

## **AIM: 1.1 Familiarization with installation of any DBMS.**

### **Description:**

#### **Introduction to Oracle 10g Express Edition and Installation**

##### **a. Overview of Oracle 10g Express Edition:**

Oracle Database 10g Express Edition (Oracle Database XE) is a free, downloadable version of the world's most capable relational database. Oracle Database XE is easy to install and easy to manage.

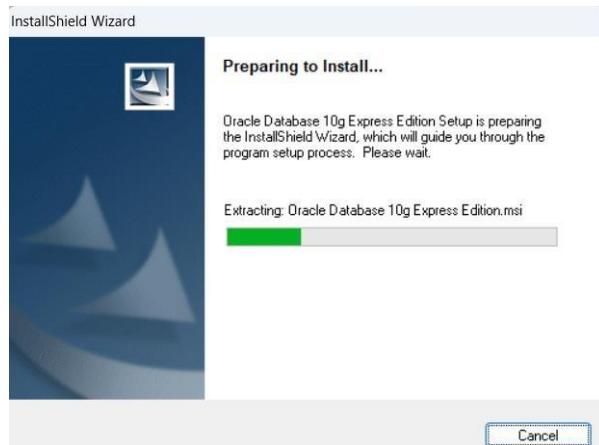
#### **Interfaces**

**Database Home Page:** Oracle XE included a web-based interface accessible through a browser, typically at <http://localhost:8080/apex>.

**SQL Command Line:** A simple command-line interface called "SQL Command Line" was included for direct SQL execution and basic administration tasks.

##### **b. Installation Steps**

Click on the first file named OracleXE.exe to begin your downloading



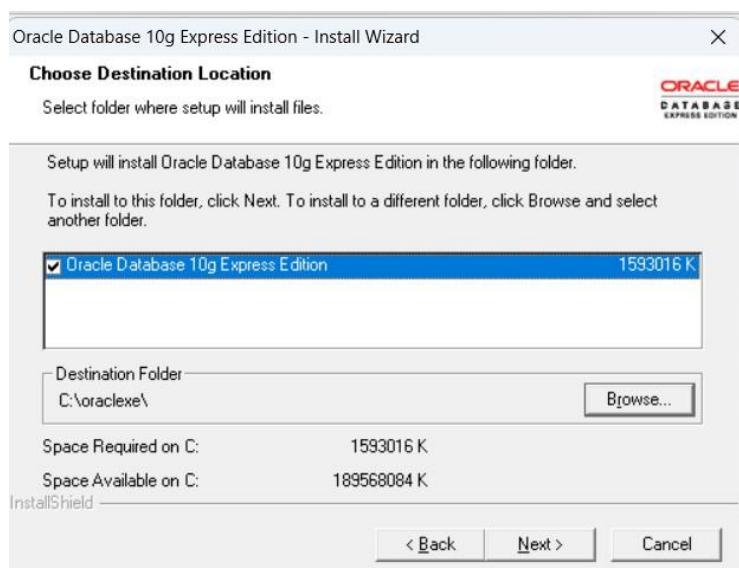
Click the Next button to begin to install the Oracle Database 10g Express Edition.



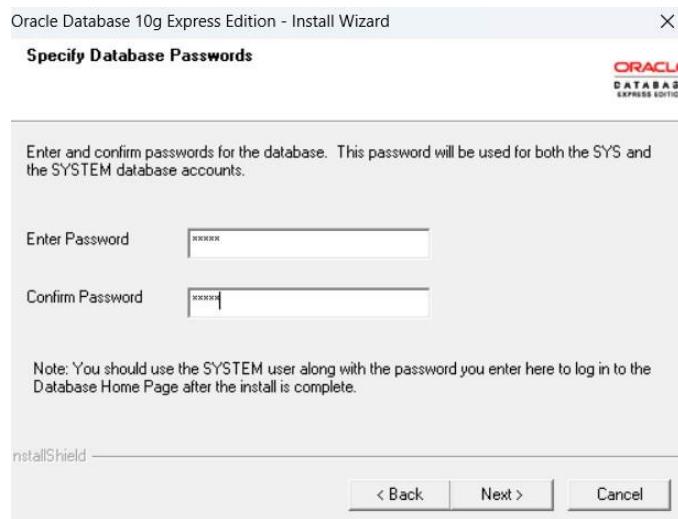
Select and click the Accept radio button followed by clicking the Next button to go to the next page.



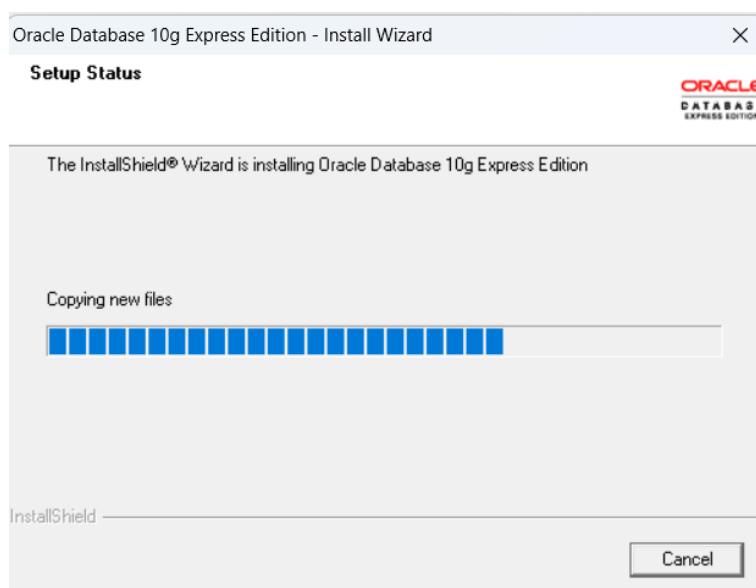
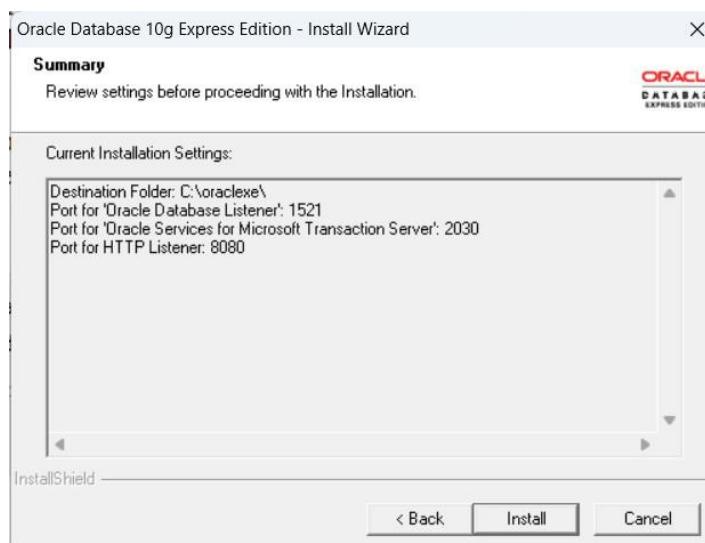
Select folder where setup will install files.



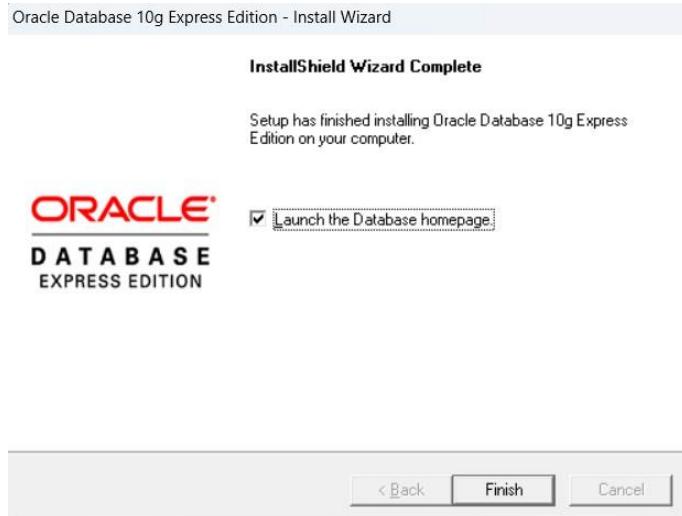
In the next dialog, you need to enter your password. You must remember this password since you need to use it to access your database installed in the Oracle server from your client computer later. The username is SYSTEM by default. This means that you need to use the SYSTEM as your username and the password you selected to open and access the database you created and installed in the Oracle server later.



Click the Next button after you entered your desired password, and then click the Install button to begin this installation.

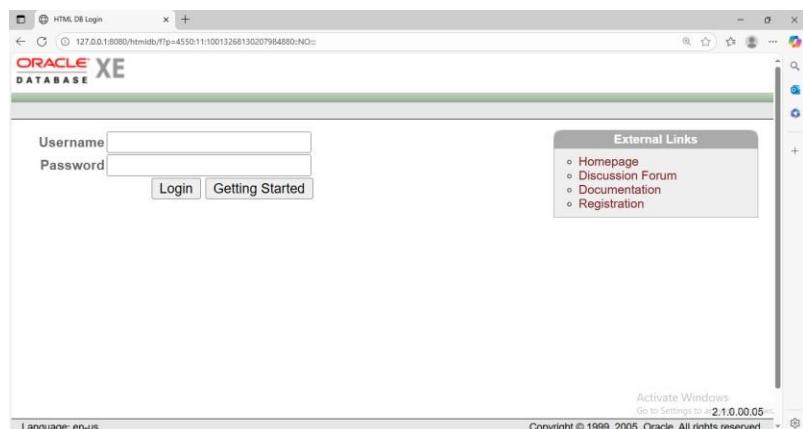


When this installation is complete, a finish dialog is displayed



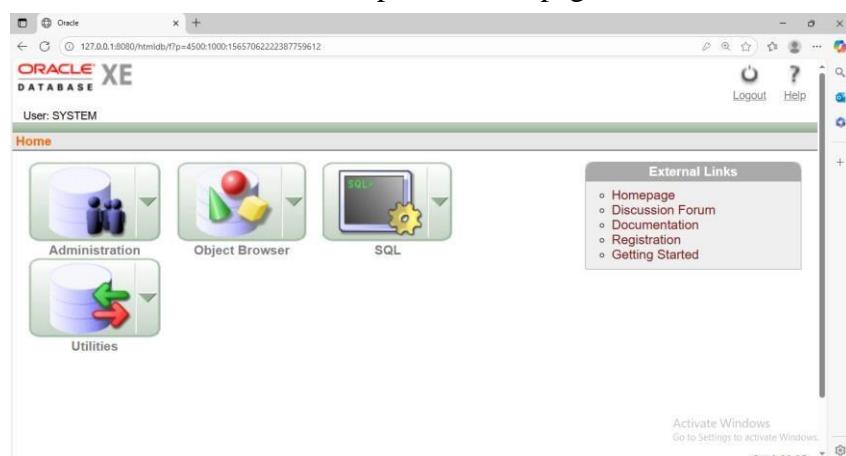
Click the Finish button to complete this installation.

The Database Login page is opened to allow you to enter your username and password to access the Oracle server to create, access and manipulate the database you created in this server.



Enter the SYSTEM as your username and the password you selected when you installed the server, and then click the Login button to open the home page.

There are four icons on the opened home page



At this point, you have successfully finished the download and installation of the Oracle Database 10g Express Edition, which include the Oracle Database 10g Express server and the client. A Database Started Guide icon has been automatically added to your desktop on your computer. You can open this icon to follow the guide to access your database if you like. To connect your client to the server and start creating, accessing and manipulating your database, you need to open the Database home page by clicking the Start>All Program Files\Oracle Database 10g Express Edition\Go To Database Home Page. Then you can create and manipulate your database staring from the Object Browser icon.

## **AIM: 1.2 Implementing a University Database System.**

### **Description:**

#### **a. Schema**

It is overall design of the database. It tells how many tables are there, what are their attributes, and how they are related. It includes table names, column names(attributes), datatypes, and keys.

#### **Syntax of Schema:**

Table Name (column1 : datatype, column2: datatype, column3: datatype, ......., columnN: datatype)

#### **b. Attributes per Table**

- Students Table**

Attributes: StudentID, StudentName, Major

- Courses Table**

Attributes: CourseID, CourseName, Credits

- Enrollments Table**

Attributes: StudentID, CourseID, EnrollmentDate

- Instructors Table**

Attributes: InstructorID, InstructorName, Phone

- Course\_Instructors Table**

Attributes: CourseID, InstructorID

#### **c. Schema for University Database:**

Students (StudentID:string, StudentName:string, Major:string)

Courses (CourseID:string, CourseName:string, Credits:integer)

Enrollments (StudentID:string, CourseID:string, EnrollmentDate:date)

Instructors (InstructorID:integer, InstructorName:string, Phone:integer)

Course\_Instructors (CourseID:string, InstructorID:integer)

## **AIM:2.1 Querying and modifying the database using Data Manipulation Language commands -select, insert, update, delete**

### **Description:**

DML COMMANDS are INSERT, UPDATE, DELETE and SELECT.

#### **INSERT COMMAND:**

This command is used to create data into the table which is already defined through DDL commands. The data can be entered in the form of rows and columns.

**Syntax:** INSERT INTO <Table name>

```
(column1, [column2, ..... ,columnN])  
values (column1value,column2value,..... ,columnNvalue);  
OR
```

```
INSERT INTO <Table name>  
Values (column1value,column2value,.....,columnNvalue);
```

#### **UPDATE COMMAND:**

This command is used to modify or change or replace the existing data of a table.

**Syntax:** UPDATE <Table\_name>  
Set <column1>=<column1value>  
[,<column2>=<column2value>,.....  
,<columnN>=<columnNvalue>]  
[where<condition>];

#### **DELETE COMMAND:**

This command is used to remove a single row or multiple rows of a table.

**Syntax:** DELETE FROM<Table\_name>  
[where<condition>];

#### **SELECT COMMAND:**

This command is used to view a single row or multiple rows or single column or multiple columns of a table.

#### **Syntax:**

Select [distinct] column1,column2 as newname from table1,table2

Where condition

Group by columnname

Having condition

Order by columnname asc|desc

#### **Creating Students Table:**

create table students

(

```
rollno varchar2(30),  
name varchar2(30)
```

);

Table Created

#### **Inserting Data into the table**

```
insert into students values('24B11CS234','Bala');
```

1 row(s) inserted.

```
insert into students values('24B11CS381','Kiran');
```

1 row(s) inserted.

**Displaying Data from the table**

select \* from students;

ROLLNO	NAME
24B11CS234	Bala
24B11CS381	Kiran

select name from students;

NAME
Bala
Kiran

select \* from students9 where rollno='24B11CS234'

ROLLNO	NAME
24B11CS234	Bala

**Deleting a row from the table**

delete from students where rollno='24B11CS381';  
1 row(s) deleted.

**Updating a row in the table**

update students set name='Bala Raju' where rollno='24B11CS381';  
1 row(s) updated.

## **AIM: 2.2 Implementation of Aggregate Functions – sum, avg, min, max, count.**

### **Use group-by and having clause.**

#### **Description:**

#### **Aggregate Functions**

**AVG function:** It can be used on numeric data or character data that contains only numeric's.

**MAX function:** It is used to find the maximum value of x. It can be used on any type of data.

**MIN function:** It is used to find the minimum value of x

**SUM function:** It sums the values and can be used on numeric data also.

**COUNT(\*):** It counts the number of rows in the table or the number of row in the group including NULL.

**Group by:** The attribute or attributes given in the clauses are used to form groups. Tuples with the same value on all attributes in the group by clause are placed in one group.

**Having:** SQL applies predicates (conditions) in the having clause after groups have been formed, so aggregate function be used.

#### **Source Table**

```
select * from company;  
companyn      amount  
wipro          5000  
ibm            8000  
dell           9000  
wipro          2000  
dell           10000
```

#### **Queries**

**Find the average salary of company**

Select AVG(amount) from company;

AVG(AMOUNT)
6800

**Find the Sum of salaries of company**

Select SUM(amount) from company;

SUM(AMOUNT)
34000

**Find the Maximum amount of company**

Select Max(amount) from company;

MAX(AMOUNT)
10000

**Find the Minimum amount of company**

Select Min(amount) from company;

MIN(AMOUNT)
2000

**Find the number of rows in a company**

Select Count(\*) from company;

COUNT(*)
5

**Find the sum of amount of each company.**

select companyn,sum(amount) from company group by companyn;

COMPANYN	SUM(AMOUNT)
wipro	7000
dell	19000
ibm	8000

**Find the minimum amount of each company.**

select companyn,min(amount) from company group by companyn;

COMPANYN	MIN(AMOUNT)
wipro	2000
dell	9000
ibm	8000

**Find the maximum amount of each company.**

select companyn,max(amount) from company group by companyn;

COMPANYN	MAX(AMOUNT)
wipro	5000
dell	10000
ibm	8000

**Find the count of all the rows grouped by each company name.**

select companyn,count(\*) from company group by companyn;

COMPANYN	COUNT(*)
wipro	2
dell	2
ibm	1

**Find the count of all the rows grouped by each company name & having count greater than 1.**  
select companyn,count(\*) from company group by companyn having count(\*)>1;

COMPANYN	COUNT(*)
wipro	2
dell	2

**Find the sum of amount of each company and having sum of amount greater than 10000.**  
select companyn,sum(amount) from company group by companyn having sum(amount)>10000;

COMPANYN	SUM(AMOUNT)
dell	19000

## **AIM: 3.1 Perform Join Operations-Natural Join, Equi-Join, Outer Join, Left Outer Join, Right Outer Join, Inner Join and assess the impact of query plans on the performance of join heavy queries.**

### **Description:**

JOIN Keyword is used in SQL queries for joining two or more tables.

### **Types of Joins**

#### **Inner Join**

The INNER JOIN keyword selects records that have matching values in both tables.

Syntax: select \* from tablename1 inner join tablename2 on condition

#### **Outer Join**

Outer Join is based on both matched and unmatched data.

It is divided into

- 1) Left outer join

The left outer join returns a resultset table with the matched data from the two tables and then the remaining rows of the left table and null from the right table's columns.

Syntax:

```
SELECT column-name-list FROM  
table-name1 LEFT OUTER JOIN table-name2  
ON table-name1.column-name = table-name2.column-name;
```

- 2) Right outer join

The right outer join returns a resultset table with the matched data from the two tables being joined, then the remaining rows of the right table and null for the remaining left table's columns.

Syntax:

```
SELECT column-name-list FROM  
table-name1 RIGHT OUTER JOIN table-name2  
ON table-name1.column-name = table-name2.column-name;
```

#### **Natural join**

It is based on column having same name and same datatype present in both the tables to be joined.

Syntax:

```
SELECT * FROM table-name1 NATURAL JOIN table-name2;
```

#### **Cross join**

It will return a table which consists of records which combines each row from the first table -with each row of the second table.

Syntax:

```
select * from tablename1,tablename2
```

#### **Self Join**

A **self join** is a **join** in which a table is joined with itself.

#### **EQUI Join**

An Equi Join in SQL is a type of join that combines rows from two or more tables based on a common column or set of columns, using only the equality operator (=) to compare the values in those columns.

Syntax

```
SELECT column-name-list FROM table1, table2....  
WHERE table1.column_name = table2.column_name;
```

### Source Tables

select \*from tb1;

RNO	NAME	MARKS
503	Suma	40
504	Raju	70
505	Ramu	45
501	Abhi	50
502	Ravi	60

select \* from tb2;

RNO	FEE
501	15000
502	5000
503	10000
504	25000

### Inner Join

select \*from tb1 inner join tb2 on tb1.rno=tb2.rno;

Or use the below query also

select \*from tb1 join tb2 on tb1.rno=tb2.rno;

RNO	NAME	MARKS	RNO	FEE
503	Suma	40	503	10000
504	Raju	70	504	25000
501	Abhi	50	501	15000
502	Ravi	60	502	5000

### Left Outer Join

select \* from tb1 left outer join tb2 on tb1.rno=tb2.rno;

RNO	NAME	MARKS	RNO	FEE
501	Abhi	50	501	15000
502	Ravi	60	502	5000
503	Suma	40	503	10000
504	Raju	70	504	25000
505	Ramu	45	-	-

### Right outer join

select \* from tb1 right outer join tb2 on tb1.rno=tb2.rno;

RNO	NAME	MARKS	RNO	FEE
503	Suma	40	503	10000
504	Raju	70	504	25000
501	Abhi	50	501	15000
502	Ravi	60	502	5000

### Natural join

select \* from tb1 natural join tb2;

RNO	NAME	MARKS	FEE
503	Suma	40	10000
504	Raju	70	25000
501	Abhi	50	15000
502	Ravi	60	5000

### Cross join

select \* from tb1 cross join tb2;

or use the below query

select \* from tb1,tb2;

RNO	NAME	MARKS	RNO	FEE
503	Suma	40	501	15000
504	Raju	70	501	15000
505	Ramu	45	501	15000
501	Abhi	50	501	15000
502	Ravi	60	501	15000
503	Suma	40	502	5000
504	Raju	70	502	5000
505	Ramu	45	502	5000
501	Abhi	50	502	5000
502	Ravi	60	502	5000
503	Suma	40	503	10000
504	Raju	70	503	10000
505	Ramu	45	503	10000
501	Abhi	50	503	10000
502	Ravi	60	503	10000
503	Suma	40	504	25000
504	Raju	70	504	25000
505	Ramu	45	504	25000
501	Abhi	50	504	25000
502	Ravi	60	504	25000

**Self Join:**

```
select t1.rno, t2.name from tb1 t1,tb1 t2 where t1.rno=t2.rno;
```

RNO	NAME
503	Suma
504	Raju
505	Ramu
501	Abhi
502	Ravi

**EQUI Join:**

```
select * from tb1,tb2 where tb1.rno=tb2.rno;
```

RNO	NAME	MARKS	RNO	FEE
503	Suma	40	503	10000
504	Raju	70	504	25000
501	Abhi	50	501	15000
502	Ravi	60	502	5000

## **AIM: 3.2 Perform Set Operations-Union, Intersection, Set Difference**

### **Description:**

Set Operators

- 1) UNION is used to combine the results of two or more SELECT statements. It will -eliminate duplicate rows from its resultset.

Syntax:-

```
SELECT column_name FROM table1  
UNION  
SELECT column_name FROM table2;
```

- 2) UNION ALL This is similar to Union. But it also shows the duplicate rows.

Syntax:-

```
SELECT column_name FROM table1  
UNION ALL  
SELECT column_name FROM table2;
```

- 3) Intersect operation is used to combine two SELECT statements, but it only retuns the records which are common from both SELECT statements.

Syntax:-

```
SELECT column_name FROM table1  
INTERSECT  
SELECT column_name FROM table2;
```

- 4) The Minus operation combines results of two SELECT statements and return only those in the final result, which belongs to the first set of the result.

Syntax:-

```
SELECT column_name FROM table1  
MINUS  
SELECT column_name FROM table2;
```

### **Source Tables**

#### **Sailors Table**

SID	SNAME	AGE	RATING
22	Dustin	45	7
29	Brutus	33	1
31	Lubber	55.5	8
32	Andy	25.5	8
64	Horatio	35	7
71	Zobra	16	10
74	Ravi	40	9
85	Art	26.5	3
95	Bob	63.5	3
58	Rusty	35	10

### Boats Table

BID	BNAME	BCOLOR
101	Interlake	Blue
102	Interlake	Red
104	Marine	Red
103	Clipper	Green

### Reserves Table

SID	BID	RDATE
22	101	10-OCT-98
22	102	10-OCT-98
22	103	08-OCT-98
22	104	07-OCT-98
31	102	10-NOV-98
31	103	06-NOV-98
31	104	12-NOV-98
64	101	05-SEP-98
64	102	08-SEP-98
74	103	08-SEP-98

### Find the names of sailors who have reserved a red or a green 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'  
UNION

select s1.sname from sailors s1,reserves r1,boats b1  
where s1.sid=r1.sid and r1.bid=b1.bid and b1.bcolor='Green';

SNAME
Dustin
Horatio
Lubber
Ravi

### Find the names of sailors who have reserved a red or a green 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'  
UNION all  
select s1.sname from sailors s1,reserves r1,boats b1  
where s1.sid=r1.sid and r1.bid=b1.bid and b1.bcolor='Green';

SNAME
Dustin
Lubber
Horatio
Dustin
Lubber
Dustin
Lubber
Ravi

**Find the names of sailors who have reserved both a red and green 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'
```

INTERSECT

```
select s1.sname from sailors s1,reserves r1,boats b1
where s1.sid=r1.sid and r1.bid=b1.bid and b1.bcolor='Green';
```

SNAME
Dustin
Lubber

**Find the names of sailors who have reserved a red boat but not a green 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'
```

MINUS

```
select s1.sname from sailors s1,reserves r1,boats b1
where s1.sid=r1.sid and r1.bid=b1.bid and b1.bcolor='Green';
```

SNAME
Horatio

## **AIM:3.3 Implementation of Correlated sub-queries and Nested queries**

### **Description:**

#### **Nested Queries**

A query within another SQL query and embedded within the WHERE clause. In Nested Query, Inner query runs first, and only once. Outer query is executed with result from Inner query.

##### **1) IN and NOT IN**

It tests whether a value is in a given set of elements

Syntax:-

Select column\_names from table\_name

Where column\_name IN/NOT IN (Select column\_name from table\_name  
Where condition);

##### **2) ALL and ANY**

It is used to compare a value to a list. It is preceded by comparison operator and followed by a list.

Syntax:-

Select column\_name from table\_name

Where column\_name comparison operator ALL/ANY(subquery);

### **Correlated Sub query**

In Correlated sub query, a query is nested inside another query and an inner query uses values from the outer query

Syntax:-

SELECT *column\_names* FROM *table\_name*

WHERE EXISTS/NOT EXISTS

(SELECT *column\_name* FROM *table\_name* WHERE *condition*);

**Find the names of sailors who have reserved boat no 103.**

```
select s.sname from sailors s  
where s.sid IN( select r.sid from reserves r  
where r.bid=103);
```

SNAME
Dustin
Lubber
Ravi

**Find the names of sailors who have not reserved boat no 103.**

```
select s.sname from sailors s  
where s.sid NOT IN( select r.sid from reserves r  
where r.bid=103);
```

SNAME
Brutus
Andy
Horatio
Zobra
Art
Bob
Rusty

**Find the sailor id with the highest rating**

```
select s.sid from sailors s where s.rating>=all(select s1.rating from sailors s1);
```

SID
71
58

**Find the sailor id whose rating is better than some sailor called andy**

```
select s.sid from sailors s  
where s.rating>ANY(select s1.rating from sailors s1 where s1.sname='Andy');
```

SID
71
58
74

### **Correlated Sub query**

#### **Find the names of sailors who have reserved boat no 103**

```
select s.sname from sailors s where EXISTS(select * from reserves r where s.sid=r.sid and r.bid=103);
```

SNAME
Dustin
Lubber
Ravi

#### **Find the names of sailors who have not reserved boat no 103**

```
select s.sname from sailors s where NOT EXISTS(select * from reserves r where s.sid=r.sid and r.bid=103);
```

SNAME
Brutus
Andy
Horatio
Zobra
Art
Bob
Rusty

## **AIM: 3.4 Creating and Querying views and Materialized views.**

### **Description:**

#### **View**

A view is a logical table based on the result set of an SQL Statement. A view contains rows and columns, just like a table. The fields in a view are fields from one or more base tables in the database. We can apply all DDL and DML statements on views.

#### **Syntax:**

Create or replace force/noforce view viewname as

Select column\_list from table\_list

Where condition with read only|check option

#### **Dropping a view syntax:**

Drop view viewname;

#### **Source Table**

ROLLNO	NAME	MARKS
501	jyothi	90
502	sai	95
504	yamuna	70
505	padma	60
503	ravi	80

#### **Creating View**

create view myview as select rollno,name from st1;

view created

#### **Display Views and Tables in your login**

select \* from tab;

ST1	TABLE
MYVIEW	VIEW
STUDENTS	TABLE
COURSES	TABLE
INSTRUCTORS	TABLE
TEACHES	TABLE
ENROLLMENTS	TABLE
STU_VIEW	VIEW
STUDENT_LOG	TABLE
COMPANY	TABLE
LOC	TABLE
EMPLOYEE	TABLE
JOB	TABLE
DEP	TABLE
STU2	TABLE

### **Inserting a row in view**

```
insert into myview values(506,'prathisha');
```

1 row(s) inserted

### **Display view**

```
select * from myview;
```

ROLLNO	NAME
501	jyothi
502	sai
506	prathisha
504	yamuna
505	padma
503	ravi

### **Displaying Table**

```
select * from st1;
```

ROLLNO	NAME	MARKS
501	jyothi	90
502	sai	95
506	prathisha	-
504	yamuna	70
505	padma	60
503	ravi	80

### **Deleting a row in a view**

```
delete from myview where rollno=506;
```

1 row(s) deleted

### **Display view**

```
select * from myview;
```

ROLLNO	NAME
501	jyothi
502	sai
504	yamuna
505	padma
503	ravi

## **Displaying Table**

```
select * from st1;
```

ROLLNO	NAME	MARKS
501	jyothi	90
502	sai	95
504	yamuna	70
505	padma	60
503	ravi	80

## **Change the Structure of the View**

```
create or replace view myview as select * from st1;
```

view created.

## **Creating View when base table doesn't exist**

```
create or replace force view abc as select * from dummy_table;
```

Warning: View created with compilation errors.

## **Creating Read only view**

```
create view myview1 as select * from st1 with read only;
```

view created

## **Inserting Data in Read only view**

```
insert into myview1 values(503,'prathisha',80);
```

ORA-01733: virtual column not allowed here

```
update myview1 set name='suma' where rollno=505;
```

ORA-01733: virtual column not allowed here

## **Displaying view**

```
select * from myview1;
```

ROLLNO	NAME	MARKS
501	jyothi	90
502	sai	95
504	yamuna	70
505	padma	60
503	ravi	80

## **Creating View with check option**

```
create view myview2 as select * from st1 where marks<101 with check option;
```

view created.

### **Inserting a row into view**

```
insert into myview2 values(504,'siri',101);
```

**ORA-01402: view WITH CHECK OPTION where-clause violation**

### **Dropping view**

```
Drop View myview1;
```

```
View dropp
```

