**INTRODUCTION**

### Database Management System

A database management system (DBMS) is a software package designed to define, manipulate, retrieve and manage data in a database..

### Relational Database Management System

RDBMS is acronym for Relation Database Management System. Dr. E.F.Codd first introduced the Relational Database Model in 1970. The Relational model allows data to be represented in a simple row- column. Each data field is considered as a column and each record is considered as a row. Relational Database is more or less similar to Database Management System.

**Introduction to ORACLE:**

ORACLE is a powerful RDBMS product that provides efficient and effective solutions for major database features. This includes:

* Large databases and space management control
* Many concurrent database users
* High transaction processing performance
* Manageable security
* Database enforced integrity
* Client/Server environment
* Distributed database systems
* Portability

**Exp No: 1**

**AIM: Working with ER Diagram and Normalization**

**Example: ER Diagram for Sailors Database Entities:**

1. **Sailor**
2. **Boat Relationship: Reserves Primary Key Attributes:**
   1. **Sailor Entity - SID**
   2. **Boat Entity – BID**
   3. **Reserves Relationship -SID, BID, DAY**

**Following steps are involved in the process:**

1. Analyzing the problem and identifying the Entities and Relationships
2. E-R Model
3. Normalized Relational Model
4. Creating the database
5. Querying.
6. Triggers and Stored procedures on the tables.

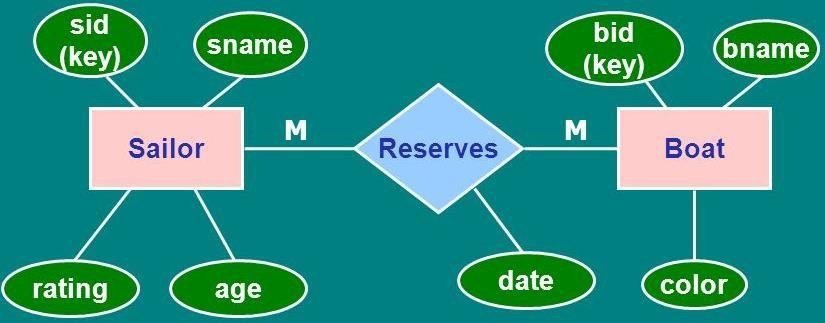
### E-R Model:

Analyze the problem carefully and come up with the entities in it. Identify what data has to be persisted in the database. This contains the entities, attributes etc. In this we will analyze different types of entities with attributes of “Roadways Travels”.

**Entity:** An Entity is an object or concept about which you want to store information

**Relationship:** A relationship is an association between several entities.

**Attributes:** An object is characterized by its properties or attributes. In relational database systems attributes correspond to fields.



### Attributes of the Sailor Entity

|  |  |
| --- | --- |
| **Attribute** | **Description** |
| Sid | A sailor—each sailor is assigned a unique ID |
| Sname | 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) |
| Bname | The name of the boat (also painted on the bow) |
| Color | The color of the boat |

The notation {PK} is used to denote primary keys. You can see that Sid and bid are both primary keys, meaning that values of these attributes can be used to uniquely identify each row of their respective tables

The "Reservation" table is a little more complex. It consists of the attributes in below table.

### Attributes of the Reservation Table

|  |  |
| --- | --- |
| **Attribute** | **Description** |
| Sid | Sailor ID |
| Bid | Boat ID |
| Date | The date and time of the reservation |
| duration | The expected duration of the reservation in minutes |

Notice that the "Reservation" table is a bridge table because it allows data from two other tables to be combined. Recall that these tables, "Sailor" and "Boat," are known as base tables.

The attributes SID and BID are called foreign keys. The notation {FK} is used to indicate this fact. In combination, the attributes "sid," "bid," and "date" make up the primary key for the table. The notation {PK} is used to indicate this. Therefore, "sid" and "bid" are individually foreign keys, plus they are part of the primary key.

The label "Makes" tells us how to read the relationship between "Sailor" and "Reservation." Thus, we see that a sailor "makes" a reservation. We can also see that a boat is "allocated" to a reservation.

The \* and 1 indicate the multiplicity of the relationships. In short, the model states that 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. In other words, these are one-to-many relationships.

**The Schemas:**

Sailors(sid: integer, sname: string, rating: integer, age: real) Boats(bid: integer, bname: string, color: string) Reserves(sid: integer , bid: integer , day: date)

**Exp No: 2 and 3**

**AIM**: **Working with DDL, DML, DCL and Key Constraints**

**Data types in Oracle. Creation (Use Constraints like Primary key, Foreign key and Check while Creating Tables), Altering and Dropping of Tables.**

**Inserting records into a Table, Updating and Deleting records from table. Simple queries using Select Command.**

**COMMIT and ROLLBACK commands.**

### SQL is structure query language. SQL contains different data types those are

**SQL commands**

**Create Database**

**Syntax**

CREATE DATABASE Database\_name;

Example

Create Database Mydatabase;

**Showing all Databases**

SHOW DATABASES;

**Deleting an database**

DROP DATABASE Database\_name;

**Create table**

**Syntax**

CREATE TABLE table\_name

(column1 datatype,

column2 datatype,

column3 datatype,

...

columnN datatype,

PRIMARY KEY (one or more columns)

);

**Sailor Table**

CREATE TABLE Sailors

(SID INT NOT NULL,

SNAME VARCHAR(15),

RATING INT CHECK (RATING >= 1 AND RATING <= 10),

AGE REAL,

PRIMARY KEY (SID));

**Boat Table**

CREATE TABLE Boats

(BID INT NOT NULL,

BNAME VARCHAR(20),

COLOR VARCHAR(8),

PRIMARY KEY (BID));

**Reserves Table**

CREATE TABLE Reserves

(

SID INT NOT NULL,

BID INT NOT NULL,

DAY DATE NOT NULL,

PRIMARY KEY (SID, BID, DAY),

FOREIGN KEY (SID) REFERENCES Sailors(SID),

FOREIGN KEY (BID) REFERENCES Boats(BID)

);

**Inserting values into table**

**Inserting values for all columns in order**:

Syntax

INSERT INTO TableName

VALUES (value1, value2, value3, ...);

Example

INSERT INTO Sailors

VALUES ( 22, 'Dustin', 7, 45.0)

**Inserting values for specific columns**:

Syntax

INSERT INTO TableName (column1, column2, ...)

VALUES (value1, value2, ...);

Examples

INSERT INTO Sailors (sid, age, rating, sname)

VALUES (29, 33.0, 1, 'Brutus');

INSERT INTO Sailors (sid, sname, rating, age)

VALUES (31, 'Lubber', 8, 55.5);

**Inserting multiple rows at once**:

Syntax

INSERT INTO TableName (column1, column2, ...)

VALUES (value1\_1, value1\_2, ...),

(value2\_1, value2\_2, ...),

...;

Examples

INSERT INTO Sailors (sid, sname, rating, age)

VALUES

(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);

**Displaying an table**

Select \* from Sailors;

**Inserting values into Boat table**

INSERT INTO boats (bid, bname, color) VALUES (101, 'Interlake', 'blue');

INSERT INTO boats (bid, bname, color) VALUES (102, 'Interlake', 'red');

INSERT INTO boats (bid, bname, color) VALUES (103, 'Clipper', 'green');

INSERT INTO boats (bid, bname, color) VALUES (104, 'Marine', 'red');

**Inserting values into Reserves table**

INSERT INTO Reserves (sid, bid, day) VALUES (22, 101, '1998-10-10');

INSERT INTO Reserves (sid, bid, day) VALUES (22, 102, '1998-10-10');

INSERT INTO Reserves (sid, bid, day) VALUES (22, 103, '1998-08-10');

INSERT INTO Reserves (sid, bid, day) VALUES (22, 104, '1998-07-10');

INSERT INTO Reserves (sid, bid, day) VALUES (31, 102, '1998-10-11');

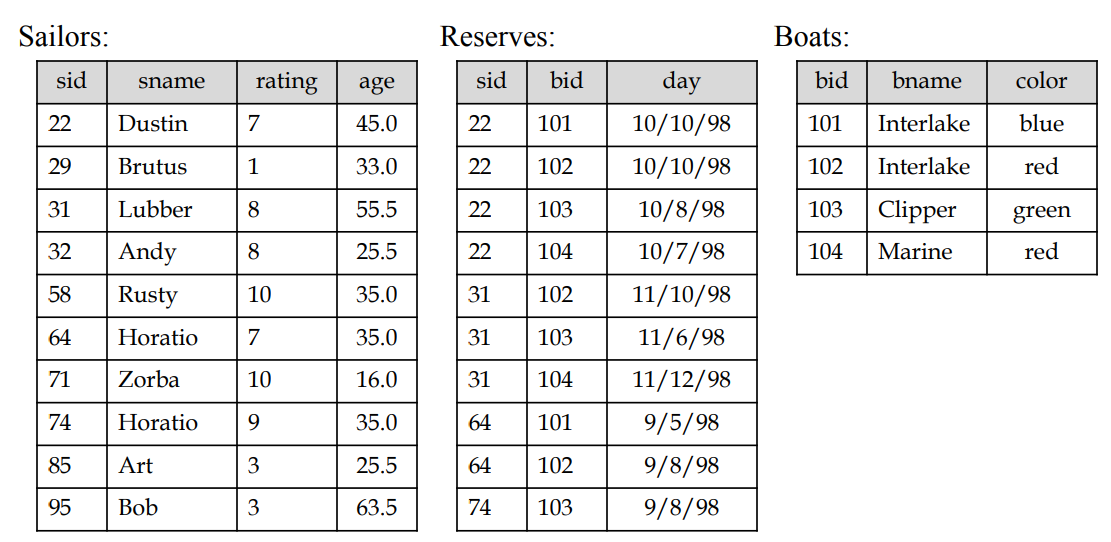
INSERT INTO Reserves (sid, bid, day) VALUES (31, 103, '1998-06-11');

INSERT INTO Reserves (sid, bid, day) VALUES (31, 104, '1998-12-11');

INSERT INTO Reserves (sid, bid, day) VALUES (64, 101, '1998-05-09');

INSERT INTO Reserves (sid, bid, day) VALUES (64, 102, '1998-08-09');

INSERT INTO Reserves (sid, bid, day) VALUES (74, 103, '1998-08-09');



**Alter command**

**Add column**

if you want to alter the structure of the existing tables in the database, you can use the ALTER TABLE command.

Here's an example of altering the **Sailors** table to add a new column named **gender**:

ALTER TABLE Sailors

ADD COLUMN PhoneNo VARCHAR(15);

If you want to modify an existing column, such as changing its data type or modifying constraints, you can use the **ALTER TABLE** command with the **MODIFY COLUMN** or **ALTER COLUMN** clause.

For example, to change the data type of the rating column in the Sailors table to INTEGER:

ALTER TABLE Sailors

ALTER COLUMN rating INTEGER;

**DROP :**

This statement is used to drop an existing database. When you use this statement, complete information present in the database will be lost.

Syntax

DROP DATABASE DatabaseName;

Example DROP DATABASE Mydatabase;

**The ‘DROP TABLE’ Statement**

This statement is used to drop an existing table. When you use this statement, complete information present in the table will be lost.

Syntax

DROP TABLE TableName;

Example DROP Table Sailors;

**TRUNCATE**

This command is used to delete the information present in the table but does not delete the table. So, once you use this command, your information will be lost, but not the table.

Syntax:

TRUNCATE TABLE table\_name;

Example:

TRUNCATE TABLE Sailors;

**Data Manuplation Languages**

**Insert :** Already discussed

**UPDATE:**

This command is used to update or modify the value of a column in the table.

Syntax:

UPDATE table\_name

SET column1= values

column2= values

columnN = value

WHERE CONDITION;

**For example:**

UPDATE Sailors SET sname = 'Bob Ray' WHERE sid = '95';

**DELETE:**

It is used to remove one or more row from a table.

Syntax1:

DELETE FROM table\_name;

Syntax2

DELETE FROM table\_name WHERE condition;

Example1: Delete all rows from boat table

DELETE FROM Boats;

Example2: Delete all rows from boat whose boatname is Interlake

DELETE FROM Boats WHERE bname="Interlake";

**TCL commands:**

TCL commands can only use with DML commands like INSERT, DELETE and UPDATE only.

These operations are automatically committed in the database that's why they cannot be used while creating tables or dropping them.

Here are some commands that come under TCL:

o COMMIT

o ROLLBACK

o SAVEPOINT

**a. Commit:**

Commit command is used to save all the transactions to the database.

Syntax:

COMMIT;

Example:

DELETE FROM CUSTOMERS WHERE AGE = 25;

COMMIT;

**b. Rollback:**

Rollback command is used to undo transactions that have not already been saved to the database.

Syntax:

ROLLBACK;

Example:

DELETE FROM CUSTOMERS WHERE AGE = 25;

ROLLBACK;

**c. SAVEPOINT:**

It is used to roll the transaction bacs to a certain point without rolling bacs the entire transaction.

Syntax:

SAVEPOINT SAVEPOINT\_NAME;

**Data Query Language**

DQL is used to fetch the data from the database.

**SELECT**

This statement is used to select data from a database and the data returned is stored in a result table, called the result-set.

Syntax

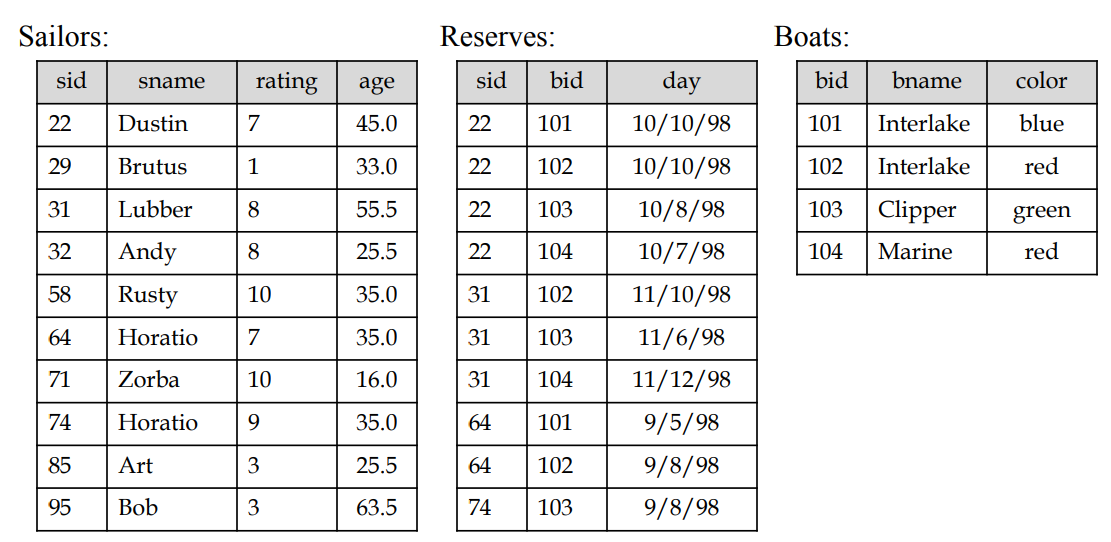
SELECT Column1, Column2, ...ColumN FROM TableName;

(\*) is used to select all from the table

SELECT \* FROM table\_name;

To select the number of records to return use:

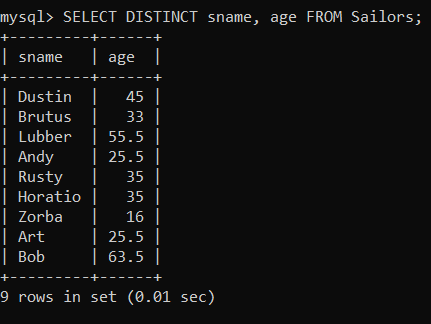
SELECT TOP 3 \* FROM TableName;



**Queries**

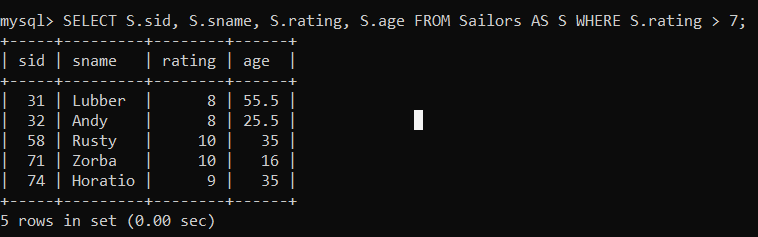
1. Find the names and ages of all Sailors.

SELECT DISTINCT sname, age FROM Sailors



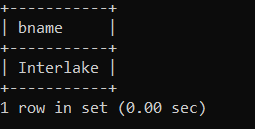
1. Find all Sailors with a rating above 7.

SELECT S.sid, S.sname, S.rating, S.age FROM Sailors AS S WHERE S.rating > 7



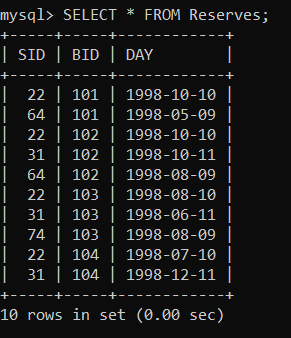
1. Find all boat name with a boat color blue.

SELECT bname FROM Boats WHERE color=blue;



1. Find details of all the reservtions.

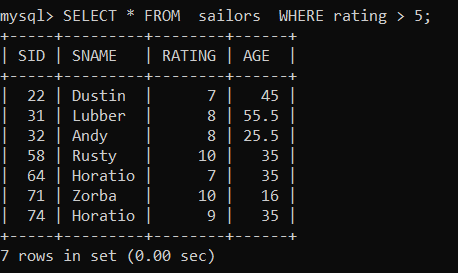
SELECT \* FROM Reserves;



**Practice**

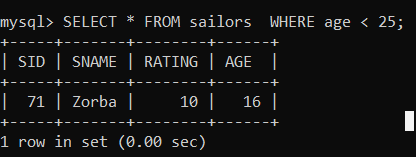
1. Selecting sailors with a rating greater than 5:

SELECT \* FROM sailors WHERE rating > 5;



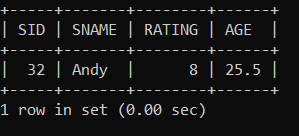
1. Selecting sailors younger than 25:

SELECT \* FROM sailors WHERE age < 25;



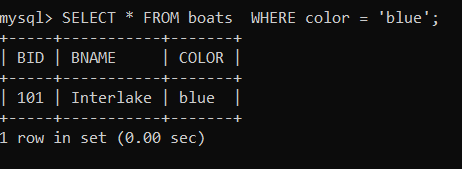
1. Selecting sailors named 'Andy':

SELECT \* FROM sailors WHERE name = 'Andy';



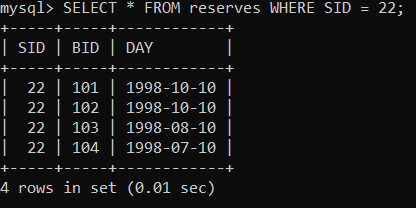
1. Selecting boats with a color of 'blue':

SELECT \* FROM boats WHERE color = 'blue';



9.Selecting reservations made by a specific sailor:

SELECT \* FROM reservations WHERE SID = 22;

****

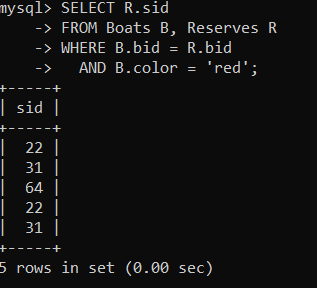
10. Find the sids of Sailors who have reserved a red boat.

SELECT R.sid

FROM Boats B, Reserves R

WHERE B.bid = R.bid

AND B.color = 'red';

****

**Exp No: 4**

### AIM: Working with Queries and Nested QUERIES

Queries using Set Operations - UNION, INTERSETION

Nested Queries using IN, NOT IN, EXISTS, NOT EXISTS, ANY, ALL

### SQL Operators: Here is a list of all the logical operators available in SQL.

|  |  |
| --- | --- |
| **Operator** | **Description** |
| IN | The IN operator is used to compare a value to a list of literal values that have been specified. |
| NOT | The NOT operator reverses the meaning of the logical operator with which it is used. e.g.: NOT EXISTS, NOT BETWEEN, NOT IN, etc. |
| EXISTS | The EXISTS operator is used to search for the presence of a row in a specified table that meets certain criteria. |
| ANY | The ANY operator is used to compare a value to any applicable value in the list according to the condition. |
| ALL | The ALL operator is used to compare a value to all values in another value set. |
| AND | The AND operator allows the existence of multiple conditions in an SQL statement's WHERE clause. |
| BETWEEN | The BETWEEN operator is used to search for values that are within a set of values, given the minimum value and the maximum value. |
| LIKE | The LIKE operator is used to compare a value to similar values using wildcard operators. |
| OR | The OR operator is used to combine multiple conditions in an SQL statement's WHERE clause. |
| IS NULL | The NULL operator is used to compare a value with a NULL value. |
| UNIQUE | The UNIQUE operator searches every row of a specified table for uniqueness (no duplicates). |

**Union:** The union operator returns all distinct rows selected by two or more queries. It combines the result of two queries with the union operator, which eliminates duplicate rows.

**Union All :** The union all operators returns all rows selected by either query including duplicates. The following example combines the result with the aid of union all operator, which does not eliminates duplicate rows.

**Intersect :** The i n t e r s e c t operator o u t p u t s o n l y r o w s produced by both the queries intersected i.e. the output in an intersect clause will include only those rows that are retrieved by both the queries.

**Minus :** The Minus operator outputs the rows produced by the first query, after filtering the rows retrieved by the second query.

### SQL UNION Statement:

SELECT column1 [, column2 ] FROM table1 [, table2 ] [WHERE condition]

##### **UNION**

SELECT column1 [, column2 ] FROM table1 [, table2 ] [WHERE condition]

### SQL INTERSECT Statement:

SELECT column1 [, column2 ] FROM table1 [, table2 ] [WHERE condition]

##### **INTERSECT**

SELECT column1 [, column2 ] FROM table1 [, table2 ] [WHERE condition]

select s.sid,s.sname

from sailors s,boats b,reserves r

where s.sid=r.sid and b.bid=r.bid and b.color='red'

### MINUS

select s.sid,s.sname

from sailors s,boats b,reserves r

where s.sid=r.sid and b.bid=r.bid and b.color='green';

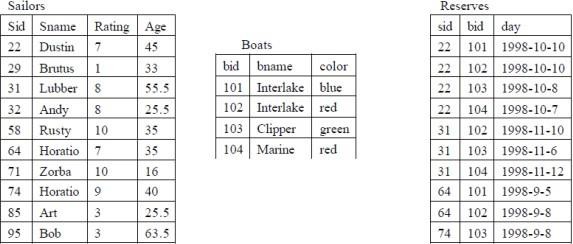
**EXAMPLES:**

Fig: Instances of Sailors, Boats, Reserves Find the ids of sailors who have reserved a red boat or a green boat.

SELECT R.sid

FROM Boats B, Reserves R

WHERE R.bid = B.bid AND B.color = ‘red’

UNION

SELECT R2.sid

FROM Boats B2, Reserves R2

WHERE R2.bid = B2.bid AND B2.color = ‘green’ The answer contains: SID 22 31 64 74

The default for UNION queries is that **duplicates are eliminated**. To retain duplicates, use UNION ALL.

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 )

select s.sname from sailors s

where s.sid **not in** (select r.sid from reserves r

where r.bid=103);

The inner subquery has been completely independent of the outer query.

**(Correlated Nested Queries)**

SELECT S.sname FROM Sailors S

WHERE **EXISTS** ( SELECT \*

FROM Reserves R WHERE R.bid = 103

AND **R.sid = S.sid** )

The inner query depends on the row that is currently being examined in the outer query.

**Find sailors who’ve 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

)

);

**An alternative solution:**

SELECT S.sname FROM Sailors S

WHERE NOT EXISTS (SELECT B.bid

FROM Boats B

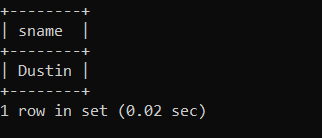
WHERE NOT EXISTS (SELECT R.bid

FROM Reserves R WHERE R.bid=B.bid

AND R.sid=S.sid))

Note that Union, Intersect and Except can be used on only two tables that are union- compatible,

that is, have the same number of columns and the columns, taken in order, have the same types.

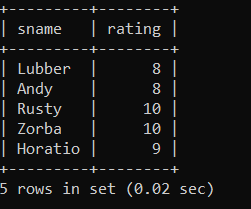


**Find the names and ratings of sailor whose rating is better than some sailor called Horatio.**

SELECT S.sname, S.rating FROM Sailors S

WHERE S.rating > ANY ( SELECT S2.rating FROM Sailors S2

WHERE S2.sname = ‘Horatio’ )

****

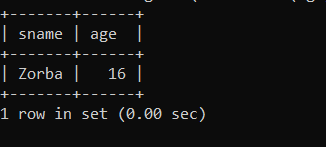
**Find the name and the age of the youngest sailor.**

SELECT S.sname, S.age

FROM Sailors S

WHERE S.age = (SELECT MIN(age) FROM Sailors);

Note that IN and NOT IN are equivalent to = ANY and <> ALL, respectively.



**Find sailors whose rating is greater than that of some sailor called Horatio:**

SELECT \*

FROM Sailors S

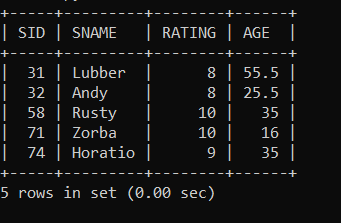
WHERE S.rating > ANY (

SELECT S2.rating

FROM Sailors S2

WHERE S2.sname = 'Horatio'

);



**Find sid’s of sailors who’ve reserved both a red and a green boat:**

SELECT S.sid

FROM Sailors S, Boats B, Reserves R

WHERE S.sid = R.sid AND R.bid = B.bid AND B.color = 'red'

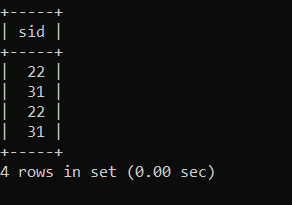
AND S.sid IN (

SELECT S2.sid

FROM Sailors S2, Boats B2, Reserves R2

WHERE S2.sid = R2.sid AND R2.bid = B2.bid AND B2.color = 'green'

);

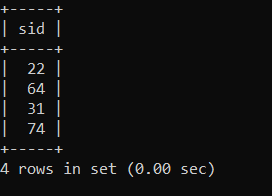


**Find sailors who’ve reserved at least one boat**

SELECT DISTINCT S.sid

FROM Sailors S, Reserves R

WHERE S.sid = R.sid;



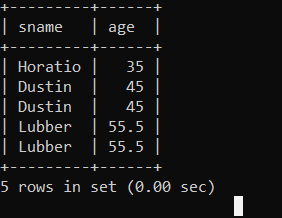
**Find the names of sailors who have reserved a red boat, and list in the order of age.**

SELECT S.sname, S.age

FROM Sailors S, Reserves R, Boats B

WHERE S.sid = R.sid AND R.bid = B.bid AND B.color = 'red'

ORDER BY S.age ASC;



**Exp No: 5 & 6**

**AIM: Working with Queries USING Aggregate Operators & views**

**Queries using Aggregate Functions like COUNT, SUM, AVG, MAX and MIN. Queries using ORDER BY Clause.**

**Queries using GROUP BY and HAVING clauses. Creation of Views, Queries on Views and Dropping Views.**

**Aggregate operators:** In addition to simply retrieving data, we often want to perform some computation or summarization.

**Group Functions:** A group functions returns a result based on a group of rows.

**To create tables**

Sailor Table

CREATE TABLE Sailors

(SID INT NOT NULL,

SNAME VARCHAR(15),

RATING INT CHECK (RATING >= 1 AND RATING <= 10),

AGE REAL,

PRIMARY KEY (SID));

Boat Table

CREATE TABLE Boats

(BID INT NOT NULL,

BNAME VARCHAR(20),

COLOR VARCHAR(8),

PRIMARY KEY (BID));

Reserves Table

CREATE TABLE Reserves

(

SID INT NOT NULL,

BID INT NOT NULL,

DAY DATE NOT NULL,

PRIMARY KEY (SID, BID, DAY),

FOREIGN KEY (SID) REFERENCES Sailors(SID),

FOREIGN KEY (BID) REFERENCES Boats(BID)

);

To insert values into sailor table

INSERT INTO Sailors

VALUES ( 22, 'Dustin', 7, 45.0);

INSERT INTO Sailors (sid, age, rating, sname)

VALUES (29, 33.0, 1, 'Brutus');

INSERT INTO Sailors (sid, sname, rating, age)

VALUES (31, 'Lubber', 8, 55.5);

INSERT INTO Sailors (sid, sname, rating, age)

VALUES

(31, '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);

Inserting values into Boat table

INSERT INTO boats (bid, bname, color) VALUES (101, 'Interlake', 'blue');

INSERT INTO boats (bid, bname, color) VALUES (102, 'Interlake', 'red');

INSERT INTO boats (bid, bname, color) VALUES (103, 'Clipper', 'green');

INSERT INTO boats (bid, bname, color) VALUES (104, 'Marine', 'red');

Inserting values into Reserves table

INSERT INTO Reserves (sid, bid, day) VALUES (22, 101, '1998-10-10');

INSERT INTO Reserves (sid, bid, day) VALUES (22, 102, '1998-10-10');

INSERT INTO Reserves (sid, bid, day) VALUES (22, 103, '1998-08-10');

INSERT INTO Reserves (sid, bid, day) VALUES (22, 104, '1998-07-10');

INSERT INTO Reserves (sid, bid, day) VALUES (31, 102, '1998-10-11');

INSERT INTO Reserves (sid, bid, day) VALUES (31, 103, '1998-06-11');

INSERT INTO Reserves (sid, bid, day) VALUES (31, 104, '1998-12-11');

INSERT INTO Reserves (sid, bid, day) VALUES (64, 101, '1998-05-09');

INSERT INTO Reserves (sid, bid, day) VALUES (64, 102, '1998-08-09');

INSERT INTO Reserves (sid, bid, day) VALUES (74, 103, '1998-08-09');

**1.Count:**

COUNT following by a column name returns the count of tuple in that column. If DISTINCT keyword is used then it will return only the count of unique tuple in the column. Otherwise, it will return count of all the tuples (including duplicates) count (\*) indicates all the tuples of the column.

**Syntax:**

COUNT (Column name)

Count(distinct column name)

This function is used to count number rows. It can take three different arguments, which mentioned below.

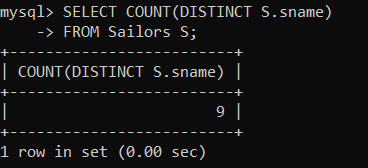
Syntax: Count(\*)

Count (\*): This will count all the rows, including duplicates and nulls.

Example: Count the number of different sailor names.

SELECT COUNT(DISTINCT S.sname)

FROM Sailors S;

****

**2.SUM:**

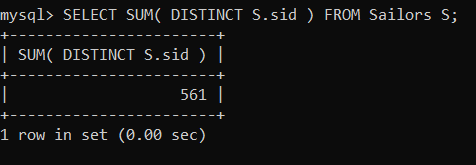
SUM followed by a column name returns the sum of all the values in that column.

**Syntax:**

SUM (Column name)

Example: sum the number of different sailor id.

SELECT SUM( DISTINCT S.sid ) FROM Sailors S;



**3.AVG:**

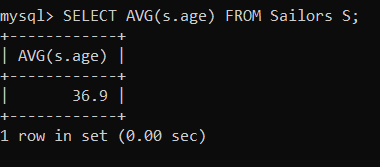
AVG followed by a column name returns the average value of that column values.

**Syntax:**

AVG (n1,n2..)

Example: Calculate the average age of all sailors.

SELECT AVG(s.age) FROM Sailors S;

****

**4. MAX:**

MAX followed by a column name returns the maximum value of that column.

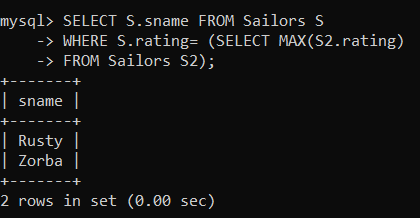
**Syntax**:

MAX (Column name) Example:

SELECT S.sname FROM Sailors S

WHERE S.rating= (SELECT MAX(S2.rating)

FROM Sailors S2);



1. **MIN:**

MIN followed by column name returns the minimum value of that column.

**Syntax:**

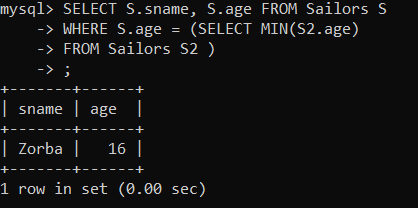
MIN (Column name)

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

SELECT S.sname, S.age FROM Sailors S

WHERE S.age = (SELECT MIN(S2.age)

FROM Sailors S2 );



**Order by Clause:**

Order by clause is used to arrange rows in either ascending or descending order. The order by clause can also be used to arrange multiple columns.

Select <column(s)> f r o m <Table Name> w h e r e [ condition(s)] [order by <column name> [asc/] desc];

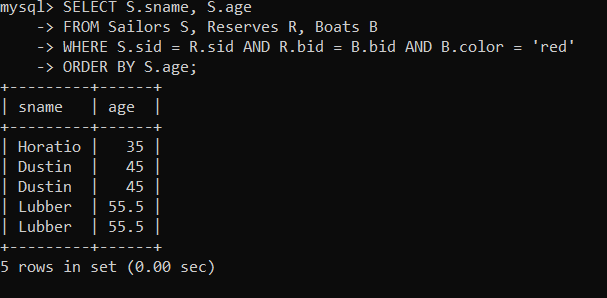
**Find the names of sailors who have reserved a red boat, and list in the order of age.**

SELECT S.sname, S.age

FROM Sailors S, Reserves R, Boats B

WHERE S.sid = R.sid AND R.bid = B.bid AND B.color = 'red'

ORDER BY S.age;Group by clause:



G r o u p by c l a u s e is used with group functions only. Normally group functions returns only one row. But group by clause will group on that column. The group by

clause tells Oracle to group rows based on distinct values for specified columns, i.e. it creates a data set, containing several sets of records grouped together based on a condition. Select group function from table name group by column name

Having Clause:

The having clause is used to satisfy certain conditions on rows, retrieved by using group by clause. Having clause should be preceding by a group by clause. Having clause further filters the rows return by group by clause.

**SYNTAX SQL GROUP BY & HAVING Statement:**

SELECT [DISTINCT] select-list FROM from-list

WHERE qualification GROUP BY grouping-list HAVING group-qualification

SQL GROUP BY Clause:

SELECT SUM (column\_name) FROM table\_name

WHERE CONDITION

GROUP BY column\_name;

SQL HAVING Clause:

SELECT SUM (column\_name) FROM table\_name

WHERE CONDITION

GROUP BY column\_name

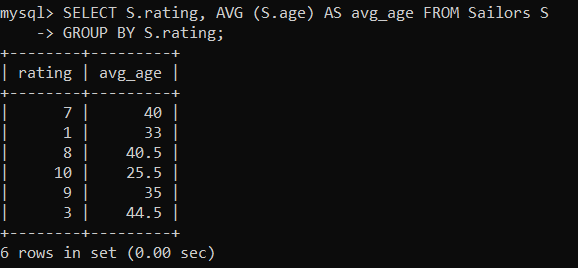
HAVING (arithmetic function condition);

EXAMPLES:

**Find the average age of sailors for each rating level.**

SELECT S.rating, AVG (S.age) AS avg\_age FROM Sailors S

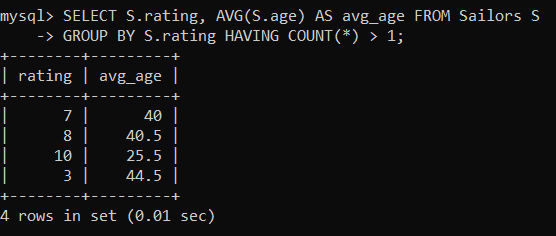
GROUP BY S.rating;



**Find the average 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;



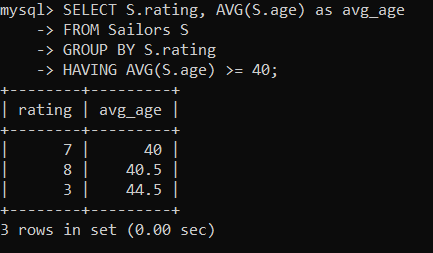
**An example shows difference between WHERE and HAVING:**

SELECT S.rating, AVG(S.age) as avg\_age

FROM Sailors S

GROUP BY S.rating

HAVING AVG(S.age) >= 40;



**VIEW:**

A view is a virtual table, which consists of a set of columns from one or more tables. It is similar to a table but it does not store in the database. View is a query stored as an object.

**Syntax:**

CREATE VIEW view\_name AS SELECT set of fields FROM relation\_name WHERE (Condition);

**SQL CREATE VIEW syntax:**

CREATE VIEW view\_name AS SELECT column1, column2..... FROM table\_name

WHERE [condition];

DROP VIEW:

This query is used to delete a view , which has been already created.

**Syntax:** DROP VIEW View\_name;

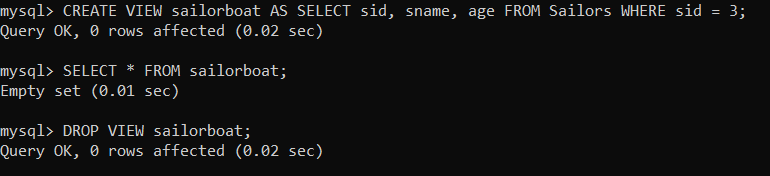
Example :

CREATE VIEW sailorboat AS SELECT sid,sname,age FROM SAILOR WHERE sid = 3;

View created.

SELECT \* FROM sailorboat;

DROP VIEW sailorboat;



**Exp No: 7 and 8**

**AIM: Working with Conversion Functions & String Functions**

**Queries using Conversion Functions - TO\_CHAR, TO\_NUMBER and TO\_DATE. Queries using String Functions - CONCAT, LPAD, RPAD, LTRIM, RTRIM,LOWER, UPPER,**

**INITCAP, LENGTH, SUBSTR, INSTR, TRANSLATE, and REPLACE.**

**Queries using Date Functions**

**SYSDATE, NEXT DAY, ADD MONTHS, LAST DAY,**

**MONTHS BETWEEN, LEAST, GREATEST, TRUNC and ROUND**.

**1. Conversion functions:**

**To\_char:**

TO\_CHAR (number) converts n to a value of VARCHAR2 data type, using the optional number format fmt. The value n can be of type NUMBER, BINARY\_FLOAT, or BINARY\_DOUBLE.

SQL> select to\_char (35,'RN') from dual;

TO\_CHAR (35,'RN')

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

XXXV

**To\_number:**  TO\_NUMBER converts expr to a value of NUMBER data type. SQL> Select To\_number ('1234.64') from Dual;

1234.64

**To\_date:**

TO\_DATE converts char of CHAR, VARCHAR2, NCHAR, or NVARCHAR2 data type to a value of DATE data type. - 55 –

SQL>SELECT TO\_DATE ('January 15, 1989, 11:00 A.M.') FROM DUAL;

TO\_DATE

15-JAN-89

**2. String functions:**

**Concat**:

CONCAT returns char1 concatenated with char2. Both char1 and char2 can be any of the data types

SQL> SELECT CONCAT(‘ORACLE’,’CORPORATION’)FROM DUAL;

ORACLECORPORATION

**Lpad:**

LPAD returns expr1, left-padded to length n characters with the sequence of characters in expr2.

SQL>SELECT LPAD(‘ORACLE’,15,’\*’)FROM DUAL;

\*\*\*\*\*\*\*\*\*\*ORACLE

**Rpad:**

RPAD returns expr1, right-padded to length n characters with expr2, replicated as many times as necessary.

SQL>SELECT RPAD (‘ORACLE’,15,’\*’)FROM DUAL;

ORACLE\*\*\*\*\*\*\*\*\*

**Ltrim:**

Returns a character expression after removing leading blanks.

SQL>SELECT LTRIM(‘SSMITHSS’,’S’)FROM DUAL;

MITHSS

**Rtrim:**

Returns a character string after truncating all trailing blanks

SQL>SELECT RTRIM(‘SSMITHSS’,’S’)FROM DUAL;

SSMITH

**Lower:**

Returns a character expression after converting uppercase character data to lowercase. SQL>SELECT LOWER(‘DBMS’)FROM DUAL;

dbms

**Upper:**

Returns a character expression with lowercase character data converted to uppercase SQL>SELECT UPPER(‘dbms’)FROM DUAL;

DBMS

**INITCAP**

SQL> SELECT INITCAP('hello world from sql') FROM DUAL;

INITCAP('HELLOWORLDFROMSQL')

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

Hello World From Sql

**Length:**

Returns the number of characters, rather than the number of bytes, of the given string expression, excluding trailing blanks.

SQL>SELECT LENGTH(‘DATABASE’)FROM DUAL;

8

**Substr:**

Returns part of a character, binary, text, or image expression.

SQL>SELECT SUBSTR(‘ABCDEFGHIJ’,3,4) FROM DUAL;

CDEF

**Instr:**

The INSTR functions search string for substring. The function returns an integer indicating the position of the character in string that is the first character of this occurrence.

SQL>SELECT INSTR('CORPORATE FLOOR','OR',3,2)FROM DUAL;

12

**TRANSLATE**

SQL> SELECT TRANSLATE('HELLO WORLD', 'HE', 'YX') FROM DUAL;

TRANSLATE('HELLOWORLD','HE','YX')

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

YXLLX WORLD

**REPLACE**

SQL> SELECT REPLACE('HELLO WORLD', 'WORLD', 'SQL') FROM DUAL;

REPLACE('HELLOWORLD','WORLD','SQL')

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

HELLO SQL

**3. Date functions:**

**Sysdate:**

SQL>SELECT SYSDATE FROM DUAL;

SYSDATE

---------

08-JUN-24

**next\_day:**

SQL>SELECT NEXT\_DAY(SYSDATE, 'MONDAY') FROM DUAL;

NEXT\_DAY(SYSDATE, 'MONDAY')

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

10-JUN-24

**add\_months:**

SELECT ADD\_MONTHS(SYSDATE, 3) FROM DUAL;

ADD\_MONTHS(SYSDATE, 3)

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

08-SEP-24

**last\_day:**

SQL>SELECT LAST\_DAY(SYSDATE)FROM DUAL;

LAST\_DAY(SYSDATE)

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

30-JUN-24

**months\_between:**

SQL> SELECT MONTHS\_BETWEEN('01-JAN-2024', '01-JUN-2023') FROM DUAL;

MONTHS\_BETWEEN('01-JAN-2024', '01-JUN-2023')

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

7

**Least:**

SQL>SELECT LEAST('10-JAN-07','12-OCT-07')FROM DUAL;

10-JAN-07

**Greatest:**

SQL>SELECT GREATEST('10-JAN-07','12-OCT-07')FROM DUAL;

12-OCT-07

**Trunc:**

SQL> SELECT TRUNC(SYSDATE, 'MM') FROM DUAL; -- Truncate to the first day of the month

TRUNC(SYSDATE, 'MM')

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

01-JUN-24

**Round:**

SQL> SELECT ROUND(SYSDATE, 'MM') FROM DUAL; -- Round to the nearest month

ROUND(SYSDATE, 'MM')

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

01-JUN-24