J.N.T.U.H. UNIVERSITY COLLEGE OF ENGINEERING, SCIENCE AND TECHNOLOGY HYDERABAD (Autonomous)

KUKATPALLY, HYDERABAD – 500 085

Certificate

Certified that this is the bonafide record of the practical work done during

the academic year _____ by

Name _____ Roll

Number _____ Class _____

in the Laboratory of _____

of the Department of _____

Signature of the Staff Member

Signature of the Head of the Department

Date of Examination_____

Signature of the Examiner/s

Internal Examiner

External Examiner

J.N.T.U.H. UNIVERSITY COLLEGE OF ENGINEERING HYDERABAD
(Autonomous)

KUKATPALLY, HYDERABAD – 500 085

S.No.	Name of the Experiment	Date of Experiment	Page Number	Remarks	Signature
1.	Concept design with E-R Model				
2.	Relational Model				
3.	Normalization				
4.	DDL Commands				
5.	DML Commands				
6	Querying Nested, Correlated Subqueries.				
7	Queries using Aggregate functions.				
8	Triggers				
9	Procedures				

1. Concept of E-R Model

Conceptual design is the first step in the database design process. It deals with converting user requirements into a high-level, understandable data model. The most commonly used high-level data model is the Entity-Relationship (E-R) model. It provides a graphical way of representing entities (things in the real world), attributes (properties), and relationships (associations between entities).

What is the E-R Model?

The Entity-Relationship Model was proposed by Peter Chen in 1976. It represents:

- Entities: Objects or concepts (e.g., Student, Course).
- Attributes: Properties of entities (e.g., Student_ID, Name).
- Relationships: Associations among entities (e.g., a Student *enrolls* in a Course).

This model uses E-R diagrams to visually represent data structures. These diagrams are later transformed into relational schemas during logical design.

Steps in Conceptual Design using E-R Model

1. Identify Entities
   These are real-world objects about which data needs to be stored.

○ Example: Student, Course, Professor.

2. Identify Relationships
   Determine how entities are associated with one another.

○ Example: Student enrolls in Course.

3. Identify Attributes
   Define the properties of entities and relationships.

○ Example: Student has attributes like Student_ID, Name.

4. Identify Primary Keys
   Choose attributes that uniquely identify entities.

○ Example: Student_ID for Student.

5. Draw E-R Diagram

Represent entities as rectangles, attributes as ovals, relationships as diamonds, and links between them with lines.

Sample E-R Diagram

Let’s consider a university database that keeps track of students, courses, and instructors.

Entities:

- Student (Student_ID, Name, Email)
- Course (Course_ID, Title, Credits)
- Instructor (Instructor_ID, Name, Department)

Relationships:

- A Student enrolls in a Course
- A Course is taught by an Instructor

2. Relational Model

The Relational Model is a fundamental model in database management introduced by E.F. Codd in 1970. It represents data in the form of tables (called relations), where each table consists of rows and columns. This model is the backbone of relational databases, such as MySQL, PostgreSQL, and Oracle.

Key Concepts of the Relational Model

1. Relation (Table):
   A relation is a table with columns and rows. Each relation represents a specific entity or concept from the real world. The columns represent attributes, and the rows represent tuples or records.

Example – A "Student" table:

Student_Id	Name	Email
101	Ramesh	ramesh@example.com
102	Sita	sita@example.com

2. Attribute (Column):
   Attributes are the properties or characteristics of an entity. For example, in the Student relation, Student_ID, Name, and Email are attributes.

3. Tuple (Row):
   A tuple represents a single record in the table. Each row in the relation is unique.

4. Domain:
   A domain is a set of valid values for an attribute. For example, the domain for Student_ID might be positive integers.

5. Primary Key:
   A primary key is a unique identifier for tuples in a relation. In the Student table, Student_ID could be a primary key.

6. Foreign Key:
   A foreign key is an attribute that creates a relationship between two tables. It refers to the primary key in another table.

Properties of the Relational Model

- Atomicity: All attribute values must be atomic, meaning they are indivisible (no arrays or nested tables).

- Uniqueness: Each tuple in a relation is unique.

- Ordering Irrelevant: Tuples and attributes are not ordered in the relational model.

Relational Integrity Constraints

1. Entity Integrity:
   Ensures the primary key cannot be NULL and must be unique.

2. Referential Integrity:
   Ensures that foreign key values must either be NULL or match a primary key value in another relation.

3. Domain Integrity:
   Ensures that attribute values must fall within the domain defined for that attribute.

3. Normalization

Normalization is a process in relational database design that organizes data to reduce redundancy and improve data integrity. It involves decomposing complex tables into simpler, smaller tables and defining relationships among them. The goal is to eliminate undesirable anomalies—such as insertion, update, and deletion anomalies—and to ensure consistency and efficient storage.

Purpose of Normalization

- To eliminate redundant data.
- To ensure data dependencies make sense (i.e., data is stored logically).
- To maintain data integrity by avoiding anomalies.

Types (Forms) of Normalization

1. First Normal Form (1NF)

○ Ensures that each column contains only atomic (indivisible) values.

○ No repeating groups or arrays are allowed in a column.

2. Second Normal Form (2NF)

- Achieved when the relation is in 1NF and every non-prime attribute is fully functionally dependent on the primary key.
- Removes partial dependencies (when a non-key attribute depends only on part of a composite key).

3. Third Normal Form (3NF)

- Achieved when the relation is in 2NF and there is no transitive dependency between non-prime attributes and the primary key.
- Ensures that all attributes are only dependent on the primary key.

4. Boyce-Codd Normal Form (BCNF)

- A stricter version of 3NF.

- A relation is in BCNF if, for every functional dependency $X \rightarrow Y$, X is a super key.

5. Fourth Normal Form (4NF):

A table is in 4NF if:

- It is in BCNF, and
- It has no multivalued dependencies (MVDs).

6. Fifth Normal Form (5NF):

A table is in 5NF if:

- It is in 4NF, and
- Every join dependency in the relation is implied by the candidate keys.

This form deals with reconstructing original data from multiple joins without redundancy

4. Practicing DDL Commands

1. CREATE: Used to create new database objects like tables, views, indexes, schemas, etc.

Ex:

Creating the 3 tables: sailors, boats and reserves.

```
CREATE TABLE Sailors (     sid INT PRIMARY
KEY,     sname VARCHAR(50),
   rating INT,     age
DECIMAL(4, 1)
);

CREATE TABLE Boats (
bid INT PRIMARY KEY,
```

```
bname VARCHAR(50),
color VARCHAR(20)
);

CREATE TABLE Reserves (
   sid INT,
bid INT,
rdate DATE,
   PRIMARY KEY (sid, bid, rdate),
   FOREIGN KEY (sid) REFERENCES Sailors(sid),
   FOREIGN KEY (bid) REFERENCES Boats(bid)
);
```

2. ALTER: Used to modify an existing database object (like adding/removing a column in a table).

Ex:

ALTER TABLE Sailors ADD email VARCHAR(100); **3.**

DROP: Deletes an existing database object permanently.

Ex:

DROP TABLE Boats;

4. **TRUNCATE:** Deletes **all rows** from a table but does **not** remove the structure.

Ex:

TRUNCATE TABLE Reserves;

5. **RENAME:** Renames a database object (table, column, etc.).

Ex:

ALTER TABLE Sailors RENAME TO Mariners;

5. Practicing DML Commands

1. INSERT: Adds new records/rows into a table.

Ex:

i)

INSERT INTO boats (bid, bname, color)

VALUES (101, 'Interlake', 'blue'),

(102, 'Interlake', 'red'),

(103, 'Clipper', 'green'),

(104, 'Marine', 'red');

ii)

INSERT INTO sailors(sid, sname, rating, age) VALUES
(22, 'Dustin', 7, 45),

(29, 'Dustin', 7, 45),

(31, 'Dustin', 7, 45),

(32, 'Dustin', 7, 45),

(58, 'Dustin', 7, 45),

(64, 'Dustin', 7, 45),

(71, 'Dustin', 7, 45),

(74, 'Dustin', 7, 45),

(85, 'Dustin', 7, 45), (95,

'Dustin', 7, 45);

VALUES

(22, 101, '1998-10-10'),

(22, 102, '1998-10-10'),

(22, 103, '1998-10-08'),

(22, 104, '1998-10-07'),

(31, 102, '1998-11-10'),

(31, 103, '1998-11-06'),

(31, 104, '1998-11-12'),

(64, 101, '1998-09-05'),

(64, 102, '1998-09-08'),

(74, 103, '1998-09-08');

2. SELECT: Retrieves data from one or more tables

Ex:

i) SELECT * FROM sailors; Output:

	sid [PK] integer	sname character varying (50)	rating integer	age double precision
1	22	Dustin	7	45
2	29	Brutus	7	33
3	31	Lubbur	7	55.5
4	32	Andy	7	25.5
5	58	Rusty	7	35
6	64	Horatio	7	35
7	71	Zorba	7	60
8	74	Horatio	7	35
9	85	Art	7	25.5
10	95	Bob	7	63.5

ii) SELECT * FROM boats;
Output:

	bid [PK] integer	bname character varying (50)	color character varying (50)
1	101	Interlake	blue
2	102	Interlake	red
3	103	Clipper	green
4	104	Marine	red

3. UPDATE: Modifies existing data in a table

Ex:

i) UPDATE Sailors SET rating = 9

WHERE sid = 22;

Output:

	sid [PK] integer	sname character varying (50)	rating integer	age double precision
1	22	Dustin	9	45

ii) UPDATE Boats

SET color = 'Green'

WHERE bid = 101;

Output:

	bid [PK] integer	bname character varying (50)	color character varying (50)
1	101	Interlake	Green

4. DELETE: Removes specific records from a table.

Ex:

i) DELETE FROM Sailors

WHERE sid = 31;

ii) DELETE FROM Reserves

WHERE rdate = '1998-10-10'';

6.

A. Querying (using ANY, ALL, UNION, INTERSECT, JOIN, Constraints etc.)

1. **ANY:** Returns true if any comparison is true

   Ex:

   i) SELECT sname FROM sailors

   WHERE rating > ANY (SELECT rating FROM sailors WHERE age < 50); Output:

	sname character varying (50)
1	Dustin

   ii) SELECT sname FROM Sailors

   WHERE sid = ANY (SELECT sid FROM Reserves WHERE bid > 103); Output:

	sname character varying (50)
1	Lubbur
2	Dustin

2. **ALL:** Returns true only if all comparisons are true.

   Ex:

   i) SELECT sname FROM Sailors

   WHERE rating > ALL (SELECT rating FROM Sailors WHERE age < 18);
   Output:

	sname character varying (50)
1	Brutus
2	Lubbur
3	Andy
4	Rusty
5	Horatio
6	Zorba
7	Horatio
8	Art
9	Bob
10	Dustin

ii) SELECT DISTINCT bname FROM Boats

WHERE bid = ALL (SELECT bid FROM Reserves WHERE sid IN

(SELECT sid FROM Sailors WHERE rating = 10)); Output:

	bname character varying (50)
1	Clipper
2	Interlake
3	Marine

3. **UNION:** Combines results from two queries, removing duplicates.

Ex:

i) SELECT sid FROM Reserves
UNION

SELECT sid FROM Sailors WHERE rating > 8; Output:

	sid integer
1	74
2	31
3	64
4	22

ii) SELECT sname AS name FROM Sailors

UNION

SELECT bname AS name FROM Boats; Output:

	name character varying (50)
1	Clipper
2	Andy
3	Zorba
4	Art
5	Dustin
6	Rusty
7	Lubbur
8	Interlake
9	Bob
10	Brutus
11	Horatio
12	Marine

4. **INTERSECT:** Returns common records between two queries.

   Ex:

   i) SELECT sid FROM Reserves WHERE bid = 101
   INTERSECT

   SELECT sid FROM Reserves WHERE bid = 102; Output:

	sid integer
1	64
2	22

ii) SELECT sid FROM Sailors WHERE rating > 7

INTERSECT

SELECT sid FROM Reserves; Output:

	sid integer
1	22

5. **JOIN:** Combines columns from multiple tables based on a related column.

   Ex:

   i) SELECT s.sname, b.bname

   FROM Sailors s

   JOIN Reserves r ON s.sid = r.sid

   JOIN Boats b ON r.bid = b.bid;

   Output:

	sname character varying (50)	bname character varying (50)
1	Dustin	Interlake
2	Dustin	Interlake
3	Dustin	Clipper
4	Dustin	Marine
5	Lubbur	Interlake
6	Lubbur	Clipper
7	Lubbur	Marine
8	Horatio	Interlake
9	Horatio	Interlake
10	Horatio	Clipper

B. Nested, Correlated subqueries

Nested queries:

i) Sailors who reserved a boat named ‘Interlake’

```
SELECT sname
FROM Sailors
WHERE sid IN (
   SELECT sid
   FROM Reserves
   WHERE bid IN (
      SELECT bid
      FROM Boats
      WHERE bname = 'Interlake' ));
```

Output:

	sname character varying (50)
1	Lubbur
2	Horatio
3	Dustin

ii) Sailors with rating higher than the average rating

```
SELECT sname
FROM Sailors
WHERE rating > (
   SELECT AVG(rating)
```

```
    FROM Sailors);
```

Output:

	sname character varying (50)
1	Dustin

Correlated Queries:

i) Sailors who have reserved more than one boat

```
SELECT s1.sid, s1.sname
FROM Sailors s1
WHERE (
    SELECT COUNT(DISTINCT r.bid)
    FROM Reserves r
    WHERE r.sid = s1.sid) > 1;
```

Output:

	sid [PK] integer	sname character varying (50)
1	31	Lubbur
2	64	Horatio
3	22	Dustin

ii) Sailors who have reserved all boats

```
SELECT s.sid, s.sname
FROM Sailors s
WHERE NOT EXISTS (
    SELECT *
```

```
FROM Boats b
WHERE NOT EXISTS (
   SELECT *
   FROM Reserves r
   WHERE r.sid = s.sid AND r.bid = b.bid)); Output:
```

	sid [PK] integer	sname character varying (50)
1	22	Dustin

7. Queries using Aggregate functions, GROUP BY, HAVING and Creation and dropping of Views.

1. Aggregate functions:

   i) Find the average age of sailors

   SELECT AVG(age) AS average_age FROM Sailors; Output:

	average_age double precision
1	41.3

   ii) Count the total number of reservations

   SELECT COUNT(*) AS total_reservations FROM Reserves; Output:

	total_reservations bigint
1	10

2. GROUP BY Clause:

i) Number of reservations made by each sailor

```
SELECT sid, COUNT(bid) AS reservation_count

FROM Reserves

GROUP BY sid;
```

Output:

	sid integer	reservation_count bigint
1	22	4
2	74	1
3	31	3
4	64	2

ii) Average rating of sailors grouped by age

```
SELECT age, AVG(rating) AS avg_rating

FROM Sailors GROUP

BY age;
```

Output:

	age double precision	avg_rating numeric
1	25.5	7.0000000000000000
2	33	7.0000000000000000
3	35	7.0000000000000000
4	63.5	7.0000000000000000
5	60	7.0000000000000000
6	55.5	7.0000000000000000
7	45	9.0000000000000000

3. HAVING Clause:

i) Sailors who made more than 2 reservations

SELECT sid, COUNT(*) AS num_reservations

FROM Reserves

GROUP BY sid
HAVING COUNT(*) > 2; Output:

	sid integer	num_reservations bigint
1	22	4
2	31	3

ii) Sailor age groups with average rating above 7

SELECT age, AVG(rating) AS avg_rating

FROM Sailors

GROUP BY age HAVING

AVG(rating) > 7; Output:

	age double precision	avg_rating numeric
1	45	9.0000000000000000

4. Views: Creation and Dropping

i) Create a view of all reservations with sailor names and boat names

CREATE VIEW SailorBoatReservations AS

SELECT s.sname, b.bname, r.rdate

FROM Sailors s

JOIN Reserves r ON s.sid = r.sid

```
JOIN Boats b ON r.bid = b.bid;
```

ii) Drop the above view

```
DROP VIEW SailorBoatReservations;
```

8. Triggers in DBMS

A trigger is a database object that automatically executes a predefined action when a specific event occurs on a table (like INSERT, UPDATE, or DELETE).
They are used to enforce business rules, audit changes, or maintain data consistency.

Example 1: INSERT Trigger on Reserves

Purpose: Logs new reservations into a log table after insertion.

```
CREATE TRIGGER after_reserve_insert
AFTER INSERT ON Reserves
FOR EACH ROW
BEGIN
  INSERT INTO Reserve_Log (sid, bid, rdate, action, log_time)
  VALUES (NEW.sid, NEW.bid, NEW.rdate, 'INSERTED', NOW());
END;
```

Output:

```
+-----+-----+------------+----------+---------------------+ |
sid | bid | rdate      | action   | log_time            |
+-----+-----+------------+----------+---------------------+
| 22  | 101 | 1998-10-10 | INSERTED | 2025-06-07 10:00:00 |
+-----+-----+------------+----------+---------------------+
```

Example 2: DELETE Trigger on Sailors

Purpose: Logs the deleted sailor's data to a log table.

```
CREATE TRIGGER after_sailor_delete
AFTER DELETE ON Sailors
FOR EACH ROW
BEGIN
```

```
  INSERT INTO Sailor_Log (sid, sname, rating, age, action, log_time)
  VALUES (OLD.sid, OLD.sname, OLD.rating, OLD.age, 'DELETED', NOW());
END;
```

Output:

```
+-----+--------+--------+------+---------+---------------------+
| sid | sname  | rating | age  | action  | log_time            |
+-----+--------+--------+------+---------+---------------------+
| 31  | Dustin | 7      | 45.0 | DELETED | 2025-06-07 10:05:00 |
+-----+--------+--------+------+---------+---------------------+
```

Example 3: UPDATE Trigger on Boats

Purpose: Logs changes in boat color into a log table.

```
CREATE TRIGGER after_boat_update
AFTER UPDATE ON Boats
FOR EACH ROW
BEGIN
  IF OLD.color != NEW.color THEN
    INSERT INTO Boat_Log (bid, old_color, new_color, changed_on)
    VALUES (OLD.bid, OLD.color, NEW.color, NOW());
  END IF;
END;
```

Output:

```
+-----+------------+-----------+---------------------+ +-----+------------+-----------+---------------------+
| bid | old_color | new_color | changed_on          | +-----+------------+-----------+---------------------+
| 102 | red        | yellow    | 2025-06-07 10:10:00 |
+-----+------------+-----------+---------------------+
```

9. Procedures in DBMS

Stored procedures are precompiled SQL statements grouped to perform a specific task. They increase reusability and performance.

Example 1: Procedure to Insert a New Sailor

```
DELIMITER //
CREATE PROCEDURE AddSailor(
   IN p_sid INT,
   IN p_sname VARCHAR(50),
   IN p_rating INT,
```

```
    IN p_age DECIMAL(4,1)
)
BEGIN
    INSERT INTO Sailors(sid, sname, rating, age)
    VALUES(p_sid, p_sname, p_rating, p_age);
END;
//
DELIMITER ;
```

CALL AddSailor(101, 'John', 5, 34.5); Output:

```
+-----+-------+--------+------+
| sid | sname | rating | age  |
+-----+-------+--------+------+
| 101 | John  | 5      | 34.5 |
+-----+-------+--------+------+
```

Example 2: Procedure to Delete a Reservation

```
DELIMITER //
CREATE PROCEDURE DeleteReservation(
    IN p_sid INT,
    IN p_bid INT
)
BEGIN
    DELETE FROM Reserves
    WHERE sid = p_sid AND bid = p_bid;
END;
//
DELIMITER ;
```

CALL DeleteReservation(22, 101); Output:

```
+-----+-----+------------+ |
sid | bid | rdate      |
+-----+-----+------------+
| 22  | 102 | 1998-10-10 | |
... remaining records ...
+-----+-----+------------+
```

Example 3: Procedure to Update Boat Color

```
DELIMITER //
```

```
CREATE PROCEDURE UpdateBoatColor(
    IN p_bid INT,
    IN p_newColor VARCHAR(20)
)
BEGIN
    UPDATE Boats
    SET color = p_newColor
    WHERE bid = p_bid;
END;
//
DELIMITER ;
```

CALL UpdateBoatColor(102, 'Yellow'); Output:

```
+-----+-----------+--------+
| bid | bname     | color  |
+-----+-----------+--------+
| 102 | Interlake | Yellow |
+-----+-----------+--------+
```

Creating Tables:

```
DROP TABLE Sailors;
DROP TABLE Boats;
DROP TABLE Reserves;
--creating sailors table CREATE
TABLE Sailors (
        sid INT PRIMARY KEY,
sname VARCHAR(50),
 rating INT,        age FLOAT
);
--inserting values
INSERT INTO Sailors(sid,sname,rating,age)
VALUES
(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);
--creating boats table CREATE TABLE
Boats(
        bid INT PRIMARY KEY,
 bname VARCHAR(50),
        color VARCHAR(50)
);
--inserting values into Boats
INSERT INTO Boats(bid,bname,color)
VALUES
(101, 'Interlake', 'blue'),
(102, 'Interlake', 'red'),
(103, 'Clipper', 'green'),
(104, 'Marine', 'red');
--creating Reserves table
CREATE TABLE Reserves(
sid INT,            bid INT,
        day DATE,
        PRIMARY KEY (sid,bid)
);
--inserting values into Reserves
INSERT INTO Reserves(sid,bid,day)
```

VALUES
(22, 101, '1998-10-10'),
(22, 102, '1998-10-10'),
(22, 103, '1998-10-08'),
(22, 104, '1998-10-07'),
(31, 102, '1998-11-10'),
(31, 103, '1998-11-06'),
(31, 104, '1998-11-12'),
(64, 101, '1998-09-05'),
(64, 102, '1998-09-08'), (74,
103, '1998-09-08');

Tables:

Sailors

	sid [PK] integer	sname character varying (50)	rating integer	age double precision
1	22	Dustin	7	45
2	29	Brutus	1	33
3	31	Lubber	8	55.5
4	32	Andy	8	25.5
5	58	Rusty	10	35
6	64	Horatio	7	35
7	71	Zorba	10	16
8	74	Horatio	9	35
9	85	Art	3	25.5
10	95	Bob	3	63.5

Boats

	bid [PK] integer	bname character varying (50)	color character varying (50)
1	101	Interlake	blue
2	102	Interlake	red
3	103	Clipper	green
4	104	Marine	red

Reserves

	sid [PK] integer	bid [PK] integer	day date
1	22	101	1998-10-10
2	22	102	1998-10-10
3	22	103	1998-10-08
4	22	104	1998-10-07
5	31	102	1998-11-10
6	31	103	1998-11-06
7	31	104	1998-11-12
8	64	101	1998-09-05
9	64	102	1998-09-08
10	74	103	1998-09-08

1. Find the names of sailors who have reserved boat 103.

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

OUTPUT

```
  sname
---------
 Dustin
 Lubber
 Horatio
(3 rows)
```

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

```
SELECT DISTINCT S.sname FROM Sailors S
JOIN Reserves R ON S.sid = R.sid
JOIN Boats B ON R.bid = B.bid
WHERE B.color = 'red';
```

OUTPUT

```
  sname
---------
 Dustin
 Horatio
 Lubber
(3 rows)
```

3. Find the colors of boats reserved by Lubber.

```
SELECT DISTINCT B.color FROM Boats B
JOIN Reserves R ON B.bid = R.bid
JOIN Sailors S ON R.sid = S.sid
WHERE S.sname = 'Lubber';
```

OUTPUT

```
 color
-------
 green
 red
(2 rows)
```

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

```
SELECT DISTINCT S.sname FROM Sailors S
JOIN Reserves R ON S.sid = R.sid;
```

OUTPUT

```
  sname
---------
 Dustin
 Horatio
 Lubber
(3 rows)
```

5. Find the names of sailors who have reserved a red or a green boat.

```
SELECT DISTINCT S.sname FROM Sailors S
JOIN Reserves R ON S.sid = R.sid
JOIN Boats B ON R.bid = B.bid
WHERE B.color = 'red' OR B.color = 'green';
```

OUTPUT

```
  sname
---------
 Dustin
 Horatio
 Lubber
(3 rows)
```

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

```
SELECT DISTINCT S.sname
FROM Sailors S
JOIN Reserves r1 ON S.sid = r1.sid
JOIN Boats b1 ON r1.bid = b1.bid
```

```
JOIN Reserves r2 ON S.sid = r2.sid
JOIN Boats b2 ON r2.bid = b2.bid
WHERE b1.color = 'red' AND b2.color = 'green';
```

OUTPUT

```
 sname
--------
 Dustin
 Lubber
(2 rows)
```

7. Find the names of sailors who have reserved at least two boats.

```
SELECT S.sname
FROM Sailors S
JOIN Reserves R ON S.sid = R.sid
GROUP BY S.sid, S.sname
HAVING COUNT(DISTINCT R.bid) >= 2;
```

OUTPUT

```
  sname
---------
 Dustin
 Lubber
 Horatio
(3 rows)
```

8. Find the sids of sailors with age over 20 who have not reserved a red boat.

```
SELECT DISTINCT S.sid
FROM Sailors S , Reserves R, Boats B
WHERE S.age > 20
AND S.sid = R.sid
AND R.bid = B.bid
AND NOT B.color = 'red';
```

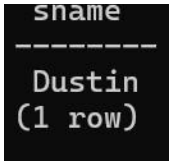
OUTPUT

```
 sid
-----
  22
  74
  31
  64
(4 rows)
```

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

```
SELECT S.sname FROM Sailors S
JOIN Reserves R ON S.sid = R.sid
GROUP BY S.sid, S.sname
HAVING COUNT(DISTINCT R.bid) = (SELECT COUNT(*) FROM Boats);
```

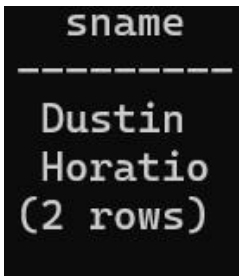
OUTPUT



10. Find the names of sailors who have reserved all boats called Interlake.

```
SELECT S.sname FROM Sailors S
JOIN Reserves R ON S.sid =R.sid
JOIN Boats B ON R.bid = B.bid
WHERE B.bname = 'Interlake'
GROUP BY S.sid, S.sname
HAVING COUNT(DISTINCT R.bid) = (
        SELECT COUNT(*) FROM Boats
        WHERE bname = 'Interlake'
);
```

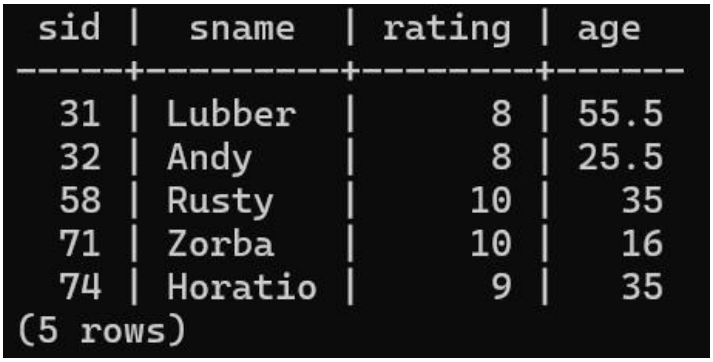
OUTPUT



11. Find all sailors with a rating above 7.

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

OUTPUT



12. Find the names and ages of sailors with a rating above 7.

SELECT sname, age FROM Sailors WHERE rating > 7;

OUTPUT

sname	age
Lubber	55.5
Andy	25.5
Rusty	35
Zorba	16
Horatio	35

(5 rows)

13. Find the sailor name, boat id, and reservation date for each reservation.

SELECT S.sname, R.bid, R.day
FROM Sailors S
JOIN Reserves R ON S.sid = R.sid;

OUTPUT

sname	bid	day
Dustin	101	1998-10-10
Dustin	102	1998-10-10
Dustin	103	1998-10-08
Dustin	104	1998-10-07
Lubber	102	1998-11-10
Lubber	103	1998-11-06
Lubber	104	1998-11-12
Horatio	101	1998-09-05
Horatio	102	1998-09-08
Horatio	103	1998-09-08

(10 rows)

14. Find sailors who have reserved all red boats.

```
SELECT S.sname FROM Sailors S
JOIN Reserves R ON S.sid = R.sid
JOIN Boats B ON R.bid = B.bid
WHERE B.color = 'red'
GROUP BY S.sid, S.sname
HAVING COUNT(DISTINCT B.bid) = (
        SELECT COUNT(*) FROM Boats WHERE color = 'red'
);
```

OUTPUT

```
 sname
--------
 Dustin
 Lubber
(2 rows)
```

15. Find the names and ages of all sailors.

SELECT sname, age FROM Sailors;

OUTPUT

```
  sname   | age
----------+------
 Dustin   |   45
 Brutus   |   33
 Lubber   | 55.5
 Andy     | 25.5
 Rusty    |   35
 Horatio  |   35
 Zorba    |   16
 Horatio  |   35
 Art      | 25.5
 Bob      | 63.5
(10 rows)
```

`

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

SELECT DISTINCT R.sid FROM Reserves R
JOIN Boats B ON R.bid = B.bid
WHERE B.color = 'red';

OUTPUT

```
 sid
-----
  22
  31
  64
(3 rows)
```

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

SELECT DISTINCT S.sid, S.sname, S.rating
FROM Sailors S
JOIN Reserves R1 ON S.sid = R1.sid

```
JOIN Reserves R2 ON S.sid = R2.sid
WHERE R1.bid <> R2.bid AND R1.day = R2.day;
```

OUTPUT

```
 sid | sname  | rating
-----+--------+--------
  22 | Dustin |      7
(1 row)
```

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

```
SELECT age FROM Sailors
WHERE sname LIKE 'B%B' AND LENGTH(sname) >= 3;
```

OUTPUT

```
 age
-----
(0 rows)
```

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

```
SELECT DISTINCT R1.sid FROM Reserves R1
JOIN Boats B1 ON R1.bid = B1.bid
WHERE B1.color = 'red'
AND R1.sid NOT IN (
        SELECT R2.sid FROM Reserves R2
        JOIN Boats B2 ON R2.bid = B2.bid
        WHERE B2.color = 'green'
);
```

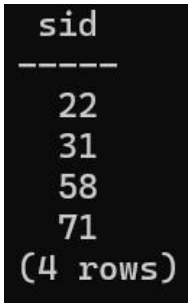
OUTPUT

```
 sid
-----
  64
(1 row)
```

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

```
SELECT sid FROM Sailors
WHERE rating = 10
UNION
SELECT sid FROM Reserves
WHERE bid = 104;
```

OUTPUT



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

```
SELECT sname FROM Sailors
WHERE sid NOT IN(
        SELECT R.sid FROM Reserves R
        JOIN Boats B ON R.bid = B.bid
        WHERE B.color = 'red'
);
```

OUTPUT

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

```
SELECT sname FROM Sailors
WHERE rating > ANY(
        SELECT rating FROM Sailors
        WHERE sname = 'Horatio'
);
```

OUTPUT

```
  sname
---------
 Lubber
 Andy
 Rusty
 Zorba
 Horatio
(5 rows)
```

23. Find sailors whose rating is better than every sailor called Horatio.

```
SELECT sname FROM Sailors
WHERE rating > ALL(
        SELECT rating FROM Sailors
        WHERE sname = 'Horatio'
);
```

OUTPUT

```
 sname
-------
 Rusty
 Zorba
(2 rows)
```

24. Find the sailors with the highest rating.

```
SELECT sname, rating FROM Sailors
WHERE rating = (SELECT MAX(rating) FROM Sailors);
```

OUTPUT

```
 sname | rating
-------+--------
 Rusty |     10
 Zorba |     10
(2 rows)
```

25. Find the average age of all sailors.

```
SELECT AVG(age) FROM Sailors;
```

OUTPUT

```
 avg
------
 36.9
(1 row)
```

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

SELECT AVG(age) FROM Sailors
WHERE rating = 10;

OUTPUT

```
 avg
------
 25.5
(1 row)
```

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

SELECT sname,age FROM Sailors
WHERE age = (SELECT MAX(age) FROM Sailors);

OUTPUT

```
 sname | age
-------+------
 Bob   | 63.5
(1 row)
```

28. Count the number of sailors.

SELECT COUNT(*) FROM Sailors;

OUTPUT

```
 count
-------
    10
(1 row)
```

29. Count the number of different sailor names.

SELECT COUNT (DISTINCT sname) FROM Sailors;

OUTPUT

```
 count
-------
     9
(1 row)
```

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

```
SELECT sname FROM Sailors
WHERE age > (
        SELECT MAX(age) FROM Sailors
        WHERE rating = 10
);
```

OUTPUT

```
 sname
--------
 Dustin
 Lubber
 Bob
(3 rows)
```

31. Find the age of the youngest sailor for each rating level.

```
SELECT rating, MIN(age) AS youngest_age
FROM Sailors
GROUP BY rating;
```

OUTPUT

```
 rating | youngest_age
--------+--------------
      9 |           35
      3 |         25.5
     10 |           16
      7 |           35
      1 |           33
      8 |         25.5
(6 rows)
```

32. Find the age of the youngest sailor who is eligible to vote (i.e., is at least 18 years old) for each rating level with at least two such sailors.

```
SELECT rating, MIN(age) AS youngest_age
FROM Sailors
WHERE age >= 18
GROUP BY rating
HAVING COUNT(*) >=2;
```

OUTPUT

```
 rating | youngest_age
--------+--------------
      3 |         25.5
      7 |           35
      8 |         25.5
(3 rows)
```

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

```
SELECT B.bid, B.bname, COUNT(R.sid) AS reservation_count
FROM Boats B
LEFT JOIN Reserves R ON B.bid = R.bid
WHERE B.color = 'red'
GROUP BY B.bid, B.bname;
```

OUTPUT

```
 bid |   bname    | reservation_count
-----+------------+-------------------
 102 | Interlake  |                 3
 104 | Marine     |                 2
(2 rows)
```

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

```
SELECT rating, AVG(age) AS average_age
FROM Sailors
GROUP BY rating HAVING COUNT(*) >=2;
```

OUTPUT

```
 rating | average_age
--------+-------------
      3 |        44.5
     10 |        25.5
      7 |          40
      8 |        40.5
(4 rows)
```

35. Find the average age of sailors who are of voting age (i.e., at least 18 years old) for each rating level that has at least two sailors.

```
SELECT S.rating , AVG(S.age) AS avg_age
FROM Sailors S
WHERE S.age >= 18
GROUP BY S.rating
HAVING 1 <(
        SELECT COUNT(*) FROM Sailors S2
        WHERE S2.rating = S.rating AND S2.age >= 18
);
```

OUTPUT

```
 rating | avg_age
--------+---------
      3 |    44.5
      7 |      40
      8 |    40.5
(3 rows)
```

36. Find the average age of sailors who are of voting age (i.e., at least 18 years old) for each rating level that has at least two such sailors.

```
SELECT Temp.rating, Temp.avg_age
FROM(
        SELECT S.rating, AVG(S.age) AS avg_age,COUNT(*) AS rating_count
        FROM Sailors S
        WHERE S.age >= 18
        GROUP BY S.rating
) AS Temp
WHERE Temp.rating_count > 1;
```

OUTPUT

```
 rating | avg_age
--------+---------
      3 |    44.5
      7 |      40
      8 |    40.5
(3 rows)
```

37. Find those ratings for which the average age of sailors is the minimum over all ratings.

```
SELECT rating
FROM (
        SELECT rating, AVG(age) AS avg_age
        FROM Sailors
        GROUP BY rating
) AS AvgAges
WHERE avg_age = (
        SELECT MIN(avg_age)
        FROM (
                SELECT AVG(age) AS avg_age
                FROM Sailors
                GROUP BY rating
        ) AS InnerAvg
);
```

OUTPUT