

Experiment-1**Date:****ER Diagram for Sailors Database**

The goal of the "BoatClub" database is to enable members of a boat club to reserve boats for trips lasting several hours.

The two major entities are:

- Sailors—members of the boat club who reserve boats; and
- Boats—boats in the club's inventory.

In this problem we need to know what boats are reserved by what sailors on a given day. Thus, "reservation" is obviously an important relationship in this simple problem.

Attributes of the Sailor Entity

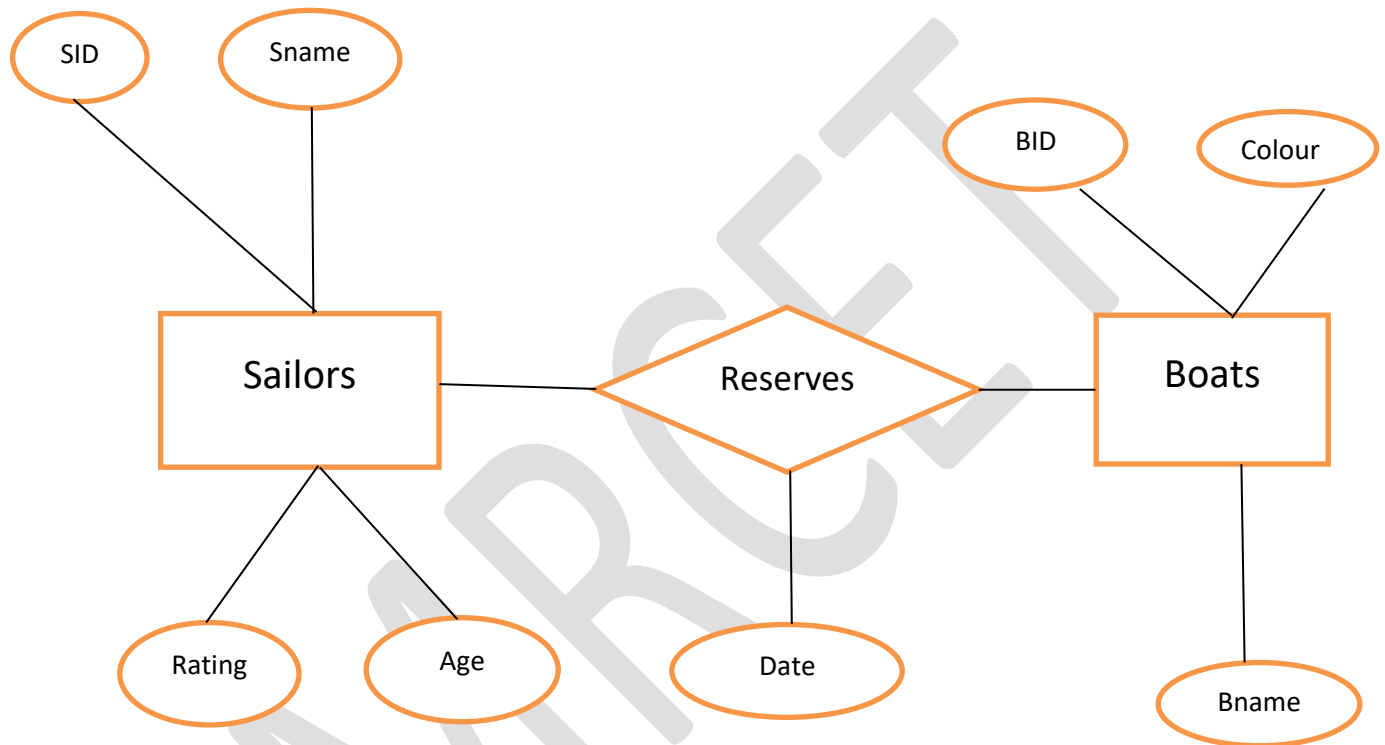
Attribute	Description
SID	- A sailor—each sailor is assigned a unique ID
name	- The sailor's name
rating	- The sailor's rating, ranging from 1 (low) to 10 (high)
age	- The sailor's age

Attributes of the Boat Entity:

Attribute	Description
BID	- A boat ID—each boat is assigned a unique ID (painted on the bow)
name	- The name of the boat (also painted on the bow)
color	- The color of the boat

A sailor can make many reservations (*) but a reservation involves only a single sailor. Similarly, a boat can be allocated to many reservations, but only one boat is allocated to a particular reservation.

Q1. Draw an ER diagram that captures the above information.



Q2. Convert above ER diagram in to relations (tables)

Sailors:

SID	SNAME	RATING	AGE

Boats:

BID	BNAME	COLOUR
-----	-------	--------

Q3. Write SQL statements to create above relations (tables).

CREATE TABLE SAILORS (SID NUMBER(3) PRIMARY KEY, SNAME VARCHAR2(20), AGE NUMBER(3), RATING NUMBER(2));

CREATE TABLE BOATS (BID NUMBER(3) PRIMARY KEY, BNAME VARCHAR2(20), BCOLOR VARCHAR(10));

CREATE TABLE RESERVES (SID NUMBER(3), BID NUMBER(3), DAY DATE, FOREIGN KEY SID REFERENCES SAILORS(SID), FOREIGN KEY BID REFERENCES BOATS(BID));

Output:

Table created.

Table created.

Table created.

Q4. Insert the following data into above created tables.

Sailors			
Sid	Sname	Rating	Age
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35
64	Horatio	7	35
71	Zorba	10	16
74	Horatio	9	35
85	Art	3	25.5
95	Bob	3	63.5

Boats		
Bid	Bname	Color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Reserves		
SID	BID	Day
22	101	10-10-1998
22	102	10-10-1998
22	103	10-08-1998
22	104	10-07-1998
31	102	10-11-1998
31	103	06-11-1998
31	104	11-12-1998
64	101	09-05-1998
64	102	09-08-1998
74	103	09-08-1998

Sailors Data:

INSERT INTO SAILORS VALUES (22,'Dustin',7,45);

1 row(s) inserted.

INSERT INTO SAILORS VALUES (29,'Brutus',1,33);

1 row(s) inserted.

INSERT INTO SAILORS VALUES (31,'Lubber',8,55.5);

1 row(s) inserted.

```
INSERT INTO SAILORS VALUES (32,'Andy',8,25.5);
```

```
1 row(s) inserted.
```

```
INSERT INTO SAILORS VALUES (58,'Rusty',10,35);
```

```
1 row(s) inserted.
```

```
INSERT INTO SAILORS VALUES (64,'Horatio',7,35);
```

```
1 row(s) inserted.
```

```
INSERT INTO SAILORS VALUES (71,'Zorba',10,16);
```

```
1 row(s) inserted.
```

```
INSERT INTO SAILORS VALUES (74,'Horatio',9,35);
```

```
1 row(s) inserted.
```

```
INSERT INTO SAILORS VALUES (85,'Art',3,25.5);
```

```
1 row(s) inserted.
```

```
INSERT INTO SAILORS VALUES (95,'Bob',3,63.5);
```

```
1 row(s) inserted.
```

Boats Data:

```
Insert into Boats values (101,'Interlake','blue');
```

```
1 row(s) inserted.
```

```
Insert into Boats values (102,'Interlake','red');
```

```
1 row(s) inserted.
```

```
Insert into Boats values (103,'Clipper','green');
```

```
1 row(s) inserted.
```

```
Insert into Boats values (104,'Marine','red');
```

```
1 row(s) inserted.
```

Reserves Data:

```
insert into Reserves values(22,101,'1998-10-10');
```

```
1 row(s) inserted.
```

```
insert into Reserves values(22,102,'1998-10-10');
```

```
1 row(s) inserted.
```

```
insert into Reserves values(22,103,'1998-08-10');
```

```
1 row(s) inserted.
```

```
insert into Reserves values(22,104,'1998-07-10');
```

```
1 row(s) inserted.
```

```
insert into Reserves values(31,102,'1998-11-10');
```

```
1 row(s) inserted.
insert into Reserves values(31,103,'1998-11-10');
```

```
1 row(s) inserted.
insert into Reserves values(31,104,'1998-12-11');
```

```
1 row(s) inserted.
insert into Reserves values(64,101,'1998-05-09');
```

```
1 row(s) inserted.
insert into Reserves values(64,102,'1998-08-09');
```

```
1 row(s) inserted.
SELECT * FROM SAILORS;
```

```
SELECT * FROM BOATS;
```

```
SELECT * FROM RESERVES;
```

Output:

SID	SNAME	RATING	AGE
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	56
32	Andy	8	26
58	Rusty	10	35
64	Horatio	7	35
71	Zorba	10	16
74	Horatio	9	35
85	Art	3	26
95	Bob	3	64

10 rows returned in 0.00 seconds

SID	BID	DAY
22	101	10-OCT-98
22	102	10-OCT-98
22	103	10-AUG-98
22	104	10-JUL-98
31	102	10-NOV-98
31	103	06-NOV-98
31	104	11-DEC-98
64	102	09-AUG-98
74	103	09-MAY-98
64	101	09-MAY-98

10 rows returned in 0.00 seconds

BID	BNAME	BCOLOR
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

4 rows returned in 0.00 seconds

Q5. Write the following Queries in SQL.

5.1 Find the names and ages of all sailors.

SELECT SNAME, AGE FROM SAILORS;

Output:

SNAME	AGE
Dustin	45
Brutus	33
Lubber	56
Andy	26
Rusty	35
Horatio	35
Zorba	16
Horatio	35
Art	26
Bob	64

10 rows returned in 0.00 seconds

5.2 Find all sailors with a rating above 7.

SELECT * FROM SAILORS WHERE RATING > 7;

Output:

SID	SNAME	RATING	AGE
31	Lubber	8	56
32	Andy	8	26
58	Rusty	10	35
71	Zorba	10	16
74	Horatio	9	35

5 rows returned in 0.00 seconds

5.3 Find the names of sailors who have reserved boat number 103.

SELECT SNAME FROM SAILORS , RESERVES WHERE SAILORS.SID = RESERVES.SID AND RESERVES.BID = 103;

Output:

SNAME
Dustin
Lubber
Horatio

3 rows returned i

5.4 Find the sids of sailors who have reserved a red boat.

SELECT S.SID FROM SAILORS S,RESERVES R,BOATS B WHERE S.SID = R.SID AND B.BID = R.BID AND B.BCOLOR = 'red';

Output:

SID
22
22
31
31
64

5.5 Find the names of sailors who have reserved a red boat.

SELECT S.SNAME FROM SAILORS S,RESERVES R,BOATS B WHERE S.SID = R.SID AND B.BID = R.BID AND B.BCOLOR = 'red';

Output:

SNAME
Dustin
Dustin
Lubber
Lubber
Horatio

5.6. Find the colors of boats reserved by Lubber.

```
SELECT B.BCOLOR FROM BOATS B,RESERVES R,SAILORS S WHERE S.SID = R.SID AND R.BID = B.BID AND S.SNAME = 'Lubber';
```

Output:

BCOLOR
red
green
red

5.7.Find the names of sailors who have reserved at least one boat.

```
SELECT S.SNAME FROM SAILORS S,RESERVES R,BOATS B WHERE S.SID = R.SID AND B.BID = R.BID;
```

Output:

SNAME
Dustin
Dustin
Dustin
Dustin
Lubber
Lubber
Lubber
Horatio
Horatio
Horatio

10 rows returned in 0.00 seconds

5.8 Compute increments for the ratings of persons who have sailed two different boats on the same day.

```
SELECT S.SNAME,S.RATING+1 AS RATING FROM SAILORS S, RESERVES R1,RESERVES R2 WHERE
```


S.SID = R1.SID AND R1.SID = R2.SID AND R1.BID!=R2.BID AND R1.DAY = R2.DAY;

Output:

SNAME	RATING
Dustin	7
Dustin	7

2 rows returned in 0.00 seconds

5.9.Find the ages of sailors whose names begins and ends with the B and has at least three characters.

SELECT * FROM SAILORS WHERE SNAME LIKE 'B_%b';

Output:

SID	SNAME	RATING	AGE
95	Bob	3	64

1 rows returned in 0.00 seconds

Experiment-2**Date :**

Creation, Altering and Dropping of Tables and Inserting Rows into a Table (Use Constraints While Creating Tables) Examples Using Select Command.

Examples Using Select Command.

Q.1: Login into Oracle data base using **system** user and create a user with your **Roll no** and grant permission to create the tables in SQL.

Ans: Create user <username> identified by <password>
Grant dba to <username>

Q.2 Login into oracle data base using your roll no and create the following tables.

1. Sailors (sid:integer)
2. Boats (bid:integer);
3. Reserves (sid:integer,bid:integer,day:date);

Ans:

```
create table Sailors_demo(sid int(10));  
  
create table Boats_demo(bid int(10));  
  
create table Reserves_demo(sid int(10),bid int(10),day date);
```

Write the following DQL statements.

1. Add new column sailor name and rating to sailors table.

```
ALTER TABLE SAILORS_DEMO ADD(SNAME VARCHAR2(20),RATING NUMBER(2));
```

Output:

Table Altered.

2. Add new column boat name and color to Boats table.

```
ALTER TABLE BOATS_DEMO ADD (BNAME VARCHAR2(20),BCOLOR VARCHAR(10));
```

Output:

Table Altered.

3. Add a primary key to sailors table after table creation

```
ALTER TABLE SAILORS_DEMO ADD CONSTRAINT SID_PK PRIMARY KEY(SID);
```

Output:

Table Altered.

4. Add a primary key to Boats table after table creation

```
ALTER TABLE BOATS_DEMO ADD CONSTRAINT BID_PK PRIMARY KEY(BID);
```

Output:

Table Altered.

5. Remove a Primary Key from Sailors table

```
ALTER TABLE SAILORS_DEMO DROP CONSTRAINT SID_PK;
```

Output:

Table dropped.

6. Remove a Primary Key from Boats table

```
ALTER TABLE BOATS_DEMO DROP CONSTRAINT BID_PK;
```

Output:

Table dropped.

7. Add a not null constrain on Sailors, Boats Table

```
ALTER TABLE SAILORS_DEMO MODIFY SNAME VARCHAR(20) NOT NULL;
```

```
ALTER TABLE BOATS_DEMO MODIFY BNAME VARCHAR(20) NOT NULL;
```

Output:

Table Altered.

8. Drop not null constraints from Sailors, Boats Table.

```
ALTER TABLE SAILORS_DEMO MODIFY SNAME VARCHAR(20) NULL;
```

```
ALTER TABLE BOATS_DEMO MODIFY BNAME VARCHAR2(20) NULL;
```

Output:

Table Altered.

9. Add a constraint check to the rating column

```
ALTER TABLE SAILORS ADD CONSTRAINT CHECK_RATING CHECK(RATING>=1 AND RATING <=10);
```

Output:

Table Altered.

10. Drop the above check constraint from rating column.

```
ALTER TABLE SAILORS DROP CONSTRAINT CHECK_RATING;
```

Output:

Table Dropped.

11. Add a primary key constraint on Sailors, Boats tables.

```
ALTER TABLE SAILORS_DEMO ADD CONSTRAINT SID_PK PRIMARY KEY(SID);
```

```
ALTER TABLE BOATS_DEMO ADD CONSTRAINT BID_PK PRIMARY KEY(BID);
```

Output:

Table Altered.

12. Drop primary key constraints from sailors ,Boats tables.

```
ALTER TABLE SAILORS_DEMO DROP CONSTRAINT SID_PK;
```

```
ALTER TABLE BOATS_DEMO DROP CONSTRAINT BID_PK;
```

Output:

Table dropped

13. Add foreign key constraints to reserves table.

```
ALTER TABLE RESERVES_DEMO ADD CONSTRAINT SID_FK FOREIGN KEY(SID) REFERENCES  
SAILORS_DEMO(SID);
```

```
ALTER TABLE RESERVES_DEMO ADD CONSTRAINT BID_FK FOREIGN KEY(BID) REFERENCES  
BOATS_DEMO(BID);
```

Output:

Table Altered.

14. Drop foreign key constraints from reserve table

```
ALTER TABLE RESERVES_DEMO DROP CONSTRAINT SID_FK;
```

```
ALTER TABLE RESERVES_DEMO DROP CONSTRAINT BID_FK;
```

Output:

Table dropped.

Experiment 3**Date :**

Queries (along with sub Queries) using ANY, ALL, IN, EXISTS, NOTEXISTS, UNION, INTERSET, Constraints

Q1.Find the Sid's of sailors who have reserved a red or a green boat.

```
SELECT S.SID FROM SAILORS S,RESERVES R,BOATS B WHERE S.SID = R.SID AND R.BID = B.BID AND  
B.BCOLOR = 'red'
```

UNION

```
SELECT S1.SID FROM SAILORS S1,RESERVES R1,BOATS B1 WHERE S1.SID = R1.SID AND R1.BID = B1.BID  
AND B1.BCOLOR = 'green';
```

Output:

SID
22
31
64
74

4 rows returned in 0.00 seconds

Q2. Find the names of sailors who have reserved a red and a green boat.

```
SELECT S.SNAME, S.SID FROM SAILORS S,RESERVES R,BOATS B WHERE S.SID = R.SID AND R.BID = B.BID  
AND B.BCOLOR = 'red'
```

UNION

```
SELECT S1.SNAME, S1.SID FROM SAILORS S1,RESERVES R1,BOATS B1 WHERE S1.SID = R1.SID AND R1.BID  
= B1.BID AND B1.BCOLOR = 'green';
```

Output:

SNAME	SID
Dustin	22
Horatio	64
Horatio	74
Lubber	31

4 rows returned in 0.00 seconds

Q3. Find the names of sailors who have reserved a red but not green boats.

SELECT S.SNAME,S.SID FROM SAILORS S,RESERVES R,BOATS B WHERE S.SID = R.SID AND R.BID = B.BID
AND B.BCOLOR = 'red'

MINUS

SELECT S1.SNAME,S1.SID FROM SAILORS S1,RESERVES R1,BOATS B1 WHERE S1.SID = R1.SID AND R1.BID
= B1.BID AND B1.BCOLOR = 'green';

Output:

SNAME	SID
Horatio	64

1 rows returned in 0.00 seconds

Q4.Find all sids of sailors who have a rating of 10 or reserved boat 104.

SELECT S.SID FROM SAILORS S,RESERVES R,BOATS B WHERE RATING = 10

UNION

SELECT S1.SID FROM SAILORS S1,RESERVES R1,BOATS B1 WHERE S1.SID=R1.SID AND R1.BID=B1.BID AND
R1.BID = 104;

Output:

SID
22
31
58
71

4 rows returned in 0.00 seconds

Q5.Find the names of sailors who have reserved boat 103 using independent nested query.

SELECT S.SNAME FROM SAILORS S WHERE S.SID IN (SELECT R.SID FROM RESERVES R WHERE R.BID = 103);

Output:

SNAME
Dustin
Lubber
Horatio

3 rows returned in 0.00 seconds

Q6.Find the names of sailors who have reserved a red boat.

SELECT S.SNAME FROM SAILORS S WHERE S.SID IN (SELECT R.SID FROM RESERVES R WHERE R.BID IN (SELECT B.BID FROM BOATS B WHERE B.BCOLOR = 'red'));

SNAME
Dustin
Lubber
Horatio

3 rows returned in 0.00 seconds

Q7.Find the names of sailors who have not reserved a red boat.

SELECT S.SNAME FROM SAILORS S WHERE S.SID NOT IN (SELECT R.SID FROM RESERVES R WHERE R.BID NOT IN (SELECT B.BID FROM BOATS B WHERE B.BCOLOR = 'red'));

Output:

SNAME
Brutus
Andy
Rusty
Zorba
Art
Bob

6 rows returned in 0.00 seconds

Q8.Find the names of sailors who have reserved boat number 103 using correlated nested query.

SELECT S.SNAME FROM SAILORS S WHERE EXISTS(SELECT R.SID FROM RESERVES R WHERE R.SID = S.SID AND R.BID = 103);

Output:

SNAME
Dustin
Lubber
Horatio

3 rows returned in 0.00 seconds

Q9.Find sailors whose rating is better than some sailor called 'Horatio'.

```
SELECT * FROM SAILORS S WHERE S.RATING > ANY(SELECT S1.RATING FROM SAILORS S1 WHERE S1.SNAME = 'Horatio');
```

Output:

SID	SNAME	RATING	AGE
31	Lubber	8	56
32	Andy	8	26
58	Rusty	10	35
71	Zorba	10	16
74	Horatio	9	35

5 rows returned in 0.00 seconds

Q10.Find the sailors with the highest rating.

```
SELECT * FROM SAILORS WHERE RATING IN(SELECT MAX(RATING) FROM SAILORS);
```

Output:

SID	SNAME	RATING	AGE
58	Rusty	10	35
71	Zorba	10	16

2 rows returned in 0.00 seconds

Q11.Find the names of sailors who have reserved both a red and a green boat using nested queries.

```
SELECT S.SNAME FROM SAILORS S WHERE S.SID IN (SELECT S.SID FROM SAILORS S,RESERVES R,BOATS B WHERE S.SID=R.SID AND R.BID = B.BID AND B.BCOLOR = 'RED') INTERSECT SELECT S.SNAME FROM SAILORS S WHERE S.SID IN (SELECT S.SID FROM SAILORS S,RESERVES R,BOATS B WHERE S.SID=R.SID AND R.BID = B.BID AND B.BCOLOR = 'GREEN');
```

Output:

SNAME
Dustin
Horatio
Lubber

3 rows returned in 0.00 seconds

Q12.Find the names of sailors who have reserved all boats.

```
SELECT S.SNAME FROM SAILORS S WHERE NOT EXISTS ( SELECT B.BID FROM BOATS B WHERE NOT EXISTS
```

```
( SELECT R.BID FROM Reserves R WHERE R.BID = B.BID AND R.SID = S.SID));
```

Output:

SNAME
Dustin

1 rows returned in 0.02 seconds

Experiment 4**Date :**

Queries using Aggregate Functions (COUNT, SUM, AVG, MAX and MIN), GROUP BY, HAVING and Creation and Dropping of Views

Q1.Find the average age of all sailors.

```
SELECT AVG(AGE) FROM SAILORS;
```

Output:

AVG(AGE)
37.1

1 rows returned in 0.00 seconds

Q2.Find the average age of sailors with a rating of 10.

```
SELECT AVG(AGE) FROM SAILORS WHERE RATING = 10;
```

Output:

AVG(AGE)
25.5

1 rows returned in 0.00 seconds

Q3.Find the name and age of the oldest sailor.

```
SELECT S.SNAME ,S.AGE FROM SAILORS S WHERE AGE=(SELECT MAX(AGE) FROM SAILORS);
```

Output:

SNAME	AGE
Bob	64

1 rows returned in 0.00 seconds

Q4.Count the number of sailors.

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

Output:

COUNT(*)
10

1 rows returned in 0.00 seconds

Q5.Count the names of different sailor names.

SELECT COUNT(DISTINCT SNAME) FROM SAILORS;

Output:

COUNT(DISTINCT SNAME)
9

1 rows returned in 0.00 seconds

Q6.Find the names of sailors who are older than the oldest sailor with a rating of 10.

SELECT S.SNAME FROM SAILORS S WHERE S.AGE>(SELECT MAX(S1.AGE) FROM SAILORS S1 WHERE S1.RATING = 10);

Output:

SNAME
Dustin
Lubber
Bob

3 rows returned in 0.00 seconds

Q7.Find the age of the youngest sailors for each rating level.

SELECT RATING ,MIN(AGE) FROM SAILORS GROUP BY RATING;

Output:

RATING	MIN(AGE)
1	33
8	26
7	35
3	26
10	16
9	35

6 rows returned in 0.00 seconds

Q8.Find the age of the youngest sailor who is eligible to vote for each rating level with at least two such sailors.

```
SELECT MIN(AGE) FROM SAILORS WHERE AGE>=18 GROUP BY RATING HAVING COUNT(*)>1;
```

Output:

MIN(AGE)
26
35
26

3 rows returned in 0.00 seconds

Q9.For each red boat, find the number of reservations for this boat.

```
SELECT R.BID,COUNT(*) FROM RESERVES R , BOATS B WHERE R.BID = B.BID AND B.BCOLOR = 'red'
GROUP BY R.BID;
```

Output:

BID	COUNT(*)
102	3
104	2

2 rows returned in 0.00 seconds

Q10.Find the average age of sailors for each rating level that has at least two sailors.

```
SELECT RATING ,AVG(AGE) FROM SAILORS GROUP BY RATING HAVING COUNT(*)>1;
```

Output:

RATING	AVG(AGE)
8	41
7	40
3	45
10	25.5

4 rows returned in 0.00 seconds

Q11. Find the average age of sailors who are of voting age for each rating level that has at least two sailors.

```
SELECT RATING ,AVG(AGE) FROM SAILORS WHERE AGE>=18 GROUP BY RATING HAVING COUNT(*)>1;
```

Output:

RATING	AVG(AGE)
8	41
7	40
3	45

3 rows returned in 0.00 seconds

Q12. Find those ratings for which the average age of sailors is the minimum overall ratings.

```
SELECT RATING,SID,SNAME FROM SAILORS WHERE AGE >=(SELECT AVG(AGE) FROM SAILORS);
```

Output:

RATING	SID	SNAME
7	22	Dustin
8	31	Lubber
3	95	Bob

3 rows returned in 0.00 seconds

Q13. Define a view for finding sailors whose rating is above 7.

```
CREATE VIEW RATING_VIE AS SELECT SID,SNAME FROM SAILORS WHERE RATING > 7;
```

```
SELECT * FROM RATING_VIEW;
```

Output:

View created.

SID	SNAME
31	Lubber
32	Andy
58	Rusty
71	Zorba
74	Horatio

5 rows returned in 0.00 seconds

Q14.Insert some rows into above view.

```
INSERT INTO RATING_VIEW VALUES (66,'Mark');
```

```
INSERT INTO RATING_VIEW VALUES (67,'Zuck');
```

Output:

1 row(s) inserted.

1 row(s) inserted.

Q.15.Restrict updates on above view.

```
INSERT INTO RATING_VIEW VALUES(34,'RAM');
```

```
SELECT * FROM RATING_VIEW;
```

Output:

SID	SNAME
31	Lubber
32	Andy
58	Rusty
71	Zorba
74	Horatio

5 rows returned in 0.00 seconds

Q16.Drop the view created in Q13.

```
DROP RATING_VIEW;
```

Output:

View dropped.

CNARCET

Experiment 5**Date:**

Queries using Conversion Functions (to_char, to_number and to_date), String Functions (Concatenation, lpad, rpad, ltrim, rtrim, lower, upper, initcap, length, substr and instr), Date Functions (Sysdate, next_day, add_months, last_day, months_between, least, greatest, trunc, round, to_char, to_date)

Q1.Create the following table:

Table Name::Staff Record

Staff_ID	Name	DOB	Sex	Salary	Award	District	Department
1001	Jeffrey Lee	23/02/1978	M	28463.40	3	Tai Kok Tsui	Sales
1002	Hugo Cheung	08/04/1976	M	14598.50	2	Central	Sales
1003	Jennifer Wong	29/03/1978	F	39850.00	6	Tai Po	Sales
1004	Melinda Ma	28/08/1982	F	7783.00	6	Tai Po	Purchase
1005	Hilda Leung	24/10/1982	F	45670.50	2	Westren	Sales
1006	Nelly Tam	10/10/1973	F	4530.80	4	Shatin	Sales
1007	Mable Mee	30/08/1979	F	3549.40	1	Tai Kok Tsui	Purchase
1008	Barnabv Nge	12/05/1980	M	8327.30	5	Hunghom	Account
1009	Luaretta Tai	23/09/1982	F	32445.42	3	Tai Wai	Account
1010	Gregory tai	22/10/1972	M	35542.40	4	Tai Wo	Purchase

Write the following SQL queries

1. Write a SQL statement to produce a list of male staff only, showing their names in upper case, department in lower case and the number of characters in the department.

Ans:- select upper(Name),lower(Department),length(Department) from StaffRecord where Sex='M';

Output :

UPPER(NAME)	LOWER(DEPARTMENT)	LENGTH(DEPARTMENT)
JEFFREY LEE	Sales	5
HUGO CHEUNG	Sales	5
BARNABV NGE	Account	7

GREGORY TAI	Purchase	8
-------------	----------	---

2. Write a SQL statement to produce a list of all staff member with their names appended with first letter of their Department.
Ex Jeffrey Lee (S)

Ans:- select concat(concat(name,' '), concat(substr(Department,1,1),'')) from StaffRecord;

Output:

CONCAT(CONCAT(NAME,' '),CONCAT(SUBSTR(DEPARTMENT,1,1),''))
Jeffrey Lee(S)
Hugo Cheung(S)
Jennifer Wong(S)
Melinda Ma(P)
Hilda Leung(S)
Nelly Tam(S)
Mable Mee(P)
Barnabv Nge(A)
Luaretta Tai(A)
Gregory tai(P)

3. Write a SQL statement to print a list of first names of staff
Ex. Jeffrey

Ans:- select substr(name,1,instr(name,' ')) from StaffRecord;

Output:

SUBSTR(NAME,1,INSTR(NAME," "))
Jeffrey
Hugo
Jennifer
Melinda
Hilda
Nelly
Mable
Barnabv
Luaretta

Gregory

4. Write a SQL statement to print a list of districts that consists of a single word. The list should not consist of repeating items and is arranged in descending alphabetical order.
Ex Central

Ans:- select distinct district from StaffRecord where instr(district, ' ') = 0 order by district desc;

Output:

DISTRICT
Westren
Shatin
Hunghom
Central

5. Given that the bonus of staff is calculated by
 $\text{Bonus} = \text{SQRT}(\text{Salary} * \text{Award})$
 Write a SQL statement to print a list of salary and bonus for each staff.

Ans:- select name, salary, to_char(sqrt(salary*award), '\$999.99') as bonus from StaffRecord;

Output :

NAME	SALARY	BONUS
Jeffrey Lee	28463.4	\$292.22
Hugo Cheung	14598.5	\$170.87
Jennifer Wong	39850	\$488.98
Melinda Ma	7783	\$216.10
Hilda Leung	45670.5	\$302.23
Nelly Tam	4530.8	\$134.62
Mable Mee	3549.4	\$59.58
Barnabv Nge	8327.3	\$204.05
Luaretta Tai	32445.42	\$311.99
Gregory tai	35542.4	\$377.05

6. Write a SQL statement to show date of birth of staff in the following format

Name Day Month Week Year

Jeffrey Lee 23 February Thursday 1978

Ans:-select name, to_char(dob,'dd') as day, to_char(dob,'mm') as month ,to_char (dob,'yy') as year from staffrecord;

Output:

NAME	DAY	MONTH	YEAR
Jeffrey Lee	23	2	78
Hugo Cheung	8	4	76
Jennifer Wong	29	3	78
Melinda Ma	28	8	82
Hilda Leung	24	10	82
Nelly Tam	10	10	73
Mable Mee	30	8	79
Barnabv Nge	12	5	80
Luaretta Tai	23	9	82
Gregory tai	22	10	72

7. Write a SQL statement to show those staff born in the months between September and December. Display the dates 10 days before these dates of birth so that manager has enough time to prepare present for the staff. Arrange dates from nearest to furthest.

Ans:-select name ,dob-10 as ten days from staffrecord where to_char (dob,'mm') in (9,10,11,12) order by to_char (dob,'mm') ,to_char(dob,'dd');

Output:

NAME	TENDAYS
Luaretta Tai	13-Sep-82
Nelly Tam	30-Sep-73
Gregory tai	12-Oct-72
Hilda Leung	14-Oct-82

Experiment- 6**Date:****Working with Triggers using PL/SQL: Develop Programs using BEFORE and AFTER Triggers, Row and Statement Triggers and INSTEAD OF Triggers**

Q1:Write a trigger that checks the value of SAL before insert or update statement and ensures that SAL below 500 is not inserted.It acts BEFORE insertion or updation.

```
create table emp_trig(ename varchar2(20),sal number(10));  
  
create or replace trigger min_sal_check before insert or update on emp_trig  
for each row  
when (new.sal<500)  
begin  
raise_application_error(-2000,'Sal must be above 500');  
end;
```

Output:

Table created.

Trigger created.

Q2:Write a trigger that keeps backup of deleted records of emp_trig table. Deleted records of emp_trig are inserted in emp_backup table.

```
create table emp_bkup(ename varchar2(20),sal number(10),sys_date date);  
  
create or replace trigger bkup_rec after delete on emp_trig for  
each row  
begin  
insert into emp_bkup values(:old.ename,:old.sal,sysdate);  
  
end;
```

Output:

Table created.

Trigger created.

Q3:Write a trigger that checks for any duplicate value and disallows insertion.

```
create or replace trigger unq_val before insert on emp_trig
for each row
declare
vcnt number(2);
begin
select count(*) into vcnt from emp_trig where
ename=:new.ename;
if vcnt=1 then
raise_application_error(-20001,'you have entered duplicate ename');
end if;
end;
```

Output:

Trigger created.

Q4:Write a trigger that disallows any DML operations performed after 13.00Hrs.on emp table.

```
create or replace trigger restrict_access before insert or update or delete on emp_trig
begin
if to_char(sysdate,'HH24')>12 then
if inserting then
raise_application_error(-20002,'insertion not allowed after 13000hrs');
elsif updating then
raise_application_error(-20002,'updation not allowed after 13000hrs');
elsif deleting then
raise_application_error(-20002,'deletion not allowed after 13000hrs');
end if;
end if;
end
```

Output: Trigger created.

Q5:Write an INSTEAD OF trigger that disallows insertion and updation of SAL if same SAL is present in any record.

```
create view v1 as select * from emp_trig where sal<10000;

create or replace trigger m_sal_check instead of insert or update on v1
for each row
declare
vcnt number(2);
begin
select count(*) into vcnt from v1 where sal=:new.sal;
if vcnt=1 then
raise_application_error(-20000,'duplicate sal not allowed');
end if;
end;
```

Output:

View created.

Trigger created.

Experiment -7**Date :****Working with PL/SQL Procedures :Programs Development using Creation of Procedures, Passing Parameters IN and OUT of PROCEDURES**

```
CREATE TABLE EMP(ENO NUMBER(10),ENAME VARCHAR(20),SAL NUMBER(10),deptno NUMBER(10));
```

Output:

Table created.

```
INSERT INTO EMP VALUES(1001,'Lokesh',100000,2);
```

```
INSERT INTO EMP VALUES(1002,'Madhu ',50000,1);
```

```
INSERT INTO EMP VALUES(1003,'Sravan',55000,1);
```

```
INSERT INTO EMP VALUES(1004,'Sam',200000,3);
```

```
SELECT * FROM EMP;
```

Output:

4 row(s) inserted.

ENO	ENAME	SAL	DEPTNO
1003	Sravan	55000	1
1004	Sam	200000	3
1001	Lokesh	100000	2
1002	Madhu	50000	1

4 rows returned in 0.00 seconds

Q1: Create a procedure which displays employee name and salary for given employee number.

```
CREATE OR REPLACE PROCEDURE MYEMPNAME (EMPNO EMP.ENO%TYPE) IS
```

```
VNAME VARCHAR2 (10);
```

```
VSAL NUMBER (6);
```

```
BEGIN
```

```
SELECT ENAME, SAL INTO VNAME, VSAL FROM EMP WHERE ENO=EMPNO;
```

```
DBMS_OUTPUT.PUT_LINE (VNAME || ' ' || VSAL);
```


END;

Output:

Procedure created.

BEGIN

MYEMPNAME(1004);

END;

Output:

Sam 200000

Statement processed.

Q2: Create a procedure which displays employee salary for given employee number using out variable

Create or replace procedure empsalary(enum emp.eno%type,salary out emp.sal%type)

Is

Begin

Select sal into salary from emp where eno=enum;

Dbms_output.put_line(salary);

End;

Output:

Procedure created.

Invocation:

declare

I number;

begin

empsalary(1002,i);

end

Output:

150000

Statement processed.

Q3: Create a procedure which displays number of sailors for given rating

Create or replace procedure no_sailors(srating sailors.rating%type,cnt out number)

Is

Begin

Select count(*) into cnt from sailors where rating=srating;

End;

Output:

Procedure created.

Invocation:

declare

I number;

begin

no_sailors(10,i);

Dbms_output.put_line(i);

end

Output:

2

Statement processed.

Q4: Create a procedure which displays average age of sailors for given rating

Create or replace procedure avgage(srating sailors.rating%type,cnt out number)

Is

Begin

Select avg(age) into cnt from sailors where rating=srating;

End;

Output:

Procedure created.

Invocation:

```
declare  
  
l number;  
  
begin  
  
avgage(10,i);  
  
Dbms_output.put_line(i);  
  
End
```

Output:

```
25.5  
  
Statement processed.
```

Q5: Create a procedure which displays day of a week for given date

Create or replace procedure date_day(d date)

```
Is  
  
b varchar2(20);  
  
Begin  
  
b:=to_char(d,'day');  
  
dbms_output.put_line(b);  
  
End;
```

Output:

Procedure created.

Invocation:

```
Begin  
  
date_day(sysdate);  
  
end
```

Output:

```
wednesday  
  
Statement processed.
```

Experiment 8

Date :

Working with LOOPS using PL/SQL and Exception Handling: Program Development using WHILE LOOPS, Numeric FOR LOOPS, Nested Loops using ERROR Handling, BUILT-IN Exceptions, USE Defined Exceptions, RAISE- APPLICATION ERROR

Q1: Write a PL/SQL program using while loop to display dept number and dept name from dept table.

Declare

l varchar2(20) := Enter_dept_Number;

Vname varchar2(10);

Vdeptno varchar2(10);

Begin

dbms_output.put_line('Dept_name Dept_no');

While i < 106 loop

Select dname, deptno into vname, vdeptno from dept where deptno = l;

Dbms_output.put_line(' ' || vname || ' ' || vdeptno);

l := i + 1;

End loop;

End;

Output:

Dept_name Dept_no

CSE 10

ECE 20

EEE 30

MECH 40

CIVIL 50

Statement processed.

Q2: Write a PL/SQL program using for loop to update location of all department to 'Hyderabad' except department Whose dept no 20.

Begin

For i in (select deptno from dept) Loop

If i.deptno!=20 then

Update dept set loc ='Hyd' where deptno=i.deptno;

End if;

End loop;

End;

Output:

Statement processed.

select * from dept;

Output:

DEPTNO	DNAME	LOC
10	CSE	Hyd
20	ECE	-
30	EEE	Hyd
40	MECH	Hyd
50	CIVIL	Hyd

5 rows returned in 0.00 seconds

Q3. Write a PL/SQL program to print all prime numbers below 50

DECLARE

i number(3);

j number(3);

BEGIN

i := 2;

```
LOOP
j:= 2;
LOOP
exit WHEN ((mod(i, j) = 0) or (j = i));
j := j +1;
END LOOP;
IF (j = i ) THEN
dbms_output.put_line(i || ' is prime');
END IF;
i := i + 1;
exit WHEN i = 50;
END LOOP;
END;
```

Output:

```
2 is prime
3 is prime
5 is prime
7 is prime
11 is prime
13 is prime
17 is prime
19 is prime
23 is prime
29 is prime
31 is prime
37 is prime
41 is prime
43 is prime
47 is prime
```

Statement processed.

0.03 seconds

Q4:Write a PL/SQL program to handle built-in exceptions

4.1

Declare

```
Vname varchar2(20);  
  
Begin  
  
Select ename into vname from emp where eno=:no;  
  
Exception  
  
When no_data_found then  
  
Dbms_output.put_line('no data found with this number');  
  
End;
```

Output:**NO:101**

```
no data found with this number  
  
Statement processed.
```

4.2

```
Declare  
  
Vempno emp.eno%type;  
Vename emp.ename%type;  
Esal emp.sal%type;  
Edeptno emp.deptno%type:=:deptno;  
  
Begin  
  
Select ename,eno,sal into vename,vempno,esal from emp where deptno=deptno;  
  
Dbms_output.put_line(vename||' '||vempno||' '||deptno||' '||esal);  
  
Exception  
  
When too_many_rows then  
  
Dbms_output.put_line('there are more than one record with same deptno');  
  
End;
```

Output:

```
there are more than one record with same deptno
```

Statement processed.

4.3

Declare

Vempname emp.ename%type;

Vempno emp.eno%type;

Vsal emp.sal%type;

Begin

Select ename,eno,sal into vempname,vempno,vsal from emp where deptno=:deptno;

Dbms_output.put_line('name is' || vempname || 'empno is' || vempno || 'salary is' || vsal);

Exception

When no_data_found then

Dbms_output.put_line('no emp is in the deptno');

When too_many_rows then

Dbms_output.put_line('more than one record is inserted ');

End;

Output:

DEPTNO : 1

more than one record is inserted

Statement processed.

4.4

Declare

M number(10) :=:m;

N number(10):=:n;

K number(10);

Begin

Select m/n into k from dual;

dbms_output.put_line(k);

Exception

When zero_divide then

Dbms_output.put_line('division with zero');

End;

Output:

division with zero

Statement processed.

4.5

Declare

N number(10);

Info varchar2(30);

Vname varchar2(10);

Begin

Select ename into vname from emp where eno=:no;

Dbms_output.put_line(vname);

Exception

When others then

N:=sqlcode;

Info:=sqlerrm;

Dbms_output.put_line(n || '**' || info);

End;

Output:

NO:1005

100**ORA-01403: no data found

Statement processed.

Q5. Write a PL/SQL program to handle user defined exceptions

Declare

M number(10):=:m;

N number(10):=:n;

K number(10);

Infinity exception;

Begin

If n=0 then

Raise infinity;

Else

Select m/n into k from dual;

Dbms_output.put_line(k);

End if;

Exception

When infinity then

Dbms_output.put_line('the result is infinity');

End;

Output:

M:10

N:0

the result is infinity

Q6. Write a PL/SQL program which use user defined error numbers.

Declare

Mempno emp.eno%type;

Mename emp.ename%type;

Msal emp.sal%type;

Mdeptno emp.deptno%type:=:deptno;

Begin

Select ename,eno,sal into mename,mempno,msal from emp where deptno=mdeptno;

Dbms_output.put_line(mename||' '||mempno||' '||mdeptno||' '||msal);

Exception

When too_many_rows then

Raise_application_error(-20001,'it contains more than one record');

End;

Output:

ORA-20001: it contains more than one record

Statement processed.

Experiment- 9**Date :**

Working with Functions Using PL/SQL: Program Development using Creation of Stored Functions, Invoke Functions in SQL Statements and Write Complex Functions.

Q1.Create a function which returns employee salary given employee number as input.

Create or replace function empsal(no number) return number is

Vsal number;

Begin

Select sal into vsal from emp where eno=no;

Return vsal;

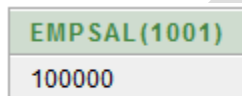
End;

Output:

Function created.

Invocation

select empsal(1001) from dual



EMP_SAL(1001)
100000

1 rows returned in 0.05 seconds

Q2. Create a function which returns employee name given employee number as input.

Create or replace function ename(enum emp.eno%type) return varchar2 is

Vname varchar2(10);

Vsal number(5);

Begin

Select ename,sal into vname,vsal from emp where eno=enum;

Return vname || ' salary is ' || vsal;

End;

Output:

Function created.

Invocation

select ename(1004) from dual

Output:

ENAME(1004)
Sam salary is 200000

1 rows returned in 0.01 seconds

Q3. Create a function which returns week day of a given date

Create or replace function dis_day(d date) return varchar2 is

Tday varchar2(10);

Begin

Tday:=to_char(d,'day');

Return tday;

End;

Output:

Function created.

Invocation

select dis_day('21-mar-2019') from dual

DIS_DAY('21-MAR-2019')
thursday

1 rows returned in 0.00 seconds

Q4. Create a function which returns number of sailors for a given rating level.

Create or replace function no_of_sailors(srating sailors.rating%type) return number is

num number;

Begin

select count(*) into num from sailors where rating=srating;

```
return num;
```

```
End;
```

Output:

Function created.

Invocation

```
select no_of_sailors(10) from dual
```

NO_OF_SAILORS(10)
2

1 rows returned in 0.01 seconds

Q5. Create a function which returns average age of sailors for a given rating level.

Create or replace function avg_sailors(srating sailors.rating%type) return number is
num number;

```
Begin
```

```
select avg(age) into num from sailors where rating=srating;
```

```
return num;
```

```
End;
```

Output:

Function created.

Invocation

```
select avg_sailors(10) from dual
```

AVG_SAILORS(10)
25.5

1 rows returned in 0.02 seconds

Experiment 10

Date :

**Working with CURSORS: Develop Programs using Features Parameters in a CURSOR, FOR UPDATE
CURSOR, WHERE CURRENT of Clause and CURSOR Variables**

Q1.

Declare

Cursor c1 is select * from emp;

Rec emp%rowtype;

Begin

Open c1;

Loop

Fetch c1 into rec;

Exit when c1%notfound;

Dbms_output.put_line(rec.ename||' '||rec.sal);

End loop;

Close c1;

End;

Output:

```

Sravan 55000
Sam 200000
Lokesh 100000
Madhu 50000
```

Statement processed.

Q2

Declare

V_name varchar2(10);

V_empno varchar2(10);

Begin

V_empno:=:no;

Select ename into v_name from emp where eno=v_empno;

If sql%found then

Dbms_output.put_line('records found name is ' || v_name);

End if;

End;

Output:

records found name is Lokesh

Statement processed.

Q3:

Declare

Vname emp.ename%type;

Begin

Delete from emp where deptno=:no;

If sql%notfound then

Dbms_output.put_line('no records found');

Else

Dbms_output.put_line('record deleted');

Dbms_output.put_line('no of records are ' || sql%rowcount);

End if;

End;

Output:

record deleted

no of records are 2

Statement processed.

Example:


```
declare
l varchar2(10);

Begin
Select ename into l from emp where eno=:no;

If sql%rowcount>0 then

Dbms_output.put_line('record found ' || i);

Else

Dbms_output.put_line('records not found');

End if;

End;
```

Output:**NO:1001**

```
record found Lokesh
Statement processed.
```

Q4:

```
declare

Cursor c1 is select * from emp where deptno=:no;
Rec emp%rowtype;

Begin

Open c1;

Loop

Fetch c1 into rec;

Dbms_output.put_line(rec.ename || ' ' || rec.eno || ' ' || rec.sal || ' ');

Exit when c1%notfound;

End loop;

Close c1;
```

End;

Output:

Sam 1004 200000

Sam 1004 200000

Statement processed.

Q5:

Declare

Cursor c1 is select ename,rownum from emp;

R emp.ename%type;

S varchar(30);

K number(10);

Begin

Open c1;

Loop

Fetch c1 into r,k;

Exit when c1%rowcount>1;

Dbms_output.put_line(r||' '||k);

End loop;

Close c1;

End;

Output:

Sam 1

Statement processed.

Q6:

Declare

Cursor c2 is select * from emp;

Begin

Open c2;

If not c2%isopen then

Dbms_output.put_line('the cursor is yet to be opened');

Else

Dbms_output.put_line('the cursor is opened');

End if;

Close c2;

End;

Output:

the cursor is opened

Statement processed.

Q7:

Declare

Cursor c2 is select * from emp;

Rec emp%rowtype;

Begin

Open c2;

Loop

Fetch c2 into rec;

Exit when c2%notfound;

Dbms_output.put_line(rec.ename);

End loop;

If c2%isopen then

Close c2;

End if;

End;

Output:

Sam

Lokesh

Statement processed.

Q8;

Declare

Cursor c1 is select * from emp;

Rec c1%rowtype;

Begin

Open c1;

Loop

Fetch c1 into rec;

If rec.sal>250 then

Dbms_output.put_line(rec.ename);

End if;

Exit when c1%notfound;

End loop;

Close c1;

End;

Output:

Sam

Lokesh

Lokesh

Statement processed.