ADITYA UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

LAB MANUAL

SUB : Database Management Systems

Sem : III Sem

UNIT I

Aim:1. Familiarization with installation of any DBMS.

Description:

Oracle Database 10g Express Edition (often referred to as Oracle XE 10g or Oracle XE) was a free, entry-level version of the robust Oracle Database 10g product. It was designed to be easy to download, install, and use, making it ideal for developers, students, independent software vendors (ISVs), and small to mid-sized businesses.

Features of Oracle Database

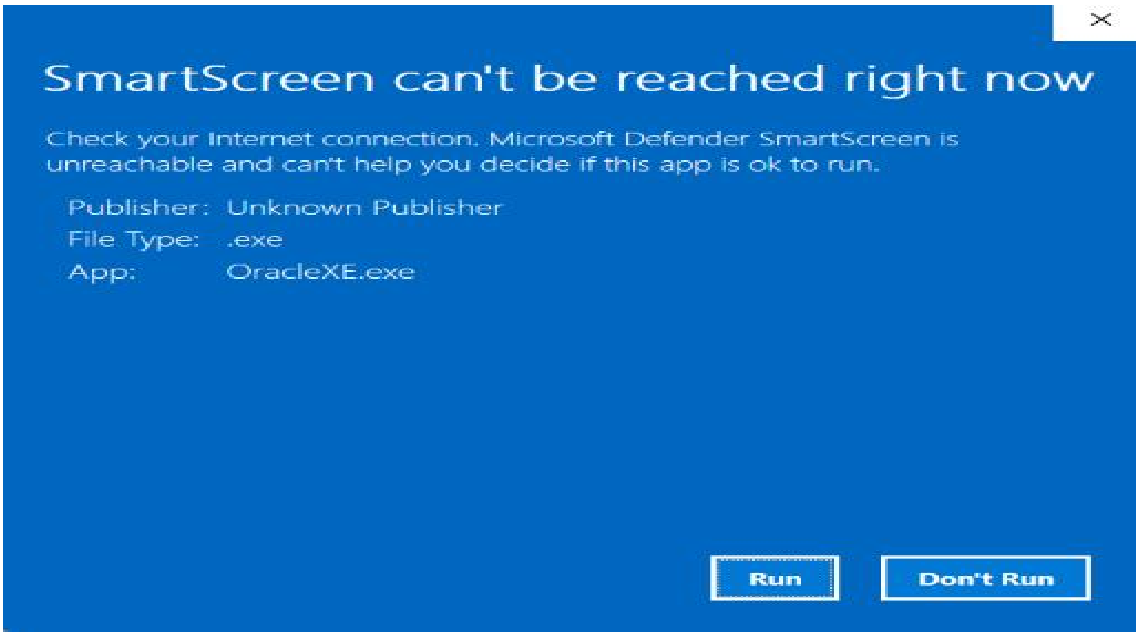
Relational Database Management System (RDBMS): At its core, it was a full-fledged relational database, supporting SQL, PL/SQL, and standard database operations.

Programming Interfaces: It offered connectivity to various programming languages and development environments, including Java, .NET, PHP, C, and JDBC.

Oracle Application Express (APEX): It often came with Oracle APEX (HTML DB at the time), a low-code development platform for building web applications directly on the database.

Basic Database Administration: It included a browser-based interface for managing database activities, users, storage, and memory.

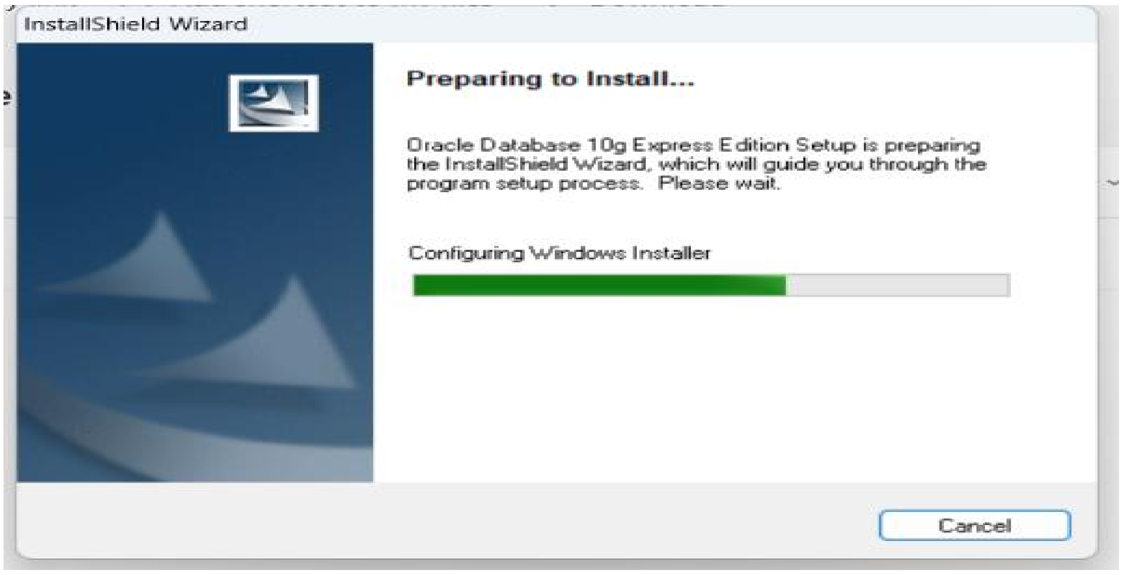
Core Database Functionality: Features like memory and statistics management, basic replication, and backup/recovery capabilities were included.



SmartScreen can't be reached right now

Check your Internet connection. Microsoft Defender SmartScreen is unreachable and can't help you decide if this app is ok to run.

Publisher:	Unknown Publisher
File Type:	.exe
App:	OracleXE.exe

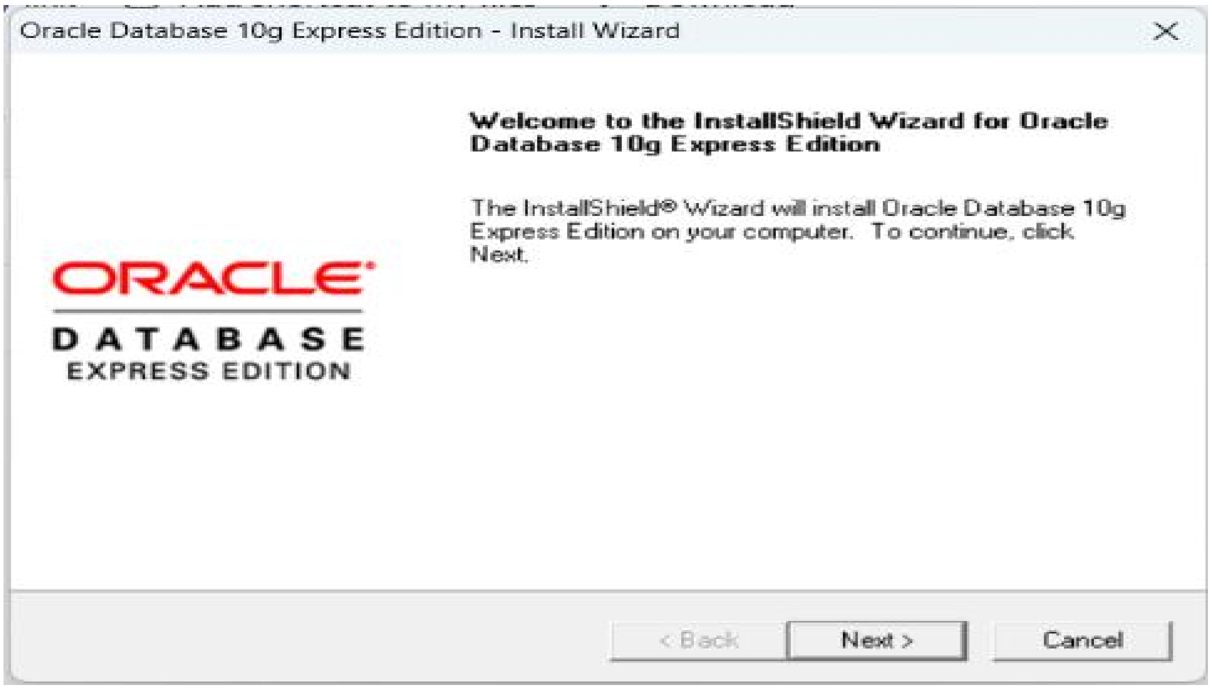


InstallShield Wizard

Preparing to Install...

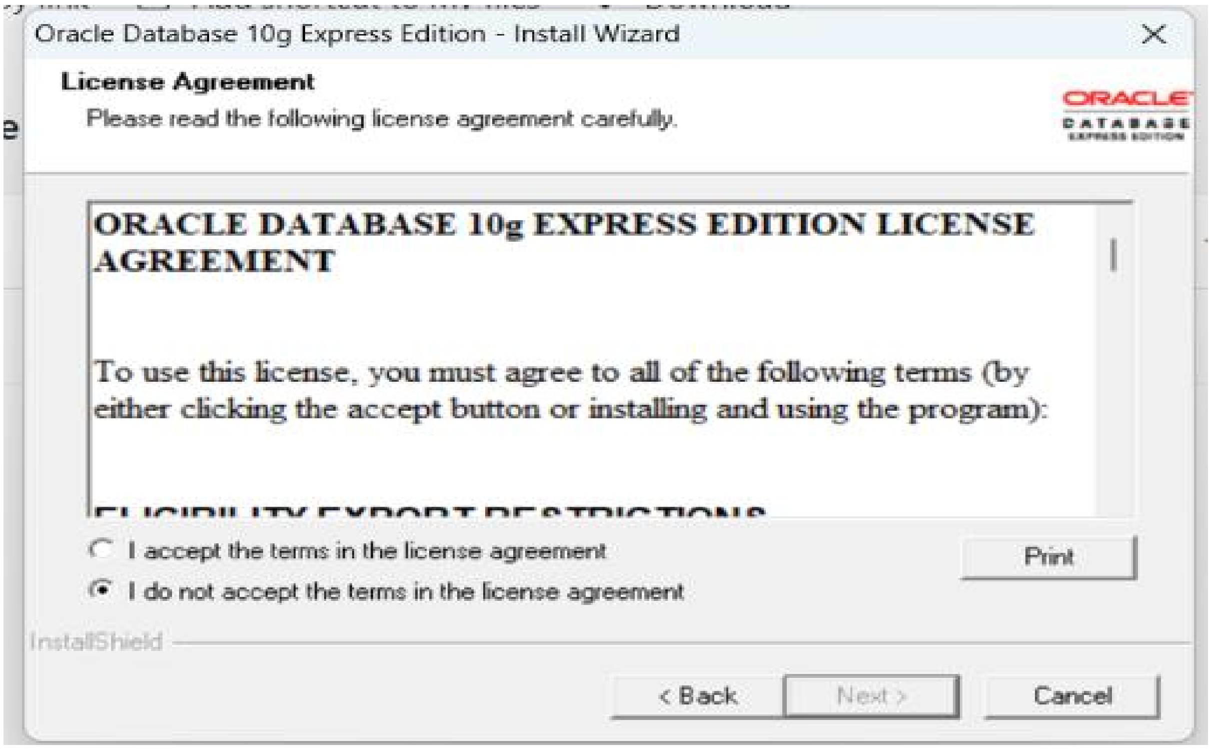
Oracle Database 10g Express Edition Setup is preparing the InstallShield Wizard, which will guide you through the program setup process. Please wait.

Configuring Windows Installer


Oracle Database 10g Express Edition - Install Wizard

Welcome to the InstallShield Wizard for Oracle Database 10g Express Edition

The InstallShield® Wizard will install Oracle Database 10g Express Edition on your computer. To continue, click Next.


Oracle Database 10g Express Edition - Install Wizard

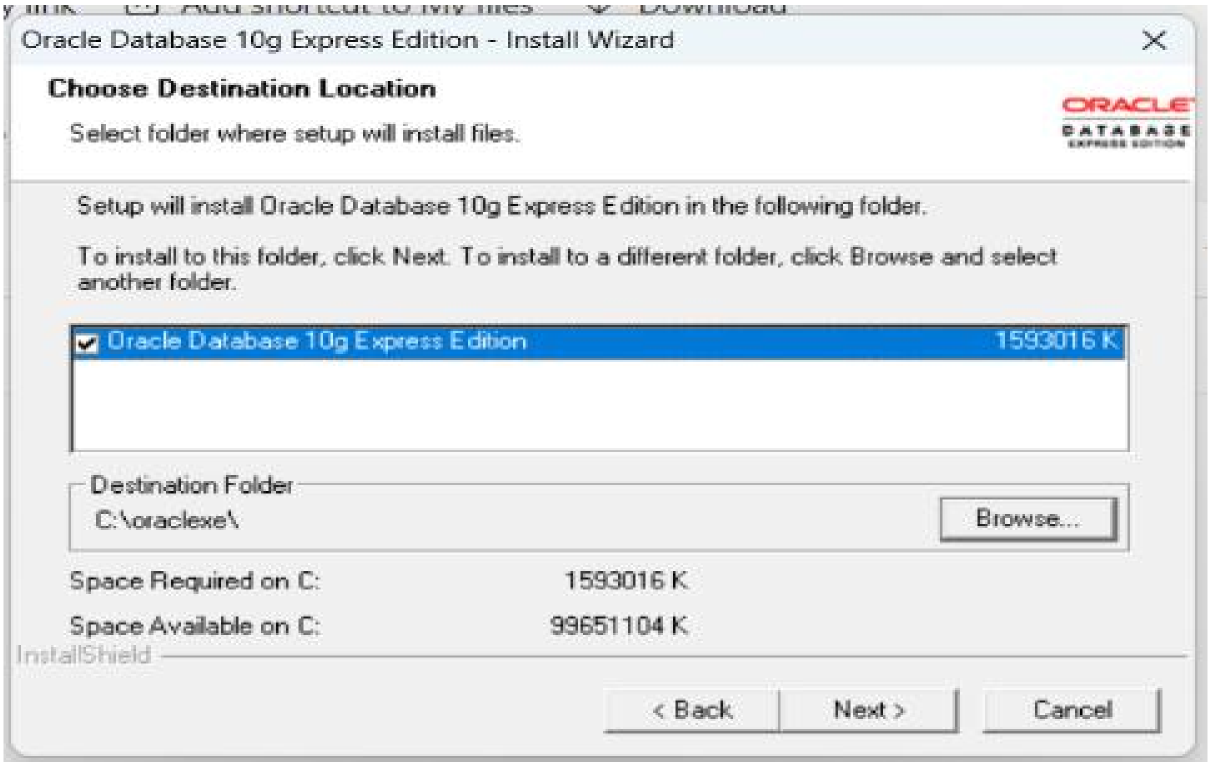
License Agreement

Please read the following license agreement carefully.

ORACLE DATABASE 10g EXPRESS EDITION LICENSE AGREEMENT

To use this license, you must agree to all of the following terms (by either clicking the accept button or installing and using the program):

( ) I accept the terms in the license agreement
[✓ I do not accept the terms in the license agreement]


Oracle Database 10g Express Edition - Install Wizard

Choose Destination Location

Select folder where setup will install files.

Setup will install Oracle Database 10g Express Edition in the following folder.

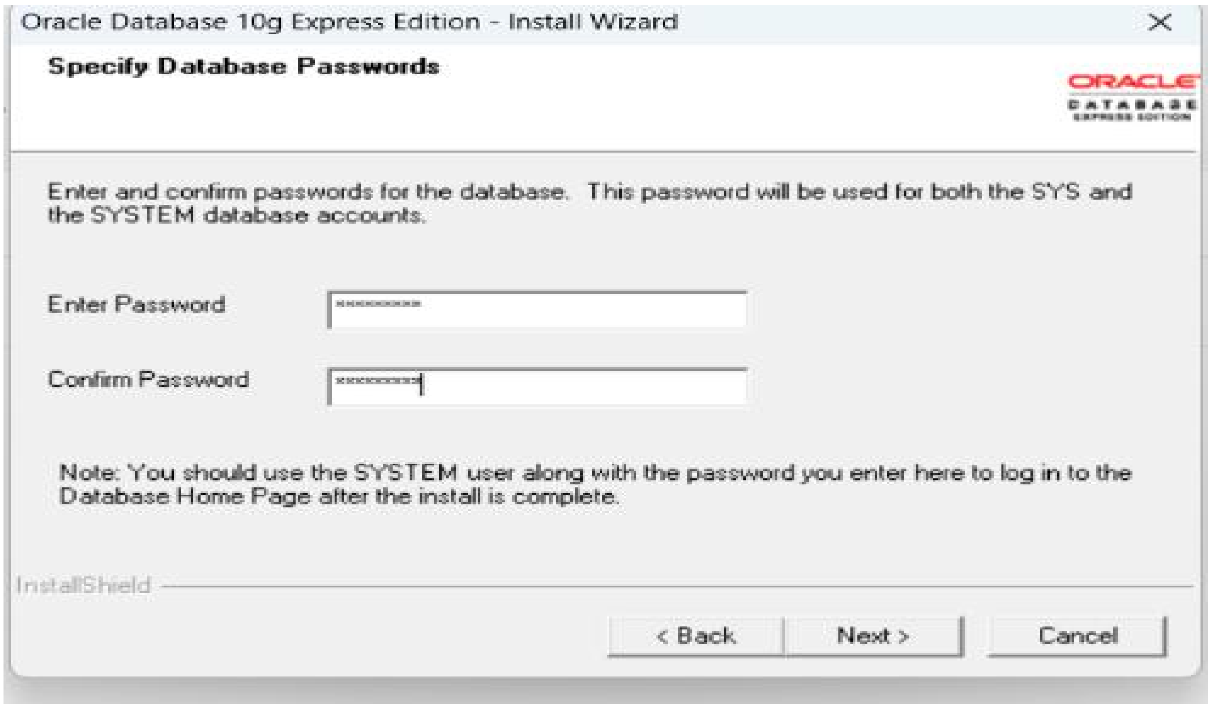
To install to this folder, click Next. To install to a different folder, click Browse and select another folder.

☑ Oracle Database 10g Express Edition 1593016 K

Destination Folder
C:\oraclexe\

Space Required on C: 1593016 K

Space Available on C: 99651104 K


Oracle Database 10g Express Edition - Install Wizard

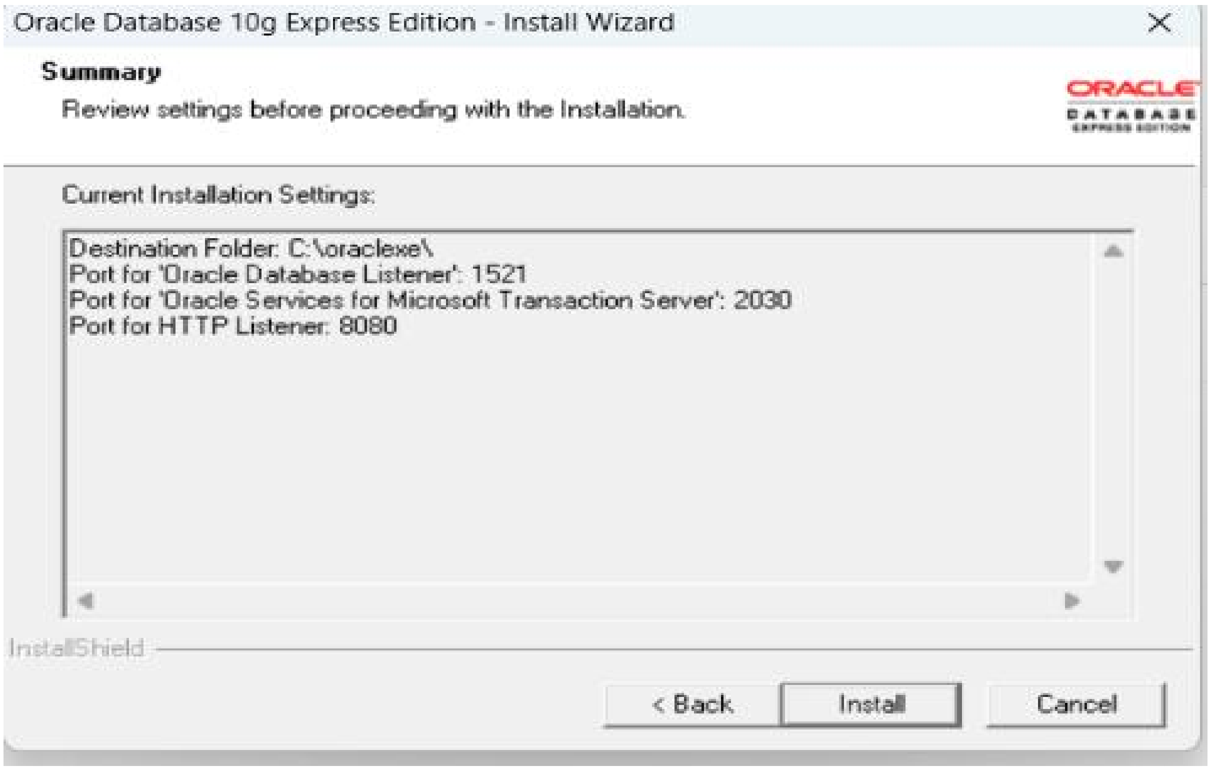
Specify Database Passwords

Enter and confirm passwords for the database. This password will be used for both the SYS and the SYSTEM database accounts.

Enter Password ********

Confirm Password ********

Note: You should use the SYSTEM user along with the password you enter here to log in to the Database Home Page after the install is complete.

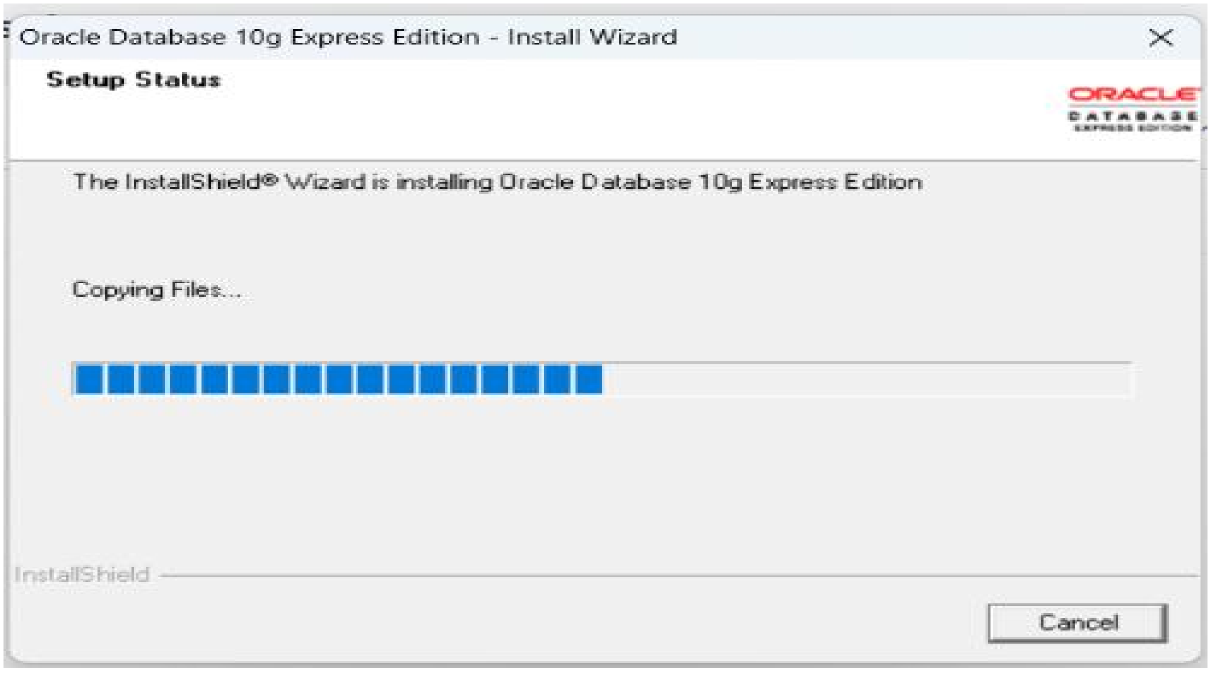

Oracle Database 10g Express Edition - Install Wizard

Summary

Review settings before proceeding with the Installation.

Current Installation Settings:

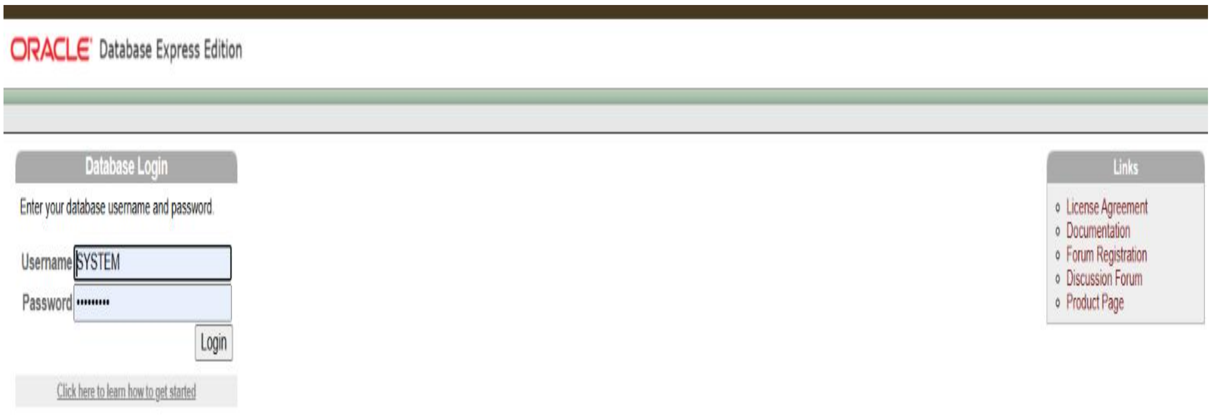
```
Destination Folder: C:\oraclexe\
Port for 'Oracle Database Listener': 1521
Port for 'Oracle Services for Microsoft Transaction Server': 2030
Port for HTTP Listener: 8080
```


Oracle Database 10g Express Edition - Install Wizard

Setup Status

The InstallShield® Wizard is installing Oracle Database 10g Express Edition

Copying Files...



Enter your database username and password.

Username SYSTEM

Password ••••••••

<u>Aim:</u> 2. Implementing a University Database System

<u>Description:</u>

What is Schema?

Schema defines the structure of a 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: datype)

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

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)

Viva -Questions

1. **Q:** What is a database schema?
   A: A database schema is the structure that defines the organization of data, including tables, fields, relationships, and constraints in a database.
2. **Q:** Name the tables used in the university database schema.
   A: The main tables are: Students, Departments, Courses, Instructors, Enrollments, and Teaches.
3. **Q:** What is a primary key?
   A: A primary key is a unique identifier for each record in a table. It must contain unique values and cannot be NULL.
4. **Q:** What is a foreign key?
   A: A foreign key is a field in one table that refers to the primary key of another table. It is used to establish relationships between tables.

UNIT II

Aim: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 existingdata 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] column|expression [alias],...}
FROM table;
```

<u>Creating the students, curses, enrollments, instructors, and teaches</u>

1. Students

```
CREATE TABLE Students (
    student_id INT PRIMARY KEY,
    name VARCHAR(50),
    major VARCHAR(50),
    age INT
);
```

Output: Table Created.

2. Courses

```
CREATE TABLE Courses (
    course_id INT PRIMARY KEY,
    course_name VARCHAR(100),
    credits INT
);
```

Output: Table Created.

3. Enrollments

```
CREATE TABLE Enrollments (
    enrollment_id INT PRIMARY KEY,
    student_id INT,
    course_id INT,
    grade CHAR(2),
    FOREIGN KEY (student_id) REFERENCES Students(student_id),
    FOREIGN KEY (course_id) REFERENCES Courses(course_id)
);
```

Output: Table Created.

4. Instructors

```
CREATE TABLE Instructors (
   instructor_id INT PRIMARY KEY,
   name VARCHAR(50),
   department VARCHAR(50)
);
```

Output: Table Created.

5. Teaches

```
CREATE TABLE Teaches (
   instructor_id INT,
   course_id INT,
   semester VARCHAR(10),
   PRIMARY KEY (instructor_id, course_id),
   FOREIGN KEY (instructor_id) REFERENCES Instructors(instructor_id),
   FOREIGN KEY (course_id) REFERENCES Courses(course_id)
);
```

Output: Table Created.

```
--Students
INSERT INTO Students VALUES (1, 'Alice', 'CSE', 20);
```

Output: 1 row(s) inserted.

```
INSERT INTO Students VALUES (2, 'Bob', 'ECE', 21);
```

Output: 1 row(s) inserted.

```
INSERT INTO Students VALUES (3, 'Charlie', 'CSE', 19);
```

Output: 1 row(s) inserted.

```
-- Courses

INSERT INTO Courses VALUES (101, 'DBMS', 4);
```

Output: 1 row(s) inserted.

```
INSERT INTO Courses VALUES (102, 'Operating Systems', 3);
```

Output: 1 row(s) inserted.

```
INSERT INTO Courses VALUES (103, 'Networks', 3);
```

Output: 1 row(s) inserted.

```
 -- Enrollments

INSERT INTO Enrollments VALUES (1, 1, 101, 'A');
```

Output: 1 row(s) inserted.

```
INSERT INTO Enrollments VALUES (2, 2, 102, 'B');
```

Output: 1 row(s) inserted.

```
INSERT INTO Enrollments VALUES (3, 1, 103, 'A');
```

Output: 1 row(s) inserted.

```
INSERT INTO Enrollments VALUES (4, 3, 101, 'B');
```

Output: 1 row(s) inserted.

```
 -- Instructors

INSERT INTO Instructors VALUES (1001, 'Dr. Kumar', 'CSE');
```

Output: 1 row(s) inserted.

```
INSERT INTO Instructors VALUES (1002, 'Dr. Rao', 'ECE');
```

Output: 1 row(s) inserted.

```
-- Teaches

INSERT INTO Teaches VALUES (1001, 101, 'Fall');
```

Output: 1 row(s) inserted.

```
INSERT INTO Teaches VALUES (1002, 102, 'Spring');
```

Output: 1 row(s) inserted.

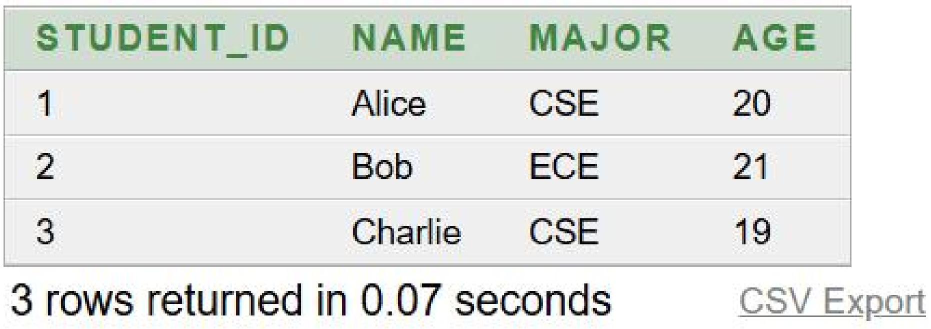
SQL Queries to Practice

Basic Queries

1. List all students.

SELECT * FROM Students;

Output:



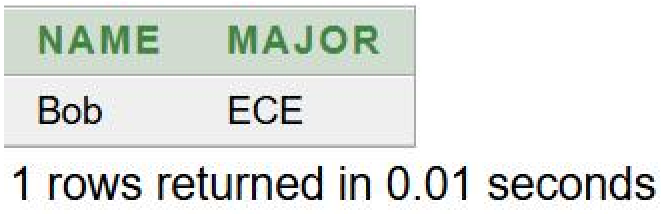
STUDENT_ID	NAME	MAJOR	AGE
1	Alice	CSE	20
2	Bob	ECE	21
3	Charlie	CSE	19

3 rows returned in 0.07 seconds   CSV Export

2. Show names and majors of students aged above 20.

SELECT name, major FROM Students WHERE age > 20;

Output:



NAME	MAJOR
Bob	ECE

1 rows returned in 0.01 seconds

3. Update students age in students table

   Update Students

   Set age=20

   Where student_id=3;

   Output:

   1 row(s) updated.

4. delete a row in Enrollments table

```
delete from Enrollments
where enrollment_id=4;
```

Output:

```
1 row(s) deleted.
```

Aim:2. Implementation of Aggregate Functions – sum, avg, min, max, count. Use group-by and having clause.

Description:

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;
```

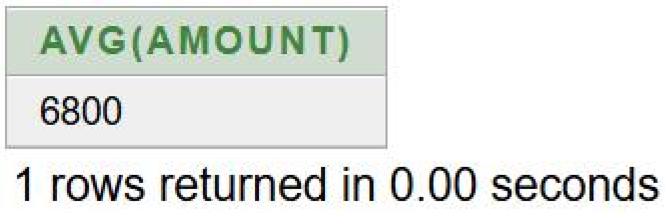
```
companyn      amount

wipro           5000

ibm             8000

dell            9000

wipro           2000

dell            10000
```

Queries

Find the average salary of company

Select AVG(amount) from company;

Output:

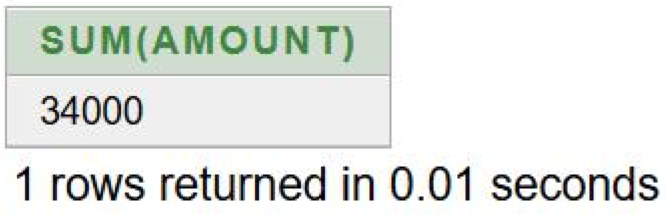


AVG(AMOUNT)
6800

1 rows returned in 0.00 seconds

Find the Sum of salaries of company

Select SUM(amount) from company;

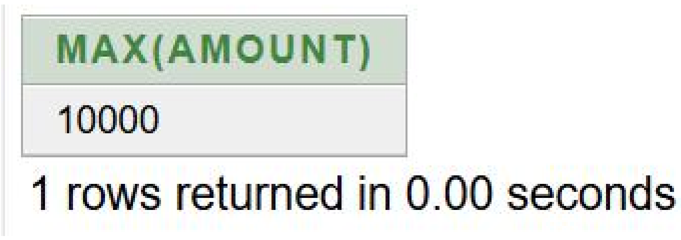


SUM(AMOUNT)
34000

1 rows returned in 0.01 seconds

Find the Maximum amount of company

Select Max(amount) from company;

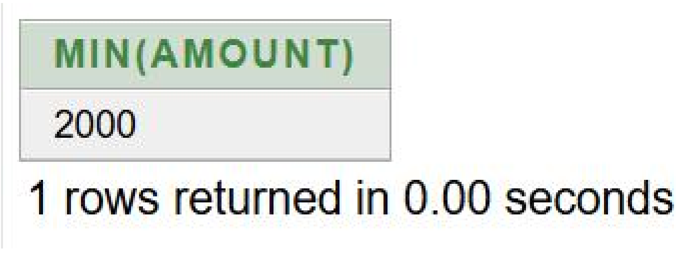


MAX(AMOUNT)
10000

1 rows returned in 0.00 seconds

Find the Minimum amount of company

Select Min(amount) from company;

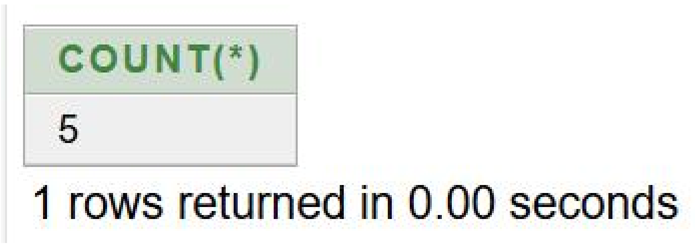


MIN(AMOUNT)
2000

1 rows returned in 0.00 seconds

Find the number of rows in a company

Select Count(*) from company;

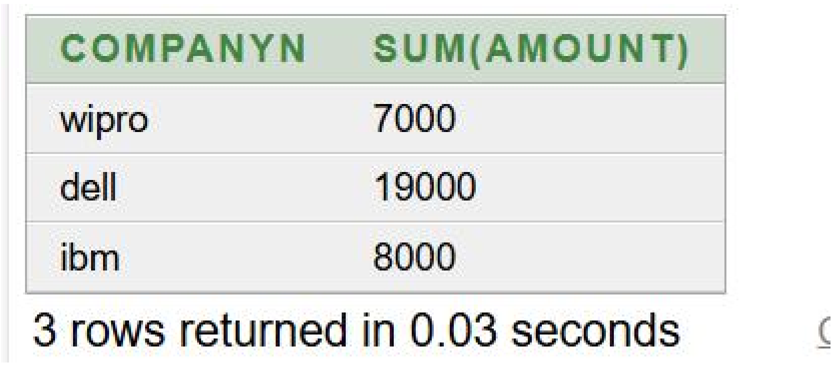


COUNT(*)
5

1 rows returned in 0.00 seconds

Find the sum of amount of each company.

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

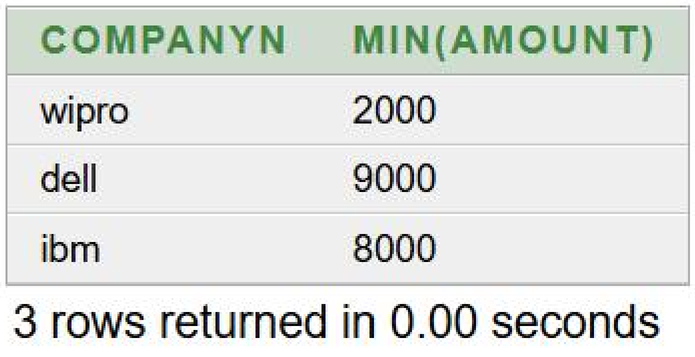


COMPANYN	SUM(AMOUNT)
wipro	7000
dell	19000
ibm	8000

3 rows returned in 0.03 seconds

Find the minimum amount of each company.

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

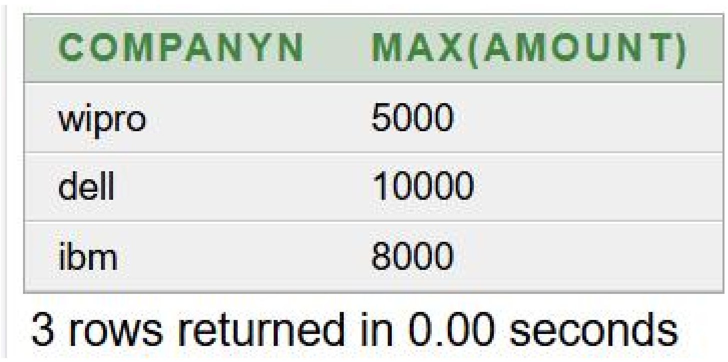


COMPANYN	MIN(AMOUNT)
wipro	2000
dell	9000
ibm	8000

3 rows returned in 0.00 seconds

Find the maximum amount of each company.

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

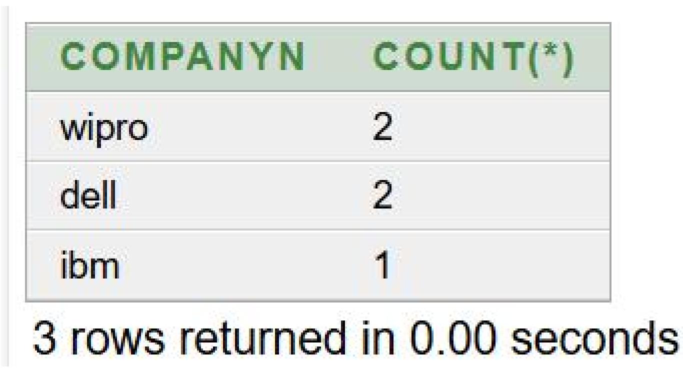


COMPANYN	MAX(AMOUNT)
wipro	5000
dell	10000
ibm	8000

3 rows returned in 0.00 seconds

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

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

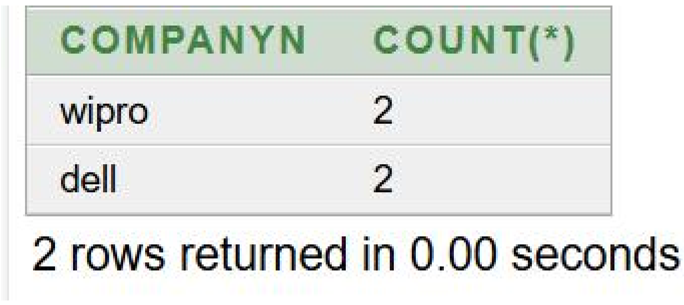


COMPANYN	COUNT(*)
wipro	2
dell	2
ibm	1

3 rows returned in 0.00 seconds

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;

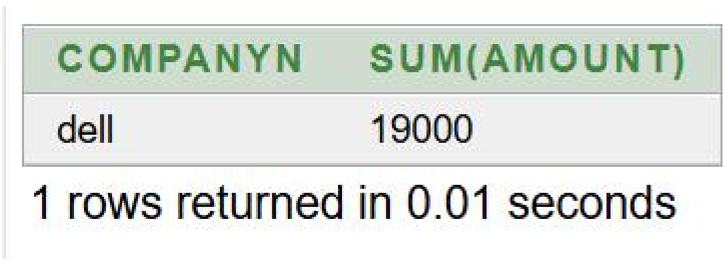


COMPANYN	COUNT(*)
wipro	2
dell	2

2 rows returned in 0.00 seconds

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
1 rows returned in 0.01 seconds
```

Viva-Questions:

1. **Q:** What is SQL?
   A: SQL (Structured Query Language) is a standard language used to communicate with relational databases for storing, retrieving, and manipulating data.
2. **Q:** Write a query to display all records from a table named Students.
   A:SELECT * FROM Students;
3. **Q:** How do you rename a column in the result of a query?
   A:SELECT FirstName AS Name FROM Students;
4. **Q:** How can you retrieve unique values from a column?
   A:SELECT DISTINCT DepartmentID FROM Students;
5. **Q:** How do you sort results in SQL?
   A:SELECT * FROM Students ORDER BY FirstName ASC;

6. **Q:** What is a subquery?
   A: A subquery is a query nested inside another SQL query. It can be used in SELECT, FROM, or WHERE clauses.
7. **Q:** Write a query to find students whose department has more than 50 students.
   A:SELECT * FROM Students

```
WHERE DepartmentID IN (
   SELECT DepartmentID
   FROM Students
   GROUP BY DepartmentID
   HAVING COUNT(*) > 50
);
```

8. **Q:** What is the difference between **correlated** and **non-correlated** subqueries?
   A: A correlated subquery depends on the outer query for its value, whereas a non-correlated subquery runs independently of the outer query.
9. **Q:** Find names of instructors who teach at least one course.
   A:

```
SELECT FirstName, LastName

FROM Instructors

WHERE InstructorID IN (

   SELECT DISTINCT InstructorID
```

```
    FROM Teaches
);
```

10. **Q:** What are some common SQL operators?
    A:

- Arithmetic: +, -, *, /
- Comparison: =, <>, <, >, <=, >=
- Logical: AND, OR, NOT
- Range: BETWEEN, IN, LIKE

11. **Q:** Write a query to find students between ages 18 and 22.
    A:

```
SELECT * FROM Students
WHERE YEAR(CURDATE()) - YEAR(DOB) BETWEEN 18 AND 22;
```

12. **Q:** Find instructors whose name starts with 'A'.
    A:

```
SELECT * FROM Instructors
WHERE FirstName LIKE 'A%';
```

13. **Q:** What does the IN operator do?
    A: It checks if a value exists within a specified set of values.

14. **Q:** What are SQL set operators?
    A: Set operators combine results of two or more queries:

- UNION: Removes duplicates
- UNION ALL: Includes duplicates
- INTERSECT: Common records
- MINUS / EXCEPT: Records in the first query but not in the second

15. **Q:** Example: Get list of all unique student and instructor emails.
    A:

```
SELECT Email FROM Students
UNION
SELECT Email FROM Instructors;
```

16. **Q:** Difference between UNION and UNION ALL?
    A: UNION removes duplicates; UNION ALL retains them.

E. GROUP BY and HAVING Clauses

17. **Q:** What is the use of the GROUP BY clause?
    A: GROUP BY is used to group rows that have the same values in specified columns for aggregation.

18. **Q:** What is the difference between WHERE and HAVING?
    A: WHERE filters rows before grouping; HAVING filters groups after aggregation.
19. **Q:** Find the number of students in each department.
    A:

```
SELECT DepartmentID, COUNT(*) AS StudentCount
FROM Students
GROUP BY DepartmentID;
```

20. **Q:** Find departments having more than 100 students.
    A:

```
SELECT DepartmentID, COUNT(*) AS StudentCount
FROM Students
```