DRONACHARYA
Group of Institutions

DATABASE MANAGEMENT SYSTEMS LAB

LABORATORY MANUAL

B.Tech. Semester – V

Subject Code: BCS-551

Session: 2024-25, Odd Semester

Name:	
Roll. No.:	
Group/Branch:	

DRONACHARYA GROUP OF INSTITUTIONS
DEPARTMENT OF ECE
#27 KNOWLEDGE PARK 3
GREATER NOIDA

AFFILATED TO Dr. ABDUL KALAM TECHNICAL UNIVERSITY, LUCKNOW

Table of Contents

1. Vision and Mission of the Institute
2. Vision and Mission of the Department
3. Programme Educational Objectives (PEOs)
4. Programme Outcomes (POs)
5. Programme Specific Outcomes (PSOs)
6. University Syllabus
7. Course Outcomes (COs)
8. CO- PO and CO-PSO mapping
9. Course Overview
10. List of Experiments
11. DOs and DON‟‟Ts
12. General Safety Precautions
13. Guidelines for students for report preparation
14. Lab assessment criteria
15. Details of Conducted Experiments
16. Lab Experiments

Vision and Mission of the Institute

Vision:

Instilling core human values and facilitating competence to address global challenges by providing Quality Technical Education.

Mission:

M1: Enhancing technical expertise through innovative research and education, fostering creativity and excellence in problem-solving.

M2: Cultivating a culture of ethical innovation and user-focused design, ensuring technological progress enhances the well-being of society.

M3: Equipping individuals with the technical skills and ethical values to lead and innovate responsibly in an ever-evolving digital landscape.

Vision and Mission of the Department

VISION

To achieve excellence in Electronics and Computer engineering through quality education, research contributing to the emerging technologies and innovation to serve industry and society**.**

MISSION

M1: To help students achieve their goals by recognizing, identifying, and to bring up their unique strengths through quality education and cutting-edge research training.

M2: To facilitate adequate exposure to the students through training in the state of-the art technologies.

M3: To imbibe ability in the students to solve real life problems as per need of the society through nurturing their skills, creative thinking, and research acumen.

Programme Educational Objectives (PEOs)

PEO 1.

To develop a strong foundation of engineering fundamentals to build successful careers maintaining high ethical standards.

PEO 2.

To prepare graduates for higher studies and research activities, facilitating a commitment to lifelong learning.

PEO 3.

To prepare graduates for higher studies and research activities, facilitating a commitment to lifelong learning.

Programme Outcomes (POs)

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

PO6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Specific Outcomes (PSOs)

PSO1:

To analyses electronics systems applying principles of mathematics and engineering sciences, to develop innovative ethical solutions to complex engineering problems with team spirit and social commitment.

PSO2:

To develop solution for real world problems based on principles of computer hardware, advanced software and simulation tools with a focus to devise indigenous, eco-friendly and energy efficient projects.

University Syllabus

DR. A.P.J. ABDUL KALAM TECHNICAL UNIVERSITY UTTAR PRADESH, LUCKNOW

ELECTRONICS AND COMPUTER ENGINEERING

Database Management Systems Lab (BCS551)		
Course Outcome (CO)		**Bloom's Knowledge Level (KL)**
At the end of course , the student will be able to:		
CO 1	Understand and apply oracle 11 g products for creating tables, views, indexes, sequences and other database objects.	K_2, K_4
CO 2	Design and implement a database schema for company data base, banking data base, library information system, payroll processing system, student information system.	K_3, K_5
CO 3	Write and execute simple and complex queries using DDL, DML, DCL and TCL.	K_4, K_5
CO 4	Write and execute PL/SQL blocks, procedure functions, packages and triggers, cursors.	K_4, K_5
CO 5	Enforce entity integrity, referential integrity, key constraints, and domain constraints on database.	K_3, K_4

DETAILED SYLLABUS

1. Installing oracle/ MYSQL
2. Creating Entity-Relationship Diagram using case tools.
3. Writing SQL statements Using ORACLE /MYSQL:
   - a)Writing basic SQL SELECT statements.
   - b) Restricting and sorting data.
   - c)Displaying data from multiple tables.
   - d)Aggregating data using group function.
   - e)Manipulating data.
   - e)Creating and managing tables.
4. Normalization
5. Creating cursor
6. Creating procedure and functions
7. Creating packages and triggers
8. Design and implementation of payroll processing system
9. Design and implementation of Library Information System
10. Design and implementation of Student Information System
11. Automatic Backup of Files and Recovery of Files
12. Mini project (Design & Development of Data and Application) for following :
    - a) Inventory Control System.
    - b) Material Requirement Processing.
    - c) Hospital Management System.
    - d) Railway Reservation System.
    - e) Personal Information System.
    - f) Web Based User Identification System.
    - g) Timetable Management System.
    - h) Hotel Management System

Course Outcomes (COs)

CO1	Understand and apply oracle 11 g products for creating tables, views, indexes, sequences and other database objects
CO2	Design and implement a database schema for company data base, banking data base, library information system, payroll processing system, student information system.
CO3	Write and execute simple and complex queries using DDL, DML, DCL and TCL.
CO4	Write and execute PL/SQL blocks, procedure functions, packages and triggers, cursors.
CO5	Enforce entity integrity, referential integrity, key constraints, and domain constraints on database

CO-PO Mapping

CO-PO Matrix												
Course Outcomes	**PO1**	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**
CO1	2	2	3	3	3	-	-	-	-	-	-	2
CO2	3	3	3	2	2	-	-	-	-	-	-	3
CO3	2	3	3	3	3	-	-	-	-	-	-	2
CO4	2	3	2	2	2	-	-	-	-	-	-	2
CO5	2	3	2	2	2	-	-	-	-	-	-	3

CO-PSO Mapping

	PSO1	**PSO2**
CO1	2	2
CO2	2	3
CO3	2	3
CO4	2	2
CO5	2	3

Course Overview

This course offers a practical foundation in Database Management Systems, essential for computer science and engineering.

- Understand and apply core DBMS concepts, including relational models, SQL, normalization, and transactions.
- Develop proficiency in database design and implementation using various DBMS tools.
- Explore the role of databases in software applications, emphasizing performance, security, and scalability.
- Analyze and solve real-world database problems, enhancing problem-solving and analytical thinking skills.

Course Objectives

- **Practical Application**: Equip students with the ability to apply database concepts in real-world scenarios, reinforcing theoretical knowledge through hands-on lab exercises.
- **Database Design**: Foster the development of strong skills in designing efficient and normalized databases, enabling students to construct and manage robust database systems.
- **Problem-Solving Skills**: Enhance students' ability to use database technologies to model and solve complex data management problems effectively.
- **Critical Thinking**: Cultivate a deep understanding of DBMS principles to analyze and optimize the performance and security of database systems.

List of Experiments

S.No	Experiments
1	Installing oracle/ MYSQL.
2	Creating Entity-Relationship Diagram using case tools.
3	Writing SQL statements Using ORACLE /MYSQL: a) Writing basic SQL SELECT statements. b) Restricting and sorting data. c) Displaying data from multiple tables. d) Aggregating data using group function. e) Manipulating data. f) Creating and managing tables.
4	Creating procedure and functions
5	Design and implementation of Student Information System.
6	Write a CURSOR to display list of clients in the client Master Table.
7	Execute the queries related to Group By and having Clause on tables SALES_ORDER.
8	Execute the following queries: a) The NOT NULL b) The UNIQUE Constraint c) The PRIMARY KEY Constraint d) The CHECK Constraint e) Define Integrity Constraints in ALTER table Command
9	Execute Nested Queries on tables CLIENT_MASTER, PRODUCT_MASTER, SALESMAN_MASTER, SALES_ORDER, SALES_ORDER_DETAILS.
10	Execute Queries related to Exists, Not Exists, Union, Intersection, Difference, Join on tables CLIENT_MASTER, PRODUCT_MASTER, SALESMAN_MASTER, SALES_ORDER_DETAILS>

DOs and DON'Ts

DOs

1. Login-on with your username and password.
2. Log off the computer every time when you leave the Lab.
3. Arrange your chair properly when you are leaving the lab.
4. Put your bags in the designated area.
5. Ask permission to print.

DON'Ts

1. Do not share your username and password.
2. Do not remove or disconnect cables or hardware parts.
3. Do not personalize the computer setting.
4. Do not run programs that continue to execute after you log off.
5. Do not download or install any programs, games or music on computer in Lab.
6. Personal Internet use chat room for Instant Messaging (IM) and Sites is strictly prohibited.
7. No Internet gaming activities allowed.
8. Tea, Coffee, Water & Eatables are not allowed in the Computer Lab.

General Safety Precautions

Precautions (In Case of Injury or Electric Shock)

1. **Break Contact Safely**: If a person is in contact with a live electrical source, use an insulator such as a plastic chair or a wooden object to break the contact. Avoid touching the victim with bare hands to prevent electric shock to yourself.
2. **Disconnect Power**: Unplug the affected equipment or turn off the main circuit breaker if accessible. Ensure all systems are powered down to prevent further risk.
3. **Provide Immediate Aid**: If the victim is unconscious, begin CPR immediately. Perform chest compressions and use mouth-to-mouth resuscitation if necessary.
4. **Seek Emergency Help**: Call emergency services and campus security immediately. Time is critical, so act swiftly and efficiently.

Precautions (In Case of Fire)

1. **Power Down Equipment**: Turn off the affected system immediately. If the power switch is not accessible, unplug the device.
2. **Contain the Fire**: If safe to do so, use a fire extinguisher or cover the fire with a heavy cloth to smother it. Ensure the fire does not spread to other devices or components in the lab.
3. **Raise the Alarm**: Activate the nearest fire alarm switch located in the hallway to alert others in the building.
4. **Contact Security and Emergency Services**: Call the security office and emergency services without delay to report the fire and get professional help on site.

Guidelines for Report Preparation for Students in Database Management Systems Lab

All students are required to maintain a comprehensive record of the experiments conducted in the Database Management Systems Lab. The guidelines for preparing this record are as follows:

1. **File Structure**:
    - Each file must begin with a **title page**, followed by an **index page**. Faculty will not sign the file unless there is an entry on the index page.
2. **Student Information**:
    - **Student's Name**, **Roll Number**, and **Date of Conduction** of the experiment must be clearly mentioned on all pages of the record.
3. **Experiment Documentation**:
    - For each algorithm experiment, the record must include the following sections:
        - **Program Name**
        - **Code Implementation**
        - **Output and Observations**
4. **Additional Notes**:
    - Students must bring their **lab record** to every lab session.
    - Ensure that the lab record is **regularly evaluated** by the faculty. Consistent updates and evaluations are essential for accurate assessment.

Lab Assessment Criteria for Database Management Systems Lab

In a semester, approximately 10 lab classes are conducted for each lab course. These classes are assessed continuously based on five assessment criteria. The performance in each experiment contributes to the computation of Course Outcome (CO) attainment and internal marks. The grading criteria are detailed in the following table:

Grading Criteria	Exemplary (4)	Competent (3)	Needs Improvement (2)	Poor (1)
AC1: Understanding and Applying Database Concepts	The student demonstrates a deep understanding of database concepts (e.g., relational models, normalization, transactions) and applies them accurately to solve complex problems.	The student applies database concepts correctly but may lack in-depth understanding or innovative approaches.	The student shows basic understanding but struggles with correct application in problem-solving.	The student does not demonstrate an understanding of discrete structures or fails to apply them correctly.
AC2: Implementing SQL Queries	Develops correct and efficient SQL queries with adherence to best practices and optimized for performance.	Constructs SQL queries correctly with minor issues in efficiency or adherence to best practices.	Implements SQL queries with significant errors or inefficiencies in syntax or logic.	Implementation of logical constructs is incomplete or incorrect, with major errors or omissions.
AC3: Analyzing Database Design and Normalization	Correctly interprets and validates database design and normalization, providing thorough and accurate analysis.	Provides reasonable analysis of database design and normalization with minor inaccuracies.	Attempts analysis but with several inaccuracies or a lack of comprehensive understanding.	Lacks understanding or does not attempt to analyze logical validity or proofs.
AC4: Optimizing Database Performance	Thoroughly interprets and analyzes performance metrics, proposing	Provides performance optimizations that are somewhat effective but	Attempts to optimize performance, but efforts are inaccurate,	Does not provide logical conclusions or the conclusions are irrelevant or incorrect.

	insightful optimizations or improvements.	lack depth or thorough analysis.	incomplete, or superficial.	
AC5: Lab Record	Well-organized, clear, and thoroughly presented record that integrates theoretical concepts with practical exercises and results.	The record is clear and organized but may have minor issues in presentation or completeness.	The record lacks clarity, organization, and completeness, showing significant gaps.	The record is poorly presented or incomplete, with minimal effort exhibited.

Additional Notes:

- **Continuous Assessment**: Each experiment is evaluated during the lab sessions.
- **CO Attainment**: Performance in each experiment contributes to the Course Outcome (CO) attainment.
- **Internal Marks**: The cumulative performance determines the internal marks for the lab course.

Details of Conducted Experiments

The following experiments are conducted to provide hands-on experience in implementing and analyzing core concepts of Database Management Systems Lab. Each experiment focuses on practical applications, enhancing problem-solving skills, and understanding theoretical foundations essential to computer science.

1. Installing Oracle/MySQL

- **Objective:** Guide the user through the installation process of Oracle Database or MySQL Server on their system.
- **Implementation:**
  - Provide clear, step-by-step instructions for downloading and installing the chosen database system.
  - Include troubleshooting tips for common installation issues.
  - Consider creating a script or a set of commands to automate parts of the installation process.

2. Creating Entity-Relationship Diagram (ERD)

- **Objective:** Introduce the concept of ERDs and demonstrate their creation using a case tool (e.g., Lucidchart, Draw.io).
- **Implementation:**
  - Select a sample scenario (e.g., a library system, an e-commerce store).
  - Create an ERD for the chosen scenario, identifying entities, attributes, and relationships.
  - Explain the different types of relationships (one-to-one, one-to-many, many-to-many) and how they are represented in an ERD.

3. Writing SQL Statements

- **Objective:** Practice writing various SQL queries to retrieve, manipulate, and manage data in a database.
- **Implementation:**
  - **a) Basic SELECT Statements:**
    - Write queries to retrieve specific columns from a table.
    - Use the WHERE clause to filter data based on conditions.
  - **b) Restricting and Sorting Data:**
    - Use the WHERE clause with different operators (>, <, =, !=, BETWEEN, LIKE).
    - Use the ORDER BY clause to sort data in ascending or descending order.
  - **c) Displaying Data from Multiple Tables:**
    - Use the JOIN clause (INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL OUTER JOIN) to combine data from multiple tables.
  - **d) Aggregating Data using Group Functions:**
    - Use functions like SUM, AVG, COUNT, MIN, MAX to perform calculations on groups of data.

        - Use the GROUP BY clause to group data based on one or more columns.
    - **e) Manipulating Data:**
        - Use the INSERT, UPDATE, and DELETE statements to modify data in tables.
    - **f) Creating and Managing Tables:**
        - Use the CREATE TABLE, ALTER TABLE, and DROP TABLE statements to create, modify, and delete tables.

4. Creating Procedures and Functions

- **Objective:** Learn how to create and use stored procedures and functions in a database.
- **Implementation:**
    - Write a stored procedure to perform a specific task (e.g., insert a new customer record).
    - Write a user-defined function to calculate a value (e.g., calculate the total amount of an order).
    - Demonstrate how to call and execute stored procedures and functions.

5. Design and Implementation of Student Information System

- **Objective:** Design a database schema for a student information system and implement it using SQL.
- **Implementation:**
    - Identify the entities and attributes involved (e.g., students, courses, professors, grades).
    - Create tables to store the data.
    - Define relationships between tables using primary and foreign keys.
    - Write SQL queries to perform various operations on the student data (e.g., add a new student, enroll a student in a course, update grades).

6. Writing a Cursor to Display List of Clients

- **Objective:** Learn how to use cursors to process data row-by-row.
- **Implementation:**
    - Write a PL/SQL block that uses a cursor to fetch each client record from the client master table.
    - Process each record and display the client information.

7. Executing Queries Related to Group By and Having Clause

- **Objective:** Practice writing SQL queries using the GROUP BY and HAVING clauses to analyze sales data.
- **Implementation:**
    - Write queries to find the total sales for each product.
    - Find the customers who have placed the most orders.
    - Identify the salespersons with the highest sales volume.

8. Executing Queries Related to Constraints

- **Objective:** Understand and apply various constraints in SQL to ensure data integrity.
- **Implementation:**
  - **a) NOT NULL:** Enforce that a column cannot have null values.
  - **b) UNIQUE:** Ensure that each value in a column is unique.
  - **c) PRIMARY KEY:** Designate a column or a set of columns as the primary key of a table.
  - **d) CHECK:** Validate that data in a column meets specific conditions.
  - **e) Define Integrity Constraints in ALTER TABLE Command:** Modify existing tables to add or remove constraints.

9. Executing Nested Queries

- **Objective:** Learn how to write nested queries to retrieve complex data.
- **Implementation:**
  - Write queries to find customers who have placed orders for a specific product.
  - Find the salespersons who have sold products to a specific customer.
  - Find the products that have been ordered by all customers.

10. Executing Queries Related to Set Operators

- **Objective:** Understand and apply set operators (UNION, INTERSECTION, DIFFERENCE) in SQL.
- **Implementation:**
  - Find the list of customers who have placed orders or have made inquiries.
  - Find the list of products that have been ordered by both customers A and B.
  - Find the list of customers who have placed orders but have not made any inquiries.

Lab Experiments

Experiment No: 1

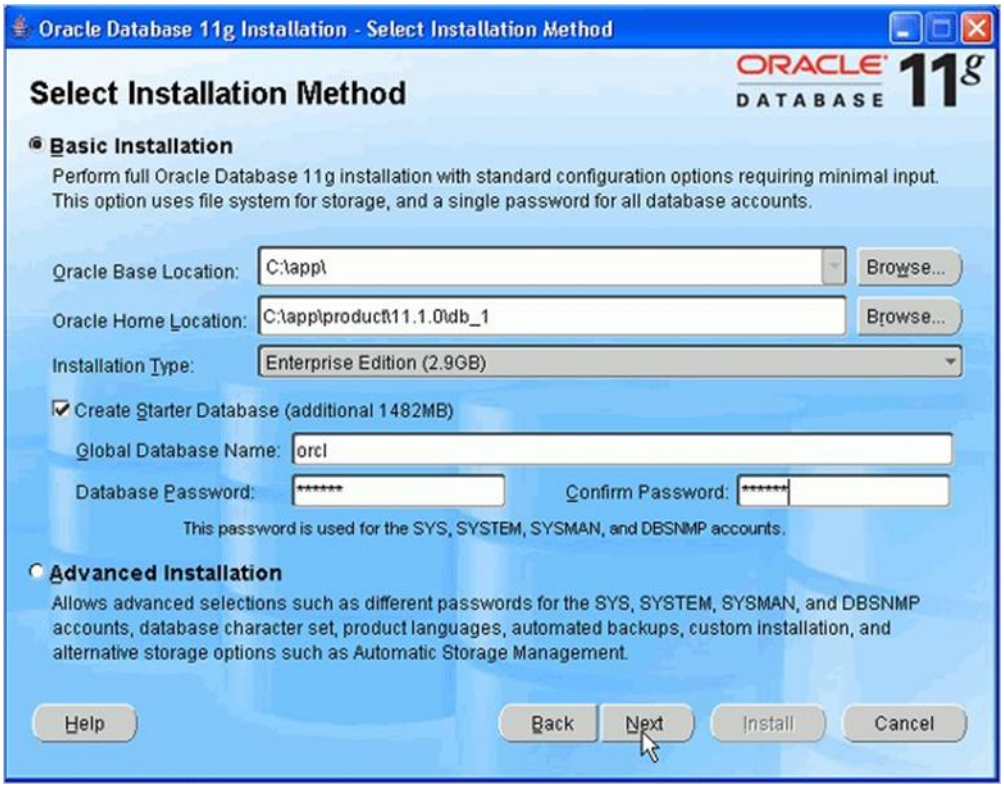
Program Name: Installing Oracle

Theory Concept: To install the software, you must use the Universal installer.

Implementation:

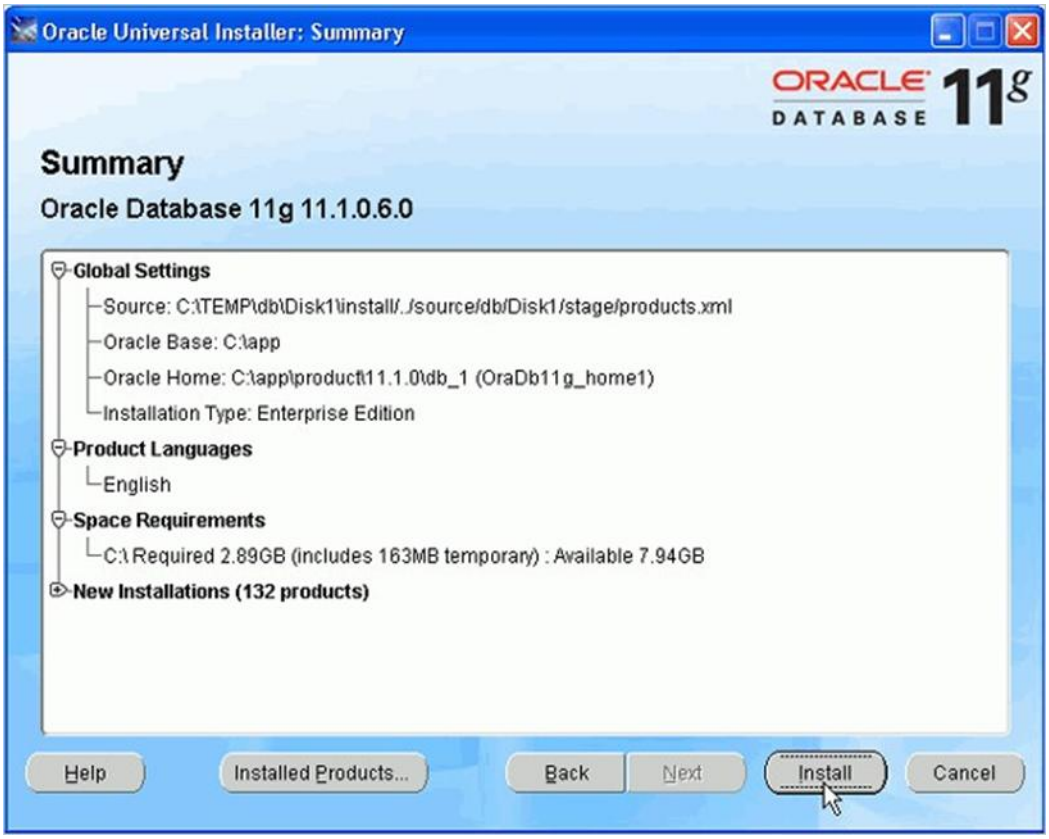
1. For this installation, you need either the DVDs or a downloaded version of the DVDs. In this tutorial, you install from the downloaded version. From the directory where the DVD files were unzipped, open Windows Explorer and double-click on setup.exe from the \db\Disk1 directory.
2. The product you want to install is Database 11g. Make sure the product is selected and click Next.

3. You will perform a basic installation with a starter database. Enter orcl for the Global Database Name and for Database Password and Confirm Password. Then, click **Next.**

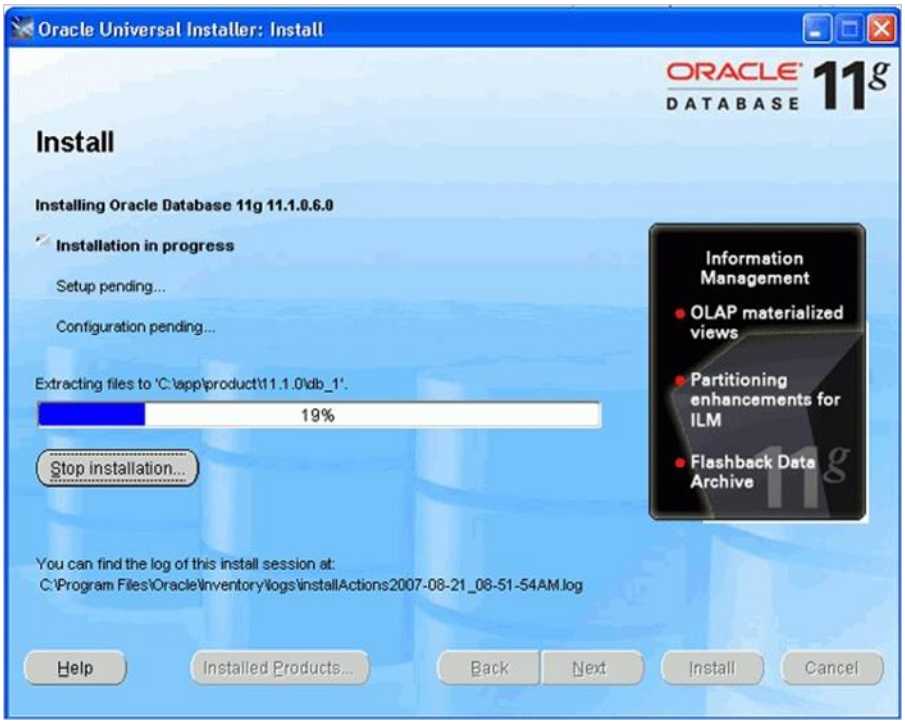


4. Configuration Manager allows you to associate your configuration information with your Metalink account. You can choose to enable it on this window. Then, click **Next**.

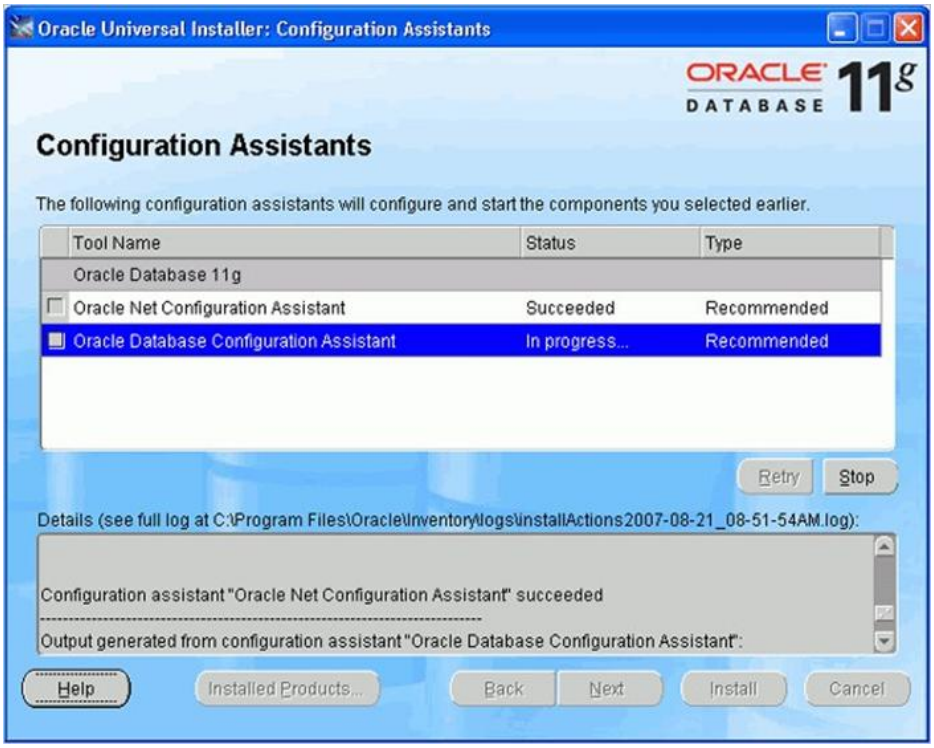
5. Review the Summary window to verify what is to be installed. Then, click **Install**.



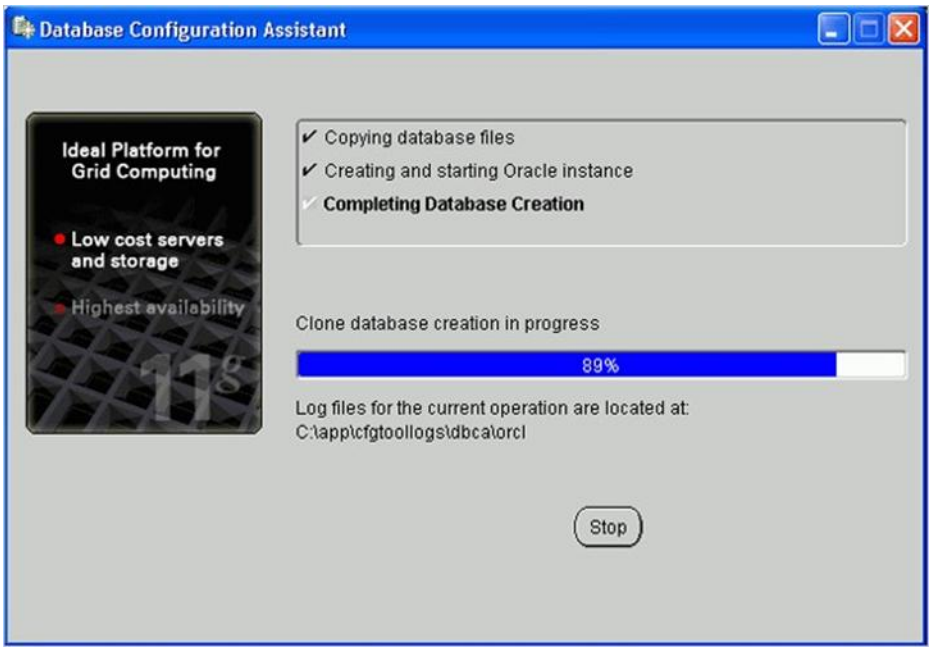
6. The progress window appears.



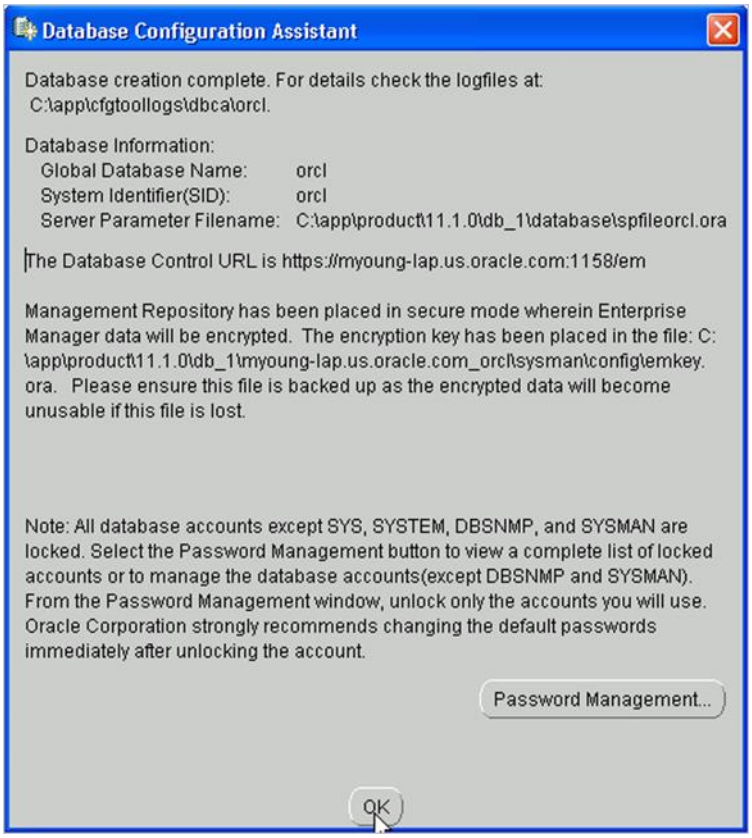
7. The Configuration Assistants window appears.



8. Your database is now being created.



9. When the database has been created, you can unlock the users you want to use. Click OK.



10. Click **Exit**. Click **Yes** to confirm exit.

Experiment No: 2

Program Name: Creating Entity-Relationship Diagram using case tools.

Steps:

Step 1: Install MySQL Workbench

If you don't already have MySQL Workbench installed, you can download it from the official MySQL website: https://www.mysql.com/products/workbench/

Step 2: Launch MySQL Workbench

After installation, launch MySQL Workbench on your computer.

Step 3: Create a New EER Diagram

Click on "File" in the menu bar.
Select "New Model" to create a new Entity-Relationship Diagram (ERD).

Step 4: Add Entities and Attributes

In the diagram canvas, you can add entities by clicking on the "Entity" button in the toolbar and then clicking on the canvas to place the entity.
Double-click on the entity to give it a name.
To add attributes to an entity, right-click on the entity and select "Add Attribute."

Step 5: Define Relationships

To define relationships between entities, select the "Relationship" tool from the toolbar.
Click on one entity and then click on the related entity to establish a relationship.
Specify the cardinality and other properties of the relationship.

Step 6: Save Your ERD

It's important to save your work. Click on "File" and then "Save" to save the model.

Step 7: Generate SQL Script (Optional)

MySQL Workbench allows you to generate SQL scripts from your ERD. You can do this by clicking on "Database" and then "Forward Engineer..." to create a database schema based on your ERD.

Step 8: Review and Export (Optional)

You can review your ERD, make any necessary changes, and then export it in different formats, such as PNG or PDF.

Output Examples:

1. First make sure you have a **Database** and **Tables** created on the MySQL server.

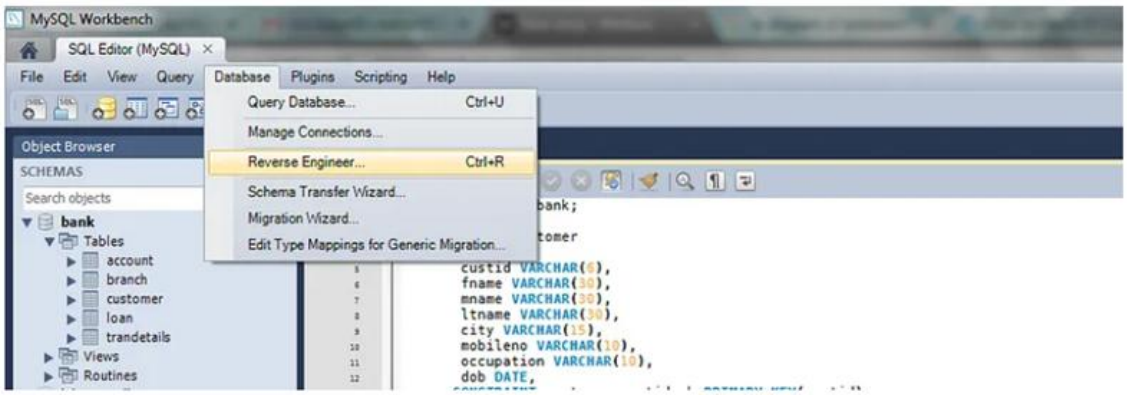
Example :-

Database - *bank.*

Tables - *account, branch, customer, loan, trandetails.*

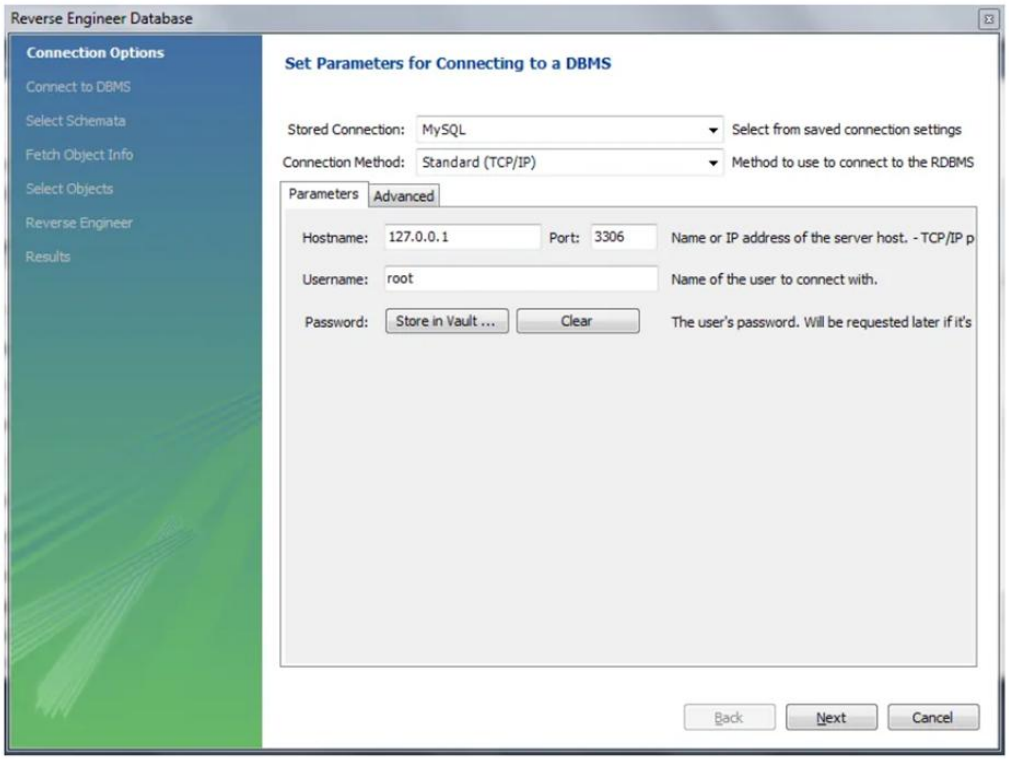
2. Click on **Database** -> **Reverse Engineer.**

2. Click on **Database** -> **Reverse Engineer.**

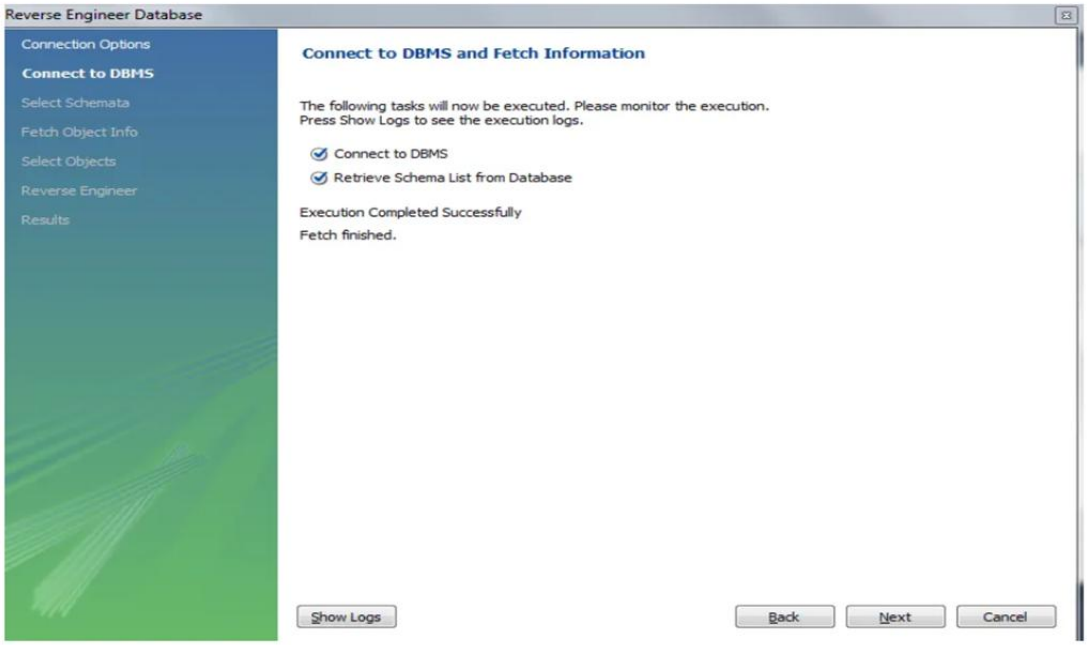


3. Select your **stored connection** *(for connecting to your MySQL Server in which database is present)* from the dropdown. Then click **Next.**

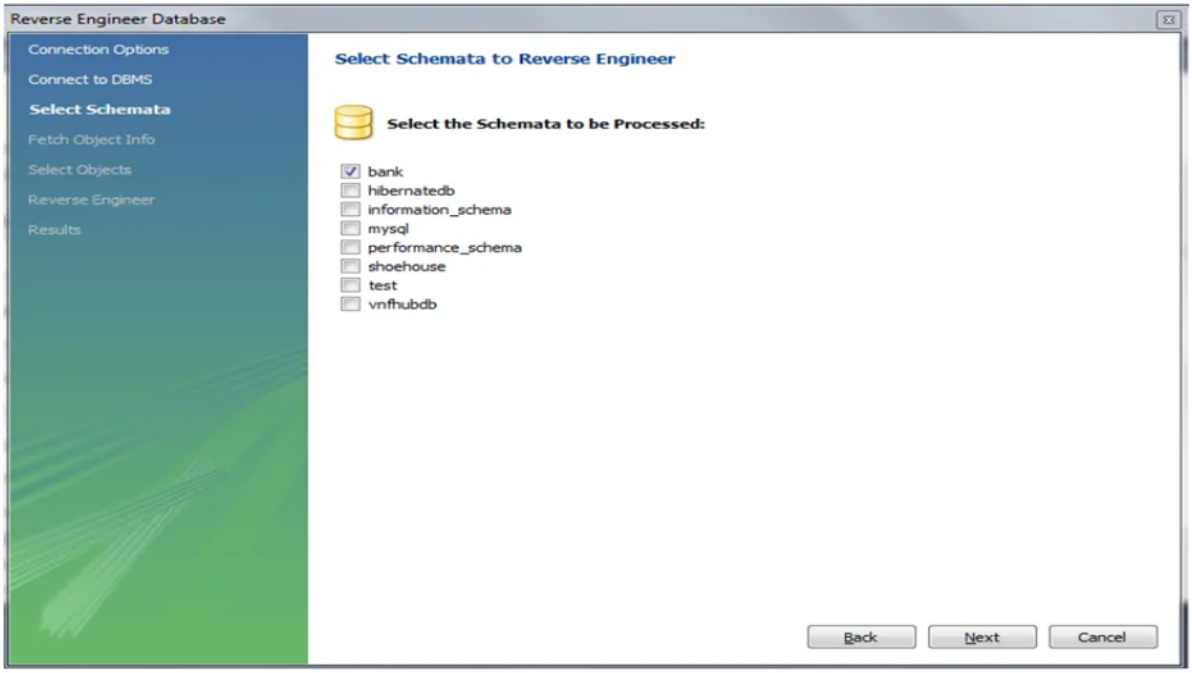
3. Select your **stored connection** *(for connecting to your MySQL Server in which database is present)* from the dropdown. Then click **Next.**



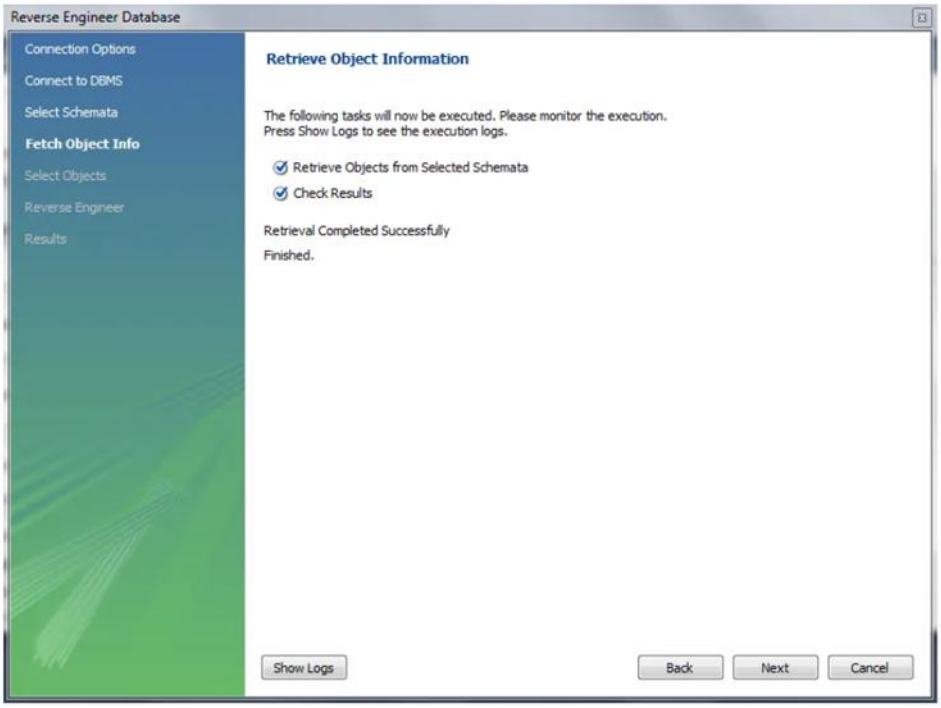
4. After the execution gets completed successfully *(connection to DBMS)*, click **Next.**



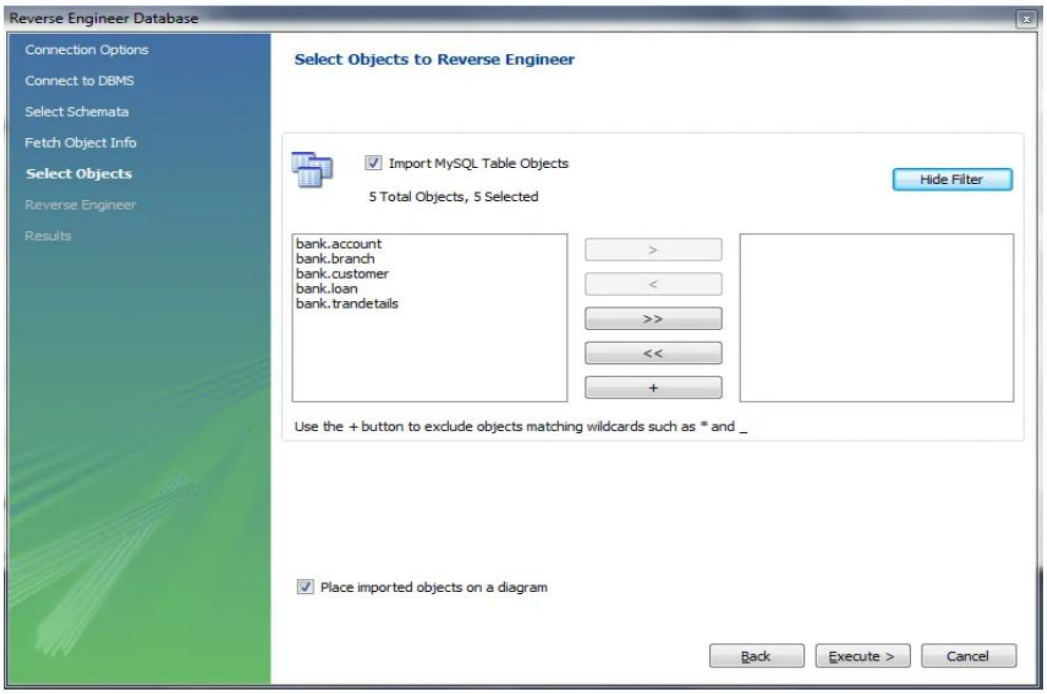
5. Select your Database from the MySQL Server for which you want to create the ER Diagram (*in our case the database name is "**bank**"*), then click **Next.**



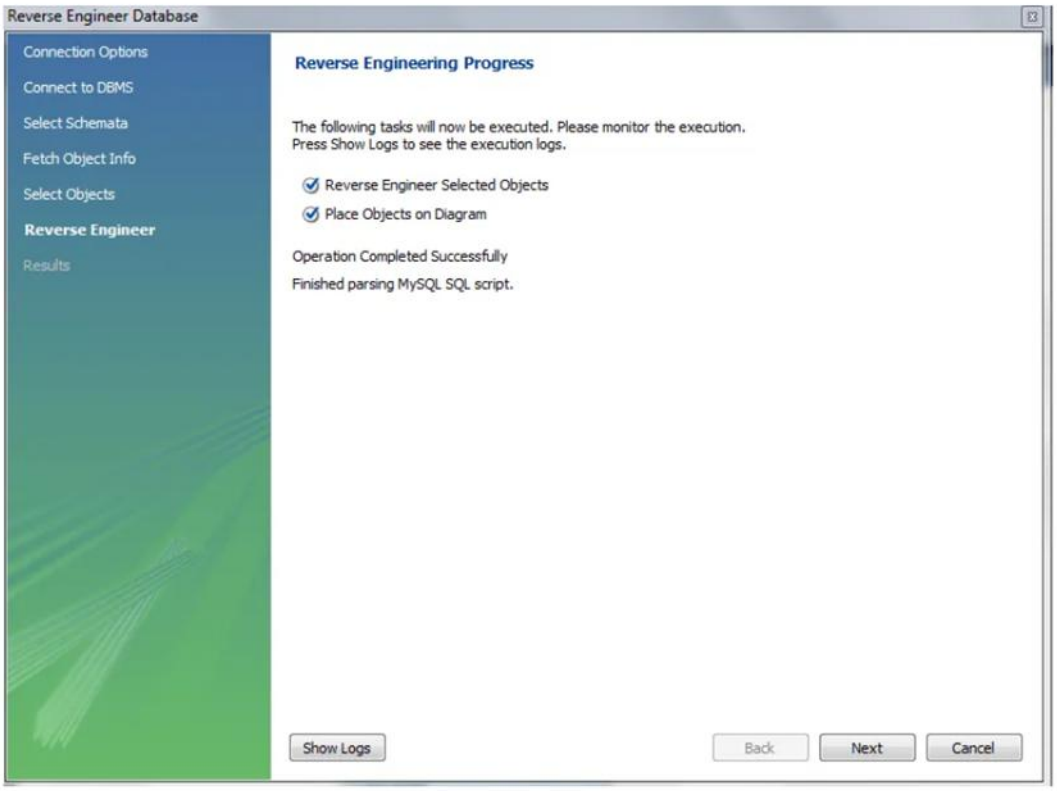
6. After the retrieval gets **completed** successfully for the selected Database, click **Next.**



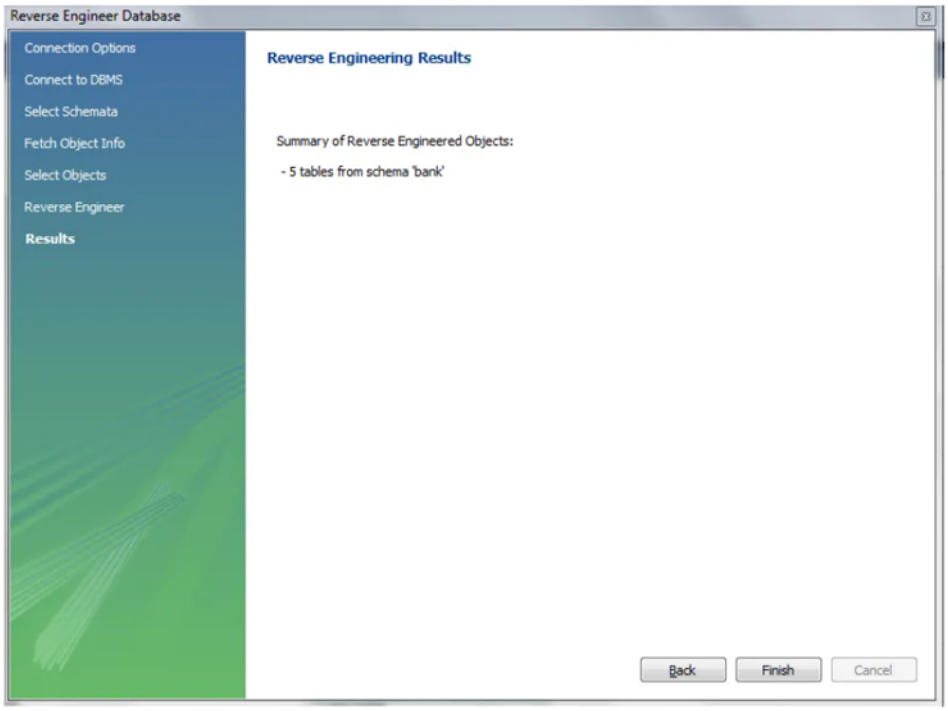
7. Select the Tables of the Database which you want to be visible on the ER Diagram *(In this case I am importing all the tables of the DB),* then click **Execute>.**

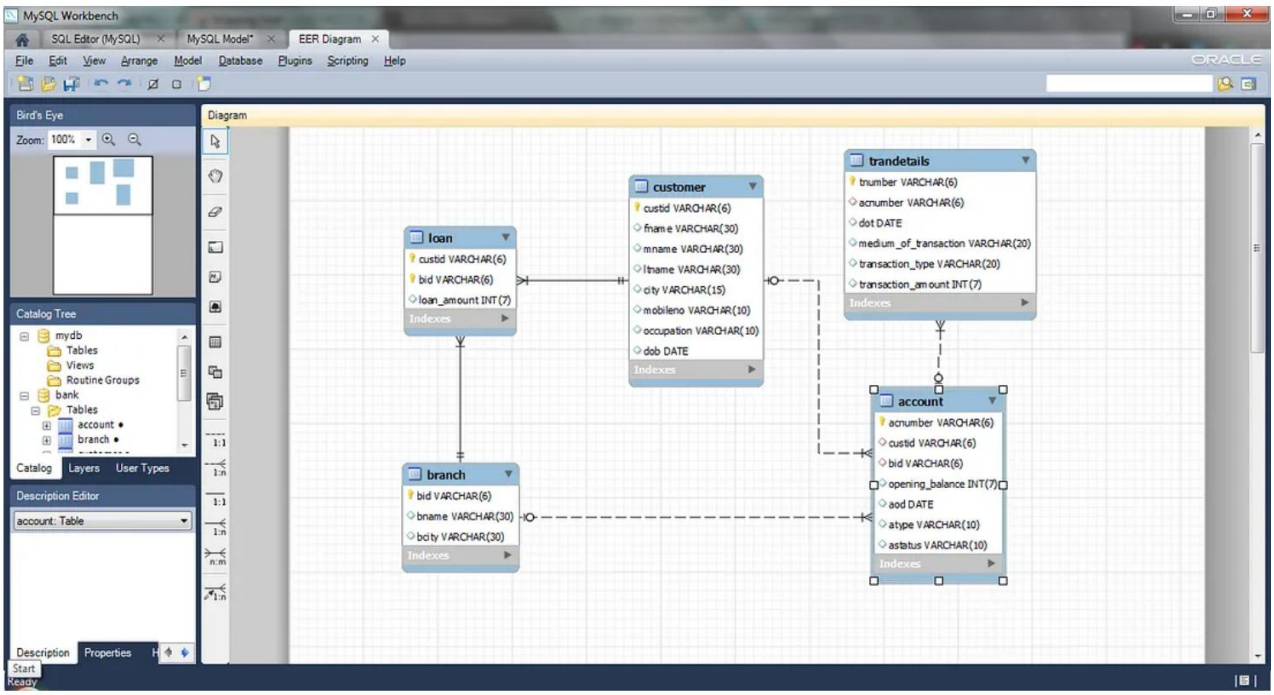


8. After the Reverse Engineering Process gets completed successfully, click **Next.**



9. Click **Finish.**





Experiment No: - 3

Program Name: Writing SQL statements Using ORACLE /MYSQL:

a) Writing basic SQL SELECT statements.
b) Restricting and sorting data.
c) Displaying data from multiple tables.
d) Aggregating data using group function.
e) Manipulating data.
f) Creating and managing tables.

SQL statements using MYSQL:

a) **Writing basic SQL SELECT statements.**

```
-- Select all columns from a table

SELECT * FROM employees;


-- Select specific columns from a table

SELECT first_name, last_name FROM employees;


-- Select distinct values from a column

SELECT DISTINCT department_id FROM employees;


-- Select data with a filter (WHERE clause)

SELECT * FROM employees WHERE salary > 50000;


-- Select data with a combination of conditions

SELECT * FROM employees WHERE department_id = 2 AND salary >
50000;
```

b**) Restricting and sorting data.**

```
-- Sorting data in ascending order
SELECT * FROM employees ORDER BY last_name;

-- Sorting data in descending order
SELECT * FROM employees ORDER BY hire_date DESC;
```

```
-- Limiting the number of rows returned
SELECT * FROM employees LIMIT 10;

-- Limiting the number of rows with an offset
SELECT * FROM employees LIMIT 10 OFFSET 20;
```

c) Displaying data from multiple tables (JOIN).

```
-- Inner Join
SELECT orders.order_id, customers.customer_name
FROM orders
INNER JOIN customers ON orders.customer_id =
customers.customer_id;

-- Left Join
SELECT employees.first_name, departments.department_name
FROM employees
LEFT JOIN departments ON employees.department_id =
departments.department_id;
```

d) Aggregating data using group function.

```
-- Calculate the total salary for each department
SELECT department_id, SUM(salary) AS total_salary
FROM employees
GROUP BY department_id;

-- Calculate the average salary
SELECT AVG(salary) AS average_salary
FROM employees;
```

e) Manipulating data (INSERT, UPDATE, DELETE):

```
-- Inserting a new record
INSERT INTO employees (first_name, last_name, salary)
VALUES ('John', 'Doe', 60000);

-- Updating an existing record
UPDATE employees
SET salary = 65000
WHERE employee_id = 101;

-- Deleting a record
DELETE FROM employees
WHERE employee_id = 102;
```

e) Creating and managing tables:

```
-- Creating a new table
CREATE TABLE products (
    product_id INT PRIMARY KEY,
    product_name VARCHAR(255),
    price DECIMAL(10, 2)
);

-- Modifying a table (adding a new column)
ALTER TABLE employees
ADD COLUMN email VARCHAR(255);
-- Dropping a table
DROP TABLE products;
```

Experiment No: - 4

Program Name: Creating procedure and functions.

Theory Concept:

Normalization is a database design process used to organize data in a relational database efficiently and reduce data redundancy. It is a multi-step process that sets the data into tabular form and removes the duplicated data from the relational tables. Normalization typically involves dividing a database into two or more tables and defining relationships between them. Let's go through an example of normalizing a database with sample data and MySQL queries. We'll start with an unnormalized table and normalize it step by step.

Step 1: Create an Unnormalized Table

Suppose we have a table called "CustomerOrders" that stores information about customers and their orders. This table is not normalized because it contains repeating groups and data redundancy:

```
CREATE TABLE CustomerOrders (
    customer_id INT PRIMARY KEY,
    customer_name VARCHAR(255),
    order_id INT,
    order_date DATE,
    total_amount DECIMAL(10, 2)
);
INSERT INTO CustomerOrders (customer_id, customer_name, order_id, order_date,
total_amount)
VALUES
    (1, 'Alice', 101, '2023-01-15', 100.00),
    (1, 'Alice', 102, '2023-02-20', 150.00),
    (2, 'Bob', 201, '2023-03-10', 75.50),
    (3, 'Charlie', 301, '2023-04-05', 200.00);
```

Step 2: Normalize the Data

We'll normalize the data by creating two separate tables: "Customers" and "Orders." The "Customers" table will store customer information, and the "Orders" table will store order information.

```
-- Create the Customers table
CREATE TABLE Customers (
    customer_id INT PRIMARY KEY,
    customer_name VARCHAR(255)
);
```

```
-- Create the Orders table
CREATE TABLE Orders (
    order_id INT PRIMARY KEY,
    customer_id INT,
    order_date DATE,
    total_amount DECIMAL(10, 2),
    FOREIGN KEY (customer_id) REFERENCES Customers(customer_id)
);

-- Populate the Customers table with unique customer information
INSERT INTO Customers (customer_id, customer_name)
SELECT DISTINCT customer_id, customer_name FROM CustomerOrders;

-- Populate the Orders table with order information
INSERT INTO Orders (order_id, customer_id, order_date, total_amount)
SELECT order_id, customer_id, order_date, total_amount FROM CustomerOrders;
```

Step 3: Query the Normalized Tables

Now that we have normalized our data, we can query the "Customers" and "Orders" tables to retrieve information:

```
-- Query to retrieve customer information
SELECT * FROM Customers;
-- Query to retrieve order information
SELECT * FROM Orders;


-- Query to retrieve customer names and their total order amounts
SELECT c.customer_name, SUM(o.total_amount) AS total_order_amount
FROM Customers c
JOIN Orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_name;
```

Output:

These queries demonstrate the result of normalizing the data. The "Customers" table contains unique customer information, and the "Orders" table stores order details with a reference to the customer. The last query retrieves the total order amount for each customer, demonstrating the power of relational databases and normalization.

Experiment No-5

Program Name: Design and implementation of Student Information System.

Theory Concept:

Designing and implementing a Student Information System (SIS) experiment in a Database Management System (DBMS) is a practical way to learn about database design and development. Below, I'll outline a simplified experiment scenario for creating a basic SIS using a relational DBMS (e.g., MySQL, PostgreSQL). This experiment assumes you have basic knowledge of SQL and database concepts.

Experiment Scenario:

You are tasked with creating a Student Information System (SIS) for a university. The system should store information about students, courses, and grades. Students can enroll in courses, and teachers can enter grades for students in those courses.

Experiment Steps:

1. Database Design:

Define the database schema with tables for students, courses, and grades. Here's a simplified schema:

```
-- Students table
CREATE TABLE students (
    student_id INT PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    birthdate DATE,
    email VARCHAR(100)
);

-- Courses table
CREATE TABLE courses (
    course_id INT PRIMARY KEY,
    course_name VARCHAR(100),
    teacher VARCHAR(100)
);

-- Grades table
CREATE TABLE grades (
    grade_id INT PRIMARY KEY,
    student_id INT,
    course_id INT,
    grade VARCHAR(2),
    FOREIGN KEY (student_id) REFERENCES students(student_id),
    FOREIGN KEY (course_id) REFERENCES courses(course_id)
);
```

2. Data Population:

Insert sample data into the tables for testing purposes.

```
-- Insert sample students
INSERT INTO students (student_id, first_name, last_name, birthdate, email)
VALUES
   (1, 'John', 'Doe', '1995-01-15', 'john@example.com'),
   (2, 'Jane', 'Smith', '1996-03-22', 'jane@example.com');

-- Insert sample courses
INSERT INTO courses (course_id, course_name, teacher)
VALUES
   (101, 'Mathematics 101', 'Dr. Smith'),
   (102, 'Computer Science 101', 'Prof. Johnson');

-- Enroll students in courses
INSERT INTO grades (student_id, course_id, grade)
VALUES
   (1, 101, 'A'),
   (1, 102, 'B'),
   (2, 101, 'B');
```

3. Querying the Database:

Practice querying the database to retrieve information. For example, you can retrieve a student's grades or find courses taught by a specific teacher.

```
-- Get a student's grades
SELECT s.first_name, s.last_name, c.course_name, g.grade
FROM students s
JOIN grades g ON s.student_id = g.student_id
JOIN courses c ON g.course_id = c.course_id
WHERE s.student_id = 1;

-- Find courses taught by a specific teacher
SELECT course_name
FROM courses
WHERE teacher = 'Dr. Smith';
```

4. CRUD Operations:

Practice performing CRUD (Create, Read, Update, Delete) operations on the database. For example, you can add a new student, update a student's information, or delete a course.

```
-- Create: Add a new student
INSERT INTO students (student_id, first_name, last_name, birthdate, email)
VALUES (3, 'Alice', 'Johnson', '1997-05-10', 'alice@example.com');

-- Update: Change a student's email
UPDATE students
SET email = 'new_email@example.com'
WHERE student_id = 3;

-- Delete: Remove a course
DELETE FROM courses
WHERE course_id = 102;
```

Experiment No: 6

Program Name: Write a CURSOR to display list of clients in the client Master Table.

Theory Concept: The following example would illustrate the concept of CURSORS. We will be using the CLIENT_MASTER table and display records.

Implementation:

```
DECLARE
   CURSOR client_cur is SELECT id, name, address
FROM client_master;
client_rec client_cur%rowtype; BEGIN
OPEN client_cur;
LOOP
FETCH client_cur into client_rec; EXIT WHEN client_cur%notfound;
DBMS_OUTPUT.put_line(client_rec.id||''||client_rec.name);END LOOP;
      END;
/
```

Output:

When the above code is executed at SQL prompt, it produces the following result:

1 Ramesh
2 Khilan
3 kaushik
4 Chaitali
5 Hardik
6 Komal

PL/SQLprocedure successfully completed.

Experiment No -7

Program Name: Execute the queries related to Group By and having Clause on tables SALES_ORDER.

Theory Concept:

The program aims to familiarize the user with grouping of databased on conditions to ensure better usability of data.

Implementation:

GROUPBY

Q1) Create table sales_order with attributes product_no and Qty. Insert records into the table and find the total qty ordered foreach product_no.

Ans: Create table sales_order (product_novarchar(10), Qty numbe(4));

Output: Table created.

insert into sales_order values(&product_no, &qty);
select* from sales_order;

Output:

PRODUCT_NO	QTY
p1	12
p2	112
p1	9
p2	23
p3	23
p3	23

6 rows selected.

Select product_no, sum(qty) from sales_order group by product_no;

Output:

PRODUCT_NO	SUM(QTY)
p1	21
p2	135
p3	46

3 rows selected.

HAVING clause

Q2) Find the total Qty for product_no ‘p1’ and ‘p2’ from the Table sales_order

Ans: select product_no, sum(qty) from sales_order group by product_no having product_no = 'p1' OR product_no = 'p2';

Output:

PRODUCT_NO	SUM(QTY)
P1	21
P3	46

2 rows selected

Experiment No -8

Program Name: Execute the following queries:

a) The NOT NULL
b) The UNIQUE Constraint
c) The PRIMARY KEY Constraint
d) The CHECK Constraint
e) Define Integrity Constraints in ALTER table Command

a) The NOT NULL Constraint:

The NOT NULL constraint ensures that a column cannot contain NULL (empty) values.
Here's an example:

```
-- Create a table with a NOT NULL constraint
CREATE TABLE employees (
employee_id INT PRIMARY KEY,
first_name VARCHAR(50) NOT NULL,
last_name VARCHAR(50) NOT NULL,
hire_date DATE NOT NULL
);
```

b) The UNIQUE Constraint:

The UNIQUE constraint ensures that the values in a column are unique across all rows in a table. Here's an example:

```
-- Create a table with a UNIQUE constraint
CREATE TABLE products (
    product_id INT PRIMARY KEY,
    product_name VARCHAR(100) UNIQUE,
    price DECIMAL(10, 2)
);
```

```
-- Insert rows with unique product names
INSERT INTO products (product_id, product_name, price)
VALUES (1, 'Laptop', 1000.00),
      (2, 'Smartphone', 600.00);

-- Attempt to insert a row with a duplicate product name, which will result in an error
INSERT INTO products (product_id, product_name, price)
VALUES (3, 'Laptop', 1200.00);
```

c) The PRIMARY KEY Constraint:

The PRIMARY KEY constraint defines a unique identifier for each row in a table. Here's an example:

```
-- Create a table with a PRIMARY KEY constraint
CREATE TABLE students (
   student_id INT PRIMARY KEY,
   first_name VARCHAR(50),
   last_name VARCHAR(50),
   birth_date DATE
);

-- Insert rows with unique student IDs
INSERT INTO students (student_id, first_name, last_name, birth_date)
VALUES (1, 'John', 'Doe', '1995-01-15'),
      (2, 'Jane', 'Smith', '1996-03-22');
```

d) The CHECK Constraint:

The CHECK constraint allows you to specify a condition that must be met for data to be valid. Here's an Example:

```
-- Create a table with a CHECK constraint
CREATE TABLE orders (
   order_id INT PRIMARY KEY,
   order_date DATE,
   total_amount DECIMAL(10, 2),
   payment_status VARCHAR(20) CHECK (payment_status IN ('Paid',
'Unpaid', 'Pending'))
);

-- Insert rows with valid payment statuses
INSERT INTO orders (order_id, order_date, total_amount, payment_status)
VALUES (1, '2022-01-01', 500.00, 'Paid'),
      (2, '2022-02-01', 750.00, 'Unpaid');

-- Attempt to insert a row with an invalid payment status, which will result in
an error
INSERT INTO orders (order_id, order_date, total_amount, payment_status)
VALUES (3, '2022-03-01', 300.00, 'InvalidStatus');
```

e) Define Integrity Constraints in ALTER TABLE Command:

You can also define integrity constraints using the ALTER TABLE command. Here's an example of adding a NOT NULL constraint to an existing table:

```
-- Add a NOT NULL constraint to an existing column
ALTER TABLE employees
ALTER COLUMN hire_date DATE NOT NULL;
```

Experiment No: 9

Program Name:

Execute Nested Queries on tables CLIENT_MASTER, PRODUCT_MASTER, SALESMAN_MASTER, SALES_ORDER, SALES_ORDER_DETAILS

Theory Concept:

The program in tends to familiarize nested queries so as to retrieve data from are cord by using filtered data from another record.

Implementation:

Q1) Retrieve the order numbers, client names and their order dates from client_master and sales_ordertables.

Ans: Selectorder_no, order_date, name from sales order, client_master where client_master. client_no=sales_order. client_no order by order_date;

OUTPUT:

Order_no	order_date	name
1	1999/12/05	akansha
2	1999/12/12	divya

Q2) Retrieve the product numbers, description and total quantity ordered for each product

Ans: Selectsales_order_details.product_no, description, sum(qty_ordered) from sales_order_details, product_master where product_master.product_no = sales_order_details.product_no group by sales_order_details.product_no, description;

OUTPUT:

product_no	description	sum(qty_ordered)
1	chair	2
2	pen	5

Q3) **Retrieve the names of employees and names of their respective managers from the employee table.**

Ans: Select employee.name, employee.name from employee where employee.manager_no =employee.employee_no;

OUTPUT:

Name	Name
Akansha	Divya
Akshita	Divya

UNION, INTERSECT and MINUS CLAUSE

Q1) Retrieve the names of all clients and salesmen in the city of Mumbai from the tables client_master and salesman_master.

Ans: Select salesman_no from salesman_master where city = 'Mumbai' UNION Select client_no from client_master where city = 'Mumbai';

OUTPUT:

Name
Akansha
Akshita
Divya

Q2) Retrieve the sales man name in Mumbai whose efforts have resulted into atleast one sales transaction

Ans: Select sales man_no, name from salesman_master where city = 'Mumbai' INTERSECT Select salesman_master.salesman_no, name from salesman_master, sales_order where salesman_master. salesman_no = sales_order.salesman_no;

OUTPUT:

Saleman_no	Name
1	akansha
2	divya

Q3) Retrieve all the product numbers of non-moving items from the product_master table

Ans: Select product_no from product_master Minus
Select product_no from sales_order_details;

OUTPUT:

product_no
3
4

VIEWS

Q1) Create a view on salesman_master table for the sales department

Ans: Create view vw_sales as select * from salesman_master;

OUTPUT:

View created

Q2) Create a view on client_master table

Ans: Create view vw_client as select name, address1, address2, city, pincode, state, bal_due from client_master;

OUTPUT:

View created

Q3) Perform insert, modify and delete operations on the view created in Q2

Ans:

a) Insert into vw_client values('C001', 'Robert', 'AAAAAA', 'BBB', 'Delhi', 2000000, 'MMM');

OUTPUT:

1 rows created

b) Update vw_client set bal_due = 10000 where client_no = 'C001';

OUTPUT:

1 row updated

c) Delete from vw_client where client_no = 'C001';

OUTPUT:

1 row deleted

Experiment No-10

Program Name: Execute queries related to Exists, Not Exists, Union, Intersection, Difference, Join on tables CLIENT_MASTER, PRODUCT_MASTER, SALESMAN_MASTER, SALES_ORDER, SALES_ORDER_DETAILS

Theory Concept:
The program retrieves data from records by defining relation between two tables so as to retrieve filtered records.

Implementation:

Correlated queries with EXISTS/ NOT EXISTS clause

1) **Select all products and order_no where order_status is 'in Process'**
Ans: Select order_no.,product_no. from sales_order_details where exists(select * from sales_order ,order_no = sales_order_details,order_no and order_status='in process');
Output:

Order_no	Product_no
0003	3

2) **Select order_no and order_date for all orders which include product_no 'P001' and quantity_ordered>10**
Ans: Select order_no, order_data from sales_order where exists(select * from sales_order_details where sales_order_details, order_no = sales_order.Order_no and product-no='p001' and quantity-ordered>10;**Output:**

Order_no	Product_no
0002	05/feb/13

3) **Find all order_no for salesman rashmi.**
Ans: Select order_no from sales_order where exists(select * from salesman_master where sales man_master.saleman-no=sales_order-salesman_no and name='rashmi');
Output:

Order_no
3

4) **Select all clients who have not placed any orders.**
Ans: Select * from client_master where not exists(select * from sales_order.client_no=client_master.client_no);

Output:

Client_no	Name	City	Pincode	State
6	Divya	Hapur	35498	U.P.
7	Dorothy	Noida	32547	U.P.

5) Select all orders with order_date for 'acrylic colors'

Ans: Select order_no,order_date from sales_order where exists(select * fromsales_order_details.oder_no=sales_order.order_noANDexists(select*frompr oduct1wheresales_order_details.product_no=product_noAND description='acrylic colors');

Output:

Order_no	Order_date
0001	23/jan/13

Union,Intersect and minus clause:

1) List all the clients and salesman and their names

Ans: Select client_no, name from client_master UNION select salesman_no, name from salesman_master;

Output:

Client_no	Name
3	Akshita
4	Dhawal

2) List all the clients and their names who are also salesman.

Ans: Select name from client_master INTERSECT,
select name from salesman_master;

Output:

No rows selected

3) List all the clients who are not salesman.

Ans: Select name from client_master MINUS select name from salesman_master;

Output:

Name
Akshita
Dhawal
Akansha
Divya
Dorothy

4) **List all the clients who have placed orders**

Ans: Select client_no from client_masterINTERSECTselectclient_no from sales_order;

Output:

Client_no
6
7

5) **List all the clients who have not placed any order.**

Ans: Select client_no from client_master MINUS
select client_no from sales_order;

Output:

Client_no
3
4
5

6) **List all the clients in UP who have placed orders**

Ans: Select client_no from client_master where state='UP'
INTERSECT select client_no from sales_order;

Output:

Client_no
3
4
5

7) **Findalltheclientsandtheirnamesfromcity Ghaziabadwhohavedeliverydateoftheirordersastoday.**

Ans: Select client_no from client_master where city='Ghaziazbad'
INTERSECT select client_no fromsales_order where delivery_date='09-MAR-13'

Output:

Client_no
5

Queries on Joins

1) **List the product_no and description of products sold.**

Ans: Select product_no, description from (product1 natural join sales_order_details)

Output:

Product_no	Description
1	Chair
1	Chair
2	Table
3	Sofa

2) **Find the products which have been sold to 'akansha'**

Ans: Select product_no, description from (product1 natural join sales_order details natural joins ales_order natural join client_master) where name='akansha';

Output:

Product_no	Description
3	Sofa

3) **Find the products and their quantities that will have to be delivered in the current month.**

Ans: Select sales_order_details product_no, product1, description, sum(sales_order_details, quantity_ordered) from sales_order_details, sales_order, product1 where product1, product_no=sales_order_details, product_no and sales_order, order_no=sales_order_details, order_no and to_char (delivery_date,'mon-yy') = to_char(sysdate,'mon-yy')group by sales_order_details, product_no, product1, description ;

Output:

no rows selected

4) Find the names of client who have purchased 'chair'

Ans: Select name from(client_master natural join sales_order natural join sales_order_details natural join product1) where description= 'chair';

Output:

Name
Akshita
Akansha

5) **List the orders for less than 5 units of sale of chair'**

Ans: Select product_no, order_no from (sales_order_details natural join product1) where(description='chair' and qty_ordered<5);

Output:

Product_no	Order_no
1	0001
1	0001

6) **Find the products and their quantities placed by 'akansha'or 'akshita'.**

Ans: Select product_no, description, qty_ordered from(product1 natural join sales_order_details natural join sales_order_natural join client_master) where (name='akansha'or name='akshita');

Output :

Product_no	Description	Qty_ordered
1	Chair	4
1	Chair	3
2	Sofa	2

7) Find the products and their quantities for the orders placed by the client_no '3'and '5'

Ans: Select product_no, description, qty_ordered from(product1 natural join sales_order_details natural join sales_order natural join client_master) where (client_no=3 OR client_no=5);

Output:

PRODUCT_NO	DESCRIPTION	QTY_ORDERED
1	Chair	4
1	Chair	3