BONAFIDE CERTIFICATE

Register No.: RA1911003010414 Date:

Certified to be the Bonafide record of work done in 18CSC303J – DATABASE MANAGEMENT SYSTEMS by

Saket Kumar Baranwal RA1911003010414

of Third Year B.Tech. Degree course in Computer Science & Engineering in SRMIST, Kattankulathur, Chennai, during the academic year 2021-22.

Faculty-in-Charge Head of the Department

Submitted for the University Examination held in
…….....
..... at SRMIST, Kattankulathur, Chennai.

Internal Examiner - I Internal Examiner -II

Index

Exp.No	Date	Title	Page No	Marks
1	21-01-22	SQL DDL Commands		
2	01-02-22	SQL DML Commands		
3	08-02-22	SQL DCL and TCL commands		
4	15-02-22	SQL In Built Functions		
5	08-03-22	Design and Draw ER Diagrams in DBMS		
6	15-03-22	SQL Sub Queries and Nested Queries		
7	29-03-22	SQL Join Operations		
8	29-03-22	SQL Set Operations and View		
9	05-04-22	PL / SQL Conditional and Iterative Statement		
10	05-04-22	Pl / SQL Procedures		
11	11-04-22	Pl / SQL Functions		
12	19-04-22	PL / SQL Cursors		
13	26-04-22	PL / SQL Exception Handling		
14	26-04-22	PL / SQL Triggers		
15	03-05-22	SQL FRAMING AND EXECUTING CURSORS & EXCEPTION HANDLING		

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

FACULTY OF ENGINEERING & TECHNOLOGY

(Formerly SRM University, Under section 3 of UGC Act, 1956)

S.R.M. NAGAR, KATTANKULATHUR –603 203, KANCHEEPURAM DISTRICT

SCHOOL OF COMPUTING AND TECHNOLOGY

Course Code: 18CSC303J
Course Name: Database Management Systems

LAB REPORT

NAME: SAKET KUMAR BARANWAL
REG. NO.: RA1911003010414
SECTION: G1
CSE – CORE.

Exp. No	Conducted on	Submitted On	Date of Late Submission (if Any)	Max Marks Allotted	Marks Obtained	Faculty Signature
1				10		
Exp Title	SQL Data Definition Language Commands (DDL)					

Aim :- To write the SQL queries using DDL Commands with and without constraints.

Syntax (for all DDL Commands)

1- Create Table

2- Alter Table

3- Drop Table

Queries (for all DDL Commands)

1. CREATE THE TABLE (with no constraint
2. ALTER THE TABLE
   a) Add
   b) Modify
   c) Drop
   d) Rename

Result

```
SQL>Spool exp2.2
SQL>Spool off
SQL> edit exp2.2
SQL> @exp2.2
CREATE TABLE employ
(
        empno NUMBER,
        empname VARCHAR2(25),
        dob DATE,
        salary NUMBER,
        designation VARCHAR2(20)
);

Table created.

SQL> DESC employ;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMPNO                                              NUMBER
 EMPNAME                                            VARCHAR2(25)
 DOB                                                DATE
 SALARY                                             NUMBER
 DESIGNATION                                        VARCHAR2(20)

SQL> ALTER TABLE emp ADD department VARCHAR2(70);

Table altered.
```

```
SQL> DESC employ;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMPNO                                              NUMBER
 EMPNAME                                            VARCHAR2(25)
 DOB                                                DATE
 SALARY                                             NUMBER
 DESIGNATION                                        VARCHAR2(20)

SQL> ALTER TABLE emp MODIFY (department VARCHAR2(100));

Table altered.

SQL> DESC employ;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMPNO                                              NUMBER
 EMPNAME                                            VARCHAR2(25)
 DOB                                                DATE
 SALARY                                             NUMBER
 DESIGNATION                                        VARCHAR2(20)

SQL> ALTER TABLE emp DROP(department);

Table altered.

SQL> DESC employ;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMPNO                                              NUMBER
 EMPNAME                                            VARCHAR2(25)
 DOB                                                DATE
 SALARY                                             NUMBER
 DESIGNATION                                        VARCHAR2(20)

SQL> ALTER TABLE emp RENAME TO emp1 ;

Table altered.

SQL> DESC emp1oy;
ERROR:
ORA-04043: object emp1oy does not exist


SQL> DESC employ;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMPNO                                              NUMBER
 EMPNAME                                            VARCHAR2(25)
 DOB                                                DATE
```

```
SALARY                                    NUMBER
DESIGNATION                                   VARCHAR2(20)

SQL> DROP TABLE emp1;

Table dropped.

SQL> DESC emp1;
```

Exp. No	Conducted on	Submitted On	Date of Late Submission (if Any)	Max Marks Allotted	Marks Obtained	Faculty Signature
1	02/02/2022	09/02/2022		10		
Exp Title	SQL Data Manipulation Language (DML) Commands Saket Kumar Baranwal RA1911003010414					

Aim :- To write SQL Data Manipulation Language (DML) Commands.

Syntax (for all DML Commands)

SELECT - Used to query or fetch selected fields or columns from a database

Syntax,
SELECT column_name1, column_name2, …
FROM table_name
WHERE condition_ expression;
Example:
Select customer_id, sale_date, order_id, store_state from customers;
Select * from customers;

- **INSERT -** Used to insert new data records or rows in the database table

Syntax,
INSERT INTO table_name (column_name_1, column_name_2, column_name_3, ...)
VALUES (value1, value2, value3, ...)
Example:
INSERT INTO sales(
customer_id, sale_date, sale_amount, salesperson, store_state, order_id)
VALUES (1005,TO_DATE('2019-12-12','YYYY-MM-DD),4200,'R K Rakesh','MH','1007');
(or)
INSERT INTO customers
VALUES ('1006','2020-03-04',3200,'DL', '1008');

- **UPDATE -** Used to set the value of a field or column for a particular record to a new value

Syntax,
UPDATE table_name
SET column_name_1 = value1, column_name_2 = value2, ...
WHERE condition;
Example,
UPDATE customers
SET store_state = 'DL'
WHERE store_state = 'NY';

- **DELETE -** Used to remove one or more rows from the database table

Syntax,
DELETE FROM table_name WHERE condition;
Example,
DELETE FROM customers

WHERE store_state = 'MH'
AND customer_id = '1001';

Queries (for all DDL Commands)
1-Select
2- Insert
3- Update
4-Delete

Result

```
SQL>  create table students(st_id int,st_name varchar(10),st_ph_no varchar(10));

Table created.

SQL> desc students;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ST_ID                                              NUMBER(38)
 ST_NAME                                            VARCHAR2(10)
 ST_PH_NO                                           VARCHAR2(10)

SQL> insert into students values(101,'Saket','8340463394');

1 row created.

SQL> insert into students values(102,'Tanmay','746463394');

1 row created.

SQL> insert into students values(103,'Alinjar','98463394');

1 row created.

SQL> insert into students values(104,'Rucha','8974685210');

1 row created.

SQL> select*from students;

    ST_ID ST_NAME    ST_PH_NO
---------- ---------- ----------
       101 Saket      8340463394
       102 Tanmay     746463394
       103 Alinjar    98463394
       104 Rucha      8974685210

SQL> select st_name from students;

ST_NAME
----------
```

```
Saket
Tanmay
Alinjar
Rucha

SQL> update students set st_name='Sidharth'w where st_id=102;
update students set st_name='Sidharth'w where st_id=102
                                        *
ERROR at line 1:
ORA-00933: SQL command not properly ended


SQL> update students set st_name='Sidharth' where st_id=102;

1 row updated.

SQL> select * from students
  2
SQL> select*from students;

    ST_ID ST_NAME    ST_PH_NO
---------- ---------- ----------
       101 Saket      8340463394
       102 Sidharth   746463394
       103 Alinjar    98463394
       104 Rucha      8974685210

SQL> delete from students where st_id=102;

1 row deleted.

SQL> select*from students;

    ST_ID ST_NAME    ST_PH_NO
---------- ---------- ----------
       101 Saket      8340463394
       103 Alinjar    98463394
       104 Rucha      8974685210
```

Exp. No	Conducted on	Submitted On	Date of Late Submission (if Any)	Max Marks Allotted	Marks Obtained	Faculty Signature
1	16/02/2022	16/02/2022		10		
Exp Title	To write SQL queries to execute different DCL and TCL commands					

Aim:- To write SQL queries to execute different DCL and TCL commands

Syntax (for all DDL Commands)

GRANT: This command gives users access privileges to the database.

Syntax,

GRANT privileges_names ON object TO user;

Example:

Create user first identified by passwd;

Grant select on customers to first;

REVOKE: This command withdraws the user's access privileges given by using the GRANT command.

Syntax,

REVOKE privileges ON object FROM user;

Example:

Revoke select on customers from first;

Transaction Control Language (TCL) Commands:

COMMIT: Commits a Transaction.

Syntax:

COMMIT;

Example:

INSERT INTO customers

VALUES ('1006','2020-03-04',3200,'DL', '1008');

Commit;

Select * from customers;

ROLLBACK: Rollbacks a transaction in case of any error occurs.

Syntax:

Rollback;

Example:

DELETE FROM customers

WHERE store_state = 'MH'

AND customer_id = '1002';

Select * from customers;

Rollback;

Select * from customers;

SAVEPOINT:Sets a savepoint within a transaction.

Syntax:

SAVEPOINT SAVEPOINT_NAME;

This command is used only in the creation of SAVEPOINT among all the transactions.

Queries (for all DDL Commands)

```
SQL> create table stds(st_no int,st_name varchar(10),st_ph_no varchar(10));

Table created.

SQL> insert into stds values(110, 'Rachit',88548789);

1 row created.

SQL> desc stds
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ST_NO                                              NUMBER(38)
 ST_NAME                                            VARCHAR2(10)
 ST_PH_NO                                           VARCHAR2(10)

SQL> commit;

Commit complete.

SQL> insert into stds values(111, 'Vikramt',88548789);

1 row created.

SQL> insert into stds values(112, 'Paul',88548789);

1 row created.

SQL> insert into stds values(113, 'Rashmi',88548789);

1 row created.

SQL> commit;

Commit complete.
```

```
SQL> select*from stds;

    ST_NO ST_NAME    ST_PH_NO
---------- ---------- ----------
       110 Rachit     88548789
       111 Vikramt    88548789
       112 Paul       88548789
       113 Rashmi     88548789

SQL> delete from stds where st_no=110;

1 row deleted.

SQL> rollback
  2
SQL> Rollback;

Rollback complete.

SQL> select*from stds
  2
SQL> select*from stds;

    ST_NO ST_NAME    ST_PH_NO
---------- ---------- ----------
       110 Rachit     88548789
       111 Vikramt    88548789
       112 Paul       88548789
       113 Rashmi     88548789

SQL> savepoint s1;

Savepoint created.

SQL> delete from stds where st_no=111;

1 row deleted.

SQL> select*from stds;

    ST_NO ST_NAME    ST_PH_NO
---------- ---------- ----------
       110 Rachit     88548789
       112 Paul       88548789
       113 Rashmi     88548789

SQL> rollback to s1;

Rollback complete.
```

SQL> select*from stds;

```
    ST_NO ST_NAME    ST_PH_NO
---------- ---------- ----------
       110 Rachit     88548789
       111 Vikramt    88548789
       112 Paul       88548789
       113 Rashmi     88548789
```

Result:-

Thus the DCL and TCL commands are used to modify or manipulate data records present in the customer database tables.

```
D:\ORACLE CLIENT 11.2\instantclient_11_2\sqlplus.exe

SQL> create table stds(st_no int,st_name varchar(10),st_ph_no varchar(10));

Table created.

SQL> insert into stds values(110, 'Rachit',88548789);

1 row created.

SQL> desc stds
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ST_NO                                              NUMBER(38)
 ST_NAME                                            VARCHAR2(10)
 ST_PH_NO                                           VARCHAR2(10)

SQL> commit;

Commit complete.

SQL> insert into stds values(111, 'Vikramt',88548789);

1 row created.

SQL> insert into stds values(112, 'Paul',88548789);

1 row created.

SQL> insert into stds values(113, 'Rashmi',88548789);

1 row created.

SQL> commit;

Commit complete.

SQL> select*from stds;

     ST_NO ST_NAME    ST_PH_NO
---------- ---------- ----------
       110 Rachit     88548789
       111 Vikramt    88548789
       112 Paul       88548789
       113 Rashmi     88548789
```

```
SQL> delete from stds where st_no=110;

1 row deleted.

SQL> rollback
  2
SQL> Rollback;

Rollback complete.

SQL> select*from stds
  2
SQL> select*from stds;

     ST_NO ST_NAME    ST_PH_NO
---------- ---------- ----------
       110 Rachit     88548789
       111 Vikramt    88548789
       112 Paul       88548789
       113 Rashmi     88548789

SQL> savepoint s1;

Savepoint created.

SQL> delete from stds where st_no=111;

1 row deleted.

SQL> select*from stds;

     ST_NO ST_NAME    ST_PH_NO
---------- ---------- ----------
       110 Rachit     88548789
       112 Paul       88548789
       113 Rashmi     88548789

SQL> rollback to s1;

Rollback complete.
```

```
SQL> select*from stds;

     ST_NO ST_NAME    ST_PH_NO
---------- ---------- ----------
       110 Rachit     88548789
       111 Vikramt    88548789
       112 Paul       88548789
       113 Rashmi     88548789
```

Exp. No	Conducted on	Submitted On	Date of Late Submission (if Any)	Max Marks Allotted	Marks Obtained	Faculty Signature
4				10		
Exp Title	In Built Functions in SQL					

Aim –To check the inbuild functions in SQL.

Syntax (for all DDL Commands) –

Queries (for all DDL Commands) –

```
CREATE TABLE person(ID NUMBER, Name VARCHAR(45))
  2  ;

Table created.

SQL> desc person
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ID                                                 NUMBER
 NAME                                               VARCHAR2(45)

SQL> insert into person values(1,'JACK MA');

1 row created.

SQL> insert into person values(2,'ELON MUSK');

1 row created.

SQL> insert into person values(3,'MUKESH AMBANI');

1 row created.

SQL> insert into person values(4,'JEFF BEJ');

1 row created.

SQL> insert into person values(5,'RAKESH JHUN');

1 row created.

SQL> select * from person
  2  ;

       ID NAME
```

```
---------- ---------------------------------------------
         1 JACK MA
         2 ELON MUSK
         3 MUKESH AMBANI
         4 JEFF BEJ
         5 RAKESH JHUN

SQL> SELECT CONCAT('JACK','MA')NAME FROM person;

NAME
------
JACKMA
JACKMA
JACKMA
JACKMA
JACKMA

SQL> SELECT CONCAT('JACK','MA')NAME FROM person where id=1;

NAME
------
JACKMA

SQL> insert into person values(6,'BILL GATES');

1 row created.

SQL> select INITCAP('BILL GATES')name from person where id=6;

NAME
----------
Bill Gates

SQL> select INSTR('BILL GATES','T')name from person where id=6;

     NAME
----------
         8

SQL> select INSTR('BILL GATES','A')position from person where id=6;

  POSITION
----------
         7

SQL> select LENGTH('BILL GATES')length12 from person where id=6;

  LENGTH12
----------
        10
```

```
SQL> select LOWER('BILL GATES')lower12 from person where id =6;

LOWER12
----------
bill gates

SQL> select RPAD('BILL GATES',15,'dfg')output from person where id=6;

OUTPUT
---------------
BILL GATESdfgdf

SQL> select LPAD('BILL GATES',15,'ghj')outpust from person where id=6;

OUTPUST
---------------
ghjghBILL GATES

SQL> insert into person values(7,'  JACKI CHAN');

1 row created.

SQL> select trim('  JACKI CHAN')output from person where id =7;

OUTPUT
----------
JACKI CHAN

SQL> delete from person where id =7;

1 row deleted.

SQL> insert into person values(7,'  JACKI CHAN');

1 row created.

SQL> select * from person;

        ID NAME
---------- --------------------------------------------
         1 JACK MA
         2 ELON MUSK
         3 MUKESH AMBANI
         4 JEFF BEJ
         5 RAKESH JHUN
         6 BILL GATES
         7   JACKI CHAN

7 rows selected.
```

```
SQL> select id, name from person order by id desc;

        ID NAME
---------- ----------------------------------------------
         7   JACKI CHAN
         6 BILL GATES
         5 RAKESH JHUN
         4 JEFF BEJ
         3 MUKESH AMBANI
         2 ELON MUSK
         1 JACK MA

7 rows selected.

SQL> select * from student;

 STUDENTID SNAME                          DEPAR       SEM
---------- ------------------------------ ----- ----------
EMAIL_ID                       COLLEGE
------------------------------ ------------------------------
       965 PARTH                            CSE           8
parth@xyz.com                  SRM

         1 SIDDHARTH                        CSE           6
siddharth@xyz.com              SRM

       659 MAYANK                           CSE           6
mayank@xyz.com                 SRM


 STUDENTID SNAME                          DEPAR       SEM
---------- ------------------------------ ----- ----------
EMAIL_ID                       COLLEGE
------------------------------ ------------------------------
      4060 DAS                              CSE           6
das@xyz.com                    SRM

       569 AMIT                             EEE           6
amit@xyz.com                   SRM

      7850 PADI                             IT            2
padi@xyz.com                   SRM


 STUDENTID SNAME                          DEPAR       SEM
---------- ------------------------------ ----- ----------
EMAIL_ID                       COLLEGE
------------------------------ ------------------------------
      5846 JAY                              CSE           6
```

```
jay@xyz.com                 SRM

    89650 ROHIT                     MEC          4
roh@xyz.com                 SRM

      254 RONIT                     EEE          6
ron@xyz.com                 SRM


 STUDENTID SNAME                     DEPAR       SEM
---------- ------------------------------ ----- ----------
EMAIL_ID                    COLLEGE
------------------------------ ------------------------------
      908 PALLAV                     BIOT         6
pal@xyz.com                 SRM


10 rows selected.

SQL> select STUDENTID, SNAME from studen order by SEM desc;
select STUDENTID, SNAME from studen order by SEM desc
                         *
ERROR at line 1:
ORA-00942: table or view does not exist


SQL> select STUDENTID, SNAME from student order by SEM desc;

 STUDENTID SNAME
---------- ------------------------------
      965 PARTH
      659 MAYANK
     4060 DAS
      908 PALLAV
        1 SIDDHARTH
      569 AMIT
      254 RONIT
     5846 JAY
    89650 ROHIT
     7850 PADI

10 rows selected.
```

Result - Thus the inbuild functions were used to modify the tables

Exp. No	Conducted on	Submitted On	Date of Late Submission (if Any)	Max Marks Allotted	Marks Obtained	Faculty Signature
1	07/03/2022	09/03/2022		10		
Exp Title	ER Diagrams Saket Kumar Baranwal (RA1911003010414)					

Aim To construct the ER Diagram for online food ordering system
Tool used: - Draw.io
Procedure:-

A workflow created to facilitate the creation of the diagram
1. Mention all the entities involved - **Restaurants**, **Items** (menu Items for all the restaurants), **Customers**, **Reviews** (by customers for different restaurants), **Transaction.** Represent them as rectangles in the diagram.

2. For each entity, all its attributes are drawn inside an ellipse and a primary key is represented as underlined text inside an ellipse.

3. All the Relationships drawn are as follows:
● Each restaurant has many menu items and each menu item might belong to many restaurants and hence many to many cardinality.

● Each restaurant has many reviews and each review belongs to one restaurant only hence it's one to many relationship.

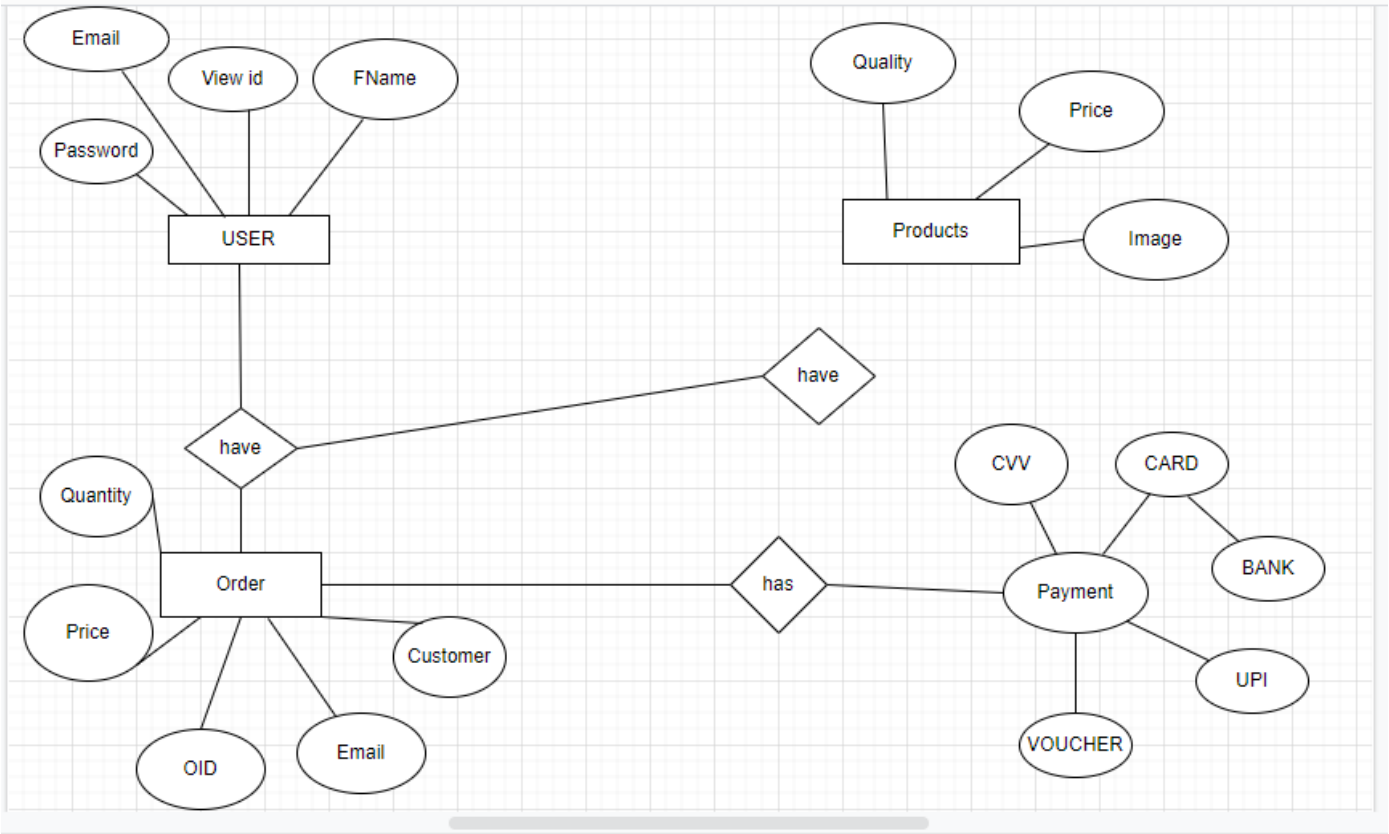
● One menu item can be selected by many customers and one customer can

select more than one menu items hence it's many to many cardinality.
● Each customer has a single cart and each cart belongs to a single customer hence it's one to one relationship.

● From one cart, multiple transactions are possible but a given transaction belongs to an individual cart hence it's one to many transactions.

4. Cart is a weak entity as it's unique for each individual so we don't need to assign a primary key to it
.

Output:-


Email
View id
FName
Password
USER
Quality
Price
Products
Image
have
have
Quantity
Order
Price
Customer
OID
Email
has
CVV
CARD
BANK
Payment
UPI
VOUCHER

Exp. No	Conducted on	Submitted On	Date of Late Submission (if Any)	Max Marks Allotted	Marks Obtained	Faculty Signature
1				10		
Exp Title	NESTED QUERIES					

Aim To study and implement the Nested Queries.

Syntax (for all DDL Commands)

Finding **C_ID** for **C_NAME** ='DSA' or 'DBMS'
Select **C_ID** from **COURSE** where **C_NAME** = 'DSA' or **C_NAME**

Using **C_ID** of step 1 for finding **S_ID**
Select **S_ID** from **STUDENT_COURSE** where **C_ID** IN
(SELECT **C_ID** from **COURSE** where **C_NAME** = 'DSA'
or **C_NAME**='DBMS');

Queries (for all DDL Commands)

```
CREATE TABLE student_info
(   S_ID VARCHAR(30) NOT NULL,
    S_NAME   VARCHAR(30) NOT NULL,
    S_ADDRESS  VARCHAR(30) NOT NULL,
    S_PHONE bigint,
    S_AGE INT
);
INSERT INTO student_info(S_ID,S_NAME,S_ADDRESS,S_PHONE,S_AGE)
values("S1","Ram","Delhi",8789546524,28),
      ("S2","Sam","Delhi",5789546524,30),
      ("S3","Preeti","Delhi",2589546524,41),
      ("S4","Khusi","Delhi",5789546524,24),
      ("S5","Sejal","Delhi",9589546524,22);
select*from student_info;
CREATE TABLE course
(   C_ID VARCHAR(30) NOT NULL,
    C_NAME   VARCHAR(30) NOT NULL);
    INSERT INTO course(C_ID,C_NAME)
    values("C1","DSA"),
          ("C3","PROGRAMMING"),
          ("C2","DBMS");
select*from course;
CREATE TABLE STUDENT_COURSE
( S_ID VARCHAR(30) NOT NULL,
  C_ID VARCHAR(30) NOT NULL
);
INSERT INTO STUDENT_COURSE(S_ID,C_ID)
    values("S1","C1"),
          ("S2","C2");

Select S_NAME from student_info where S_ID IN

(Select S_ID from STUDENT_COURSE where C_ID IN
```

(SELECT C_ID from COURSE where C_NAME="DSA" or C_NAME="DBMS"));
Select S_NAME from student_info where S_ID IN

(Select S_ID from STUDENT_COURSE where C_ID IN

(SELECT C_ID from COURSE where C_NAME="DBMS" or C_NAME="PROGRAMMING"));

Result

```
S1|Ram|Delhi|8789546524|28
S2|Sam|Delhi|5789546524|30
S3|Preeti|Delhi|2589546524|41
S4|Khusi|Delhi|5789546524|24
S5|Sejal|Delhi|9589546524|22
C1|DSA
C3|PROGRAMMING
C2|DBMS
Ram
Sam
Sam
```

```sql
CREATE TABLE student_info
(    S_ID VARCHAR(30) NOT NULL,
     S_NAME   VARCHAR(30) NOT NULL,
     S_ADDRESS  VARCHAR(30) NOT NULL,
     S_PHONE bigint,
     S_AGE INT
);
INSERT INTO student_info(S_ID,S_NAME,S_ADDRESS,S_PHONE,S_AGE)
values("S1","Ram","Delhi",8789546524,28),
       ("S2","Sam","Delhi",5789546524,30),
       ("S3","Preeti","Delhi",2589546524,41),
       ("S4","Khusi","Delhi",5789546524,24),
       ("S5","Sejal","Delhi",9589546524,22);
select*from student_info;
CREATE TABLE course
(    C_ID VARCHAR(30) NOT NULL,
     C_NAME   VARCHAR(30) NOT NULL);
     INSERT INTO course(C_ID,C_NAME)
     values("C1","DSA"),
            ("C3","PROGRAMMING"),
            ("C2","DBMS");
select*from course;
CREATE TABLE STUDENT_COURSE
( S_ID VARCHAR(30) NOT NULL,
  C_ID VARCHAR(30) NOT NULL
```

```sql
);
INSERT INTO STUDENT_COURSE(S_ID,C_ID)
    values("S1","C1"),
          ("S2","C2");

Select S_NAME from student_info where S_ID IN

(Select S_ID from STUDENT_COURSE where C_ID IN

(SELECT C_ID from COURSE where C_NAME="DSA" or C_NAME="DBMS"));
Select S_NAME from student_info where S_ID IN

(Select S_ID from STUDENT_COURSE where C_ID IN

(SELECT C_ID from COURSE where C_NAME="DBMS" or C_NAME="PROGRAMMING"));
```

Output

```
S1|Ram|Delhi|8789546524|28
S2|Sam|Delhi|5789546524|30
S3|Preeti|Delhi|2589546524|41
S4|Khusi|Delhi|5789546524|24
S5|Sejal|Delhi|9589546524|22
C1|DSA
C3|PROGRAMMING
C2|DBMS
Ram
Sam
Sam

[Execution complete with exit code 0]
```

DBMS
EXPERIMENT - 7

Exp. No	Conducted on	Submitted On	Date of Late Submission (if Any)	Max Marks Allotted	Marks Obtained	Faculty Signature
1	16/03/2022	23/03/2022		10		
Exp Title	**Join Queries**					

Aim

To join tables using the Join in SQL queries.

Syntax

SELECT table1.column, table2.column

FROM table1, table2

WHERE table1.column1 = table2.column2;

Queries:

```
CREATE TABLE  "ORDERS"
  (
      "ORD_NUM" NUMBER(6,0) NOT NULL PRIMARY KEY,
         "ORD_AMOUNT" NUMBER(12,2) NOT NULL,
         "ADVANCE_AMOUNT" NUMBER(12,2) NOT NULL,
         "ORD_DATE" DATE NOT NULL,
         "CUST_CODE" VARCHAR2(6) NOT NULL REFERENCES CUSTOMER,
         "AGENT_CODE" CHAR(6) NOT NULL REFERENCES AGENTS,
         "ORD_DESCRIPTION" VARCHAR2(60) NOT NULL
  );
INSERT INTO ORDERS VALUES('200100', '1000.00', '600.00', '08/01/2008', 'C00013', 'A003', 'SOD');
```

```
INSERT INTO ORDERS VALUES('200110', '3000.00', '500.00', '04/15/2008', 'C00019', 'A010', 'SOD');
INSERT INTO ORDERS VALUES('200107', '4500.00', '900.00', '08/30/2008', 'C00007', 'A010', 'SOD');
INSERT INTO ORDERS VALUES('200112', '2000.00', '400.00', '05/30/2008', 'C00016', 'A007', 'SOD');
INSERT INTO ORDERS VALUES('200113', '4000.00', '600.00', '06/10/2008', 'C00022', 'A002', 'SOD');
INSERT INTO ORDERS VALUES('200102', '2000.00', '300.00', '05/25/2008', 'C00012', 'A012', 'SOD');
INSERT INTO ORDERS VALUES('200114', '3500.00', '2000.00', '08/15/2008', 'C00002', 'A008',
'SOD');
INSERT INTO ORDERS VALUES('200122', '2500.00', '400.00', '09/16/2008', 'C00003', 'A004', 'SOD');
INSERT INTO ORDERS VALUES('200118', '500.00', '100.00', '07/20/2008', 'C00023', 'A006', 'SOD');
INSERT INTO ORDERS VALUES('200119', '4000.00', '700.00', '09/16/2008', 'C00007', 'A010', 'SOD');
INSERT INTO ORDERS VALUES('200121', '1500.00', '600.00', '09/23/2008', 'C00008', 'A004', 'SOD');
INSERT INTO ORDERS VALUES('200130', '2500.00', '400.00', '07/30/2008', 'C00025', 'A011', 'SOD');
INSERT INTO ORDERS VALUES('200134', '4200.00', '1800.00', '09/25/2008', 'C00004', 'A005',
'SOD');
INSERT INTO ORDERS VALUES('200108', '4000.00', '600.00', '02/15/2008', 'C00008', 'A004', 'SOD');
INSERT INTO ORDERS VALUES('200103', '1500.00', '700.00', '05/15/2008', 'C00021', 'A005', 'SOD');
INSERT INTO ORDERS VALUES('200105', '2500.00', '500.00', '07/18/2008', 'C00025', 'A011', 'SOD');
INSERT INTO ORDERS VALUES('200109', '3500.00', '800.00', '07/30/2008', 'C00011', 'A010', 'SOD');
INSERT INTO ORDERS VALUES('200101', '3000.00', '1000.00', '07/15/2008', 'C00001', 'A008',
'SOD');
INSERT INTO ORDERS VALUES('200111', '1000.00', '300.00', '07/10/2008', 'C00020', 'A008', 'SOD');
INSERT INTO ORDERS VALUES('200104', '1500.00', '500.00', '03/13/2008', 'C00006', 'A004', 'SOD');
INSERT INTO ORDERS VALUES('200106', '2500.00', '700.00', '04/20/2008', 'C00005', 'A002', 'SOD');
INSERT INTO ORDERS VALUES('200125', '2000.00', '600.00', '10/10/2008', 'C00018', 'A005', 'SOD');
INSERT INTO ORDERS VALUES('200117', '800.00', '200.00', '10/20/2008', 'C00014', 'A001', 'SOD');
INSERT INTO ORDERS VALUES('200123', '500.00', '100.00', '09/16/2008', 'C00022', 'A002', 'SOD');
INSERT INTO ORDERS VALUES('200120', '500.00', '100.00', '07/20/2008', 'C00009', 'A002', 'SOD');
INSERT INTO ORDERS VALUES('200116', '500.00', '100.00', '07/13/2008', 'C00010', 'A009', 'SOD');
INSERT INTO ORDERS VALUES('200124', '500.00', '100.00', '06/20/2008', 'C00017', 'A007', 'SOD');
```

INSERT INTO ORDERS VALUES('200126', '500.00', '100.00', '06/24/2008', 'C00022', 'A002', 'SOD');

INSERT INTO ORDERS VALUES('200129', '2500.00', '500.00', '07/20/2008', 'C00024', 'A006', 'SOD');

INSERT INTO ORDERS VALUES('200127', '2500.00', '400.00', '07/20/2008', 'C00015', 'A003', 'SOD');

INSERT INTO ORDERS VALUES('200128', '3500.00', '1500.00', '07/20/2008', 'C00009', 'A002', 'SOD');

INSERT INTO ORDERS VALUES('200135', '2000.00', '800.00', '09/16/2008', 'C00007', 'A010', 'SOD');

INSERT INTO ORDERS VALUES('200131', '900.00', '150.00', '08/26/2008', 'C00012', 'A012', 'SOD');

INSERT INTO ORDERS VALUES('200133', '1200.00', '400.00', '06/29/2008', 'C00009', 'A002', 'SOD');

select cust_name,ord_amount

from CUSTOMER,ORDERS

where CUSTOMER.cust_code=ORDERS.cust_code

Run Save

Output

```
Rangarappa|800
Avinash|500
Ramesh|500
Charles|500
Srinivas|500
Avinash|500
Cook|2500
Stuart|2500
Ramesh|3500
Ramanathan|2000
Steven|900
Ramesh|1200

[Execution complete with exit code 0]
```

select cust_name,ord_amount,advance_amount

from CUSTOMER,ORDERS

where CUSTOMER.cust_code=ORDERS.cust_code

and advance_amount>500;

Run Save

Output

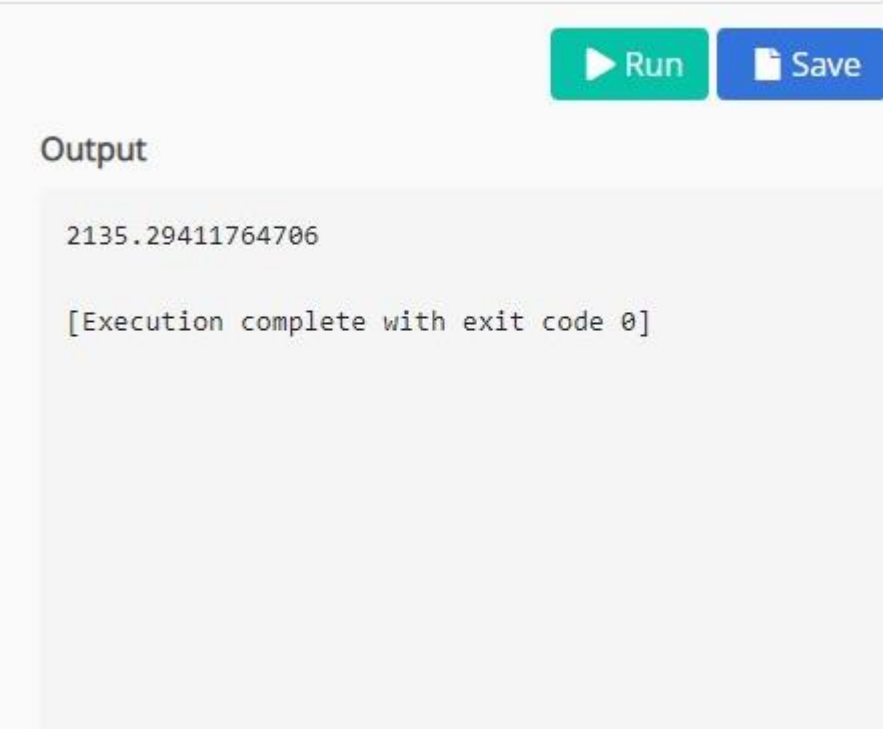
```
Bolt|3500|2000
Ramanathan|4000|700
Karolina|1500|600
Winston|4200|1800
Karolina|4000|600
Jacks|1500|700
Sundariya|3500|800
Micheal|3000|1000
Sasikant|2500|700
Fleming|2000|600
Ramesh|3500|1500
Ramanathan|2000|800

[Execution complete with exit code 0]
```

select avg(ord_amount)

from CUSTOMER,ORDERS

where customer.cust_code=orders.cust_code



Result

Thus the JOIN queries were successfully executed and verified.

Exp. No	Conducted on	Submitted On	Date of Late Submission (if Any)	Max Marks Allotted	Marks Obtained	Faculty Signature
1				10		
Exp Title	Exp No 8: Set Operation and Views					

Aim :- To study the implementation of Set Operators and Views.

Syntax (for all DDL Commands)

SQL Set Operation

The SQL Set operation is used to combine the two or more SQL SELECT statements.

Types of Set Operation

1. Union
2. UnionAll
3. Intersect
4. Minus

Queries (for all DDL Commands)

```
SELECT column_name FROM table1
UNION
SELECT column_name FROM table2;

SELECT column_name FROM table1
UNION ALL
SELECT column_name FROM table2;

SELECT column_name FROM table1
INTERSECT
SELECT column_name FROM table2;

SELECT column_name FROM table1
MINUS
SELECT column_name FROM table2;
```

Code:-

```
 create table First(id varchar(5), name varchar(10));
create table Second(id varchar(5), name varchar(10));
```

insert into First values('E1', 'Abhi');
insert into First values('E2', 'Adam');
insert into Second values('E2', 'Adam');
insert into Second values('E3', 'Chester');
SELECT * FROM First
UNION
SELECT * FROM Second;
SELECT * FROM First
UNION ALL
SELECT * FROM Second;
SELECT * FROM First
INTERSECT
SELECT * FROM Second;
SELECT * FROM First
MINUS
SELECT * FROM Second;

SQL

```sql
create table First(id varchar(5), name varchar(10));
create table Second(id varchar(5), name varchar(10));
insert into First values('E1', 'Abhi');
insert into First values('E2', 'Adam');
insert into Second values('E2', 'Adam');
insert into Second values('E3', 'Chester');
SELECT * FROM First
UNION
SELECT * FROM Second;
SELECT * FROM First
UNION ALL
SELECT * FROM Second;
SELECT * FROM First
INTERSECT
SELECT * FROM Second;
SELECT * FROM First
MINUS
SELECT * FROM Second;
```

Result:-

```
E1|Abhi        ┐
E2|Adam        ├ Union
E3|Chester     ┘

E1|Abhi        ┐
E2|Adam        ├Union All
E2|Adam        ┘
E3|Chester

E2|Adam        ──────> Intersect
```

E1|Abhi ──────>Union

ID	NAME
E1	Abhi
E2	Adam
E3	Chester

Download CSV

3 rows selected.

ID	NAME
E1	Abhi
E2	Adam
E2	Adam
E3	Chester

Download CSV

4 rows selected.

ID	NAME
E2	Adam

Download CSV

ID	NAME
E1	Abhi

Download CSV

DBMS
EXPERIMENT – 9

Saket Kumar Baranwal RA1911003010414

Exp. No	Conducted on	Submitted On	Date of Late Submission (if Any)	Max Marks Allotted	Marks Obtained	Faculty Signature
1	23/03/2022	02/05/2022		10		
Exp Title	Pl / SQL Conditional and Iterative Statements					

Aim

To study the various basic PL/SQL conditional and iterative operations on the database.

PL/ SQL GENERAL SYNTAX FOR IF CONDITION:

```
SQL> DECLARE ;
BEGIN
IF(CONDITION)THEN ;
END;
```

Coding for If Statement:

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

/

PL/ SQL GENERAL SYNTAX FOR IF AND ELSE CONDITION:

SQL> DECLARE ;

BEGIN

IF (TEST CONDITION)THEN

ELSE ;

ENDIF;

END;

Queries

```
declare
a number;
b number;
c number;
begin
a:=10;
b:=20;
c:= a+b;
dbms_output.put_line('sum of ' || a || ' and ' || b || ' is  = '|| c);
end;
/
```

Oracle Live SQL - SQL Worksheet

livesql.oracle.com/apex/f?p=590:1:7949650457113::NO:::

Live SQL

SQL Worksheet

```
declare
a number;
b number;
c number;
begin
a:=10;
b:=20;
c:= a+b;
dbms_output.put_line('sum of ' || a || ' and ' || b || ' is  = '|| c);
end;
/
```

Live SQL

SQL Worksheet

```
Statement processed.
sum of 10 and 20 is  = 30
```

```
declare
a number;
b number;
c number;
begin
a:=10;
b:=20;
c:=30;
if(a>b)and(a>c)then
dbms_output.put_line('A is maximum');
elsif (b > a) and (b > c) then
dbms_output.put_line('B is maximum');
else
dbms_output.put_line('C is maximum');
end if;
end;
```

Live SQL

SQL Worksheet

```
declare
a number;
b number;
c number;
begin
a:=10;
b:=20;
c:=30;
if(a>b)and(a>c)then
dbms_output.put_line('A is maximum');
elsif (b > a) and (b > c) then
dbms_output.put_line('B is maximum');
else
dbms_output.put_line('C is maximum');
end if;
end;
```

SQL Worksheet

```
Statement processed.
C is maximum
```

```
declare
n number;
sum1 number default 0;
endvalue number;

begin

endvalue:=30;
n:=1;
for n in 1..endvalue  loop
if mod(n,2)=1 then
sum1:=sum1+n;
end if;
end loop;
dbms_output.put_line('sum = ' || sum1);
end;
/
```

Oracle Live SQL - SQL Worksheet

livesql.oracle.com/apex/f?p=590:1:7949650457113::NO:::

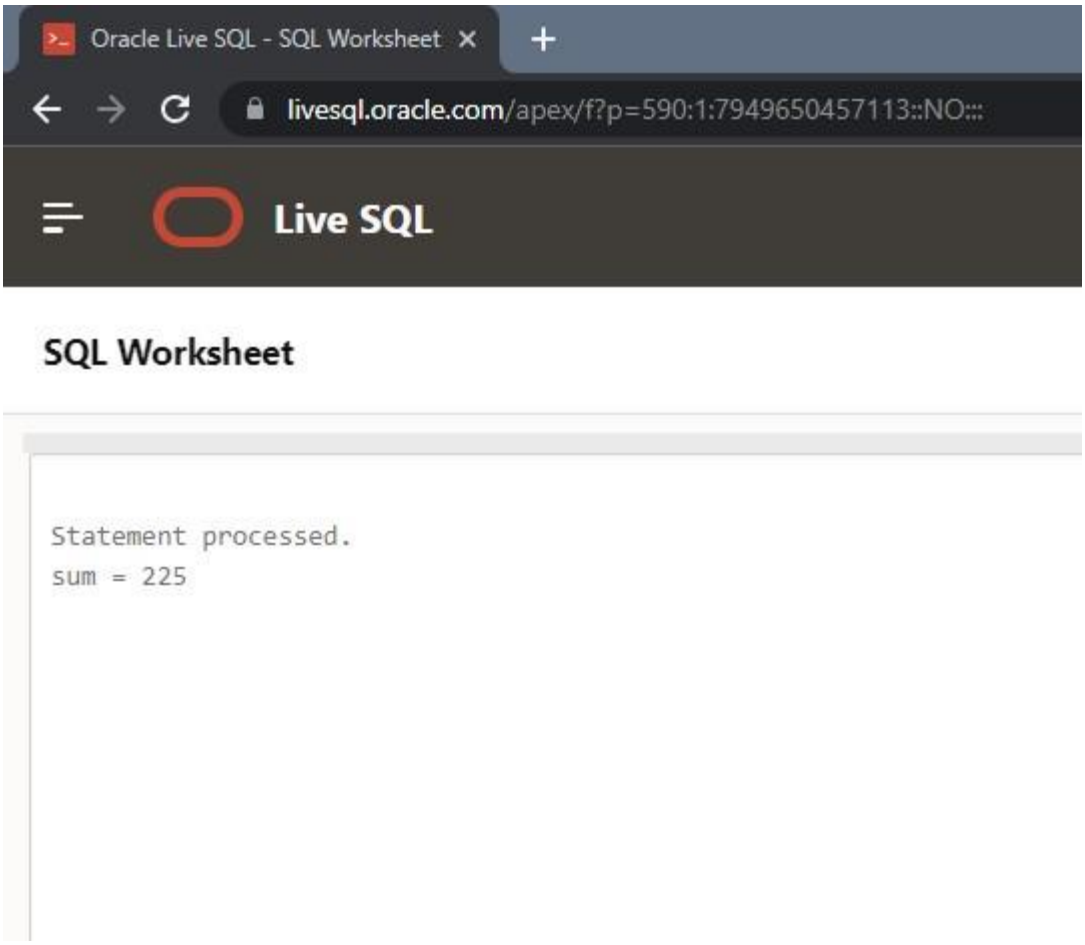
Live SQL

SQL Worksheet

```
declare
n number;
sum1 number default 0;
endvalue number;

begin

endvalue:=30;
n:=1;
for n in 1..endvalue  loop
if mod(n,2)=1 then
sum1:=sum1+n;
end if;
end loop;
dbms_output.put_line('sum = ' || sum1);
end;
/
```



Result:

Thus the pl/sql have been executed successfully.

DBMS
EXPERIMENT - 10

RA1911003010414
Saket Kumar Baranwal

Exp. No	Conducted on	Submitted On	Date of Late Submission (if Any)	Max Marks Allotted	Marks Obtained	Faculty Signature
1	30/03/2022	02/05/2022		10		
Exp Title	PI / SQL Procedures					

Aim

To study the various PL/SQL procedures operations on the database.

PL/ SQL GENERAL SYNTAX FOR LOOPING STATEMENT:

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

Queries

```
SQL> declare
  2  cursor c1 is
  3  select id, name, age, address, salary from employee;
  4  emp_id number;
  5  emp_name varchar(10);
  6  emp_age number;
  7  emp_address varchar(20);
  8  emp_salary number;
  9  begin
 10  open c1;
 11  loop
 12  fetch c1 into emp_id, emp_name, emp_age, emp_address, emp_salary;
 13  exit when c1%notfound;
 14  dbms_output.put_line(emp_id || ' ' || emp_name || ' ' || emp_age || ' ' ||
emp_address || ' ' || emp_salary);
 15  end loop;
 16  close c1;
 17  end;
 18  /
1 John 20 US 2000
2 Stephan 26 Dubai 1500
3 David 27 Bangkok 2000
4 Alina 29 UK 6500
5 Kathrin 34 Bangalore 8500
6 Harry 42 China 4500
7 Jackson 25 Mizoram 10000

PL/SQL procedure successfully completed.

SQL> select * from employee;
```

```
        ID NAME               AGE ADDRESS        SALARY
---------- ---------- ---------- ---------- ----------
         1 John               20 US               2000
         2 Stephan            26 Dubai            1500
         3 David              27 Bangkok          2000
         4 Alina              29 UK               6500
         5 Kathrin            34 Bangalore        8500
         6 Harry              42 China            4500
         7 Jackson            25 Mizoram         10000

7 rows selected.

SQL> spool off;
```

Result:

Thus the pl/sql have been executed successfully.

DBMS
EXPERIMENT - 11

RA1911003010414
Saket Kumar Baranwal

Exp. No	Conducted on	Submitted On	Date of Late Submission (if Any)	Max Marks Allotted	Marks Obtained	Faculty Signature
1	06/04/2022	02/05/2022		10		
Exp Title	PL / SQL Functions					

Aim

To study the various PL/SQL functions operations on the database.

Queries

```
SQL> set serveroutput on;
SQL> declare
  2  a number;
  3  b number;
  4  c number;
  5  procedure findMax(x in number, y in number, z out number) is
  6  begin
  7  if x > y then
  8  z:=x;
  9  else
```

```
 10  z:=y;
 11  end if;
 12  end;
 13  begin
 14  a:=154;
 15  b:=145;
 16  findMax(a,b,c);
 17  dbms_output.put_line('Maximum of (154, 145) : ' || c);
 18  end;
 19  /
Maximum of (154, 145) : 154

PL/SQL procedure successfully completed.

SQL> spool off;
```

Result:

Thus the pl/sql have been executed successfully.

DBMS
EXPERIMENT - 12

RA1911003010414
Saket Kumar Baranwal

Exp. No	Conducted on	Submitted On	Date of Late Submission (if Any)	Max Marks Allotted	Marks Obtained	Faculty Signature
1	13/04/2022	02/05/2022		10		
Exp Title	PL / SQL Cursors					

Aim

To study the various PL/SQL cursors operations on the database.

Queries

```
SQL> select * from employee;

        ID NAME               AGE ADDRESS         SALARY
---------- ---------- ---------- ---------- ----------
         1 John               20 US               2000
         2 Stephan            26 Dubai            1500
         3 David              27 Bangkok          2000
         4 Alina              29 UK               6500
         5 Kathrin            34 Bangalore        8500
         6 Harry              42 China            4500
```

```
         7 Jackson            25 Mizoram         10000

7 rows selected.

SQL> declare
  2  e_id employee.id%type:=8;
  3  e_name employee.name%type;
  4  e_addr employee.address%type;
  5  begin
  6  select name, address into e_name, e_addr
  7  from employee
  8  where id = e_id;
  9  dbms_output.put_line('Name:' || e_name);
 10  dbms_output.put_line('Address:' || e_addr);
 11  exception
 12  when no_data_found then
 13  dbms_output.put_line('No such employee');
 14  when others then
 15  dbms_output.put_line('Error');
 16  end;
 17  /
No such employee

PL/SQL procedure successfully completed.

SQL> spool off;
```

Result:

Thus the pl/sql have been executed successfully.

DBMS
EXPERIMENT - 13

RA1911003010414
Saket Kumar Baranwal

Exp. No	Conducted on	Submitted On	Date of Late Submission (if Any)	Max Marks Allotted	Marks Obtained	Faculty Signature
1	22/04/2022	02/05/2022		10		
Exp Title	PL / SQL Exception Handling					

Aim

To study the various PL/SQL exception handling operations on the database.

Syntax

```
DECLARE
declarations section;
BEGIN
executable command(s);
EXCEPTION
WHEN exception1 THEN
statement1;
WHEN exception2 THEN
statement2;
[WHEN others THEN]
```

```
/* default exception handling code */
END;
```

Queries

```
create table student(s_id int , s_name varchar(20), marks int);
insert into student values(1, 'Suraj',100);
insert into student values(2, 'Praveen',97);
insert into student values(3, 'Jessie', 99);

DECLARE
   temp varchar(20);

BEGIN
   SELECT s_id into temp from student where s_name='Shashwat';

exception
   WHEN no_data_found THEN
      dbms_output.put_line('ERROR');
      dbms_output.put_line('there is no name as');
      dbms_output.put_line('Shashwat in student table');
end;
```

livesql.oracle.com/apex/f?p=590:1:9215732284075::NO:::

Live SQL

SQL Worksheet

```
create table student(s_id int , s_name varchar(20), marks int);
insert into student values(1, 'Suraj',100);
insert into student values(2, 'Praveen',97);
insert into student values(3, 'Jessie', 99);

DECLARE
   temp varchar(20);

BEGIN
   SELECT s_id into temp from student where s_name='Shashwat';

exception
   WHEN no_data_found THEN
      dbms_output.put_line('ERROR');
      dbms_output.put_line('there is no name as');
      dbms_output.put_line('Shashwat in student table');
end;
```

livesql.oracle.com/apex/f?p=590:1:9215732284075::NO:::

Live SQL

SQL Worksheet

```
Table created.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

Statement processed.
ERROR
there is no name as
Shashwat in student table
```

livesql.oracle.com/apex/f?p=590:1:9215732284075::NO:::

Live SQL

SQL Worksheet

```
DECLARE
temp number;

BEGIN
SELECT s_name into temp from student where s_name='Suraj';
dbms_output.put_line('the s_name is '||temp);

EXCEPTION
WHEN value_error THEN
dbms_output.put_line('Error');
dbms_output.put_line('Change data type of temp to varchar(20)');

END;
```

livesql.oracle.com/apex/f?p=590:1:9215732284075::NO:::

Live SQL

SQL Worksheet

```
Statement processed.
Error
Change data type of temp to varchar(20)
```

livesql.oracle.com/apex/f?p=590:1:9215732284075::NO:::

Live SQL

SQL Worksheet

```
DECLARE
a int:=10;
b int:=0;
answer int;

BEGIN
answer:=a/b;
dbms_output.put_line('the result after division is'||answer);

exception
WHEN zero_divide THEN
    dbms_output.put_line('dividing by zero please check the values again');
    dbms_output.put_line('the value of a is '||a);
    dbms_output.put_line('the value of b is '||b);
END;
```

livesql.oracle.com/apex/f?p=590:1:9215732284075::NO:::

Live SQL

SQL Worksheet

```
Statement processed.
dividing by zero please check the values again
the value of a is 10
the value of b is 0
```

Result:

Thus the pl/sql have been executed successfully.

DBMS
EXPERIMENT - 14

RA1911003010414
Saket Kumar Baranwal

Exp. No	Conducted on	Submitted On	Date of Late Submission (if Any)	Max Marks Allotted	Marks Obtained	Faculty Signature
1	22/04/2022	02/05/2022		10		
Exp Title	PL / SQL Triggers					

Aim

To study the various PL/SQL triggers on the database.

Syntax

The syntax for creating a trigger is –

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

```
DECLARE
  Declaration-statements
BEGIN
  Executable-statements
EXCEPTION
  Exception-handling-statements
END;
```

Queries

livesql.oracle.com/apex/f?p=590:1:3243149987145::NO:::

Live SQL

SQL Worksheet

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

livesql.oracle.com/apex/f?p=590:1:3243149987145::NO:::

Live SQL

SQL Worksheet

```
Trigger created.
```

livesql.oracle.com/apex/f?p=590:1:3243149987145::NO:::

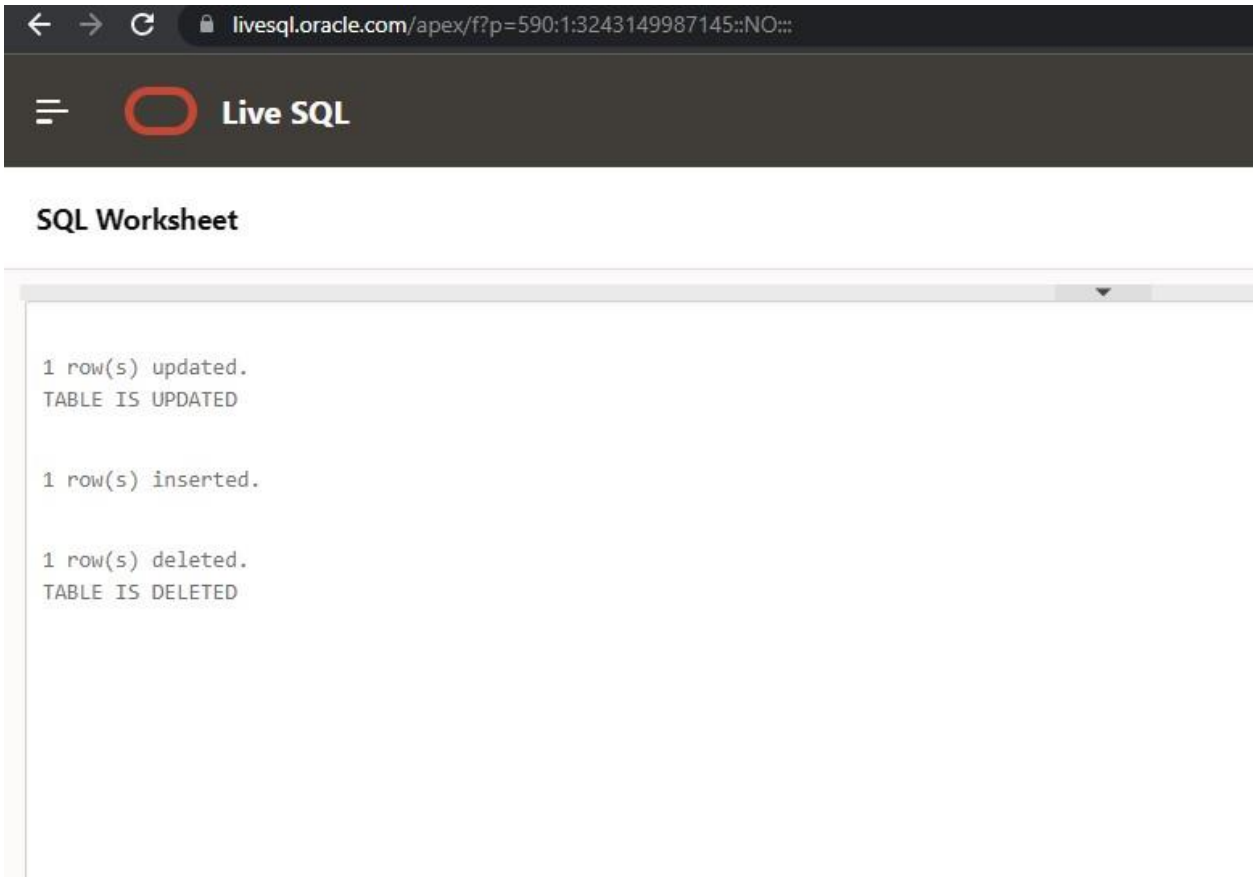
Live SQL

SQL Worksheet

```
update student set marks=90 where s_name='Suraj';

insert into geeks values(1, 'Rohan',100);

delete from student where s_name='Praveen';
```



Result:

Thus the pl/sql have been executed successfully.

DBMS
EXPERIMENT - 15

RA1911003010414
Saket Kumar Baranwal

Exp. No	Conducted on	Submitted On	Date of Late Submission (if Any)	Max Marks Allotted	Marks Obtained	Faculty Signature
1	22/04/2022	02/05/2022		10		
Exp Title	PL / SQL Cursors and Exception Handling for Project					

Aim

To execute appropriate PL/SQL cursors and Exception Handling for the Project.

Queries

```
CREATE OR REPLACE PROCEDURE add_to_wish_list (
   buyer_id     IN   VARCHAR,
   product_id   IN   INTEGER
) AS
BEGIN
   INSERT INTO wish_list VALUES (
      buyer_id,
      sysdate
   );

   INSERT INTO product_wishlist VALUES (
```

```
        product_id,
        buyer_id
    );

END add_to_wish_list;

CREATE OR REPLACE PROCEDURE place_order (
    order_id       IN   INTEGER,
    buyer_id_var   IN   VARCHAR
) AS

    card_id_var          INTEGER;
    address_id_var       INTEGER;
    total_price_var      NUMBER := 0;
    curr_price_var       NUMBER;
    total_qty_var        NUMBER := 0;
    available_units_var  NUMBER(1);
    shipping_price_var   NUMBER := 10;
    is_prime_var         NUMBER := 0;
    CURSOR products_cur IS
    SELECT
        product_id
    FROM
        product_shoppingcart
    WHERE
```

```
        buyer_id = buyer_id_var;

    product_id_var       INTEGER;
BEGIN
    OPEN products_cur;
    LOOP
        FETCH products_cur INTO product_id_var;
        EXIT WHEN products_cur%notfound;
        SELECT
            price,
            available_units
        INTO
            curr_price_var,
            available_units_var
        FROM
            product
        WHERE
            product_id = product_id_var;

        IF available_units_var > 0 THEN
            total_price_var := ( total_price_var + curr_price_var );
            total_qty_var := total_qty_var + 1;
            INSERT INTO order_product VALUES (
                order_id,
```

```
            product_id_var
        );

      END IF;

--      DELETE FROM product_shoppingcart
--      WHERE product_id = product_id_var AND buyer_id = buyer_id_var;

    END LOOP;

    CLOSE products_cur;
    SELECT
      is_prime
    INTO is_prime_var
    FROM
      buyer
    WHERE
      buyer_id = buyer_id_var;

    IF is_prime_var = 1 THEN
      shipping_price_var := 0;
    END IF;
    SELECT
      card_id
```

```
INTO card_id_var
FROM
   card_info
WHERE
   buyer_id = buyer_id_var
   AND is_default = 1;

SELECT
   address_id
INTO address_id_var
FROM
   contact_detail
WHERE
   user_id = buyer_id_var
   AND is_default = 1;

total_price_var := total_price_var + shipping_price_var + 10;
INSERT INTO amz_order VALUES (
   order_id,
   buyer_id_var,
   card_id_var,
   total_price_var,
   sysdate,
   10,
```

```
        shipping_price_var,
        address_id_var,
        add_months(DATE '2019-11-28', 1),
        'c',
        total_qty_var
    );

END place_order;
BEGIN
    add_to_wish_list('shashwatraj@gmail.com', 1);
    add_to_wish_list('shashwatraj@gmail.com', 4);
    add_to_wish_list('rohankr@gmail.com', 3);
    add_to_wish_list('rohankr@gmail.com', 2);
END;

BEGIN
    add_to_shopping_cart('shashwatraj@gmail.com', 1);
    add_to_shopping_cart('shashwatraj@gmail.com', 3);
    add_to_shopping_cart('rohankr@gmail.com', 2);
    add_to_shopping_cart('rohankr@gmail.com', 1);
END;
```

Procedure with exception handling

```
create or replace procedure prod_details(p_id in varchar)
is
quan number(2);
begin
select quantity into quan from product where product_id=p_id;
exception
when no_data_found then
dbms_output.put_line('Sorry no such product exist !!');
end;
/
```

Result:

Thus the pl/sql have been executed successfully.