Marks)	Satisfactory (7-14 Marks)	Fair (0-6 Marks)
<ul style="list-style-type: none"> <li>• Student executes the queries with correct output.</li> <li>• Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>• Student executes the queries having correct output with external guidance.</li> <li>• Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>• Student executes the queries with incorrect output.</li> <li>• Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>• Student is not able to execute the queries.</li> <li>• Student does not submit the lab report in specified time limit.</li> </ul>

**Sign with Date**

Date: .....

### Practical No.14:

Perform queries to Create synonyms, sequence and index.

1. Create a synonym for the employees table in the HR schema
2. Create a synonym for the departments table in the HR schema
3. Create a sequence named seq\_emp\_id that starts at 1 and increments by 1
4. Create a sequence for generating department IDs
5. Create an index named idx\_emp\_last\_name on the last\_name column of the employees table.
6. Create an index for the department names

#### A. Objective:

- The objective of creating synonyms is to provide an alternative name or alias for an existing database object, such as a table, view, or procedure. Synonyms can be useful in simplifying the syntax of SQL statements and in providing an additional layer of security by hiding the underlying object names.
- The objective of creating sequences is to generate unique sequential numbers or values that can be used as primary keys or other identifiers in tables. Sequences can be used to simplify the process of inserting new records into a table, as they can automatically generate the next available value.
- The objective of creating indexes is to improve the performance of database queries by creating a data structure that allows for faster searching and retrieval of data. Indexes can be created on one or more columns in a table, and they can greatly improve the performance of queries that use those columns in a WHERE clause or JOIN condition.

#### B. Expected Program Outcomes (POs): PO1, PO2, PO3, PO5, PO6, PO7

#### C. Expected Skills to be developed based on competency:

Compilation, debugging, executing SQL queries related to database objects.

#### D. Expected Course Outcomes(Cos)

Use Database Objects like Synonym, Views, Sequence and Index for efficient database handling.

#### E. Practical Outcome(Pro)

Write data objects, compilation, debugging, executing using oracle software

#### F. Expected Affective Domain Outcome (ADos)

- Follow safety practices.
- Practice good housekeeping.
- Demonstrate working as a leader/a team member.
- Maintain tools and equipment
- Follow ethical practices.

#### G. Prerequisite Theory:

##### • Synonym –

A synonym is an alternative name for objects such as tables, views, sequences, stored procedures and other database objects.



Creating a Synonym for an object: -

**SYNTAX: -**

CREATE [PUBLIC] SYNONYM [SCHEMA.] synonym\_name FOR [SCHEMA.] object\_name;

If the word PUBLIC is written in query, then the synonym can be accessible by all the users. But user must have appropriate privileges to the object.

If the SCHEMA phrase is not used, then reference is made to user's own schema.

**Dropping Synonym:-**

A synonym can be dropped by using DROP SYNONYM. When the synonym is dropped, the changes made in table through synonym remains unchanged.

**SYNTAX:-**

DROP [PUBLIC] SYNONYM [SCHEMA.] synonym\_name [FORCE];

While dropping synonym, if the word PUBLIC is written, then there is no need to write a schema name.

The FORCE word will force oracle to drop synonym, even if it has dependencies.

**Use of Synonym:-**

- To hide the actual identity of the object.
- To abbreviated the longer name.

**Sequence –**

A sequence is an automatic counter which generates sequential numbers whenever required.

**Creating a Sequence:-**

**SYNTAX:-**

CREATE SEQUENCE sequence\_name

INCREMENT BY n

START WITH n

MAXVALUE n / NOMAXVALUE

MINVALUE n / NOMINVALUE

CACHE n / NOCACHE

CYCLE / NOCYCLE

ORDER / NOORDER;

Here, in syntax, **START WITH** specifies the first sequence number, **INCREMENT BY** specifies the interval between sequence numbers, **MAXVALUE** specifies the sequence maximum value, **CYCLE** specifies to repeat cycle of generating values after reaching maximum value, **and CACHE** specifies how many values to generate in advance and to keep in memory for faster access.

The value generated by a sequence can be used in insert and update operations. The minimum information required for generating numbers using a sequence is

- The starting number
- The maximum number that can be generated by a sequence
- The increment value for generating the next number

**Altering Sequence:-**

An already created sequence can be altered by following command. The start value of the sequence cannot be altered.

**SYNTAX:-**

```
ALTER SEQUENCE sequence_name  
INCREMENT BY n  
MAXVALUE n / NOMAXVALUE  
MINVALUE n / NOMINVALUE  
CACHE n / NOCACHE  
CYCLE / NOCYCLE  
ORDER / NOORDER;
```

**Dropping Sequence:-**

A sequence can be dropped by using DROP SEQUENCE command.

**SYNTAX:-**

```
DROP SEQUENCE sequence_name;
```

**Referencing Sequence:-**

Oracle provides two pseudo columns NEXTVAL and CURRVAL to access the values generated by the sequence.

To view sequence value, SELECT statement is used as below:

```
SELECT sequence_name.CURRVAL FROM DUAL;
```

This will display the current value of the sequence. To display the next value held in the cache, following SELECT statement is used:

```
SELECT sequence_name.NEXTVAL FROM DUAL;
```

**Index –**

An index is an ordered list of content of a column or group of columns in a table.

An index created on the single column of the table is called simple index. When multiple table columns are included in the index, it is called composite index.

**Creating an Index for a table:-**

**SYNTAX: -**

**Simple Index:-**

```
CREATE INDEX index_name ON tablename (column_name);
```

**Composite Index:-**

```
CREATE INDEX index_name ON tablename (column_name, column_name);
```

Above indexes do not enforce uniqueness so the columns included in the index can hold duplicate values. Indexes that deny duplicate values for the indexed columns are called Unique Index.

**Creating a unique Index for a table: -**

To create unique index, the keyword UNIQUE should be included in the CREATE INDEX command.

**SYNTAX:-**

**Simple Unique Index:-**

CREATE UNIQUE INDEX index\_name ON tablename (column\_name);

**Composite Unique Index: -**

CREATE UNIQUE INDEX index\_name ON tablename (column\_name, column\_name);

**Dropping Indexes:-**

An index associated with the table can be removed by using DROP INDEX command.

**SYNTAX: -** DROP INDEX index\_name;

**H. Resources/Equipment Required**

Computer or laptop with a database management system (DBMS) software installed, such as Oracle

**I. Safety and necessary Precautions followed**

Shutdown computer system properly once the Lab hours are finished

**J. Source code / Solutions to queries:**





### K. Practical related Quiz.

1. What is a synonym in sql?
  - A. an alternative name for a table, view, or other database object.
  - B. a temporary table that is created in memory.
  - C. a table that is created and managed by the dbms.
  - D. a table that is used to store metadata about other tables in the database.
2. How do you create a synonym in sql?
  - A. create synonym synonym\_name for table\_name;
  - B. create synonym table\_name for synonym\_name;
  - C. alter synonym synonym\_name for table\_name;
  - D. you cannot create a synonym in sql.
3. What is a sequence in sql?
  - A. a table that is used to store metadata about other tables in the database.
  - B. an object that generates a series of unique numbers or values.
  - C. an object that is used to improve query performance by reducing the amount of data retrieved.
  - D. an object that is used to update data in multiple tables at once.
4. How do you create a sequence in sql?
  - A. create sequence sequence\_name start with 1 increment by 1;
  - B. create table sequence\_name start with 1 increment by 1;
  - C. alter sequence sequence\_name start with 1 increment by 1;
  - D. you cannot create a sequence in sql.
5. What is an index in sql?
  - A. a table that is used to store metadata about other tables in the database.
  - B. an object that generates a series of unique numbers or values.
  - C. an object that is used to improve query performance by reducing the amount of data retrieved.
  - D. an object that is used to update data in multiple tables at once.
6. How do you create an index in sql?
  - A. create index index\_name on table\_name (column\_name);
  - B. create table index\_name on table\_name (column\_name);
  - C. alter index index\_name on table\_name (column\_name);
  - D. you cannot create an index in sql.

### L. References / Suggestions

<https://www.geeksforgeeks.org/ddl-commands-syntax/>  
<https://www.w3schools.com/sql/>  
<https://www.tutorialspoint.com/sql/index.htm>

### M. Assessment-Rubrics

Excellent (21-25 Marks)	Good (15-20 Marks)	Satisfactory (7-14 Marks)	Fair (0-6 Marks)
<ul style="list-style-type: none"> <li>• Student executes the queries with correct output.</li> <li>• Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>• Student executes the queries having correct output with external guidance.</li> <li>• Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>• Student executes the queries with incorrect output.</li> <li>• Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>• Student is not able to execute the queries.</li> <li>• Student does not submit the lab report in specified time limit.</li> </ul>

**Sign with Date**

**Date:** .....

### **Practical No.15:**

Implement PL/SQL programs using control structures.

1. Write a PL/SQL block which takes student information as input and then display it.
2. Write a PL/SQL block to calculate the area of a circle for a value 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 using while loop.
3. Write a PL/SQL block to print the values 1 to 10 using a FOR loop.
4. Write a PL/SQL block to print the values 1 to 10 in reverse order using for loop.
5. Write a PL/SQL block to make calculator using GOTO statement. (Accept two number then enter choice – 1. Addition 2. Subtraction 3. Multiplication 4. Division and display result)
6. Write a PL/SQL block that will accept an account number from the user. Check if the user's balance is less than the minimum balance, only then deduct Rs.100/- from the balance. The process is fired on the ACCT\_MSTR table. (minimum balance is 5000)

#### **A. Objective:**

- To implement PL/SQL programs using control structures is to develop programs that can make decisions and perform different actions based on those decisions. Control structures in PL/SQL include conditional statements, loops, and exception handling.
- Using these control structures, PL/SQL programmers can create more sophisticated and robust programs that can handle a wide range of scenarios and user inputs. Additionally, control structures help to improve the efficiency and readability of PL/SQL code by reducing the amount of repetitive code that needs to be written.

#### **B. Expected Program Outcomes (POs): PO1, PO2, PO3, PO4, PO5, PO6, PO7**

#### **C. Expected Skills to be developed based on competency:**

Compilation, debugging, executing PL-SQL Blocks

#### **D. Expected Course Outcomes(Cos)**

Write PL/SQL block using concept of Cursor Management, Error Handling, Package and Triggers

#### **E. Practical Outcome(Pro)**

Write PL SQL Blocks, compilation, debugging, executing using oracle software

#### **F. Expected Affective Domain Outcome (ADos)**

- Follow safety practices.
- Practice good housekeeping.
- Demonstrate working as a leader/a team member.
- Maintain tools and equipment
- Follow ethical practices.

#### **G. Prerequisite Theory:**



### Basics of PL/SQL

Programming language/structured query language. It provides facility of procedural and function. It provides programming techniques like branching, looping, and condition check. It is fully structured programming language. It decreases the network traffic because entire block of code is passed to the DBA at one time for execution.

### Basic Structure of PL/SQL

PL/SQL stands for Procedural Language/SQL. PL/SQL extends SQL by adding constructs found in procedural languages, resulting in a structural language that is more powerful than SQL. The basic unit in PL/SQL is a block. A block has the following structure:

#### *Syntax:*

**DECLARE**

```
/* Declarative section: variables, types, and local subprograms. */
```

**BEGIN**

```
/* Executable section: procedural and SQL statements go here. */
```

```
/* This is the only section of the block that is required. */
```

**EXCEPTION**

```
/* Exception handling section: error handling statements go here. */
```

**END;**

### Control Structures

PL/SQL Control Structures are used to control flow of execution. PL/SQL provides different kinds of statements to provide such type of procedural capabilities. These statements are almost same as that of provided by other languages.

The flow of control statements can be classified into the following categories:

- Conditional Control
- Iterative Control
- Sequential Control

#### Conditional Control:

PL/SQL allows the use of an IF statement to control the execution of a block of code.

In PL/SQL, the IF - THEN - ELSIF - ELSE - END IF construct in code blocks allow specifying certain conditions under which a specific block of code should be executed.

#### *Syntax:*

```
IF < Condition > THEN
    < Statement >
ELSIF <Condition> THEN
    < Statement >
ELSE < Statement >
END IF;
```

#### *Example:*

Create file named "condi.sql"

```
DECLARE
```

```

    a Number := 30;
    b Number;
BEGIN
    IF a > 40 THEN
        b := a - 40;
        DBMS_OUTPUT.PUT_LINE('b=' || b);
    elsif a = 30 then
        b := a + 40;
        DBMS_OUTPUT.PUT_LINE('b=' || b);
    ELSE
        b := 0;
        DBMS_OUTPUT.PUT_LINE('b=' || b);
    END IF;
END;
/

```

**Output:**


```

Run SQL Command Line

SQL>set serveroutput on

SQL>start d://condi.sql
b=70

PL/SQL successfully completed.

```

**NOTE: Remember to use SET SERVEROUTPUT ON at start of PL-SQL Session to see the output.**

**Iterative Control:**

Iterative control indicates the ability to repeat or skip sections of a code block. A **loop** marks a sequence of statements that has to be repeated. The keyword loop has to be placed before the first statement in the sequence of statements to be repeated, while the keyword end loop is placed immediately after the last statement in the sequence.

Once a loop begins to execute, it will go on forever. Hence a conditional statement that controls the number of times a loop is executed always accompanies loops.

PL/SQL supports the following structures for iterative control:

**Simple loop:**

In simple loop, the key word loop should be placed before the first statement in the sequence and the keyword end loop should be written at the end of the sequence to end the loop.

*Syntax:***Loop**

**< Sequence of statements >**

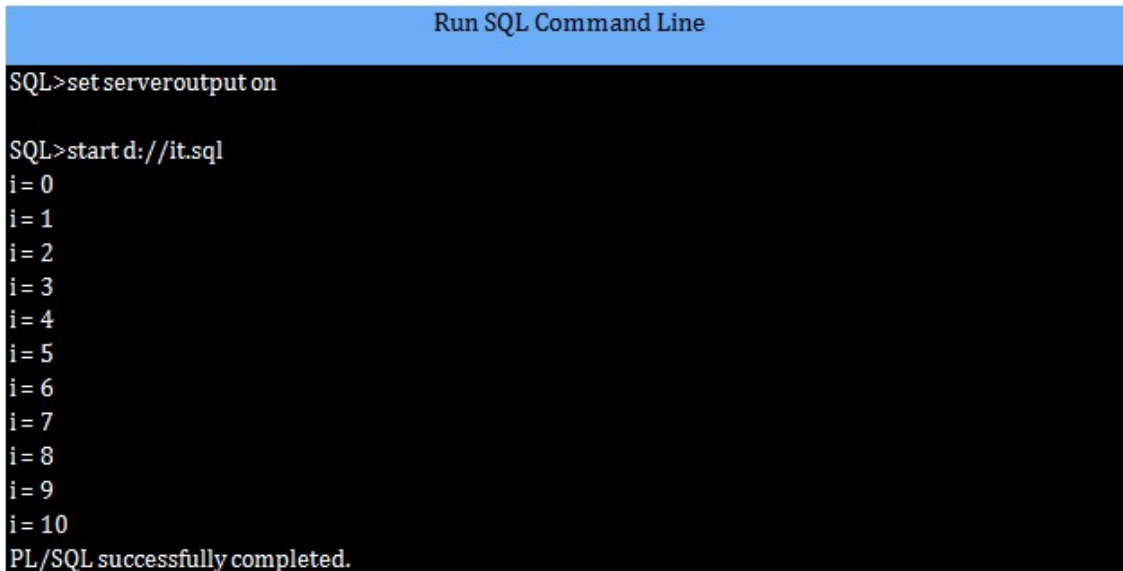
**End loop;**

*Example:*

Create file named it.sql

```
DECLARE
    i number := 0;
BEGIN
    LOOP
        dbms_output.put_line ('i = '||i);
        i:=i+1;
        EXIT WHEN i>=11;
    END LOOP;
END;
/
```

*Output:*

A screenshot of a SQL Command Line window titled "Run SQL Command Line". The window has a black background with white text. The text shows the following sequence of commands and output: "SQL>set serveroutput on", "SQL>start d://it.sql", followed by a list of values "i = 0" through "i = 10" on separate lines, and finally "PL/SQL successfully completed." at the bottom.

```
Run SQL Command Line

SQL>set serveroutput on

SQL>start d://it.sql
i = 0
i = 1
i = 2
i = 3
i = 4
i = 5
i = 6
i = 7
i = 8
i = 9
i = 10
PL/SQL successfully completed.
```

**WHILE loop**

The **while** loop executes commands in its body as long as the condition remains true

*Syntax:*

```
while < condition >
loop
    < action >
end loop
```

*Example:*

Find reverse of given number using while loop

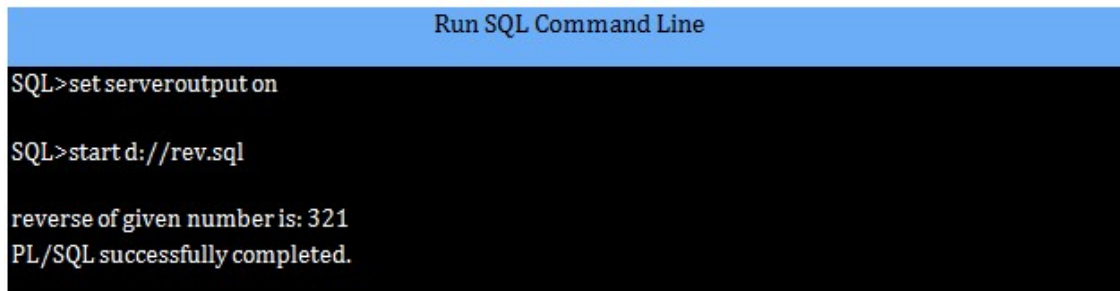
```
declare
    num number(3) :=123;
    ans number(3) :=0;
    i number(3) :=0;
begin
```

```

while num != 0
loop
    i:=mod(num,10);
    ans:=(ans * 10 ) + i;
    num:=floor(num/10);
end loop;
dbms_output.put_line('reverse of given number is: ' || ans);
end;
/

```

*Output :*



The screenshot shows a terminal window titled "Run SQL Command Line". The commands entered are:
   
SQL>set serveroutput on
   
SQL>start d://rev.sql
   
The output displayed is:
   
reverse of given number is: 321
   
PL/SQL successfully completed.

### The FOR Loop

The **FOR** loop can be used when the number of iterations to be executed are known.

#### *Syntax:*

The variable in the For Loop need not be declared. Also the increment value cannot be specified. The For Loop variable is always incremented by 1.

```

FOR variable IN [REVERSE] start..end
LOOP
    < Action >
END LOOP;

```

#### *Example:*

```

declare
    i number ;
begin
    for i in 1 .. 10
    loop
        dbms_output.put_line ('i = '||i);
    end loop;
end;
/

```

*Output:*

```

Run SQL Command Line
SQL>set serveroutput on

SQL>start d:/it.sql
i= 1
i= 2
i= 3
i= 4
i= 5
i= 6
i= 7
i= 8
i= 9
i= 10
PL/SQL successfully completed.

```

Sequential Control:

### The GOTO Statement

The GOTO statement changes the flow of control within a PL/SQL block. This statement allows execution of a section of code, which is not in the normal flow of control. The entry point into such a block of code is marked using the tags «userdefined name». The GOTO statement can then make use of this user-defined name to jump into that block of code for execution.

*Syntax :*

GOTO jump;

....

<<jump>>

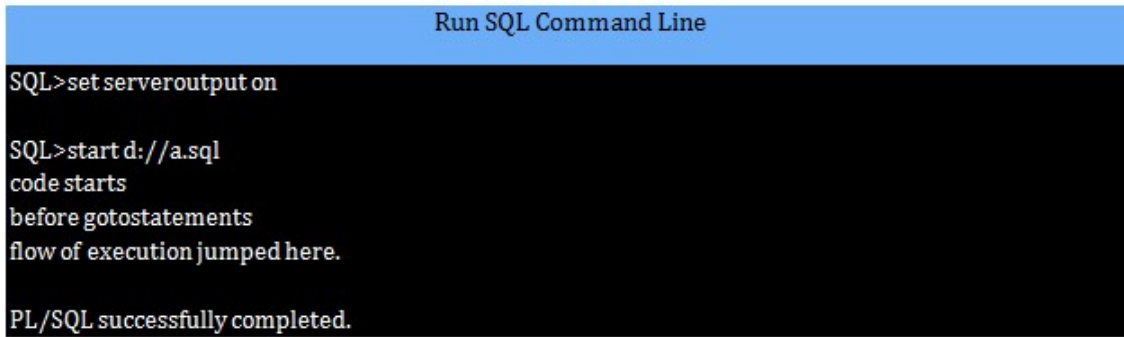
*Example :*

```

declare
begin
  dbms_output.put_line ('code starts');
  dbms_output.put_line ('before gotostatements');
  goto down;
  dbms_output.put_line ('statement will not get executed..');
<<down>>
  dbms_output.put_line ('flow of execution jumped here. ');
end;
/

```

*Output :*



```
Run SQL Command Line
SQL>set serveroutput on
SQL>start d://a.sql
code starts
before gotostatements
flow of execution jumped here.
PL/SQL successfully completed.
```

**H. Resources/Equipment Required**

Computer or laptop with a database management system (DBMS) software installed, such as Oracle

**I. Safety and necessary Precautions followed**

Shutdown computer system properly once the Lab hours are finished

**J. Source code / Solutions to queries:**









### K. Practical related Quiz.

1. What is pl/sql?
  - A. a database management system.
  - B. a programming language used to access and manipulate oracle databases.
  - C. a web development framework.
  - D. an operating system.
2. What are the basic control structures used in pl/sql?
  - A. if-else, for, while, and goto.
  - B. select, from, where, and order by.
  - C. insert, update, and delete.
  - D. none of the above.
3. What is the syntax for an if-else statement in pl/sql?
  - A. if condition then statement; else statement; end if;
  - B. if condition then statement; else end if;
  - C. if condition then statement; end if; else statement;
  - D. if condition then statement; end;
4. What is the syntax for a for loop in pl/sql?
  - A. for counter in lower\_bound..upper\_bound loop statement; end loop;
  - B. for counter in lower\_bound to upper\_bound loop statement; end loop;
  - C. for counter = lower\_bound to upper\_bound loop statement; end loop;
  - D. for counter = lower\_bound..upper\_bound loop statement; end loop;
5. What is the syntax for a while loop in pl/sql?
  - A. while condition loop statement; end loop;
  - B. do while condition loop statement; end loop;
  - C. while condition do statement; end loop;
  - D. none of the above.
6. What is the purpose of the goto statement in pl/sql?
  - A. to transfer control to a specified label within the same program.
  - B. to exit the program.
  - C. to skip over a block of code.
  - D. none of the above.

### L. References / Suggestions

<https://www.geeksforgeeks.org/ddl-commands-syntax/>  
<https://www.w3schools.com/sql/>  
<https://www.tutorialspoint.com/sql/index.htm>

**M. Assessment-Rubrics**

Excellent (21-25 Marks)	Good (15-20 Marks)	Satisfactory (7-14 Marks)	Fair (0-6 Marks)
<ul style="list-style-type: none"> <li>• Student executes the queries with correct output.</li> <li>• Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>• Student executes the queries having correct output with external guidance.</li> <li>• Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>• Student executes the queries with incorrect output.</li> <li>• Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>• Student is not able to execute the queries.</li> <li>• Student does not submit the lab report in specified time limit.</li> </ul>

**Sign with Date**

**Date:** .....

### **Practical No.16:**

Implement PL/SQL programs using Cursors.

1. Develop a program to retrieve and display employee details based on a given department ID.
2. Create a program to update the salary of employees based on a given job title.
3. Create a PL/SQL procedure that uses a cursor to fetch and display employees who have been hired within a specific date range in the HR schema

#### **A. Objective:**

- To provide a way for the PL/SQL programmer to manipulate and process data stored in a relational database. Cursors in PL/SQL allow programmers to retrieve data from one or more database tables and process it one row at a time, which can be very useful in scenarios where it's not feasible or efficient to retrieve all the data at once.
- Cursors can be used in a wide range of PL/SQL applications, from simple database queries to complex data processing tasks, and can help improve the performance and efficiency of database operations

#### **B. Expected Program Outcomes (POs): PO1,PO2,PO3,PO4,PO5,PO6,PO7**

#### **C. Expected Skills to be developed based on competency:**

Compilation, debugging, executing SQL queries

#### **D. Expected Course Outcomes(Cos)**

Write PL/SQL block using concept of Cursor Management, Error Handling, Package and Triggers.

#### **E. Practical Outcome(Pro)**

Write data definition queries, compilation, debugging, executing using oracle software

#### **F. Expected Affective Domain Outcome (ADos)**

- Follow safety practices.
- Practice good housekeeping.
- Demonstrate working as a leader/a team member.
- Maintain tools and equipment
- Follow ethical practices.

#### **G. Prerequisite Theory:**

##### **Cursors**

A cursor is a temporary work area created in the system memory when a SQL statement is executed. A cursor contains information on a select statement and the rows of data accessed by it. This temporary work area is used to store the data retrieved from the database, and manipulate this data. A cursor can hold more than one row, but can process only one row at a time. The set of rows the cursor holds is called the active set. There are two types of cursors in PL/SQL:

1. Implicit cursors.

## 2. Explicit cursors.

Both implicit and explicit cursors have the same functionality, but they differ in the way they are accessed.

Cursor Attributes	
Name	Description
%FOUND	Returns TRUE if record was fetched successfully, FALSE otherwise.
%NOTFOUND	Returns TRUE if record was not fetched successfully, FALSE otherwise.
%ROWCOUNT	Returns number of records fetched from cursor at that point in time.
%ISOPEN	Returns TRUE if cursor is open, FALSE otherwise.

Implicit cursor:

These are created by default when DML statements like, INSERT, UPDATE, and DELETE statements are executed. They are also created when a SELECT statement that returns just one row is executed. When you execute DML statements like DELETE, INSERT, UPDATE and SELECT statements, implicit statements are created to process these statements. Oracle provides few attributes called as implicit cursor attributes to check the status of DML operations.

The cursor attributes available are

- %FOUND
- %NOTFOUND
- %ROWCOUNT
- %ISOPEN

For example, when you execute INSERT, UPDATE, or DELETE statements the cursor attributes tell us whether any rows are affected and how many have been affected. When a SELECT... INTO statement is executed in a PL/SQL Block, implicit cursor attributes can be used to find out whether any row has been returned by the SELECT statement. PL/SQL returns an error when no data is selected.

Consider the PL/SQL Stock that uses implicit cursor attributes as shown below:

*Example:*

```

DECLARE
  Eid number(3);
BEGIN
  UPDATE emp set eid=&eid where salary=&salary;
  eid:=sql%rowcount;
  IF SQL%found then
    dbms_output.put_line('success');
  ELSE
    dbms_output.put_line ( ' not' );
  END IF;
  dbms_output.put_line( 'rowcount'||eid);
END;
```

```
Run SQL Command Line
SQL>STARTD://c.sql
success
PL/SQL Procedure successfully completed.
```

### Explicit Cursors:

They must be created when you are executing a SELECT statement that returns more than one row. Even though the cursor stores multiple records, only one record can be processed at a time, which is called as current row. When you fetch a row the current row position moves to next row. An explicit cursor is defined in the declaration section of the PL/SQL Block. It is created on a SELECT Statement which returns more than one row. We can provide a suitable name for the cursor.

### Syntax:

```
CURSOR cursor_name IS select_statement;
```

### Here,

**cursor\_name** -A suitable name for the cursor.

**Select\_statement** - A select query which returns multiple rows.

### How to use Explicit Cursor?

There are four steps in using an Explicit Cursor.

1. **DECLARE** the cursor in the declaration section
2. **OPEN** the cursor in the Execution Section.
3. **FETCH** the data from cursor into PL/SQL variables or records in the Execution Section.
4. **CLOSE** the cursor in the Execution Section before you end the PL/SQL Block.

Declaring a Cursor in the Declaration Section:

### Syntax:

```
CURSOR CURSORNAME ISSELECT.....;
```

### How to access an Explicit Cursor?

These are the three steps in accessing the cursor.

### OPEN THE CURSOR:

### Syntax:

```
OPEN cursor_name;
```

### FETCH THE RECORDS IN THE CURSOR ONE AT A TIME:

### Syntax:

```
FETCH cursor_name INTO record_name;
OR
FETCH cursor_name INTO variable_list;
```

### CLOSE THE CURSOR:

**Syntax:**

```
CLOSE cursor_name;
```

When a cursor is opened, the first row becomes the current row. When the data is fetched it is copied to the record or variables and the logical pointer moves to the next row and it becomes the current row. On every fetch statement, the pointer moves to the next row. If you want to fetch after the last row, the program will throw an error. When there is more than one row in a cursor we can use loops along with explicit cursor attributes to fetch all the records.

**Points to remember while fetching a row:**

1. We can fetch the rows in a cursor to a PL/SQL Record or a list of variables created in the PL/SQL Block.
2. If you are fetching a cursor to a PL/SQL Record, the record should have the same structure as the cursor.
3. If you are fetching a cursor to a list of variables, the variables should be listed in the same order in the fetch statement as the columns are present in the cursor.

**General Form of using an explicit cursor is:***Syntax:*

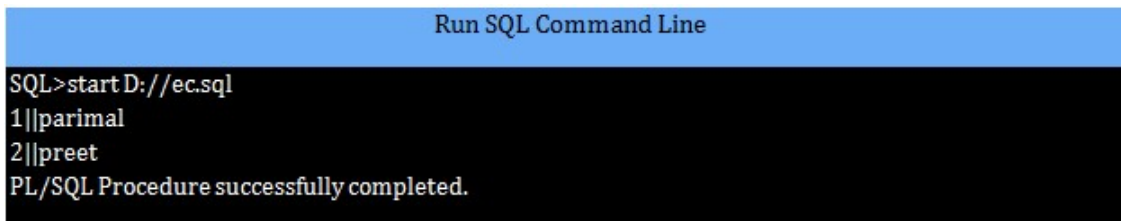
```
DECLARE
  variables;
  records;
  create a cursor;
BEGIN
  OPEN cursor;
  FETCH cursor;
  process the records;
  CLOSE cursor;
END;
/
```

*Example:*

```
DECLARE
  CURSOR er IS select eid,name from emp order by name ;
  id emp.eid%type;
  ename emp.name%type;
BEGIN
  OPEN er;
  Loop
  FETCH er into id,ename;
  Exit when er%notfound;
  dbms_output.put_line (id || ename);
```

```
end loop;  
close er;  
END;  
/
```

*Output:*



The screenshot shows a window titled "Run SQL Command Line". The command prompt shows the following text: "SQL>start D://ec.sql", "1||parimal", "2||preet", and "PL/SQL Procedure successfully completed."

**CURSOR FOR LOOP:**

Oracle provides other loop-statements to control loops specifically for cursors. This statement is a variation of the basic FOR loop, and it is known as cursor for loops.

*Syntax:*

```
FOR VARIABLE IN CURSORNAME  
LOOP  
<EXECUTE COMMANDS>  
END LOOP
```

**PARAMETERIZED CURSORS:**

Oracle allows passing parameters to cursors that can be used to provide condition with WHERE clause. If parameters are passed to cursor, that cursor is called **parameterized cursors**.

*Syntax:*

```
CURSOR CURSORNAME (VARIABLENAME DATATYPES ) IS SELECT.....
```

And, parameters can be passed to cursor while opening it using syntax-

*Syntax:*

```
OPEN CURSORNAME (VALUE / VARIABLE / EXPRESSION );
```

#### **H. Resources/Equipment Required**

Computer or laptop with a database management system (DBMS) software installed, such as Oracle

#### **I. Safety and necessary Precautions followed**

Shutdown computer system properly once the Lab hours are finished

#### **J. Source code / Solutions to queries :**











### Practical related Quiz.

1. What is a cursor in PL/SQL?
  - A. A temporary table used to store data.
  - B. A pointer to a result set returned by a query.
  - C. An object used to connect to a database.
  - D. A block of code used to loop through a result set.
2. What are the two types of cursors in PL/SQL?
  - A. Implicit and explicit.
  - B. Forward-only and scrollable.
  - C. Static and dynamic.
  - D. Simple and complex.
3. What is the syntax for declaring an explicit cursor in PL/SQL?
  - A. `CURSOR cursor_name IS SELECT column_name FROM table_name;`
  - B. `CURSOR cursor_name(column_name data_type) IS SELECT column_name FROM table_name;`
  - C. `DECLARE cursor_name CURSOR FOR SELECT column_name FROM table_name;`
  - D. None of the above.
4. What is the purpose of the OPEN statement in PL/SQL cursors?
  - A. To execute a query and retrieve the result set.
  - B. To allocate memory for the cursor.
  - C. To close the cursor.
  - D. None of the above.
5. What is the purpose of the FETCH statement in PL/SQL cursors?
  - A. To retrieve the next row from the result set.
  - B. To close the cursor.
  - C. To deallocate memory for the cursor.
  - D. None of the above.
6. What is the purpose of the CLOSE statement in PL/SQL cursors?
  - A. To close the cursor and deallocate memory.
  - B. To open the cursor.
  - C. To fetch the previous row from the result set.
  - D. None of the above.

### K. References / Suggestions

<https://www.geeksforgeeks.org/ddl-commands-syntax/>

<https://www.w3schools.com/sql/>

<https://www.tutorialspoint.com/sql/index.htm>

### L. Assessment-Rubrics

Excellent (21-25 Marks)	Good (15-20 Marks)	Satisfactory (7-14 Marks)	Fair (0-6 Marks)
<ul style="list-style-type: none"> <li>• Student executes the queries with correct output.</li> <li>• Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>• Student executes the queries having correct output with external guidance.</li> <li>• Student submits the lab report in specified time limit</li> </ul>	<ul style="list-style-type: none"> <li>• Student executes the queries with incorrect output.</li> <li>• Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>• Student is not able to execute the queries.</li> <li>• Student does not submit the lab report in specified time limit.</li> </ul>

**Sign with Date**

Date: .....

### Practical No.17:

Implement PL/SQL programs using exception handling.

1. Write a program to convert a non-numeric string to number and display its value. If string is given it should raise INVALID\_NUMBER error and display appropriate message.
2. Write a program queries the employees table in the HR schema for a non-existing employee, which will raise a NO\_DATA\_FOUND exception.
3. Create a PL/SQL program that inserts a new employee record into the HR schema's employees table. However, if a record already exists with the same employee ID, the program should raise a named exception called "DUP\_VAL\_ON\_INDEX" and handle it appropriately. The program should output a message indicating whether the record was successfully inserted or if the exception was raised.
4. Create a PL/SQL program that inserts data into the employees table of the HR schema. The program should handle exceptions raised on encountering a NULL value for the primary key column, employee\_id, or a NOT NULL value for the first\_name column.

#### A. Objective:

To improve the reliability and robustness of the code by handling unexpected or exceptional situations that may occur during program execution.

#### B. Expected Program Outcomes (POs): PO1,PO2,PO3,PO4,PO5,PO6,PO7

#### C. Expected Skills to be developed based on competency:

Compilation, debugging, executing SQL queries

#### D. Expected Course Outcomes(Cos)

Write PL/SQL block using concept of Cursor Management, Error Handling, Package and Triggers.

#### E. Practical Outcome(PRo)

Write PL-SQL programs with Error Handling.

#### F. Expected Affective domain Outcome (ADos)

- Follow safety practices.
- Practice good housekeeping.
- Demonstrate working as a leader/a team member.
- Maintain tools and equipment
- Follow ethical practices.

#### G. Prerequisite Theory:

### EXCEPTIONS

An Exception is an error situation, which arises during program execution. When an error occurs exception is raised, normal execution is stopped and control transfers to exception handling part. Exception handlers are routines written to handle the exception. The exceptions can be internally defined (system-defined or pre-defined) or User-defined exception.

**EXCEPTION**

WHEN <ExceptionName> THEN  
<User Defined Action To Be Carried Out>

*Syntax:*

**PREDEFINED EXCEPTION:**

**Predefined exception** is raised automatically whenever there is a violation of Oracle coding rules. Predefined exceptions are those like ZERO\_DIVIDE, which is raised automatically when we try to divide a number by zero. Other built-in exceptions are given below. You can handle unexpected Oracle errors using OTHERS handler. It can handle all raised exceptions that are not handled by any other handler. It must always be written as the last handler in exception block.

Exception	Raised when....
DUP_VAL_ON_INDEX	When you try to insert a duplicate value into a unique column.
INVALID_CURSOR	It occurs when we try accessing an invalid cursor.
INVALID_NUMBER	On usage of something other than number in place of number value.
LOGIN_DENIED	At the time when user login is denied.
TOO_MANY_ROWS	When a select query returns more than one row and the destination variable can take only single value.
VALUE_ERROR	When an arithmetic, value conversion, truncation, or constraint error occurs.
CURSOR_ALREADY_OPEN	Raised when we try to open an already open cursor.

Predefined exception handlers are declared globally in package STANDARD. Hence we need not have to define them rather just use them. The biggest advantage of exception handling is it improves readability and reliability of the code. Errors from many statements of code can be handles with a single handler. Instead of checking for an error at every point we can just add an exception handler and if any exception is raised it is handled by that. For checking errors at a specific spot it is always better to have those statements in a separate begin – end block.

*Example:*

```
declare
  n number;
begin
  n:=10/0;
exception when zero_divide then
  dbms_output.put_line ('divide by zero error occurs...');
end;
/
```



```

Run SQL Command Line
SQL>set serveroutput on
SQL>start D://d.sql
divide by zero error occurs...
PL/SQL procedure successfully completed.

```

**USER-DEFINED EXCEPTIONS:**

The technique that is used is to bind a numbered exception handler to a name using Pragma Exception\_init (). This binding of a numbered exception handler, to a name (i.e. a String), is done in the Declare section of a PL/SQL block. The Pragma action word is a call to a pre-compiler, which immediately binds the numbered exception handler to a name when encountered.

The function Exception\_init() takes two parameters the first is the user defined exception name the second is the Oracle engine's exception number. These lines will be included in the Declare section of the PL/SQL block.

The user defined exception name must be the statement that immediately precedes the Pragma Exception\_init() statement.

*Syntax:*

```

DECLARE
< ExceptionName > EXCEPTION ;
    PRAGMA EXCEPTION_INIT (< ExceptionName >, <ErrorCodeNo>);
BEGIN

```

Using this technique it is possible to bind appropriate numbered exception handlers to names and use these names in the Exception section of a PL/SQL block. When this is done the default exception handling code of the exception handler is overridden and the user-defined exception handling code is executed.

*Syntax:*

```

DECLARE
< ExceptionName > EXCEPTION ;
    PRAGMA EXCEPTION_INIT (< ExceptionName >, <ErrorCodeNo>);
BEGIN
    . . . .
EXCEPTION
    WHEN < ExceptionName > THEN
< Action >
END;

```

*Example:*

Create table named emp have column like id with notnull constraint,name and etc.

```

DECLARE
    e_MissingNull EXCEPTION;
    PRAGMA EXCEPTION_INIT(e_MissingNull, -1400);
BEGIN

```

```

INSERT INTO emp(id) VALUES (NULL);
EXCEPTION
    WHEN e_MissingNull then
        DBMS_OUTPUT.put_line('ORA-1400 occurred');
END;
/

```

```

Run SQL Command Line
SQL>set serveroutput on
SQL>start D://e.sql
ORA-1400 occurred
PL/SQL procedure successfully completed.

```

#### USER DEFINED EXCEPTION HANDLING:

To trap business rules being violated the technique of raising user-defined exceptions and then handling them, is used.

User-defined error conditions must be declared in the declarative part of any PL/SQL block. In the executable part, a check for the condition that needs special attention is made. If that condition exists, the call to the user-defined exception is made using a RAISE statement. The exception once raised is then handled in the Exception handling section of the PL/SQL code block.

#### *Syntax:*

```

DECLARE
< EXCEPTIONNAME > EXCEPTION;
BEGIN
<SQL SENTENCE >;
    IF < CONDITION > THEN
        RAISE <EXCEPTIONNAME>;
    END IF;
EXCEPTION
    WHEN <EXCEPTIONNAME> THEN {USER DEFINED ACTION TO BE TAKEN};
END;

```

#### *EXAMPLE:*

```

DECLARE
EX EXCEPTION;
BEGIN
    IF TO_CHAR(SYSDATE,'DY')=='SUN' THEN
        RAISE EX;
    END IF;
EXCEPTION
    WHEN EX THEN
        DBMS_OUTPUT.PUT_LINE('NO TRANSCATIONS TODAY');
END;

```

/

*Output:*

```
Run SQL Command Line
SQL>set serveroutput on
SQL>start D://ex.sql
No Transactions Today
PL/SQL procedure successfully completed.
```

**H. Resources/Equipment Required**

Computer or laptop with a database management system (DBMS) software installed, such as Oracle

**I. Safety and necessary Precautions followed**

Shutdown computer system properly once the Lab hours are finished

**J. Source code / Solutions to queries :**





**K. Practical related Quiz.**

1. What is exception handling in PL/SQL?
  - A. A mechanism to handle errors that occur during program execution.
  - B. A way to optimize query performance.
  - C. A method to secure data from unauthorized access.
  - D. None of the above.
2. What is the syntax for the EXCEPTION block in PL/SQL?
  - A. EXCEPTION WHEN exception\_name THEN statement; END;
  - B. EXCEPTION WHEN exception\_name statement; END;
  - C. WHEN exception\_name THEN statement; END EXCEPTION;
  - D. None of the above.
3. What is the purpose of the RAISE statement in PL/SQL exception handling?
  - A. To handle an exception and continue program execution.
  - B. To raise an exception and terminate program execution.
  - C. To skip over a block of code.
  - D. None of the above.
4. What is the purpose of the SQLCODE function in PL/SQL exception handling?
  - A. To return the error code associated with the most recent exception.
  - B. To execute an SQL statement.
  - C. To return the number of rows affected by an SQL statement.
  - D. None of the above.
5. What is the purpose of the SQLERRM function in PL/SQL exception handling?
  - A. To return the error message associated with the most recent exception.
  - B. To execute an SQL statement.
  - C. To return the number of rows affected by an SQL statement.
  - D. None of the above.
6. What is the purpose of the WHEN OTHERS clause in PL/SQL exception handling?
  - A. To handle all exceptions not explicitly handled by other clauses.
  - B. To handle only specific exceptions.
  - C. To raise an exception and terminate program execution.
  - D. None of the above.

**L. References / Suggestions**

<https://www.geeksforgeeks.org/ddl-commands-syntax/>

<https://www.w3schools.com/sql/>

<https://www.tutorialspoint.com/sql/index.htm>

### M. Assessment-Rubrics

Excellent (21-25 Marks)	Good (15-20 Marks)	Satisfactory (7-14 Marks)	Fair (0-6 Marks)
<ul style="list-style-type: none"> <li>• Student executes the queries with correct output.</li> <li>• Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>• Student executes the queries having correct output with external guidance.</li> <li>• Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>• Student executes the queries with incorrect output.</li> <li>• Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>• Student is not able to execute the queries.</li> <li>• Student does not submit the lab report in specified time limit.</li> </ul>

**Sign with Date**

Date: .....

### Practical No.18:

Implement user defined procedures and functions using PL/SQL blocks.

1. Create a procedure that takes in a number and displays its square
2. Create a function that takes in two numbers and returns their sum
3. Create a PL/SQL block that defines a function to calculate the area of a rectangle and a procedure to calculate the perimeter of a rectangle. The user should be prompted to enter the length and width of the rectangle, and the results should be displayed
4. Write a procedure to check whether given number is odd or even. Also write the PL/SQL block to invoke it.

#### A. Objective:

To implement user-defined procedures and functions using PL/SQL blocks is to provide a way to encapsulate business logic and perform specific tasks on the database. Procedures and functions allow you to group a set of SQL statements together and give them a name, so that they can be executed as a single unit.

#### B. Expected Program Outcomes (POs): PO1,PO2,PO3,PO4,PO5,PO6,PO7

#### C. Expected Skills to be developed based on competency:

Compilation, debugging, executing SQL queries

#### D. Expected Course Outcomes(Cos)

Write PL/SQL block using concept of Cursor Management, Error Handling, Package and Triggers.

#### E. Practical Outcome(Pro)

Write PL-SQL Procedures and Functions.

#### F. Expected Affective domain Outcome (ADos)

- Follow safety practices.
- Practice good housekeeping.
- Demonstrate working as a leader/a team member.
- Maintain tools and equipment
- Follow ethical practices.

#### G. Prerequisite Theory:

#### PROCEDURES & FUNCTIONS

"A **procedures** or **function** is a group or set of SQL and PL/SQL statements that perform a specific task." A function and procedure is a named PL/SQL Block which are similar. The major difference between a procedure and a function is, a function must always return a value, but a procedure may or may not return a value.

#### PROCEDURES:

A procedure is named PL/SQL block which performs one or more specific task. This is similar to a procedure in other programming languages. A procedure has a header and a body. The header consists of the name of the procedure and the parameters or variables passed to the procedure.



The body consists of declaration section, execution section and exception section similar to a general PL/SQL Block. A procedure is similar to an anonymous PL/SQL Block but it is named for repeated usage.

We can pass parameters to procedures in three ways :

Parameters	Description
IN type	These types of parameters are used to send values to stored procedures.
OUT type	These types of parameters are used to get values from stored procedures. This is similar to a return type in functions.
IN OUT type	These types of parameters are used to send values and get values from stored procedures.

**A procedure may or may not return any value.**

*Syntax:*

```
CREATE [ORREPLACE] PROCEDURE procedure_name (<Argument> {IN, OUT, INOUT}
<Datatype>,...)
IS
    Declaration section<variable, constant>;
BEGIN
    Execution section
EXCEPTION
    Exceptionsection
END;
/
```

**IS** - marks the beginning of the body of the procedure and is similar to DECLARE in anonymous PL/SQL Blocks. The code between IS and BEGIN forms the Declaration section. The syntax within the brackets [ ] indicate they are optional. By using CREATE OR REPLACE together the procedure is created if no other procedure with the same name exists or the existing procedure is replaced with the current code.

How to execute a Procedure?

There are two ways to execute a procedure :

- From the SQL prompt : EXECUTE [or EXEC] procedure\_name;
- Within another procedure – simply use the procedure name : procedure\_name;

*Example:*

Create table named emp have two column id and salary with number datatype.

```
create or replace procedure p1(id in number, sal in number)
as
begin
insert into emp values(id, sal);
    dbmd_output.put_line('value inserted.');
```

/

```

Run SQL Command Line

SQL>set serveroutput on
SQL>start D://pr.sql
Procedure created.

SQL>exec p1(5,4);
VALUE INSERTED.
PL/SQL procudere successfully completed.

SQL>select * from emp;
   ID      SALARY
-----
    2      5000

```

**FUNCTIONS:**

A function is a named PL/SQL Block which is similar to a procedure. The major difference between a procedure and a function is, a function must always return a value, but a procedure may or may not return a value.

**Syntax:**

```

CREATE [OR REPLACE] FUNCTION function_name [parameters]
RETURN return_datatype {IS, AS}
Declaration_section <variable,constant> ;
BEGIN
    Execution_section
    Return return_variable;
EXCEPTION
    exception section
    Return return_variable;
END;
/

```

**RETURN TYPE:** The header section defines the return type of the function. The return datatype can be any of the oracle datatype like varchar, number etc.

The execution and exception section both should return a value which is of the datatype defined in the header section.

**How to execute a Function?**

A function can be executed in the following ways.

- As a part of a SELECT statement : SELECT emp\_details\_func FROM dual;
- In a PL/SQL Statements like, : dbms\_output.put\_line(emp\_details\_func);

This line displays the value returned by the function .

**Example:**

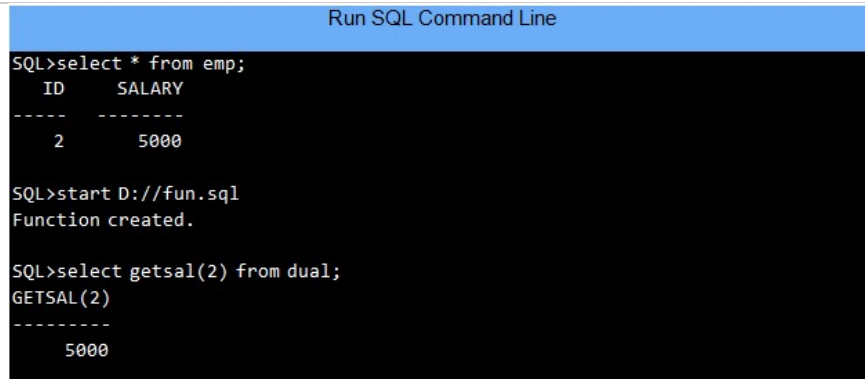
```

Create or replace function getsal (noINnumber) returnnumber
is
sal number(5);

```

```
begin
select salary into sal from emp where id=no;
return sal;
end;
/
```

**Output:**



```
Run SQL Command Line

SQL>select * from emp;
  ID      SALARY
-----
    2         5000

SQL>start D://fun.sql
Function created.

SQL>select getsal(2) from dual;
GETSAL(2)
-----
       5000
```

In the example we are retrieving the 'salary' of employee with id 2 to variable 'sal'. The return type of the function is number.

Destroying procedure and function:

*Syntax:*

**DROP PROCEDURE/FUNCTION PROCEDURE/FUNCTION\_NAME;**

Procedures VS Functions:

- A function **MUST** return a value
- A procedure cannot return a value
- Procedures and functions can both return data in OUT and IN OUT parameters
- The return statement in a function returns control to the calling program and returns the results of the function
- The return statement of a procedure returns control to the calling program and cannot return a value
- Functions can be called from SQL, procedure cannot
- Functions are considered expressions, procedure are not

#### **H. Resources/Equipment Required**

Computer or laptop with a database management system (DBMS) software installed, such as Oracle

#### **I. Safety and necessary Precautions followed**

Shutdown computer system properly once the Lab hours are finished

#### **J. Source code / Solutions to queries:**









**Practical related Quiz.**

1. What is the syntax for creating a PL/SQL procedure?
  - A. CREATE PROCEDURE procedure\_name IS BEGIN statement; END;
  - B. CREATE PROCEDURE procedure\_name AS BEGIN statement; END;
  - C. PROCEDURE procedure\_name IS BEGIN statement; END;
  - D. None of the above.
2. What is the syntax for creating a PL/SQL function?
  - A. CREATE FUNCTION function\_name IS BEGIN statement; END;
  - B. CREATE FUNCTION function\_name RETURN data\_type AS BEGIN statement; END;
  - C. FUNCTION function\_name IS BEGIN statement; END;
  - D. None of the above.
3. What is the difference between a PL/SQL procedure and a function?
  - A. A procedure returns a value, while a function does not.
  - B. A function returns a value, while a procedure does not.
  - C. A procedure can be called from within a query, while a function cannot.
  - D. None of the above.
4. What is the purpose of the RETURN statement in a PL/SQL function?
  - A. To terminate program execution.
  - B. To return a value to the calling program.
  - C. To raise an exception.
  - D. None of the above.
5. What is the purpose of the IN parameter in a PL/SQL procedure or function?
  - A. To pass a value into the program unit.
  - B. To return a value from the program unit.
  - C. To specify the data type of the program unit.
  - D. None of the above.
6. What is the syntax for creating a function in PL/SQL?
  - A. create function function\_name is begin ... end;
  - B. create function function\_name as begin ... end;
  - C. create function function\_name(parameters) is begin ... end;
  - D. None of the above.

**K. References / Suggestions**

<https://www.geeksforgeeks.org/ddl-commands-syntax/>

<https://www.w3schools.com/sql/>

<https://www.tutorialspoint.com/sql/index.htm>



**L. Assessment-Rubrics**

Excellent (21-25 Marks)	Good (15-20 Marks)	Satisfactory (7-14 Marks)	Fair (0-6 Marks)
<ul style="list-style-type: none"> <li>• Student executes the queries with correct output.</li> <li>• Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>• Student executes the queries having correct output with external guidance.</li> <li>• Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>• Student executes the queries with incorrect output.</li> <li>• Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>• Student is not able to execute the queries.</li> <li>• Student does not submit the lab report in specified time limit.</li> </ul>

**Sign with Date**

Date: .....

### Practical No.19:

Perform various operations on packages.

1. Create a package called "Employee\_Info" that contains two functions, "get\_salary" and "get\_job\_title". The "get\_salary" function takes an employee ID as input and returns the employee's salary. The "get\_job\_title" function takes an employee ID as input and returns the employee's job title
2. Create a procedure called "update\_salary" that takes an employee ID and a new salary as input and updates the employee's salary. The procedure should also log the update in a separate table called "salary\_history"
3. Create a package called "Employee\_Management" that contains a procedure called "promote\_employee". The procedure should take an employee ID and a new job title as input and update the employee's job title. The procedure should also log the promotion in the "salary\_history" table

#### A. Objective:

To perform various operations on packages in Oracle is to organize and manage code more efficiently, improve performance, and simplify application maintenance.

#### B. Expected Program Outcomes (POs): PO1,PO2,PO3,PO4,PO5,PO6,PO7

#### C. Expected Skills to be developed based on competency:

Compilation, debugging, executing SQL packages

#### D. Expected Course Outcomes(Cos)

Write PL/SQL block using concept of Cursor Management, Error Handling, Package and Triggers.

#### E. Practical Outcome(PRo)

Write code to create, compile, debug, execute packages using oracle software

#### F. Expected Affective domain Outcome (ADos)

- Follow safety practices.
- Practice good housekeeping.
- Demonstrate working as a leader/a team member.
- Maintain tools and equipment
- Follow ethical practices.

#### G. Prerequisite Theory:

#### PACKAGES

"A **package** is a container for other database objects."

A package can hold other database objects such as variables, constants, cursors, exceptions, procedures, functions and sub-programs.

A package has usually two components, a **specification** and a **body**.

A package's specification declares the types (variables of the record type), memory variables, constants, exceptions, cursors, and subprograms that are available for use.

A package's body fully defines cursors, functions, and procedures and thus implements the specification.

### PACKAGE SPECIFICATION:

The package specification contains:

- Name of the package
- Names of the data types of any arguments
- This declaration is local to the database and global to the package

This means that procedures, functions, variables, constants, cursors and exceptions and other objects, declared in a package are accessible from anywhere in the package.

Therefore, all the information a package needs, to execute a stored subprogram, is contained in the package specifications itself.

*Syntax:*

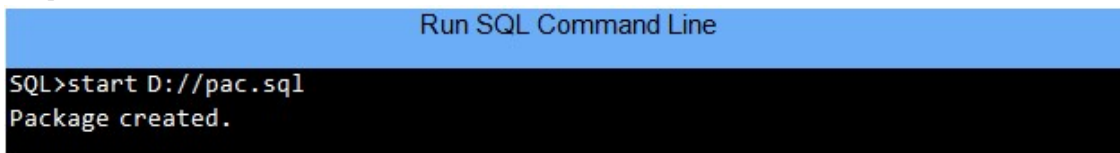
```
Create [or replace] package package_name
{is | as}
    Package_specification
End package_name;
```

*Example:*

Create package operation that contains procedure 'add' and function 'sub'.

```
Create or replace package operation
Is
    procedure addition(a in number,b in number);
    function sub(a in number,b in number)return number;
End operation;
/
```

Output:

A screenshot of a SQL Command Line window. The title bar is blue and says "Run SQL Command Line". The background is black with white text. The text shows "SQL>start D://pac.sql" followed by "Package created." on the next line.

```
Run SQL Command Line
SQL>start D://pac.sql
Package created.
```

### PACKAGE BODY:

The body of a package contains the definition of public objects that are declared in the specification.

The body can also contain other object declarations that are private to the package. The objects declared privately in the package body are not accessible to other objects outside the package.

Unlike package specification, the package body can contain subprogram bodies. After the package is written, debugged, compiled and stored in the database applications can reference the package's types, call its subprograms, use its cursors, or raise its exceptions.

*Syntax:*

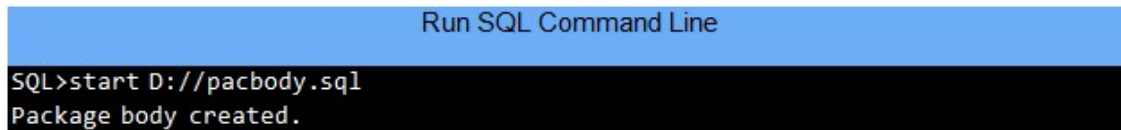
```
Create [or replace] package body package_name
{is | as}
```

```
Package_body  
End package_name;
```

*Example:*

```
Create or replace package body operation  
is  
  procedure addition(a in number,b in number)  
  is  
  begin  
    dbms_output.put_line('addition of two number :'||add(a,b));  
  end;  
  function sub(a in number,b in number) return number  
  is  
    ans number(3);  
  begin  
    ans:=a-b;  
    return ans;  
  end;  
end operation;  
/
```

*Output:*



The screenshot shows a terminal window titled "Run SQL Command Line". The command prompt is "SQL>start D://pacbody.sql". The output is "Package body created.".

**H. Resources/Equipment Required**

Computer or laptop with a database management system (DBMS) software installed, such as Oracle

**I. Safety and necessary Precautions followed**

Shutdown computer system properly once the Lab hours are finished

**J. Source code / Solutions to queries :**







### Practical related Quiz.

1. What is a package in PL/SQL?
  - A. A database object that groups related functions and procedures together.
  - B. A set of rules that govern the behavior of a database.
  - C. A block of code that performs a specific task.
  - D. None of the above.
2. What are the two parts of a package in PL/SQL?
  - A. Header and body.
  - B. Main and secondary.
  - C. Upper and lower.
  - D. None of the above.
3. What is the purpose of the header in a PL/SQL package?
  - a. To define the public interface of the package.
  - b. To define the private implementation of the package.
  - c. To define the package's database schema.
  - d. None of the above.
4. What is the purpose of the body in a PL/SQL package?
  - a. To define the public interface of the package.
  - b. To define the private implementation of the package.
  - c. To define the package's database schema.
  - d. None of the above.
5. What is the syntax for creating a package in PL/SQL?
  - a. create package package\_name as ... end;
  - b. create package package\_name as ... begin ... end;
  - c. create package body package\_name as ... end;
  - d. create package body package\_name as ... begin ... end;
6. What is the purpose of the initialization section in a PL/SQL package?
  - a. To declare global variables.
  - b. To initialize global variables.
  - c. To declare and initialize global variables.
  - d. None of the above.

### K. References / Suggestions

<https://www.geeksforgeeks.org/ddl-commands-syntax/>

<https://www.w3schools.com/sql/>

<https://www.tutorialspoint.com/sql/index.htm>



**L. Assessment-Rubrics**

Excellent (21-25 Marks)	Good (15-20 Marks)	Satisfactory (7-14 Marks)	Fair (0-6 Marks)
<ul style="list-style-type: none"> <li>• Student executes the queries with correct output.</li> <li>• Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>• Student executes the queries having correct output with external guidance.</li> <li>• Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>• Student executes the queries with incorrect output.</li> <li>• Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>• Student is not able to execute the queries.</li> <li>• Student does not submit the lab report in specified time limit.</li> </ul>

**Sign with Date**

Date: .....

## Practical No.20:

Implement various triggers.

1. Create a trigger that automatically updates the last\_update column of the employees table whenever a row is inserted or updated.
2. Create a trigger that prevents any changes to the salary column of the employees table for employees whose job title is President.
3. Create a trigger that automatically updates the salary\_grade column of the employees table based on the employee's salary.

### A. Objective:

To implement various triggers is to automate actions in response to specific events or changes in data. Triggers are used in software applications to monitor and respond to specific actions, such as user input or changes in data. By setting up triggers, you can automate certain tasks, reduce manual effort, and ensure that important actions are performed consistently and in a timely manner

### B. Expected Program Outcomes (POs): PO1,PO2,PO3,PO4,PO5,PO6,PO7

### C. Expected Skills to be developed based on competency:

Compilation, debugging, executing SQL queries

### D. Expected Course Outcomes(Cos)

Write PL/SQL block using concept of Cursor Management, Error Handling, Package and Triggers.

### E. Practical Outcome(Pro)

Write code to create, compile, debug, execute Triggers using oracle software

### F. Expected Affective domain Outcome (ADos)

- Follow safety practices.
- Practice good housekeeping.
- Demonstrate working as a leader/a team member.
- Maintain tools and equipment
- Follow ethical practices.

### G. Prerequisite Theory:

#### TRIGGERS

A trigger is a pl/sql block structure which is fired when a DML statements like Insert, Delete, Update is executed on a database table. A trigger is triggered automatically when an associated DML statement is executed.

#### Syntax:

```
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)
begin
--- sql statements
end;

```

Each statements are described below :

Statement	Description
CREATE [OR REPLACE ] TRIGGER trigger_name	This clause creates a trigger with the given name or overwrites an existing trigger with the same name.
{BEFORE   AFTER   INSTEAD OF }	This clause indicates at what time the trigger should get fired. i.e for example: before or after updating a table. INSTEAD OF is used to create a trigger on a view. before and after cannot be used to create a trigger on a view.
{INSERT [OR]   UPDATE [OR]   DELETE}	This clause determines the triggering event. More than one triggering events can be used together separated by OR keyword. The trigger gets fired at all the specified triggering event.
[OF col_name]	This clause is used with update triggers. This clause is used when you want to trigger an event only when a specific column is updated.
CREATE [OR REPLACE ] TRIGGER trigger_name	This clause creates a trigger with the given name or overwrites an existing trigger with the same name.
[ON table_name]	This clause identifies the name of the table or view to which the trigger is associated.
[REFERENCING OLD AS o NEW AS n]	This clause is used to reference the old and new values of the data being changed. By default, you reference the values as :old.column_name or :new.column_name. The reference names can also be changed from old (or new) to any other user-defined name. You cannot reference old values when inserting a record, or new values when deleting a record, because they do not exist.
[FOR EACH ROW]	This clause is used to determine whether a trigger must fire when each row gets affected ( i.e. a Row Level Trigger) or just

	once when the entire sql statement is executed(i.e.statement level Trigger).
WHEN (condition)	This clause is valid only for row level triggers. The trigger is fired only for rows that satisfy the condition specified.

**Example:**

Create or replace Trigger np before insert on account for each row

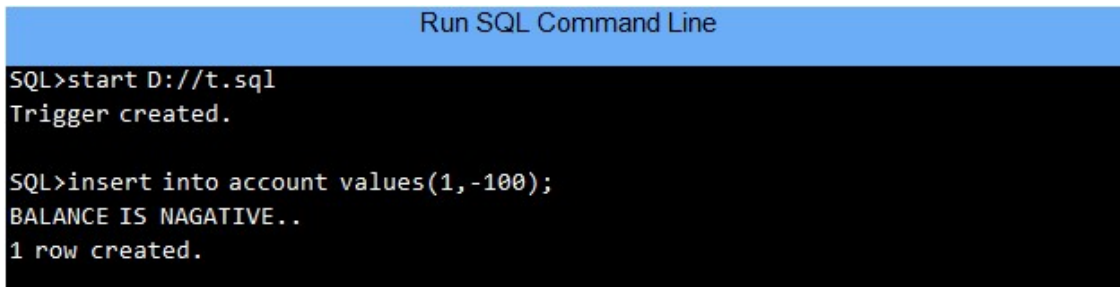
Begin

IF :NEW.bal < 0 THEN

DBMS\_OUTPUT.PUT\_LINE('BALANCE IS NAGATIVE..');

END IF;

End;

**Output:**


```

Run SQL Command Line

SQL>start D://t.sql
Trigger created.

SQL>insert into account values(1,-100);
BALANCE IS NAGATIVE..
1 row created.

```

**TYPES OF TRIGGERS**

A trigger's type is defined by the type of triggering transaction and by the level at which the trigger is executed. Oracle has the following types of triggers depending on the different applications.

- Row Level Triggers
- Statement Level Triggers
- Before Triggers
- After Triggers

**Row Level Triggers :**

Row level triggers execute once for each row in a transaction. The commands of row level triggers are executed on all rows that are affected by the command that enables the trigger.

For example, if an UPDATE statement updates multiple rows of a table, a row trigger is fired once for each row affected by the UPDATE statement. If the triggering statement affects no rows, the trigger is not executed at all. Row level triggers are created using the FOR EACH ROW clause in the CREATE TRIGGER command.

**Application:**

Consider a case where our requirement is to prevent updation of empno 100 record cannot be updated, then whenever UPDATE statement update records, there must be PL/SQL block that will be fired automatically by UPDATE statement to check that it must not be 100, so we have to use Row level Triggers for that type of applications.

**Statement Level Triggers:**

Statement level triggers are triggered only once for each transaction. For example when an UPDATE command update 15 rows, the commands contained in the trigger are executed only once, and not with every processed row. Statement level trigger are the default types of trigger created via the CREATE TRIGGER command.

**Application:**

Consider a case where our requirement is to prevent the DELETE operation during Sunday. For this whenever DELETE statement deletes records, there must be PL/SQL block that will be fired only once by DELETE statement to check that day must not be Sunday by referencing system date, so we have to use Statement level Trigger for which fires only once for above application.

**Before Trigger:**

Since triggers are executed by events, they may be set to occur immediately before or after those events.

When a trigger is defined, you can specify whether the trigger must occur before or after the triggering event i.e. INSERT, UPDATE, or DELETE commands. BEFORE trigger execute the trigger action before the triggering statement. These types of triggers are commonly used in the following situation:

1. BEFORE triggers are used when the trigger action should determine whether or not the triggering statement should be allowed to complete.
2. By using a BEFORE trigger, you can eliminate unnecessary processing of the triggering statement.
3. For example: To prevent deletion on Sunday, for this we have to use Statement level before trigger on DELETE statement.
4. BEFORE triggers are used to derive specific column values before completing a triggering INSERT or UPDATE statement.

**After Trigger:**

AFTER trigger executes the trigger action after the triggering statement is executed. AFTER triggers are used when you want the triggering statement to complete before executing the trigger action.

**For example:**

To perform cascade delete operation, it means that user delete the record fro one table, but the corresponding records in other tables are delete automatically by a trigger which fired after the execution of delete statement issued by the user. When combining the different types of triggering actions, there are mainly 4 possible Valid trigger types available to us.

The possible configurations are:

- BEFORE statement Trigger
- BEFORE row Trigger
- AFTER statement Trigger
- AFTER row Trigger

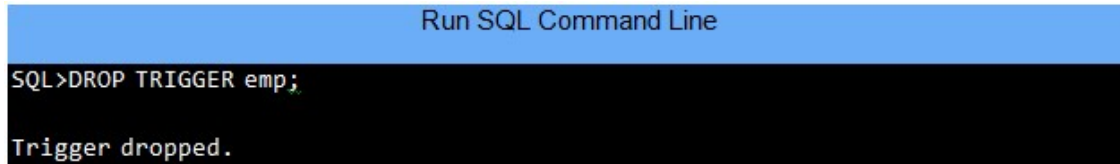
**Dropping Triggers:**

Triggers may be dropped via the drop trigger command. In order to drop a trigger, you must either own the trigger or have the DROP ANY TRIGGER system privilege.

***Syntax:***

```
DROP TRIGGER trigger_name;
```

***Example:***

A screenshot of a terminal window titled "Run SQL Command Line". The terminal has a black background with white text. The first line shows the command "SQL>DROP TRIGGER emp;". The second line shows the response "Trigger dropped.".

```
Run SQL Command Line
SQL>DROP TRIGGER emp;
Trigger dropped.
```

**H. Resources/Equipment Required**

Computer or laptop with a database management system (DBMS) software installed, such as Oracle

**I. Safety and necessary Precautions followed**

Shutdown computer system properly once the Lab hours are finished

**J. Source code /Solution to queries :**







**Practical related Quiz/Exercise.**

1. What is a trigger in SQL?
  - A. A database object that executes automatically in response to certain events.
  - B. A database object that groups related functions and procedures together.
  - C. A block of code that performs a specific task.
  - D. None of the above.
2. What is the syntax for creating a trigger in SQL?
  - A. CREATE TRIGGER trigger\_name ... BEGIN ... END;
  - B. CREATE TRIGGER trigger\_name ... FOR EACH ROW BEGIN ... END;
  - C. CREATE TRIGGER trigger\_name ... AFTER UPDATE ON table\_name BEGIN ... END;
  - D. None of the above.
3. What is the difference between a row-level trigger and a statement-level trigger?
  - A. A row-level trigger executes once for each row affected by a statement, while a statement-level trigger executes once for the entire statement.
  - B. A statement-level trigger executes once for each row affected by a statement, while a row-level trigger executes once for the entire statement.
  - C. A row-level trigger executes before the statement is executed, while a statement-level trigger executes after the statement is executed.
  - D. None of the above.
4. What is the purpose of the OLD and NEW qualifiers in a SQL trigger?
  - A. To refer to the old and new values of a row affected by an event.
  - B. To specify the order of execution of multiple triggers on the same event.
  - C. To define the conditions under which a trigger should fire.
  - D. None of the above.
5. What is the purpose of the WHEN clause in a SQL trigger?
  - A. To specify the conditions under which a trigger should fire.
  - B. To refer to the old and new values of a row affected by an event.
  - C. To define the order of execution of multiple triggers on the same event.
  - D. None of the above.

**K. References / Suggestions**

<https://www.geeksforgeeks.org/ddl-commands-syntax/>

<https://www.w3schools.com/sql/>

<https://www.tutorialspoint.com/sql/index.htm>

### L. Assessment-Rubrics

Excellent (21-25 Marks)	Good (15-20 Marks)	Satisfactory (7-14 Marks)	Fair (0-6 Marks)
<ul style="list-style-type: none"> <li>• Student executes the queries with correct output.</li> <li>• Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>• Student executes the queries having correct output with external guidance.</li> <li>• Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>• Student executes the queries with incorrect output.</li> <li>• Student submits the lab report in specified time limit.</li> </ul>	<ul style="list-style-type: none"> <li>• Student is not able to execute the queries.</li> <li>• Student does not submit the lab report in specified time limit.</li> </ul>

**Sign with Date**

**Date:** .....

**Practical No.21:**

Micro-Project(E-R Diagrams and Normalization) .

1.Design an E-R diagram for a university database management system that stores information about students, courses, and faculty members. The database should allow the university to manage course schedules, student enrollment, and faculty assignments. The database should also track student grades and maintain academic records. Convert database schema of university database management system into suitable Normal form( upto third normal form)

2. Design an E-R diagram for a library management system. The library has a collection of books, and the members can borrow books from the library. The library has multiple branches, and each branch has a different collection of books. It keeps track of books, authors, and publishers. Convert database schema of library management system into suitable Normal form( upto third normal form)

**A. Objective:**

- E-R diagrams provide a high-level view of the database's purpose and requirements, ER diagrams provide a detailed view of the database's structure and relationships. Both are essential tools in database design and development
- To reduce data redundancy, minimize the possibility of data inconsistencies, and improve the performance of the database. By following the principles of normalization, you can create a database that is more organized, efficient, and easier to maintain over time.

**B. Expected Program Outcomes (POs): PO1,PO2,PO3,PO4,PO5,PO6,PO7**

**C. Expected Skills to be developed based on competency:**

- E-R diagramming skills can improve data modeling, problem-solving, communication, analytical, and technical skills, all of which are essential in the field of database design and management
- Ability to analyze data and identify relationships between different data points.
- Understanding of normalization rules and techniques for organizing data in a database.
- Proficiency in designing and implementing a normalized database schema to ensure data consistency and eliminate redundancies.

**D. Expected Course Outcomes(Cos)**

Apply various Normalization techniques

**E. Practical Outcome(Pro)**

- E-R diagram used in data modeling, database design, communication, maintenance, and optimization.
- Normalization can improve the efficiency, consistency, flexibility, and maintenance of a database.

**F. Expected Affective domain Outcome (ADos)**

- Follow safety practices.

- Practice good housekeeping.
- Demonstrate working as a leader/a team member.
- Maintain tools and equipment
- Follow ethical practices.

#### G. Prerequisite Theory:

- The Entity Relational Model is a model for identifying entities to be represented in the database and representation of how those entities are related. The ER data model specifies enterprise schema that represents the overall logical structure of a database graphically.
- The Entity Relationship Diagram explains the relationship among the entities present in the database. ER models are used to model real-world objects like a person, a car, or a company and the relation between these real-world objects. In short, ER Diagram is the structural format of the database.
- ER Model is used to model the logical view of the system from a data perspective which consists of these symbols:

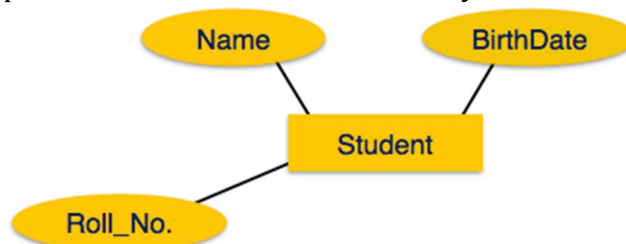
#### Entity

Entities are represented by means of rectangles. Rectangles are named with the entity set they represent.

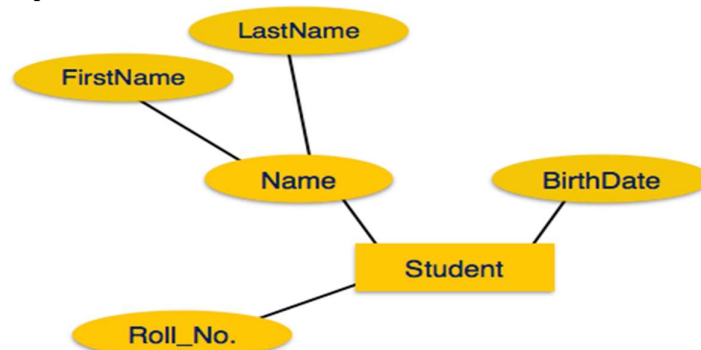


#### Attributes

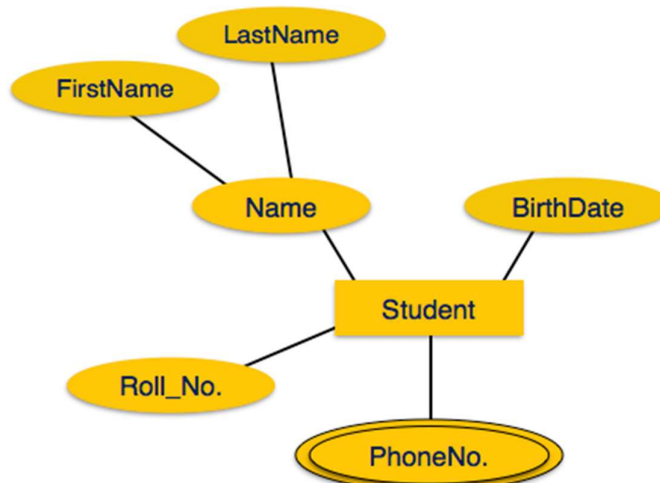
Attributes are the properties of entities. Attributes are represented by means of ellipses. Every ellipse represents one attribute and is directly connected to its entity (rectangle).



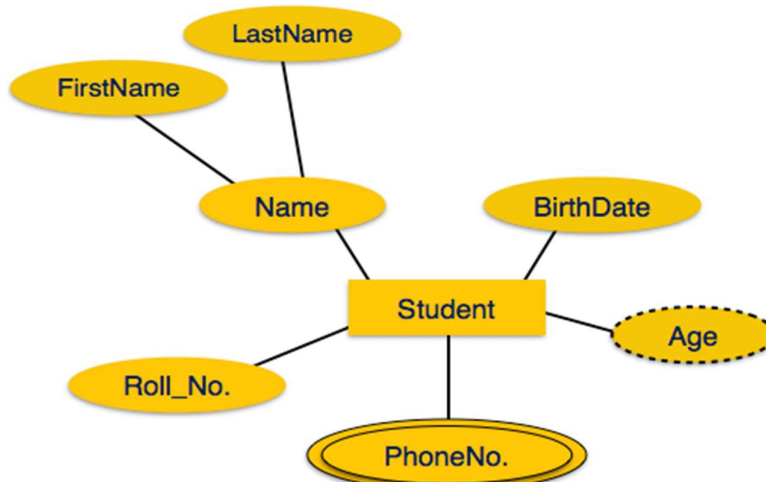
If the attributes are **composite**, they are further divided in a tree like structure. Every node is then connected to its attribute. That is, composite attributes are represented by ellipses that are connected with an ellipse.



**Multivalued** attributes are depicted by double ellipse.



**Derived** attributes are depicted by dashed ellipse.



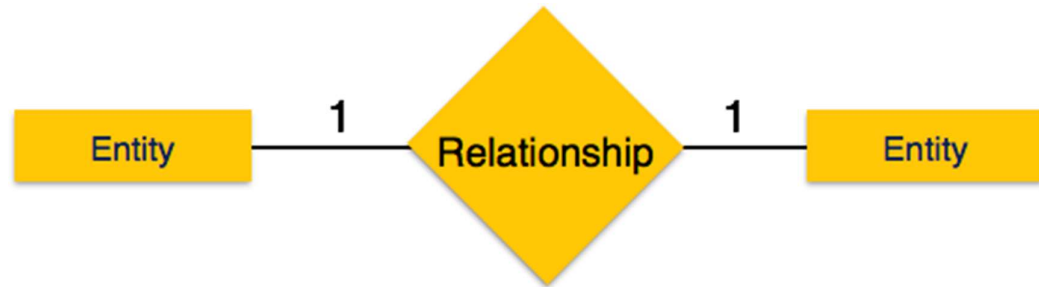
### Relationship

Relationships are represented by diamond-shaped box. Name of the relationship is written inside the diamond-box. All the entities (rectangles) participating in a relationship, are connected to it by a line.

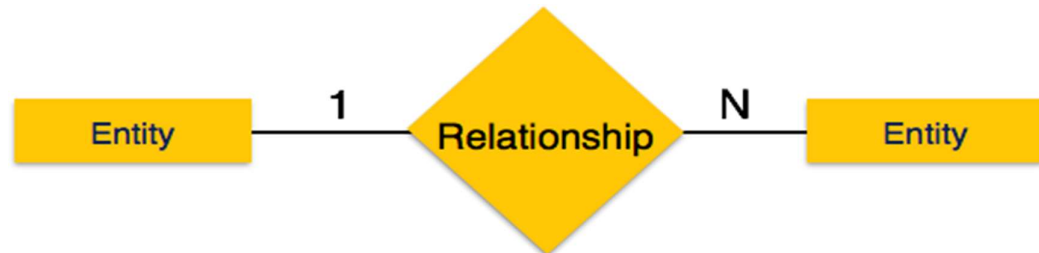
#### Binary Relationship and Cardinality

A relationship where two entities are participating is called a **binary relationship**. Cardinality is the number of instance of an entity from a relation that can be associated with the relation.

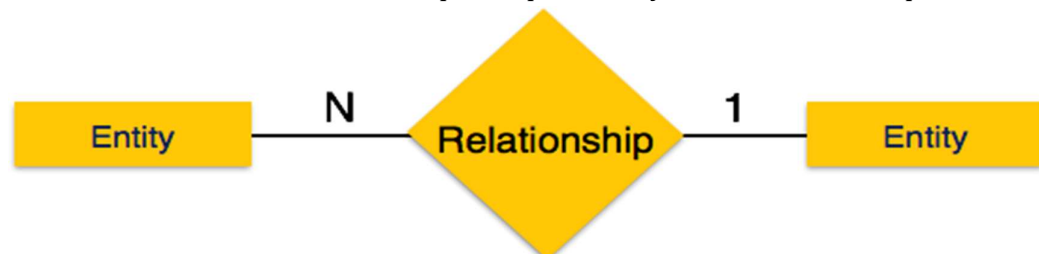
- **One-to-one** – When only one instance of an entity is associated with the relationship, it is marked as '1:1'. The following image reflects that only one instance of each entity should be associated with the relationship. It depicts one-to-one relationship.



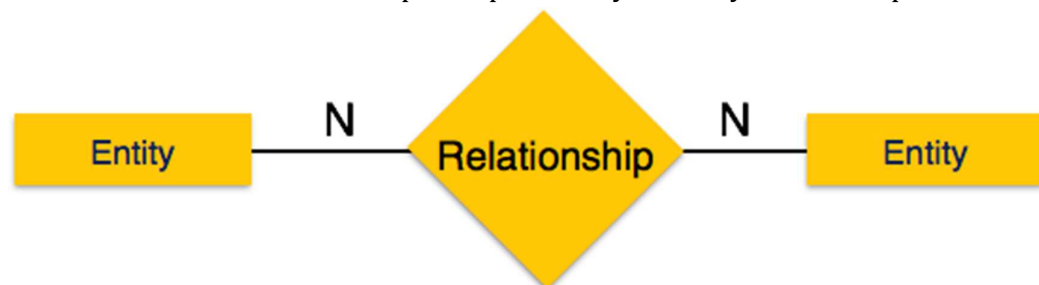
- **One-to-many** – When more than one instance of an entity is associated with a relationship, it is marked as '1:N'. The following image reflects that only one instance of entity on the left and more than one instance of an entity on the right can be associated with the relationship. It depicts one-to-many relationship.



- **Many-to-one** – When more than one instance of entity is associated with the relationship, it is marked as 'N:1'. The following image reflects that more than one instance of an entity on the left and only one instance of an entity on the right can be associated with the relationship. It depicts many-to-one relationship.



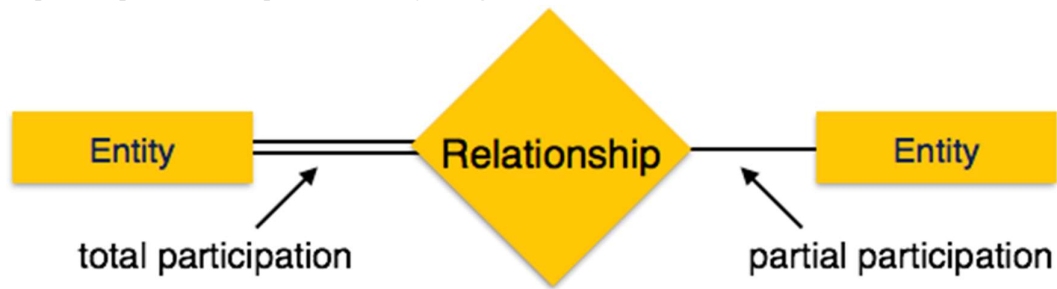
- **Many-to-many** – The following image reflects that more than one instance of an entity on the left and more than one instance of an entity on the right can be associated with the relationship. It depicts many-to-many relationship.



### Participation Constraints

- **Total Participation** – Each entity is involved in the relationship. Total participation is represented by double lines.

- **Partial participation** – Not all entities are involved in the relationship. Partial participation is represented by single lines.



### Normalization

- Normalization is the process of organizing the data in the database.
- Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate undesirable characteristics like Insertion, Update, and Deletion Anomalies.
- Normalization divides the larger table into smaller and links them using relationships.
- The normal form is used to reduce redundancy from the database table.

The main reason for normalizing the relations is removing these anomalies. Failure to eliminate anomalies leads to data redundancy and can cause data integrity and other problems as the database grows. Normalization consists of a series of guidelines that helps to guide you in creating a good database structure.

**Data modification anomalies can be categorized into three types:**

- **Insertion Anomaly:** Insertion Anomaly refers to when one cannot insert a new tuple into a relationship due to lack of data.
- **Deletion Anomaly:** The delete anomaly refers to the situation where the deletion of data results in the unintended loss of some other important data.
- **Updation Anomaly:** The update anomaly is when an update of a single data value requires multiple rows of data to be updated.

### Types of Normal Forms:

Normal Form	Description
1NF	A relation is in 1NF if it contains an atomic value.
2NF	A relation will be in 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the primary key.
3NF	A relation will be in 3NF if it is in 2NF and no transition dependency exists.

### H. Resources/Equipment Required

Computer or laptop with a database management system (DBMS) software installed, such as Oracle

### I. Safety and necessary Precautions followed

Shutdown computer system properly once the Lab hours are finished

### J. Solution







**K. Practical related Quiz.**

1. What is an E-R diagram?
  - A. A diagram that shows the relationships between tables in a database.
  - B. A diagram that shows the relationships between entities in a system.
  - C. A diagram that shows the relationships between data types in a programming language.
  - D. None of the above.
2. What is an entity in an E-R diagram?
  - A. A table in a database.
  - B. A column in a table.
  - C. A person, place, thing, or event in a system.
  - D. None of the above.
3. What is a relationship in an E-R diagram?
  - A. A connection between two tables in a database.
  - B. A connection between two columns in a table.
  - C. A connection between two entities in a system.
  - D. None of the above.
4. What is normalization in database design?
  - A. The process of organizing data into tables to minimize redundancy and ensure data integrity.
  - B. The process of organizing tables into databases to minimize redundancy and ensure data integrity.
  - C. The process of optimizing database performance by creating indexes and views.
  - D. None of the above.
5. What is the purpose of normal forms in database design?
  - A. To ensure that data is stored in the most efficient way possible.
  - B. To ensure that data is stored in a way that is easy to query and maintain.
  - C. To ensure that data is stored in a way that is free from errors and inconsistencies.
  - D. All of the above.
6. What is first normal form (1NF)?
  - A. A table is in 1NF if it has a primary key and all its columns are atomic.
  - B. A table is in 1NF if it has a primary key and all its columns are of the same data type.
  - C. A table is in 1NF if it has a primary key and all its columns are unique.
  - D. None of the above.

7. What is second normal form (2NF)?

- A. A table is in 2NF if it is in 1NF and all its non-key columns are fully dependent on the primary key.
- B. A table is in 2NF if it is in 1NF and all its columns are of the same data type.
- C. A table is in 2NF if it is in 1NF and all its columns are unique.
- D. None of the above.

8. What is third normal form (3NF)?

- A. A table is in 3NF if it is in 2NF and all its non-key columns are non-transitively dependent on the primary key.
- B. A table is in 3NF if it is in 2NF and all its columns are of the same data type.
- C. A table is in 3NF if it is in 2NF and all its columns are unique.
- D. None of the above.

**L. References / Suggestions**

<https://www.geeksforgeeks.org/ddl-commands-syntax/>

<https://www.w3schools.com/sql/>

<https://www.tutorialspoint.com/sql/index.htm>

**M. Assessment-Rubrics**

Excellent (21-25 Marks)	Good (15-20 Marks)	Satisfactory (7-14 Marks)	Fair (0-6 Marks)
<ul style="list-style-type: none"><li>• Student executes the queries with correct output.</li><li>• Student submits the lab report in specified time limit.</li></ul>	<ul style="list-style-type: none"><li>• Student executes the queries having correct output with external guidance.</li><li>• Student submits the lab report in specified time limit.</li></ul>	<ul style="list-style-type: none"><li>• Student executes the queries with incorrect output.</li><li>• Student submits the lab report in specified time limit.</li></ul>	<ul style="list-style-type: none"><li>• Student is not able to execute the queries.</li><li>• Student does not submit the lab report in specified time limit.</li></ul>

**Sign with Date**

# **Relational Database Management Systems**

## **4330702**

Lab manual is prepared by

Miss Dhara Hamirbhai Wagh  
Lecturer in Computer Engineering  
Government Polytechnic Gandhinagar

Mr Sachin D. Shah,  
Lecturer in Computer Engineering  
R.C. Technical Institute, Ahmedabad

**Branch Coordinator**  
Mr. B. H. Kantevala  
Head of the Department  
Computer Engineering  
Government Polytechnic, Ahmedabad

**Committee Chairman**  
Shri. R.D.Raghani  
(HOD-EC)  
Principal(I/C)  
Government Polytechnic, Gandhinagar