Telco Dt.:
Pg.:

Experiment - 1

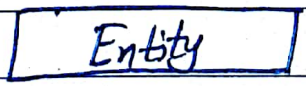
Aim: Draw an Entity Relationship Diagram & give a brief description.

Theory:

An Entity Relationship Diagram (ERD) shows the relationship of an entity set stored in a database. An entity in this context is a component of data. In other words, ER diagrams illustrate the logical structure of databases.

It is a means of visualizing how the information a system produces is related.

Entities: These are represented by rectangles. It is an object or concept about which you want to store information.



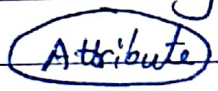
A weak entity is an entity that must be defined by a foreign key relationship with another entity as it can't be uniquely identified by its own attributes alone.

Telco Dt.: Pg.:

Actions: Which are represented by diamond shapes. They show how two entities share information in data.

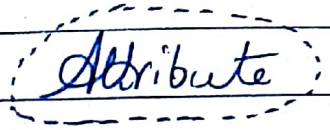
Attributes: Which are represented by ovals. A key attribute is the unique, distinguishing sequence (characteristic) of the entity.

e.g. → An employee social security no. might be the employee's key attribute.



Multivalued Attribute → Can have more than one value, e.g. Employee entity with multiple skills.

Derived Attribute → Based on another attribute. e.g. Age & DoB of an employee.



Connecting Lines → Solid lines that connect attributes to show the relationships of entities in the diagram.

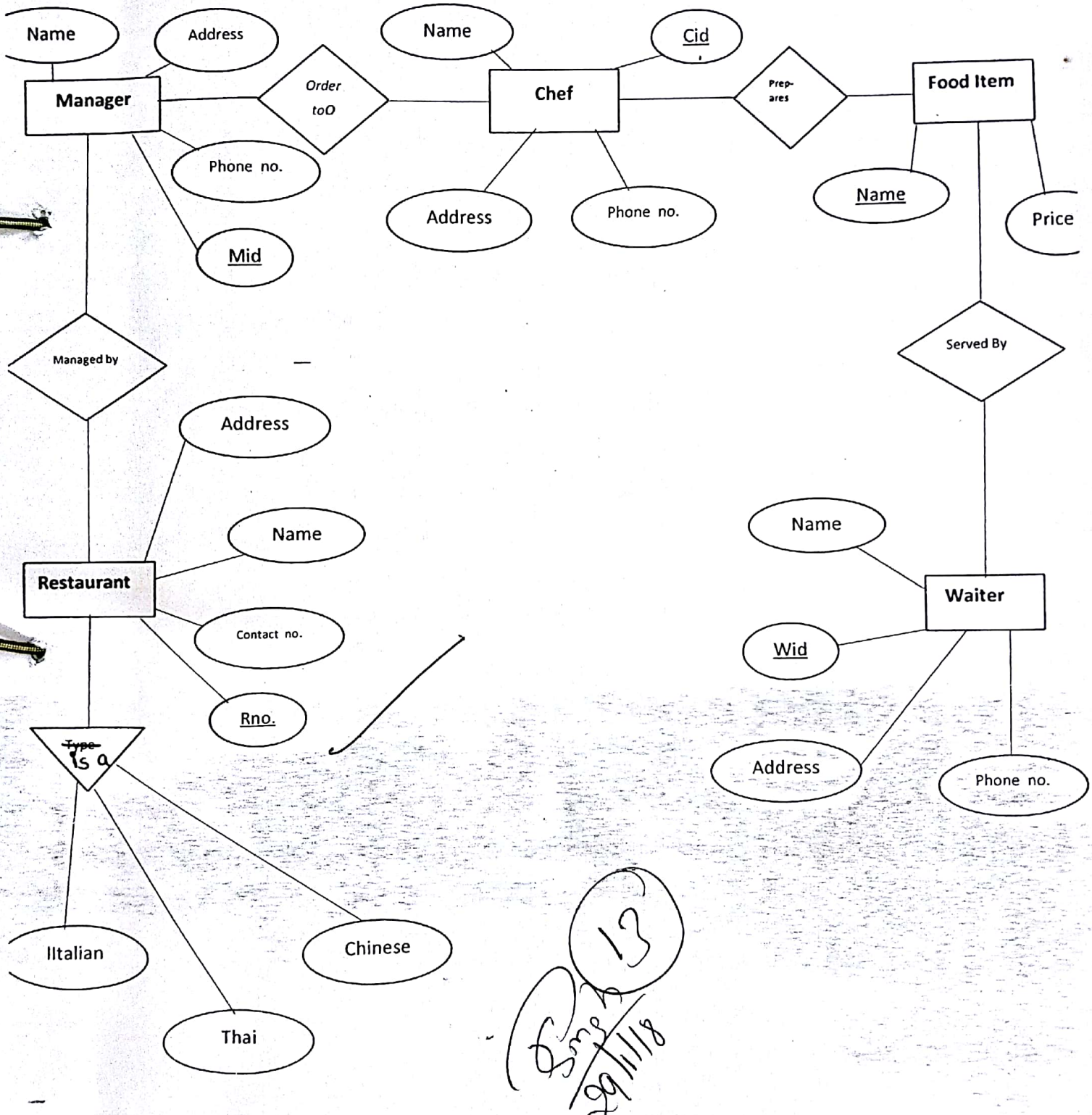
Telco Dt.: Pg.:

Cardinality → Specifies how many instances of an entity relate to one instance of another entity. Ordinality is also linked to Cardinality, while cardinality specifies the occurences of a relationship, Ordinality describes the relationship as either mandatory or optional.

Uses of ER Diagram:

When documenting a system or process, looking at system in multiple ways increases the understanding of that system. Used commonly in conjunction with a data flow diagram to display the contents of a data store. They help us to visualise that data is connected in a general way & are particularly useful for constructing a relational database.

E-R Diagram



Telco Dt.: Pg.:

Experiment 2

- **Aim**: DDL commands: To study create, alter, insert & delete commands.

1. Create table <tablename>: This command is used to create a table.
2. Alter table <tablename>: Used to add a column in existing table.
3. Insert: Used to insert data into table.
4. Delete from <tablename>: Used to delete a row from a table.
5. DROP: Used to delete a table.
6. Check: Used to add a constraint to the column.

Constraint

1. NOT NULL: A column for which value cannot be NULL
2. Unique: All values must be unique.
3. Primary Key: All values should be unique & not null.

Scanned by CamScanner

Telco Dt.:
Pg.:

Default : A value that is ~~used~~ inserted into column if no value for that column is provided.

Foreign Key : It links a table column to another column in another table.

SQL> create table shankar(sid int, name varchar(20), age int);

Table created.

SQL> desc shankar

Name	Null?	Type
SID		NUMBER(38)
NAME		VARCHAR2(20)
AGE		NUMBER(38)

SQL> create table shankar1(SID int NOT NULL, Name varchar(20) , age int);

Table created.

SQL> desc shankar1

Name	Null?	Type
SID	NOT NULL	NUMBER(38)
NAME		VARCHAR2(20)
AGE		NUMBER(38)

SQL> create table shankar2(sid int unique, Name varchar(20), age int);

Table created.

SQL> desc shankar2

Name	Null?	Type
SID		NUMBER(38)
NAME		VARCHAR2(20)
AGE		NUMBER(38)

SQL> alter table shankar1 add unique(sid);

Table altered.

Scanned by CamScanner

SQL> desc shankar1

Name	Null?	Type
SID	NOT NULL	NUMBER(38)
NAME		VARCHAR2(20)
AGE		NUMBER(38)

SQL> Insert into Shankar(sid,name,age) values(1,'shankar',20);

1 row created.

SQL> desc shankar

Name	Null?	Type
SID		NUMBER(38)
NAME		VARCHAR2(20)
AGE		NUMBER(38)

SQL> select * from shankar;

SID	NAME	AGE
1	shankar	20

SQL> commit;

Commit complete.

SQL> desc shankar1;

Name	Null?	Type
SID	NOT NULL	NUMBER(38)
NAME		VARCHAR2(20)
AGE		NUMBER(38)

SQL> Insert into Shankar(sid,name,age) values(1,'shankar',20);

1 row created.

SQL> select * from shankar;

SID	NAME	AGE
1	shankar	20
1	shankar	20

SQL> create table shankar3(sid int check(sid>0));

Table created.

SQL> desc shankar3;

Name	Null?	Type
SID		NUMBER(38)

SQL> Insert into Shankar1(sid,name,age) values(-1,'shankar',20);

1 row created.

SQL> desc shankar1

Name	Null?	Type
SID	NOT NULL	NUMBER(38)
NAME		VARCHAR2(20)
AGE		NUMBER(38)

SQL> select * from shankar1;

SID	NAME	AGE
-1	shankar	20

SQL> commit;

Commit complete.

SQL> alter table shankar1 add check(sid>0);

alter table shankar1 add check(sid>0)

*

ERROR at line 1:

ORA-02293: cannot validate (SCOTT.SYS_C005721) - check constraint violated

SQL> alter table shankar2 add (roll int);

Table altered.

SQL> desc shankar2

Name	Null?	Type
SID		NUMBER(38)
NAME		VARCHAR2(20)
AGE		NUMBER(38)
ROLL		NUMBER(38)

SQL> commit;

Commit complete.

SQL> create table shankar4(roll int unique, name varchar(20));

Table created.

SQL> alter table shankar4 add primary key(roll);

alter table shankar4 add primary key(roll)

*ERROR at line 1:

ORA-02261: such unique or primary key already exists in the table

SQL> create table shankar5(roll int primary key, name varchar(20));

Table created.

SQL> desc shankar5

Name	Null?	Type
ROLL	NOT NULL	NUMBER(38)
NAME		VARCHAR2(20)

SQL> insert into shankar5 values(1,"bob");

insert into shankar5 values(1,"bob")

*ERROR at line 1:

ORA-00984: column not allowed here

SQL> insert into shankar5 values(1,'bob');

1 row created.

SQL> create table shankar6(id int primary key, marks int, roll int);

Table created.

SQL> alter table shankar6 add foreign key(roll) references shankar5(roll);

Table altered.

SQL> create table shankar7(id int primary key, marks int, roll int, foreign key(roll) references sha

nkar5(roll));

Table created.

SQL> desc shankar7

Name	Null?	Type
ID	NOT NULL	NUMBER(38)
MARKS		NUMBER(38)
ROLL		NUMBER(38)

SQL>delete from shankar2 whereage=20;

SQL>drop table shankar2;

SQL>Alter table Shankar2 modify age bigint;

14

8/11/18

Telco Dt.: Pg.:

Experiment 3

- **Aim**: DML Commands : To study select, insert, update & delete commands.

- **Theory**

1. **Select** : This command is used to retrieve data from the database.
   Syntax → Select <attribute> from <table name>;
   Eg. → select stdID, Name from Student;

2. **Delete** : This command deletes row(s) from a table.
   Syntax → delete from <table name> where <condition>;
   E.g. → delete from student where stdID = 1998;

3. **Update** : Used to change values that are already in table.
   Syntax → update <table name> set <attribute> = <expression>; where <condition>;
   Eg. → update student SET Name = 'John' where stdID = 1998;

4. **Insert** : Used to add new row to the table.
   Syntax → insert INTO <table name> VALUES (<value1>, <value2>, ..);
   Eg → INSERT INTO Student values (1998, 'John');

SQL> create table ProductItEve(ProductId int Primary Key, ProductCode varchar(10) NOT NULL. name varchar(30) NOT NULL, Quantity int NOT NULL, Price int NOT NULL);

Table created.

SQL> desc ProductItEve

Name	Null?	Type
PRODUCTID	NOT NULL	NUMBER(38)
PRODUCTCODE	NOT NULL	VARCHAR2(10)
NAME	NOT NULL	VARCHAR2(30)
QUANTITY	NOT NULL	NUMBER(38)
PRICE	NOT NULL	NUMBER(38)

SQL> Insert into ProductItEve(ProductId,ProductCode,name,Quantity,Price) Values(1001,'Pen','Pen Red'

, 5000, 5);

1 row created.

SQL> Insert into ProductItEve(ProductId,ProductCode,name,Quantity,Price) Values(1002,'Pen','Pen Blue

', 8000,10);

1 row created.

SQL> Insert into ProductItEve(ProductId,ProductCode,name,Quantity,Price) Values(1003,'Pen','Pen Blac

k', 2000,12);

1 row created.

SQL> Insert into ProductItEve(ProductId,ProductCode,name,Quantity,Price) Values(1004,'Pec','Pencil 2

B', 10000, 16);

1 row created.

SQL> Insert into ProductItEve(ProductId,ProductCode,name,Quantity,Price) Values(1005,'Pec','Pencil 2

H', 8000, 20);

1 row created.

SQL> Insert into ProductItEve(ProductId,ProductCode,name,Quantity,Price) Values(1006,'Pec','Pencil H

B', 5000, 20);

1 row created.

SQL> select * from ProductItEve;

PRODUCTID	PRODUCTCOD	NAME	QUANTITY	PRICE
1001	Pen	Pen Red	5000	5
1002	Pen	Pen Blue	8000	10
1003	Pen	Pen Black	2000	12
1004	Pec	Pencil 2B	10000	16
1005	Pec	Pencil 2H	8000	20
1006	Pec	Pencil HB	5000	20

6 rows selected

SQL> select name,price from ProductItEve;

NAME	PRICE
Pen Red	5
Pen Blue	10
Pen Black	12
Pencil 2B	16
Pencil 2H	20
Pencil HB	20

6 rows selected.

SQL> select name,price from ProductItEve where price<10 ;

SQL> select name,price from ProductItEve where price<10 ;

NAME	PRICE
Pen Red	5

SQL> select name, quantity from ProductItEve where quantity<10000 ;

NAME	QUANTITY
Pen Red	5000

Pen Blue	8000
Pen Black	2000
Pencil 2H	8000
Pencil HB	5000

SQL> select * from ProductItEve where name like '%pen%';

no rows selected

SQL> select * from ProductItEve where name like '%Pen%';

PRODUCTID	PRODUCTCOD	NAME	QUANTITY	PRICE
1001	Pen	Pen Red	5000	5
1002	Pen	Pen Blue	8000	10
1003	Pen	Pen Black	2000	12
1004	Pec	Pencil 2B	10000	16
1005	Pec	Pencil 2H	8000	20
1006	Pec	Pencil HB	5000	20

6 rows selected.

SQL> .

SQL> select * from ProductItEve where name='Pen' AND ProductId=1001 OR Price=10;

PRODUCTID	PRODUCTCOD	NAME	QUANTITY	PRICE
1001	Pen	Pen Red	5000	5
1002	Pen	Pen Blue	8000	10

SQL> select * from ProductItEve where name='Pen Red' and ProductId=1001;

PRODUCTID	PRODUCTCOD	NAME	QUANTITY	PRICE
1001	Pen	Pen Red	5000	5

SQL> select * from ProductItEve where name in('Pen Red', 'Pencil 2B');

PRODUCTID	PRODUCTCOD	NAME	QUANTITY	PRICE
1001	Pen	Pen Red	5000	5
1004	Pec	Pencil 2B	10000	16

SQL> select * from ProductItEve where Price Between 5000 AND 10000;

no rows selected

SQL> select * from ProductItEve where Quantity Between 5000 AND 10000;

PRODUCTID	PRODUCTCOD	NAME	QUANTITY	PRICE
1001	Pen	Pen Red	5000	5
1002	Pen	Pen Blue	8000	10
1004	Pec	Pencil 2B	10000	16
1005	Pec	Pencil 2H	8000	20

1006 Pec Pencil HB 5000 20

SQL> select 1+1 from ProductItEve;

6 rows selected.

SQL> select 1+1 FROM DUAL;

1+1

2

SQL> select quantity as no from ProductItEve;

NO

5000

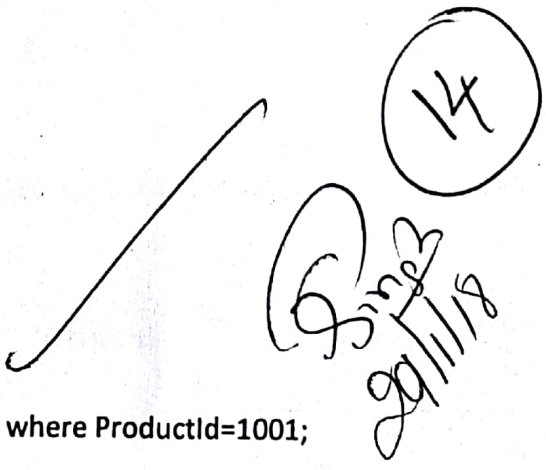
8000

2000

10000

8000

5000



SQL> update ProductIteve set price=25 where ProductId=1001;

1 row updated.

Page No.
Date

Experiment 4

- **Aim**: To implement the set operators & clauses like where, order by, group by, having like operator etc.

- **Theory**

1. **Where clause**: The where clause is used in SELECT Statement to specify the criteria for selection of rows to be returned.

Syntax –
```
SELECT <column name>
FROM <table name>
WHERE <condition>
```

2. **Order By**: The order by clause allows sorting of query results by one or more columns. The sorting can be done either in ascending or descending order.

Syntax –
```
SELECT << column name 1>..... <column name n>>
FROM <table name> WHERE <condition> ORDER BY <coln. name>
```

3. **Group by**: It is used in SELECT Statements to divide the table into groups. Grouping can be done by column name or with aggregate functions.

Syntax –
```
SELECT <column name> FROM <table name>
Group By <column name>
```

Page No.
Date

4. Having clause : It ~~helps~~ places conditions on groups in contrast to where clause that places condition on individual rows.
SELECT <column name> FROM <table name>
Group by <column name> having <condition>.

5. Like Operator : It is used for comparisions on character strings using pattern. Patterns are described using -

(a). Percent (%) - matchees any substring.
(b). underscore (_) - It matches any one character.

```
SQL> create table customer(c_id int primary key,fname varchar(10) NOT NULL,lname varchar(10),city va
rchar(20),country varchar(20),phone int default(1234));

Table created.

SQL> create table order_p(o_id int primary key,o_date date,o_no int,C_id int,total_amt int,foreign key(c_id) references customer(c_id));

Table created.

SQL> insert into customer values(1,'Vishal','Dubey','Delhi','India',9999999999);

1 row created.

SQL> insert into customer(c_id,fname,lname,city,country) values(2,'Udit','Dahima','Delhi','India');

1 row created.

SQL> insert into customer values(3,'Udita','Dahi','Mumbai','India',9989999999);

1 row created.

SQL> insert into customer values(4,'Udi','Dahi','Mumbai','India',9989999998);

1 row created.

SQL> insert into customer(c_id,fname,lname,city,country) values(5,'Bhavya','Takkar','Mumbai','India');

1 row created.

SQL> insert into customer values(6,'Smarty','Dahi','Kolkata','India',9889999998);

1 row created.

SQL> insert into customer values(7,'Honey','Singh','Kolkata','India',9889979998);

1 row created.

SQL> insert into customer values(8,'Param','Singh','Kolkata','India',9889979938);

1 row created.

SQL> select * from customer;

      C_ID FNAME      LNAME      CITY            COUNTRY         PHONE
---------- ---------- ---------- --------------- --------------- --------
         1 Vishal     Dubey      Delhi           India           9999999999
         2 Udit       Dahima     Delhi           India           1234567891
         3 Udita      Dahi       Mumbai          India           9989999999

      C_ID FNAME      LNAME      CITY            COUNTRY         PHONE
---------- ---------- ---------- --------------- --------------- --------
         4 Udi        Dahi       Mumbai          India           9989999998
         5 Bhavya     Takkar     Mumbai          India           1234567891
```

```
6 Smarty    Dahi     Kolkata          India          9889999998

C_ID FNAME     LNAME     CITY            COUNTRY    PHONE
---------- ---------- ---------- ---------- ---------- ---------

7 Honey     Singh     Kolkata          India          9889979998

8 Param     Singh     Kolkata          India          9889979938

8 rows selected.

SQL> insert into customer values(9,'Kaaliya','Dahima','Sangam Vihar','India',9871979938);

1 row created.

SQL> select * from customer;

C_ID FNAME     LNAME     CITY            COUNTRY    PHONE
---------- ---------- ---------- ---------- ---------- ---------

1 Vishal    Dubey     Delhi            India          9999999999

2 Udit      Dahima    Delhi            India          1234567891

3 Udita     Dahi      Mumbai           India          9989999999

C_ID FNAME     LNAME     CITY            COUNTRY    PHONE
---------- ---------- ---------- ---------- ---------- ---------

4 Udi       Dahi      Mumbai           India          9989999998

5 Bhavya    Takkar    Mumbai           India          1234567891

6 Smarty    Dahi      Kolkata          India          9889999998

C_ID FNAME     LNAME     CITY            COUNTRY    PHONE
---------- ---------- ---------- ---------- ---------- ---------

7 Honey     Singh     Kolkata          India          9889979998

8 Param     Singh     Kolkata          India          9889979938

9 Kaaliya   Dahima    Sangam Vihar     India          9871979938

9 rows selected.

SQL> insert into order_p values(1,DATE '2017-12-12',20,1,1200);

1 row created.

SQL> insert into order_p values(2,DATE '2017-12-08',21,2,200);

1 row created.

SQL> insert into order_p values(3,DATE '2017-11-08',22,3,2000);
```

1 row created.

SQL> insert into order_p values(4,DATE '2017-10-07',23,4,2100);

1 row created.

SQL> insert into order_p values(5,DATE '2016-10-07',24,5,1100);

1 row created.

SQL> insert into order_p values(6,DATE '2016-11-07',25,6,1200);

1 row created.

SQL> insert into order_p values(7,DATE '2016-11-30',26,7,1000);

1 row created.

SQL> insert into order_p values(8,DATE '2015-11-30',27,8,1800);

1 row created.

SQL> insert into order_p values(9,DATE '2019-10-30',27,9,1800);

1 row created.

SQL> select * from order_p;

O_ID	O_DATE	O_NO	C_ID	TOTAL_AMT
1	12-DEC-17	20	1	1200
2	08-DEC-17	21	2	200
3	08-NOV-17	22	3	2000
4	07-OCT-17	23	4	2100
5	07-OCT-16	24	5	1100
6	07-NOV-16	25	6	1200
7	30-NOV-16	26	7	1000
8	30-NOV-15	27	8	1800
9	30-OCT-19	27	9	1800

9 rows selected.

SQL> select count(c_id),city from customer group by city;

COUNT(C_ID)	CITY
2	Delhi
3	Mumbai
1	Sangam Vihar

3 Kolkata

SQL> select count(c_id),city from customer group by city;

COUNT(C_ID) CITY

---

2 Delhi

3 Mumbai

1 Sangam Vihar

3 Kolkata

SQL> select count(c_id),city from customer group by city;

COUNT(C_ID) CITY

---

2 Delhi

3 Mumbai

1 Sangam Vihar

3 Kolkata

SQL> select count(c_id),city from customer group by city order by count(c_id);

COUNT(C_ID) CITY

---

1 Sangam Vihar

2 Delhi

3 Kolkata

3 Mumbai

SQL> select count(c_id),city from customer group by city order by count(c_id),city desc;

COUNT(C_ID) CITY

---

1 Sangam Vihar

2 Delhi

3 Mumbai

3 Kolkata

SQL> select count(c_id),city from customer group by city

2 having count(c_id)>2

3 order by count(c_id),city desc;

```
COUNT(C_ID) CITY
---------- ----------
         3 Mumbai
         3 Kolkata

SQL> select count(c_id),city from customer group by city
  2  having count(c_id)<=2
  3  order by count(c_id),city desc;

COUNT(C_ID) CITY
---------- ----------
         1 Sangam Vihar
         2 Delhi

SQL> select count(c_id),city from customer group by city
  2  having count(c_id)<=2
  3  order by count(c_id),city;

COUNT(C_ID) CITY
---------- ----------
         1 Sangam Vihar
         2 Delhi

SQL> select count(c_id),city from customer group by city
  2  having count(c_id)<=2
  3  order by count(c_id) desc,city;

COUNT(C_ID) CITY
---------- ----------
         2 Delhi
         1 Sangam Vihar
```

12/2/18

Experiment 5

- **Aim**: Implement the various inbuilt functions & views.
- **S/w Used**: MySQL
- **Theory**

MySQL contains many inbuilt functions -

Count : used to count the no. of rows in table

Max : Used to select the max. value from a column.

Min : Used to select the min. value from a column.

Avg : Used to average value for a certain table col.

Sqrt : Generate a square root of a no.

Sum : Allow selecting total of a numeric column.

SQL> select min(price),max(price),avg(price),count(price),sum(price) from sakaar;

MIN(PRICE) MAX(PRICE) AVG(PRICE) COUNT(PRICE) SUM(PRICE)

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

2 10 4.83333333 6 29

SQL> select min(price),max(price),avg(price),count(price),sum(price) from sakaar group by product_code;

MIN(PRICE) MAX(PRICE) AVG(PRICE) COUNT(PRICE) SUM(PRICE)

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

5 10 6.66666667 3 20

2 4 3 3 9

SQL> select product_code,count(*) from sakaar group by product_code;

PRODUCT_CODE COUNT(*)

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

ter 3

pvc 3

SQL> select * from empl1;

ID NAME AGE SAL DNO

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

1 Bob 30 40000 10

2 Nishant 20 1000000 20

3 Nalin 21 200000 30

4 Sameer 22 1500000 10

5 Mohit 23 175000 20

6 Varun 24 50 30

6 rows selected.

ID	NAME	AGE	SAL	DNO
1	Bob	30	40000	10
2	Nishant	20	1000000	20
3	Nalin	21	200000	30
4	Sameer	22	1500000	10
5	Mohit	23	175000	20
6	Varun	24	50	30
1	Bob	30	40000	10
2	bran	45	34509	20
3	Tran	25	509	30

9 rows selected.

```
SQL> select max(sal),min(sal),avg(sal) from empl1 group by dno;

 MAX(SAL)   MIN(SAL)   AVG(SAL)
---------- ---------- ----------
    200000         50     100025
   1000000     175000     587500
   1500000      40000     770000

SQL> select dno,avg(sal) from empl1 group by dno having avg(sal)>1000;

       DNO   AVG(SAL)
---------- ----------
        30     100025
        20     587500
        10     770000

SQL> select dno,avg(sal) from empl1 group by dno having avg(sal)>(select avg(sal) from empl1);

       DNO   AVG(SAL)
---------- ----------
        20     587500
        10     770000

SQL> select * from empl1
  2  union
  3  select * from empl3;

        ID NAME                 AGE        SAL        DNO
---------- -------------------- ---------- ---------- ----------
         1 Bob                          30      40000         10
         2 Nishant                      20    1000000         20
         2 bran                         45      34509         20
         3 Nalin                        21     200000         30
         3 Tran                         25        509         30
```

14

19/3/18

```
4 Sameer          22   1500000      10
5 Mohit           23   175000       20
6 Varun           24   50           30

3 rows selected.

SQL> select * from empl1
  2  minus
  3  select * from empl3;

ID NAME          AGE      SAL      DNO
---------- ---------- ---------- ---------- ----------
2 Nishant         20   1000000      20
3 Nalin           21   200000       30
4 Sameer          22   1500000      10
5 Mohit           23   175000       20
6 Varun           24   50           30

SQL> select * from empl1
  2  intersect
  3  select * from empl3;

ID NAME          AGE      SAL      DNO
---------- ---------- ---------- ---------- ----------
1 Bob             30   40000        10

SQL> select * from empl1
  2  union all
  3  select * from empl3;
```

Page No.
Date

Experiment 6

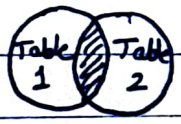
- **Aim**: Implement various operations on Joins.

- **Theory**

A join clause is used to combine rows from two or more tables, based on a related column between them.

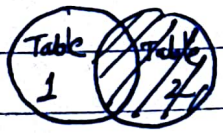
The different types of joins in SQL are -

Inner Join : Return records that have matching values in both tables



Left Outer Join : Returns all records from the left table & the matched records of right table.

Right Outer Join : Returns all records from right table & the matched records from left table.



Full Outer Join : Returns all records from left & right table.

Page No.
Date

Example showing inner join :

→ Select Orders.OrderID, customers.CustomerName, from Orders InnerJoin Customers On Orders. CustomerID = Customers.CustomerID;

Self Join : It is a special type of join that can be applied by creating 2 or more instances of the same table.

Example : Select a.eid as amp, a.ename as empname, b.eid as ~~amp~~ supid, ~~a.ename~~ from emd a, emp b where a.sid = b.eid;

SQL*Plus: Release 10.2.0.1.0 - Production on Mon Mar 26 16:47:48

2018

Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 -

Production
With the Partitioning, OLAP and Data Mining options

SQL> create table client(client_novarchar(10) primary key,namevarchar(10) not null,cityvarchar(10),balance int,check(client_no like 'C%'));

Table created.

SQL> create table product(product_novarchar(10) primary key,descrvarchar(10) not null,spint not null,cpint not null,check(product_no like 'P%'),check(spl=0),check(cpl=0));

Table created.

SQL> create table order1(client_novarchar(10),product_novarchar(10),quant int not null,rateint not null,order_statusvarchar(20),check(order_statusin('filled','inprocess','cancelled')),foreign key(client_no) references client(client_no),foreign key (product_no) references product(product_no),primary key (client_no,product_no));

Table created.

SQL> insert into client values('C1','yagya','delhi',1000);

1 row created.

SQL> insert into client values('C2','sumit','delhi',2000);

1 row created.

SQL> insert into client values('C3','harman','punjab',3000);

1 row created.

SQL> insert into client values('C4','saurav','haryana',4000);

1 row created.

SQL> insert into client values('C5','shreyansh','bhopal',5000);

1 row created.

SQL> insert into product values('P1','pen',10,5);

1 row created.

SQL> insert into product values('P2','pencil',8,5);

1 row created.

SQL> insert into product values('P3','mouse',100,80);

1 row created.

SQL> insert into product values('P4','floppies',250,150);

1 row created.

SQL> insert into product values('P5','keyboard',250,200);

1 row created.

SQL> insert into order1 values('C1','P1',20,200,'in process');

1 row created.

SQL> insert into order1 values('C2','P3',30,3000,'filled');

1 row created.

SQL> insert into order1 values('C3','P4',40,10000,'cancelled');

1 row created.

SQL> insert into order1 values('C5','P2',5,40,'filled');

1 row created.

SQL> insert into order1 values('C5','P3',50,5000,'in process');

1 row created.

SQL> insert into order1 values('C3','P5',100,25000,'filled');

1 row created.

SQL> insert into order1 values('C1','P2',10,80,'filled');

1 row created.

SQL> insert into order1 values('C1','P3',200,20000,'filled');

1 row created.

SQL> select product_no,descr from product
  2 where product_no not in(select product_no from order1);

PRODUCT_NO DESCR
---------- ----------
P5   keyboard

SQL> select name from client
  2 where client_no in(select client_no from order1 where(quant*rate)>=10000);

NAME
----
sumit
harman

SQL> select client_no,name from client where client_no in(select client_nofrom order1 where
product_no in(select product_no from product where descr='floppies'));

CLIENT_NO NAME
---------- ----------
C3   harman

SQL> select name from client where client_noin(select client_nofrom (select client_no,sum(quant) from
order1 group by client_no having sum(quant) > (select sum(quant) from order1 where client_no='C5')));

NAME
----
yagya
harman

SQL> select product_no from (select product_no,avg(quant) from order1 group by product_no having
avg(quant)=(select max(e) from (select avg(quant) as e from order1 group by product_no)));

PRODUCT_NO

P5

SQL> select name from client where balance>any(select balance from(select balance, client_no from client where city='delhi'));

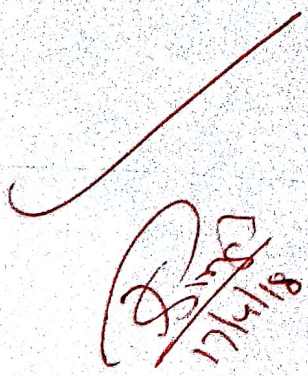
NAME
---
sumit
harman
saurav
shreyansh

SQL> select name from client where client_noin (select distinct client_no fromorder1);

NAME
---
yagya
sumit
harman
shreyansh

SQL> select name from client where client_no in(select client_no from(select count(product_no),client_no from order1 group by client_no having count(product_no)=1));

NAME
---
sumit



Page No.
Date

Experiment 7

- **Aim** : To implement various operations on nested queries.

- **Theory** : A subquery or inner query is a query within another SQL query & embedded within where clause. A subquery is used to return the data that will be used in the main query as a condition to further restrict the data to be retrieved. Subqueries can be used with the SELECT, INSERT, UPDATE & DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN etc.

1. Subqueries with the 'select' statement

Example –

```
Select column_name from customers where ID in
(select ID from customers where salary > 4500);
```

2. Subqueries with Insert statement –

Subqueries can also be used with INSERT statements. The insert statements uses the data returned from in the subquery can be modified with any of the character, date or number functions.

Page No.
Date

Example - Insert into customers_bkp
select * from customers where
ID in (select ID from customers);

3. Subqueries with Update Statement -
The Subquery can be used in conjunction with update statement.
Example - Update customers set Salary = salary * 0.25 where age ~~= 27~~); in (select age from customer_bkp where age >= 27);

4. Subqueries with Delete Statement
The Subquery can be used in conjunction with delete statement.
Example - Delete from customers where age in
Select age from customers_bkp where age >= 27);

```
SQL> create table cust(cid int primary key,fn varchar(20) NOT NULL,ln varchar(20) NOT NULL,addr
varchar(20),city varchar(20));
Table created.
SQL> insert into cust values(1,'Vishal','Dubey','Mahipalpur','Delhi');
1 row created.
SQL> insert into cust values(2,'udit','Dahima','Sangam Vihar','Delhi');
1 row created.
SQL> insert into cust values(3,'Bhavya','Takkar','Dwarka','Delhi');
1 row created.
SQL> insert into cust values(4,'Sudhanshu','Goel','Dwarka','Delhi');
1 row created.
SQL> insert into cust values(5,'Parv','Bharti','Uttam Nagar','Delhi');
1 row created.
SQL> select * from cust;
       CID FN                   LN                   ADDR
---------- -------------------- -------------------- --------------------
CITY
--------------------
         1 Vishal               Dubey                Mahipalpur
Delhi

         2 udit                 Dahima               Sangam Vihar
Delhi

         3 Bhavya               Takkar               Dwarka
Delhi


       CID FN                   LN                   ADDR
---------- -------------------- -------------------- --------------------
CITY
--------------------
         4 Sudhanshu            Goel                 Dwarka
Delhi

         5 Parv                 Bharti               Uttam Nagar
Delhi
SQL> create table ord(oid int primary key,odate date,amount int,cid int);
Table created.
SQL> insert into ord values(20,DATE'2017-10-08',2000,1);
1 row created.
SQL> insert into ord values(21,DATE'2017-11-09',3000,1);
1 row created.
SQL> insert into ord values(22,DATE'2017-05-29',3000,3);
1 row created.
SQL> insert into ord values(23,DATE'2017-03-18',4000,4);
1 row created.
SQL> insert into ord values(24,DATE'2017-11-23',5000,2);
```

```
1 row created.
SQL> select * from ord;

       OID ODATE     AMOUNT        CID
---------- -------- ---------- ----------
        20 08-OCT-17       2000          1
        21 09-NOV-17       3000          1
        22 29-MAY-17       3000          3
        23 18-MAR-17       4000          4
        24 23-NOV-17       5000          2
SQL> set line 1000;
SQL> select * from cust;

       CID FN                   LN                   ADDR                 CITY
---------- -------------------- -------------------- -------------------- --------------------
         1 Vishal               Dubey                Mahipalpur           Delhi
         2 udit                 Dahima               Sangam Vihar         Delhi
         3 Bhavya               Takkar               Dwarka               Delhi
         4 Sudhanshu            Goel                 Dwarka               Delhi
         5 Parv                 Bharti               Uttam Nagar          Delhi

SQL> select * from ord;

       OID ODATE     AMOUNT        CID
---------- -------- ---------- ----------
        20 08-OCT-17       2000          1
        21 09-NOV-17       3000          1
        22 29-MAY-17       3000          3
        23 18-MAR-17       4000          4
        24 23-NOV-17       5000          2
SQL> select a.fn,b.amount from cust a inner join ord b on a.cid=b.cid;
FN                   AMOUNT
-------------------- ----------
Vishal                     2000
Vishal                     3000
Bhavya                     3000
Sudhanshu                  4000
udit                       5000
SQL> select a.fn,a.ln,a.city,b.odate,b.amount from cust a inner join ord b on a.cid=b.cid;
FN                   LN                   CITY                 ODATE     AMOUNT
-------------------- -------------------- -------------------- --------- ----------
Vishal               Dubey                Delhi                08-OCT-17       2000
Vishal               Dubey                Delhi                09-NOV-17       3000
Bhavya               Takkar               Delhi                29-MAY-17       3000
Sudhanshu            Goel                 Delhi                18-MAR-17       4000
udit                 Dahima               Delhi                23-NOV-17       5000
SQL>  select a.fn,a.ln,a.city,b.odate,b.amount from cust a left join ord b on a.cid=b.cid;
FN                   LN                   CITY                 ODATE     AMOUNT
-------------------- -------------------- -------------------- --------- ----------
Vishal               Dubey                Delhi                08-OCT-17       2000
Vishal               Dubey                Delhi                09-NOV-17       3000
Bhavya               Takkar               Delhi                29-MAY-17       3000
Sudhanshu            Goel                 Delhi                18-MAR-17       4000
udit                 Dahima               Delhi                23-NOV-17       5000
Parv                 Bharti               Delhi
6 rows selected.
SQL> select a.fn,a.ln,a.city,b.odate,b.amount from cust a right join ord b on a.cid=b.cid;
FN                   LN                   CITY                 ODATE     AMOUNT
-------------------- -------------------- -------------------- --------- ----------
Vishal               Dubey                Delhi                08-OCT-17       2000
Vishal               Dubey                Delhi                09-NOV-17       3000
Bhavya               Takkar               Delhi                29-MAY-17       3000
Sudhanshu            Goel                 Delhi                18-MAR-17       4000
udit                 Dahima               Delhi                23-NOV-17       5000
Parv                 Bharti               Delhi
6 rows selected.
SQL> select a.fn,a.city,b.odate,b.amount from cust a full join ord b on a.cid=b.cid;
FN                   CITY                 ODATE     AMOUNT
-------------------- -------------------- --------- ----------
Vishal               Delhi                08-OCT-17       2000
Vishal               Delhi                09-NOV-17       3000
Bhavya               Delhi                29-MAY-17       3000
Sudhanshu            Delhi                18-MAR-17       4000
udit                 Delhi                23-NOV-17       5000
Parv                 Delhi
6 rows selected.
SQL> create table emp(eid int primary key,ename varchar(20),DOJ date,sid int,foreign key(sid) refere
nces emp(eid));

Table created.

SQL> insert into emp values(1,'Bob',DATE'2018-11-12',1);

1 row created.

SQL> insert into emp values(2,'Scott',DATE'2018-12-12',1);

1 row created.

SQL> insert into emp values(3,'Tiger',DATE'2018-12-13',1);

1 row created.

SQL> insert into emp values(4,'John',DATE'2018-12-14',2);
```

1 row created.

SQL> insert into emp values(5,'Sam',DATE'2018-12-20',3);

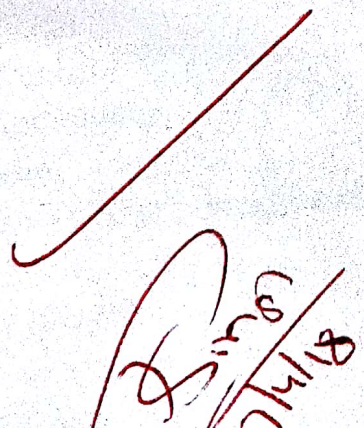
1 row created.

SQL> select * from emp;

EID	ENAME	DOJ	SID
1	Bob	12-NOV-18	1
2	Scott	12-DEC-18	1
3	Tiger	13-DEC-18	1
4	John	14-DEC-18	2
5	Sam	20-DEC-18	3

SQL> select a.eid as empid,a.ename as empname,b.eid as supid,b.ename as supname from
2 emp a ,emp b where a.sid=b.eid;

EMPID	EMPNAME	SUPID	SUPNAME
3	Tiger	1	Bob
2	Scott	1	Bob
1	Bob	1	Bob
4	John	2	Scott
5	Sam	3	Tiger



Page No.
Date

Experiment 8

- **Aim** : To implement a programme in PL/SQL.
- **Theory** : PL/SQL is a combination of SQL along with the procedural features of programming languages. It was developed by Oracle in 90's to enhance the qualities of SQL. PL/SQL is one of the 3 key programming languages embedded; Oracle database with SQL.

Declare : This section starts with the keyword 'declare'. It is an optional section & defines all variables, cursors, subprograms & other elements.

Executable Commands : This section is declared blw the keyword BEGIN and END & it is a mandatory section. It consists of executable PL/SQL statements of programs.

Initializing Variable in PL/SQL : Whenever we declare a variable, PL/SQL assign a fruit value of NULL. If we want to initialize variables with value & other known NULL, we do so by declaration using either of the following -

- Default Keyword
- Assignment Keywords.

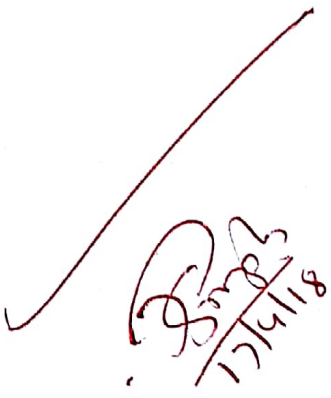
```
SQL> insert into emp values('Vishal',1,1000);
1 row created.
SQL> insert into emp values('Bhavya',2,200);
1 row created.
SQL> insert into emp values('Udit',3,50);
1 row created.
SQL> select * from emp;
NAME                 ID    SALARY
-------------------- --------- ---------
Vishal                1    1000
Bhavya                2     200
Udit                  3      50

SQL>  declare
  2   cursor c1 is select name,id from emp;
  3   v_emp emp.name %type;
  4   v_eid emp.id %type;
  5   begin
  6   open c1;
  7   loop
  8    fetch c1 into v_emp,v_eid;
  9   exit when c1 %notfound;
 10   DBMS_output.put_line(v_eid||v_emp);
 11    end loop;
 12   close c1;
 13   end;
 14   /
1 Vishal
2 Bhavya
3 Udit

SQL>  declare
  2  num integer;
  3   cursor c1 is select name,id from emp where salary = &num;
  4   v_emp emp.name %type;
  5   v_eid emp.id %type;
  6   begin
  7   open c1;
  8   loop
  9   fetch c1 into v_emp,v_eid;
 10   exit when c1 %notfound;
 11   DBMS_output.put_line(v_eid||v_emp);
 12    end loop;
 13   close c1;
 14   end;
 15   /

Enter value for num: 50
old   3:  cursor c1 is select name,id from emp where salary = &num;
new   3:  cursor c1 is select name,id from emp where salary = 50;
3 Udit

PL/SQL procedure successfully completed.
```



Page No.
Date

Experiment - 9

- **Aim**: To implement cursors using PL/SQL.
- **Theory**: Cursors are pointer to the context area. PL/SQL controls the context area through a cursor. A cursor holds the rows returned by a SQL statement. The set rows of the cursor holds referred to as active sets.
- **Types of Cursors**: 1. Implicity Cursor
  2. Explicity Cursor

FOUND : Returns to be true if a insert, update or delete statement affected rows.
NOT FOUND : It is logically opposity of % FOUND.
ROW COUNT : Returns the no. of rows affected by insert, delete or update statement.

Explicit Cursor
Declaring the cursor : Defines the cursor with a name & associated select statement.
Opening the Cursor : It allocates memory for the cursor & moves it to fetch rows.

Fetching cursor : It involves accessing a row at a time.

```
SQL> set serveroutput on;
SQL> declare
  2  e integer;
  3  i integer:=10;
  4  j integer:=10;
  5  c integer;
  6  begin
  7  c:=i+j;
  8  DBMS_output.put_line(c);
  9  end;
 10  /
 20
```

```
SQL> declare
  2  e integer;
  3  begin
  4  for e in 1..20 loop
  5  if remainder(e,2)=0
  6  then
  7  DBMS_output.put_line(e||'is even');
  8  else
  9  DBMS_output.put_line(e||'is odd');
 10  end if;
 11  end loop;
 12  end;
 13  /
1is odd
2is even
3is odd
4is even
5is odd
6is even
7is odd
8is even
9is odd
10is even
11is odd
12is even
13is odd
14is even
15is odd
16is even
17is odd
18is even
19is odd
20is even
```