```
CREATE TABLE sailors
(
sid int PRIMARY KEY,
sname varchar (20),
rating integer ,
age int
);
```

```
CREATE TABLE boats
(
bid int PRIMARY KEY, bname
varchar(20), color varchar(20)
);
```

```
CREATE TABLE reserves
(
sid int, bid
int, day date,
PRIMARY KEY (sid, bid, day),
FOREIGN KEY(sid) REFERENCES sailors(sid),
FOREIGN KEY(bid) REFERENCES boats(bid)
);

INSERT INTO sailors VALUES
(22,'Dustin',7,45), (29,'Brutus',1,33), (31,'Lubber',8,55), (32,'Andy',8,25),
(58,'Rusty', 10, 35), (64,'Horatio',7,35), (71,'Zorba',10,16), (74,'Horatio', 9,
35), (85,'Art',3,25), (95,'Bob',3,63);
```

INSERT INTO boats VALUES

(101, 'Interlake', 'Blue'), (102, 'Interlake', 'Red'),

(103, 'Clipper', 'Green'), (104, 'Marine', 'red');

INSERT INTO reserves VALUES

(22, 101, '10-10-98'), (22, 102, '10-10-98'), (22,103,'10-8-98'),

(22,104,'10-7-98'), (31, 102, '11-10-98'), (31, 103, '11-6-98'),

(31,104, '11-12-98'), (64,101,'9-5-98'), (64, 102, '9-8-98'),

(74, 103, '9-8-98');

**Sailors**

| sid | sname | ratin | age |
|---|---|---|---|
| 22 | Dustin | 7 | 45.0 |
| 29 | Brutus | 1 | 33.0 |
| 31 | Lubber | 8 | 55.5 |
| 32 | Andy | 8 | 25.5 |
| 58 | Rusty | 10 | 35.0 |
| 64 | Horatio | 7 | 35.0 |
| 71 | Zorba | 10 | 16.0 |
| 74 | Horatio | 9 | 35.0 |
| 85 | Art | 3 | 25.5 |
| 95 | Bob | 3 | 63.5 |

**Reserves**

| *sid* | *bid* | *day* |
|---|---|---|
| 22 | 101 | 10/10/98 |
| 22 | 102 | 10/10/98 |
| 22 | 103 | 10/8/98 |
| 22 | 104 | 10/7/98 |
| 31 | 102 | 11/10/98 |
| 31 | 103 | 11/6/98 |
| 31 | 104 | 11/12/98 |
| 64 | 101 | 9/5/98 |
| 64 | 102 | 9/8/98 |
| 74 | 103 | 9/8/98 |

**Boats**

| *bid* | *bname* | *color* |
|---|---|---|
| 101 | Interlake | blue |
| 102 | Interlake | red |
| 103 | Clipper | green |
| 104 | Marine | red |

Q1: Find all sailors with a rating above 7.

```
SELECT *
FROM Sailors
WHERE rating > 7
```

Q2: Find the names of sailors who have reserved boat number 103.

```
SELECT S.sname
FROM Sailors S, Reserves R
WHERE S.sid = R.sid AND R.bid=103;
```

## Q3: Find the sids of sailors who have reserved a red boat.

```
SELECT R.sid
FROM Boats B, Reserves R
WHERE B.bid = R.bid AND B.color = 'red';
```

## Q4: 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 R.bid = B.bid AND B.color = 'red';
```

## Q5: Find the colors of boats reserved by Lubber.

```
SELECT B.color
FROM Sailors S, Reserves R, Boats B
WHERE R.bid = B.bid AND S.sid = R.sid AND S.sname = 'Lubber';
```

# LIKE

## Q6: Find the sailors details whose name starts with ‘D’.

```
SELECT *
FROM Sailors
WHERE sname LIKE "D%";
```

## Q7: Find the sailors details whose name ends with ‘Y’.

```
SELECT *
FROM sailors
WHERE sname LIKE "%Y";
```

Q8: Find the sailors details whose name contains “bb”.

```
SELECT *
FROM sailors
WHERE sname LIKE "%bb%";
```

Q9: Find the sailors details whose name contain ‘o’ in second place.

```
SELECT *
FROM sailors
WHERE sname LIKE "_o%";
```

Q10: Find the sailors details whose name contain “s” in the third place and name should contain total of five characters.

```
SELECT *
FROM sailors
WHERE sname like "_ _s_ _";
```

## Q11: Find the details of sailors whose names does not starts with “a”.

```
SELECT *
FROM sailors
WHERE sname NOT LIKE "a%";
```

# UNION

Q12: Find the sid's and names of sailors who have reserved a red or a green boat.

```
SELECT S.sid, S.sname
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid AND B.color = 'red'
UNION
        SELECT S2.sid, S2.sname
        FROM Sailors S2, Boats B2, Reserves R2
        WHERE S2.sid = R2.sid AND R2.bid = B2.bid AND B2.color = 'green';
```

# INTERSECT

Q13: Find the details of sailors whose names contain the letter 'a' and do not contain the letter 'u'?

```
SELECT *
FROM sailors
WHERE sname LIKE '%a%’
INTERSECT
        SELECT *
        FROM sailors
        WHERE sname NOT LIKE '%u%';
```

# Q14: Find the sid's and names of sailors who have reserved a red and a green boat.

```
SELECT S.sid, S.sname
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid AND B.color = 'red'
INTERSECT
    SELECT S2.sid, S2.sname
    FROM Sailors S2, Boats B2, Reserves R2
    WHERE S2.sid = R2.sid AND R2.bid = B2.bid AND B2.color = 'green';
```

# EXCEPT OR MINUS

Q15: Find the sids of all sailors who have reserved red boats but not green boats.

```
SELECT S.sid
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid AND B.color = 'red'
MINUS // minus in Oracle, Except in SQL server SELECT
    S2.sid
    FROM Sailors S2, Boats B2, Reserves R2
    WHERE S2.sid = R2.sid AND R2.bid = B2.bid AND B2.color = 'green';
```

# NESTED QUERY

IN and NOT IN
EXITS and NOT EXISTS
UNIQUE and NOT UNIQUE
op ANY
op ALL

## Q16: Find the names of sailors who have reserved boat 103.

```
SELECT S.sname
FROM sailors S
WHERE S.sid IN (SELECT R.sid
                FROM reserves R
                WHERE R.bid=103);
```

## Q17: 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.color="red"));
```

# EXISTS

## Q18: Find the names of sailors who have reserved boat 103.

```
SELECT S.sname FROM
sailors S
WHERE EXISTS (      SELECT *
                    FROM reserves R
                    WHERE R.bid = 103
                          AND R.sid = S.sid );
```

## Q19: 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)
                            EXCEPT
                                    (       SELECT R.bid
                                            FROM Reserves R
                                            WHERE R.sid = S.sid ));
```

## Q20: Find the name and age of the youngest sailor.

```
SELECT S.sname, S.age FROM
sailors S
WHERE S.age <= ALL (        SELECT age
                              FROM sailors ) ;
```

Q21: Find the names and ratings of sailor whose rating is better than some sailor called Horatio.

```
SELECT S.sname, S.rating
FROM sailors S
WHERE S.rating > ANY (      SELECT S2.rating
                              FROM sailors S2
                            WHERE S2.sname = 'Horatio' );
```

# AGGREGATION OPERATIONS

COUNT

SUM

AVG

MAX

MIN

Q22 Count the number of different sailor names.

```
SELECT COUNT ( DISTNCT S.sname )
FROM sailors S
```

Q23: Calculate the average age of all sailors.

```
SELECT AVG ( S.age )
FROM sailors S
```

Q24: Find the name and age of the youngest sailor.

```
SELECT S.name, S.age
FROM sailors S
WHERE S.age = (         SELECT MIN ( S2.age )
                        FROM Sailors S2 );
```

SELECT [ DISTINCT ] select-list
FROM from-list
WHERE qualification
GROUP BY grouping-list
HAVING group-qualification

Q25: Find the average age of sailors for each rating level.

```
SELECT S.rating, AVG ( S.age ) AS avg.age
FROM sailors S
GROUP BY S.rating;
```

Q26: Find the age of sailors for each rating level that has at least two sailors.

```
SELECT S.rating, AVG ( S.age ) AS avg.age
FROM sailors S
GROUP BY S.rating
HAVING COUNT (*) > 1
```

## Q27: An example shows difference between WHERE and HAVING.

```
SELECT S.rating, AVG ( S.age ) AS avg.age
FROM sailors S
WHERE S.age >= 40
GROUP BY S.rating;

SELECT S.rating, AVG ( S.age ) AS avg.age  FROM sailors
S
GROUP BY S.rating
HAVING AVG ( S.age ) >= 40;
```

# JOIN
# OUTER JOIN

Consider the following schema for the sales information system:

PRODUCT ( Pid, Pname, Sellingprice, M_name ) CLIENT ( Clientid, Cname, Caddress, City, Pincode ) SALES ( Ordno, Odate,Clientid, Salesid )

SALESMAN ( Salesid, name, salary, Del_date, qtyordered, Pname )

Write SQL queries to

a. Retrieve the first 3 client's information one by one.
b. Retrieve the salesman id and the products sold by that salesman.
c. Display the salesman name who is selling the camera.
d. Write the SQL code which will accept salesid from the user and increase the salesman's salary by 500/- if he is selling the product camera and his current salary is less than 20000/.

```
SELECT *
FROM CLIENT
ORDER BY Clientid
FETCH FIRST 3 ROWS ONLY;
```

```
SELECT S.Salesid, P.Pname
FROM SALESMAN S
JOIN PRODUCT P ON S.Pname = P.Pname;
```

```
SELECT S.name
FROM SALESMAN S
WHERE S.Pname = 'camera';
```

```
UPDATE SALESMAN
SET salary = salary + 500
WHERE Salesid = ?
AND Pname = 'camera’
AND salary < 20000;
```

Consider the following relation schema

WORKS ( Pname, Cname, salary ) LIVES ( Pname, Street, City ) LOCATED_IN ( Cname, city ) MANAGER ( Pname, Mgrname )

Write the SQL queries for the following:

a. Find the names of all persons who live in the city of Bengaluru.
b. Retrieve the names of all persons of "Infosys" whose salary is between Rs. 50000 and Rs. 60000.
c. Find the names of all persons who live and work in the same city.
d. List the names of the people who work for “Tech M” along with the cities they live in.
e. Find the average salary of “Infosys” persons.

```
CREATE TABLE WORKS
( Pname VARCHAR(100),
Cname VARCHAR(100),
salary DECIMAL(10, 2),
PRIMARY KEY (Pname, Cname)
);
```

```
CREATE TABLE LIVES
( Pname VARCHAR(100),
Street VARCHAR(100),
City VARCHAR(100),
PRIMARY KEY (Pname, Street)
);
```

```
CREATE TABLE LOCATED_IN (
Cname VARCHAR(100),
City VARCHAR(100),
PRIMARY KEY (Cname, City)
);
```

```
CREATE TABLE MANAGER (
Pname VARCHAR(100),
Mgrname VARCHAR(100),
PRIMARY KEY (Pname)
);
```

```
INSERT INTO WORKS (Pname, Cname, salary) VALUES ('Alice Johnson', 'Tech Corp', 75000.00);
INSERT INTO WORKS (Pname, Cname, salary) VALUES ('Bob Smith', 'Finance Inc', 85000.00);
INSERT INTO WORKS (Pname, Cname, salary) VALUES ('Charlie Brown', 'Health LLC', 72000.00);
INSERT INTO WORKS (Pname, Cname, salary) VALUES ('Diana Prince', 'Tech Corp', 95000.00);
INSERT INTO WORKS (Pname, Cname, salary) VALUES ('Ethan Hunt', 'Finance Inc', 90000.00);
```

```
INSERT INTO LIVES (Pname, Street, City) VALUES ('Alice Johnson', 'Main St', 'Springfield’);
INSERT INTO LIVES (Pname, Street, City) VALUES ('Bob Smith', 'Second Ave', 'Metropolis’);
INSERT INTO LIVES (Pname, Street, City) VALUES ('Charlie Brown', 'Third Blvd', 'Gotham’);
INSERT INTO LIVES (Pname, Street, City) VALUES ('Diana Prince', 'Fourth St', 'Paradise Island’);
INSERT INTO LIVES (Pname, Street, City) VALUES ('Ethan Hunt', 'Fifth Rd', 'New York');
```

```
INSERT INTO LOCATED_IN (Cname, City) VALUES ('Tech Corp', 'San Francisco');
INSERT INTO LOCATED_IN (Cname, City) VALUES ('Finance Inc', 'New York');
INSERT INTO LOCATED_IN (Cname, City) VALUES ('Health LLC', 'Chicago');
INSERT INTO LOCATED_IN (Cname, City) VALUES ('Retail Co', 'Los Angeles');
INSERT INTO LOCATED_IN (Cname, City) VALUES ('Logistics Corp', 'Dallas');


INSERT INTO MANAGER (Pname, Mgrname) VALUES ('Alice Johnson', 'Robert
King');
INSERT INTO MANAGER (Pname, Mgrname) VALUES ('Bob Smith', 'Laura
White');
INSERT INTO MANAGER (Pname, Mgrname) VALUES ('Charlie Brown', 'James
Green');
INSERT INTO MANAGER (Pname, Mgrname) VALUES ('Diana Prince', 'Sarah
Connor');
INSERT INTO MANAGER (Pname, Mgrname) VALUES ('Ethan Hunt', 'Michael
```

Scott');

```
SELECT Pname
FROM LIVES
WHERE City = 'Bengaluru';
```

```
SELECT Pname
FROM WORKS
WHERE Cname = 'Infosys'
        AND salary BETWEEN 50000 AND 60000;
```

```
SELECT L.Pname
FROM LIVES L
JOIN WORKS W ON L.Pname = W.Pname
JOIN LOCATED_IN LI ON W.Cname = LI.Cname
WHERE L.City = LI.city;
```

```
SELECT W.Pname, L.City
FROM WORKS W
JOIN LIVES L ON W.Pname = L.Pname
WHERE W.Cname = 'Tech M';
```

```
SELECT AVG(salary) AS average_salary
FROM WORKS
WHERE Cname = 'Infosys';
```