SIR M. VISVESVARAYA INSTITUTE OF TECHNOLOGY

(Affiliated to VTU & recognized by AICTE, New Delhi)

ACCREDATED BY NBA

Department of Computer Science & Engineering

2023 - 2024

DBMS LABORATORY WITH MINI PROJECT

21CSL55

5TH SEMESTER

Prepared By: Dr. Suma Swamy and Dr. Soumya Patil

Compiled By: Radhika R

Under the Guidance of

Dr. Anitha T N, Prof. & Head

DATABASE MANAGEMENT SYSTEM LABORATORY WITH MINI PROJECT			
Course Code	**21CSL55**	CIE Marks	50
Teaching Hours/Week (L:T:P: S)	0:0:2:0	SEE Marks	50
Total Hours of Pedagogy	24	Total Marks	100
Credits	01	Exam Hours	03

Course Learning Objectives:
CLO 1. Foundation knowledge in database concepts, technology and practice to groom students into well-informed database application developers.
CLO 2. Strong practice in SQL programming through a variety of database problems.
CLO 3. Develop database applications using front-end tools and back-end DBMS..

Sl. No.	**PART-A: SQL Programming (Max. Exam Marks. 50)** Design, develop, and implement the specified queries for the following problems using Oracle, MySQL, MS SQL Server, or any other DBMS under LINUX/Windows environment. Create Schema and insert at least 5 records for each table. Add appropriate database constraints.
1	Aim: Demonstrating creation of tables, applying the view concepts on the tables. Program: Consider the following schema for a Library Database: **BOOK(Book_id, Title, Publisher_Name, Pub_Year)** **BOOK_AUTHORS(Book_id, Author_Name)** **PUBLISHER(Name, Address, Phone)** **BOOK_COPIES(Book_id, Programme_id, No-of_Copies)** **BOOK_LENDING(Book_id, Programme_id, Card_No, Date_Out, Due_Date)** **LIBRARY_PROGRAMME(Programme_id, Programme_Name, Address)** Write SQL queries to 1. Retrieve details of all books in the library – id, title, name of publisher, authors, number of copies in each Programme, etc. 2. Get the particulars of borrowers who have borrowed more than 3 books, but from Jan 2017 to Jun 2017. 3. Delete a book in BOOK table. Update the contents of other tables to reflect this data manipulation operation. 4. Partition the BOOK table based on year of publication. Demonstrate its working with a simple query. 5. Create a view of all books and its number of copies that are currently available in the Library. Reference: https://www.youtube.com/watch?v=AaSU-AOguls https://www.youtube.com/watch?v=-EwEvJxS-Fw
2	Aim: Discuss the various concepts on constraints and update operations. Program: Consider the following schema for Order Database: **SALESMAN(Salesman_id, Name, City, Commission)** **CUSTOMER(Customer_id, Cust_Name, City, Grade, Salesman_id)** **ORDERS(Ord_No, Purchase_Amt, Ord_Date, Customer_id, Salesman_id)** Write SQL queries to 1. Count the customers with grades above Bangalore's average. 2. Find the name and numbers of all salesman who had more than one customer. 3. List all the salesman and indicate those who have and don't have customers in their cities (Use UNION operation.) 4. Create a view that finds the salesman who has the customer with the highest order of a day. 5. Demonstrate the DELETE operation by removing salesman with id 1000. All his orders must also be deleted. Reference: https://www.youtube.com/watch?v=AA-KL1jbMeY https://www.youtube.com/watch?v=7S_tz1z_5bA

DBMS LABORATORY WITH MINIPROJECT 21CSL55

3	Aim: Demonstrate the concepts of JOIN operations. Program: Consider the schema for Movie Database: **ACTOR(Act_id, Act_Name, Act_Gender)** **DIRECTOR(Dir_id, Dir_Name, Dir_Phone)** **MOVIES(Mov_id, Mov_Title, Mov_Year, Mov_Lang, Dir_id)** **MOVIE_CAST(Act_id, Mov_id, Role)** **RATING(Mov_id, Rev_Stars)** Write SQL queries to 1. List the titles of all movies directed by 'Hitchcock'. 2. Find the movie names where one or more actors acted in two or more movies. 3. List all actors who acted in a movie before 2000 and also in a movie after 2015(use JOIN operation). 4. Find the title of movies and number of stars for each movie that has at least one rating and find the highest number of stars that movie received. Sort the result by movie title. 5. Update rating of all movies directed by 'Steven Spielberg' to 5. Reference: https://www.youtube.com/watch?v=hSiCUNVKJAo https://www.youtube.com/watch?v=Eod3aQkFz84
4	Aim: Introduce concepts of PLSQL and usage on the table. Program: Consider the schema for College Database: **STUDENT(USN, SName, Address, Phone, Gender)** **SEMSEC(SSID, Sem, Sec)** **CLASS(USN, SSID)** **COURSE(Subcode, Title, Sem, Credits)** **IAMARKS(USN, Subcode, SSID, Test1, Test2, Test3, FinalIA)** Write SQL queries to 1. List all the student details studying in fourth semester 'C' section. 2. Compute the total number of male and female students in each semester and in each section. 3. Create a view of Test1 marks of student USN '1BI15CS101' in all Courses. 4. Calculate the FinalIA (average of best two test marks) and update the corresponding table for all students. 5. Categorize students based on the following criterion: If FinalIA = 17 to 20 then CAT = 'Outstanding' If FinalIA = 12 to 16 then CAT = 'Average' If FinalIA< 12 then CAT = 'Weak' Give these details only for 8th semester A, B, and C section students. Reference: https://www.youtube.com/watch?v=horURQewW9c https://www.youtube.com/watch?v=P7-wKbKrAhk
5	Aim: Demonstrate the core concepts on table like nested and correlated nesting queries and also EXISTS and NOT EXISTS keywords. Program: Consider the schema for Company Database: **EMPLOYEE(SSN, Name, Address, Sex, Salary, SuperSSN, DNo)** **DEPARTMENT(DNo, DName, MgrSSN, MgrStartDate)** **DLOCATION(DNo,DLoc)** **PROJECT(PNo, PName, PLocation, DNo)** **WORKS_ON(SSN, PNo, Hours)** Write SQL queries to 1. Make a list of all project numbers for projects that involve an employee whose last name is 'Scott', either as a worker or as a manager of the department that controls the project. Show the resulting salaries if every employee working on the 'IoT' project is given a 10 percent raise. 3. Find the sum of the salaries of all employees of the 'Accounts' department, as well as the maximum salary, the minimum salary, and the average salary in this department 4. Retrieve the name of each employee who works on all the projects controlled by department number 5 (use NOT EXISTS operator).

	5. For each department that has more than five employees, retrieve the department number and the number of its employees who are making more than Rs.6,00,000.

	Reference: https://www.youtube.com/watch?v=Dk8f3ejqK
Pedagogy	For the above experiments the following pedagogy can be considered. Problem based learning, Active learning, MOOC, Chalk &Talk

PART B

Mini project: For any problem selected, make sure that the application should have five or more tables. Indicative areas include: Organization, health care, Ecommerce etc.

Course Outcomes:

At the end of the course the student will be able to:

CO 1. Create, Update and query on the database.

CO 2. Demonstrate the working of different concepts of DBMS

CO 3. Implement, analyze and evaluate the project developed for an application.

Assessment Details (both CIE and SEE)

The weightage of Continuous Internal Evaluation (CIE) is 50% and for Semester End Exam (SEE) is 50%. The minimum passing mark for the CIE is 40% of the maximum marks (20 marks). A student shall be deemed to have satisfied the academic requirements and earned the credits allotted to each course. The student has to secure not less than 35% (18 Marks out of 50) in the semester-end examination (SEE). The student has to secure a minimum of 40% (40 marks out of 100) in the sum total of the CIE (Continuous Internal Evaluation) and SEE (Semester End Examination) taken together.

Continuous Internal Evaluation (CIE):

CIE marks for the practical course is **50 Marks**.

The split-up of CIE marks for record/ journal and test are in the ratio **60:40**.

Each experiment to be evaluated for conduction with an observation sheet and record write-up. Rubrics for the evaluation of the journal/write-up for hardware/software experiments designed by the faculty who is handling the laboratory session and is made known to students at the beginning of the practical session.

Record should contain all the specified experiments in the syllabus and each experiment write-up will be evaluated for 10 marks.

Total marks scored by the students are scaled downed to 30 marks (60% of maximum marks).

Weightage to be given for neatness and submission of record/write-up on time.

Department shall conduct 02 tests for 100 marks, the first test shall be conducted after the 8th week of the semester and the second test shall be conducted after the 14th week of the semester.

In each test, test write-up, conduction of experiment, acceptable result, and procedural knowledge will carry a weightage of 60% and the rest 40% for viva-voce.

The suitable rubrics can be designed to evaluate each student's performance and learning ability. Rubrics suggested in Annexure-II of Regulation book

The average of 02 tests is scaled down to 20 marks (40% of the maximum marks).

The Sum of scaled-down marks scored in the report write-up/journal and average marks of two tests is the total CIE marks scored by the student.

Semester End Evaluation (SEE):

 SEE marks for the practical course is 50 Marks.
 SEE shall be conducted jointly by the two examiners of the same institute, examiners are appointed by the University
 All laboratory experiments are to be included for practical examination.
 (Rubrics) Breakup of marks and the instructions printed on the cover page of the answer script to be strictly adhered to by the examiners. **OR** based on the course requirement evaluation rubrics shall be decided jointly by examiners.
 Students can pick one question (experiment) from the questions lot prepared by the internal /external examiners jointly.
 Evaluation of test write-up/ conduction procedure and result/viva will be conducted jointly by examiners.
 General rubrics suggested for SEE are mentioned here, writeup-20%, Conduction procedure and result in -60%, Viva-voce 20% of maximum marks. SEE for practical shall be evaluated for 100 marks and scored marks shall be scaled down to 50 marks (however, based on course type, rubrics shall be decided by the examiners)
 Students can pick one experiment from the questions lot of PART A with an equal choice to all the students in a batch. For PART B, the project group (Maximum of 4 students per batch) should demonstrate the mini-project.
 Weightage of marks for PART A is 60% and for PART B is 40%. General rubrics suggested to be followed for part A and part B.
 Change of experiment is allowed only once and Marks allotted to the procedure part to be made zero (Not allowed for Part B).
 The duration of SEE is 03 hours

Rubrics suggested in Annexure-II of Regulation book

Textbooks:

1. Fundamentals of Database Systems, Ramez Elmasri and Shamkant B. Navathe, 7th Edition, 2017, Pearson.
2. Database management systems, Ramakrishnan, and Gehrke, 3rd Edition, 2014, McGraw Hill

Suggested Weblinks/ E Resource

https://www.tutorialspoint.com/sql/index.htm

How to Install MySql on Linux/Ubuntu:

MySQL is an open-source relational database management system that is based on SQL queries. Here, 'My" represents the name of the co-founder is Michael Widenius's daughter and "SQL" represents the Structured Query Language. This server is used for data operations like querying, filtering, sorting, grouping, modifying, and joining the tables present in the database. Before learning the queries, we are going to download and install MySQL on Linux. Some of the common features of MySQL are:

- It is easy to use and free of cost to download.
- It contains a solid data security layer to protect important data.
- It is based on client and server architecture.
- It supports multithreading which makes it more scalable.
- It is highly flexible and supported by multiple applications.
- The performance of MySQL is fast, efficient, and reliable.
- It is compatible with many operating systems like Windows, macOS, Linux, etc.

Installing MySQL on Linux

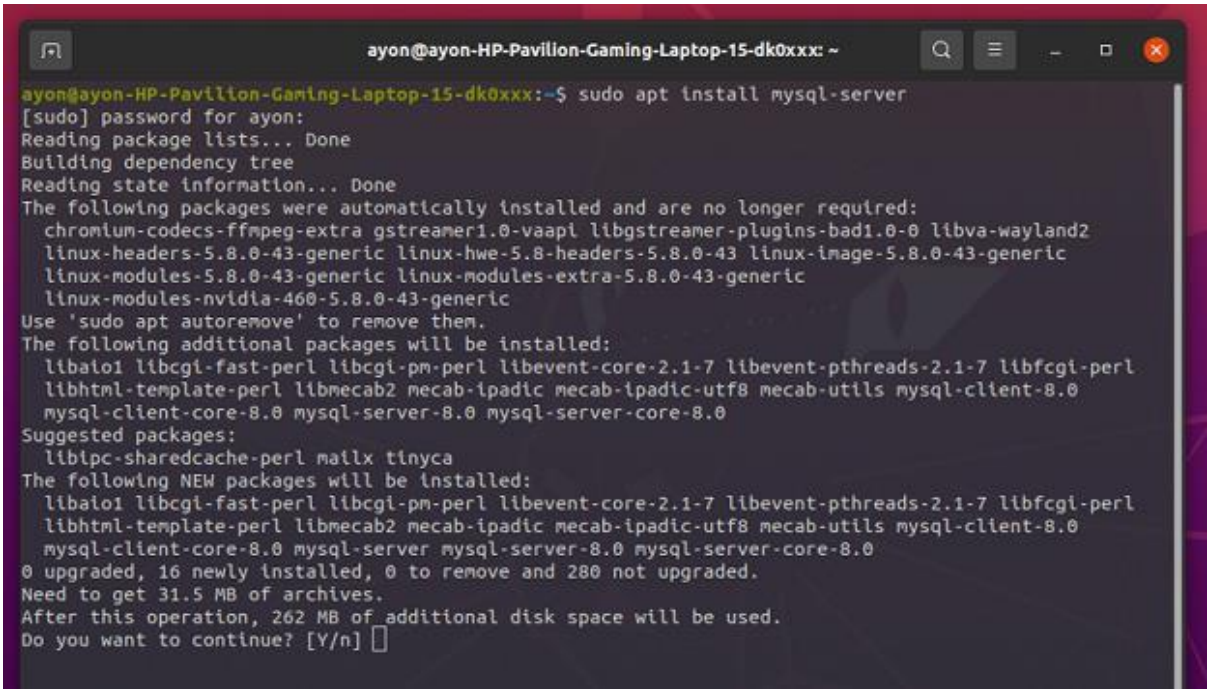
For almost every Linux system, the following commands are used to install MySQL:

Step 1: Go to the terminal using Ctrl+Alt+T. Now using the following command to install MySQL(copy and past it in terminal).

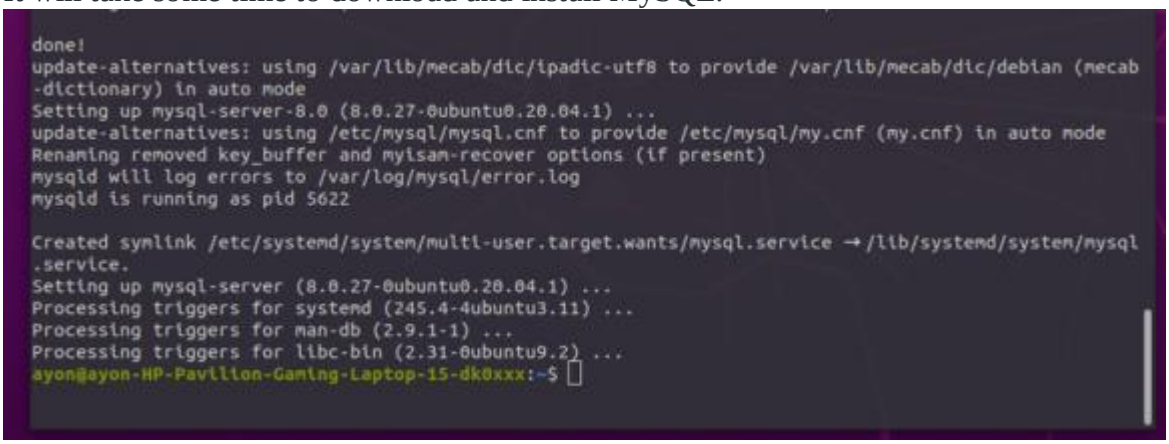
sudo apt install mysql-server

Then give your **password** and hit **ENTER**.

Step 2: Press **"y"** to continue.

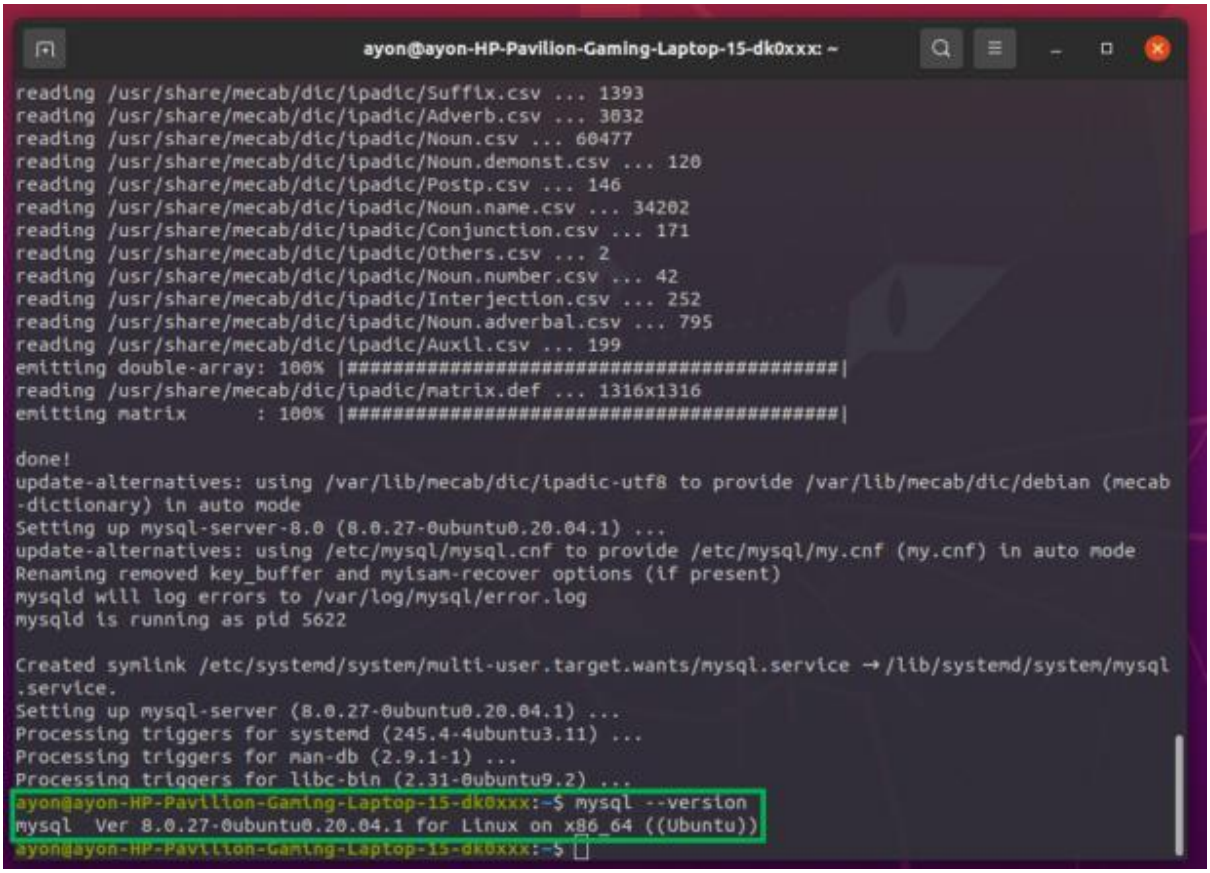


It will take some time to download and install MySQL.



Step 3: To verify the installation or to know the version enter the following commands in your Terminal.

mysql–version

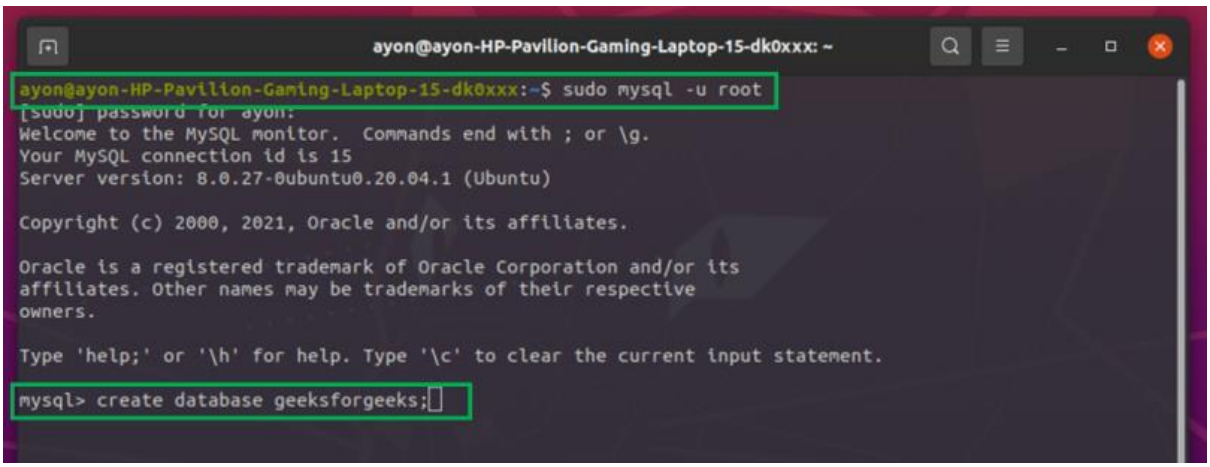


Step 4: To get started with MySQL go to the root directory.

Sudo mysql -u root

Let's create a database using the following command:

create database database_name;



show databases;

```
ayon@ayon-HP-Pavilion-Gaming-Laptop-15-dk0xxx: ~
[sudo] password for ayon:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 15
Server version: 8.0.27-0ubuntu0.20.04.1 (Ubuntu)

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database geeksforgeeks;
Query OK, 1 row affected (0.14 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| geeksforgeeks      |
| gfg                |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
6 rows in set (0.01 sec)

mysql>
```

How to Install MySql on Windows 11:

RecommendedHardware/SoftwareRequirements:

- Hardware Requirements:IntelBaseddesktopPCwithminimumof166MHZ or fasterprocessorwithatleast1GBRAMand500MBfree diskspace.

Download & Install MySQL on Windows 11

Overview

This tutorial outlines steps to download & install the MySQL database Community Server on Windows 11 operating system. Windows 11 is the latest operating system from Microsoft Corporation.

Environment

The environment used in this tutorial is as follows:

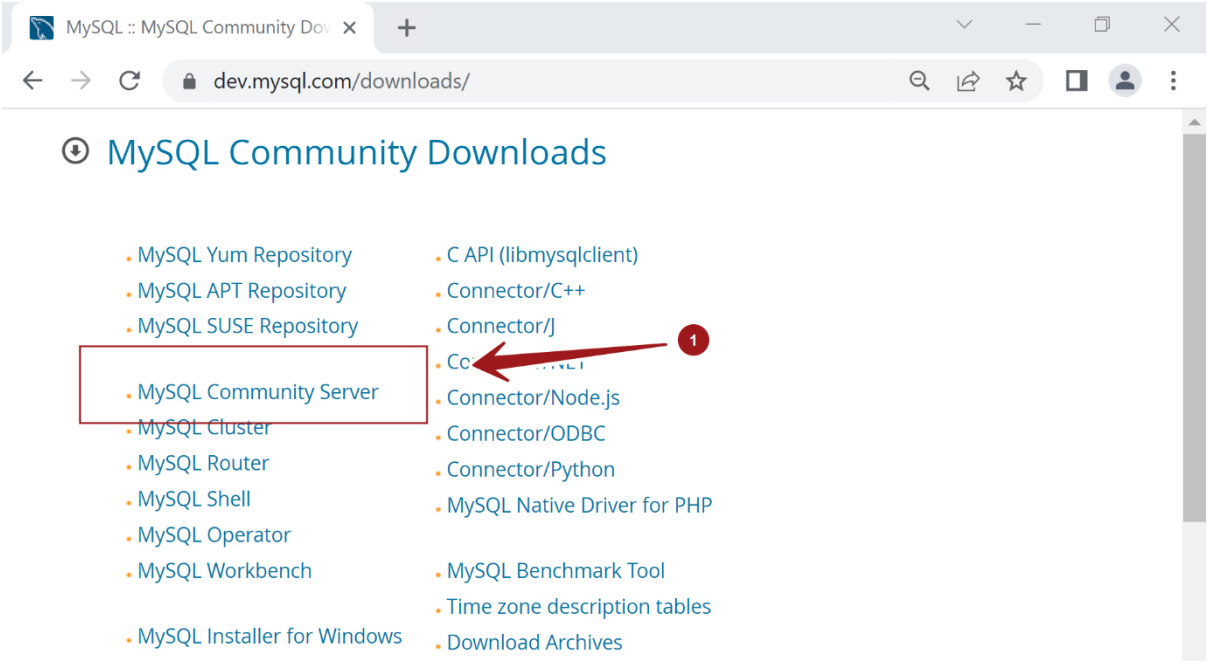
- Windows 11 Operating System
- Web Browser: Google Chrome Browser
- MySQL Server 8.x Community Server

Download

Open a web browser and navigate to the MySQL website download page:

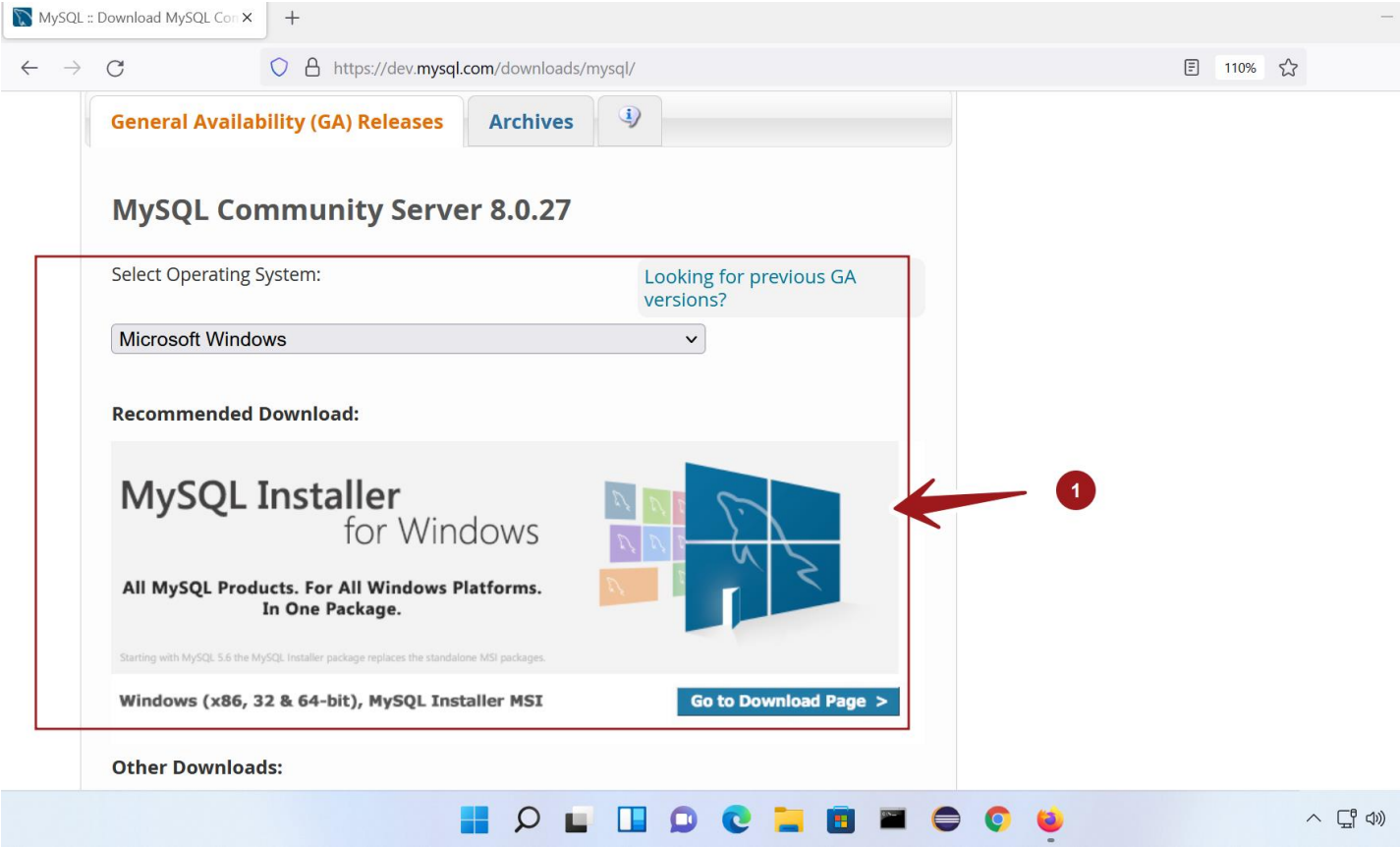
https://dev.mysql.com/downloads/

This will launch the MySQL Community downloads page. Click on the link: *MySQL Community Server*



Choose *Microsoft Windows* from the *Select Operating System* drop-down:

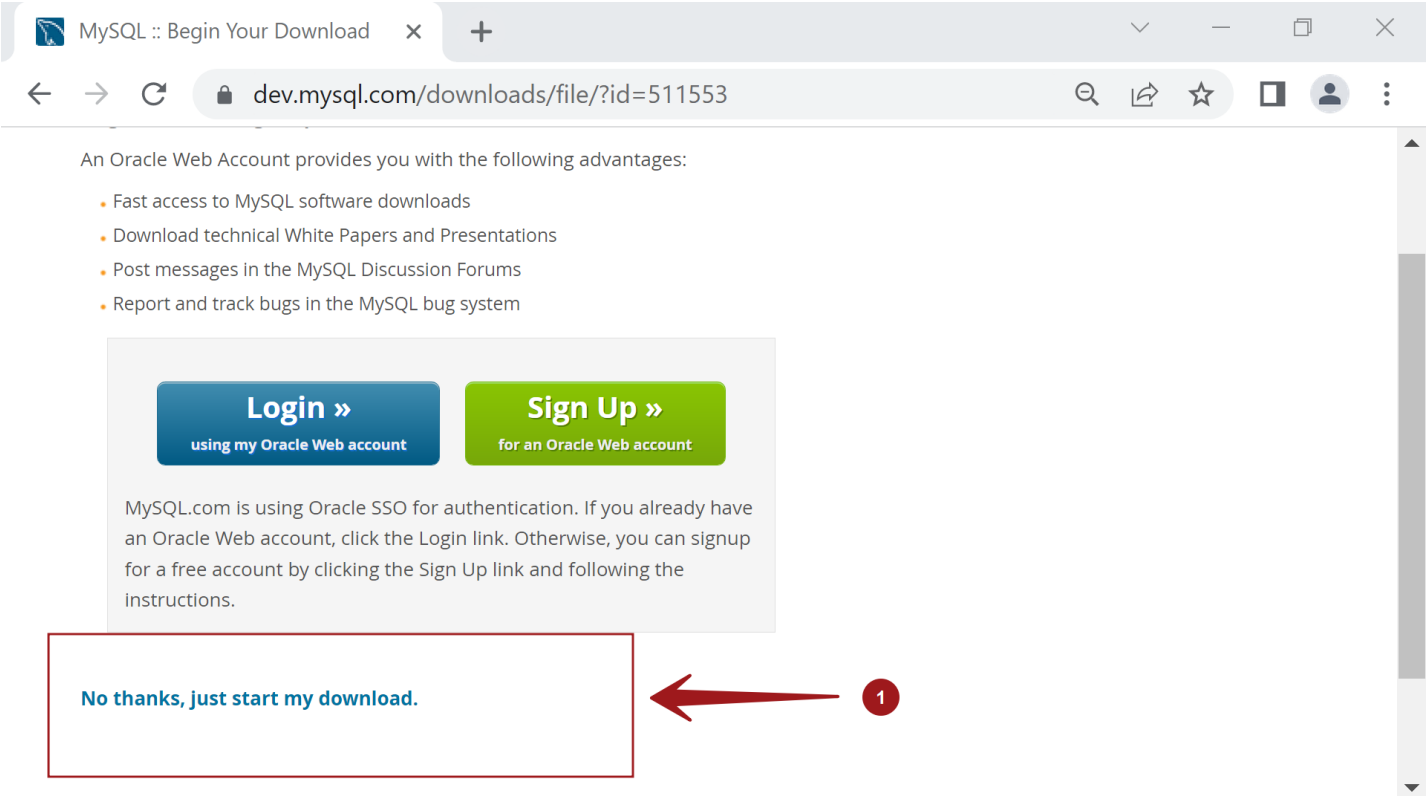
Click on the *MySQL Installer for Windows*. The installer is a single file with all the MySQL components bundled into a single package.



Click on the *Download* button and save the installer onto the Windows 11 computer.

Optional Step

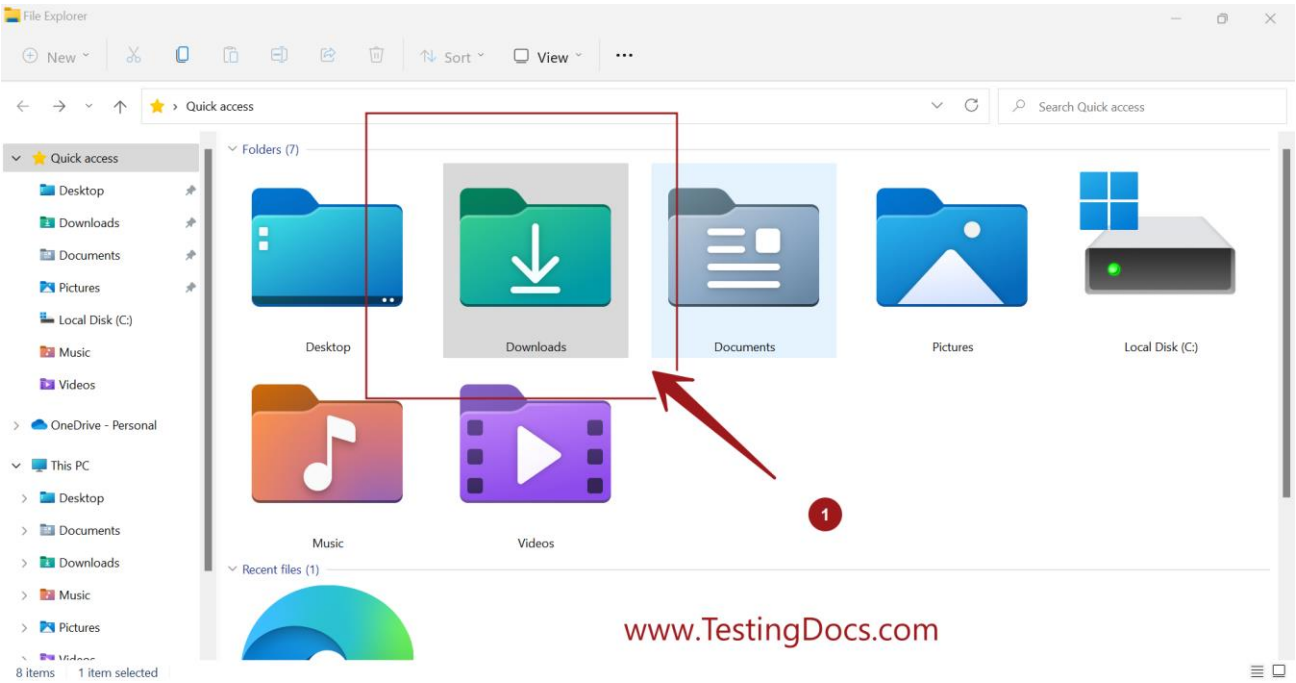
At this step, you can optionally Sign Up for an Oracle Web Account. The login or signup at this page is optional.



Click on the link '*No thanks, just start my download.*' link to continue with the download.

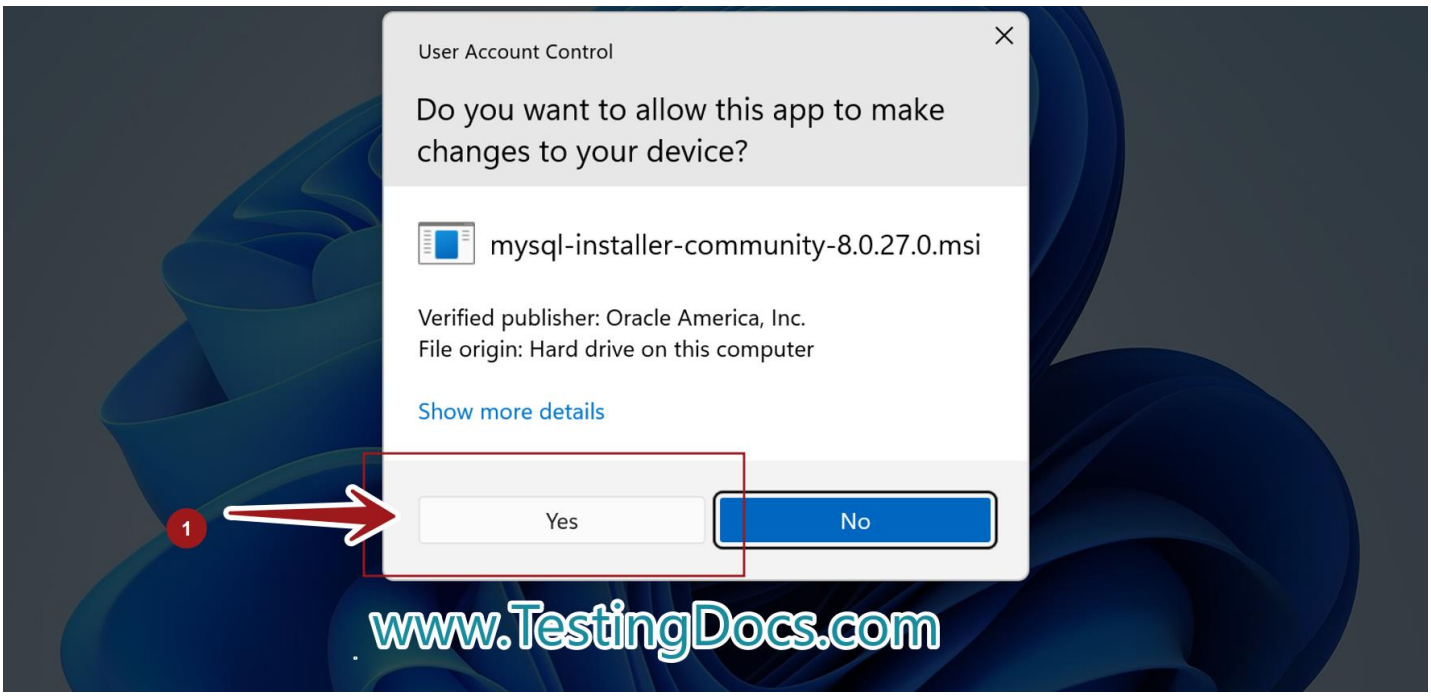
Install

Open File Explorer and locate the downloaded MySQL Installer file.



Double click on the MySQL Installer file to run it.

Click on the *Yes* button on the UAC elevation prompt. Verify the *Publisher* information in the UAC dialog prompt. MySQL server install needs administrator privileges on the machine. *UAC(User Account Control)* is a built-in Windows security feature to alert you if any program attempts to perform an administrative task. Tampered or malicious installer would not be signed or stomped with legitimate *Publisher information.*

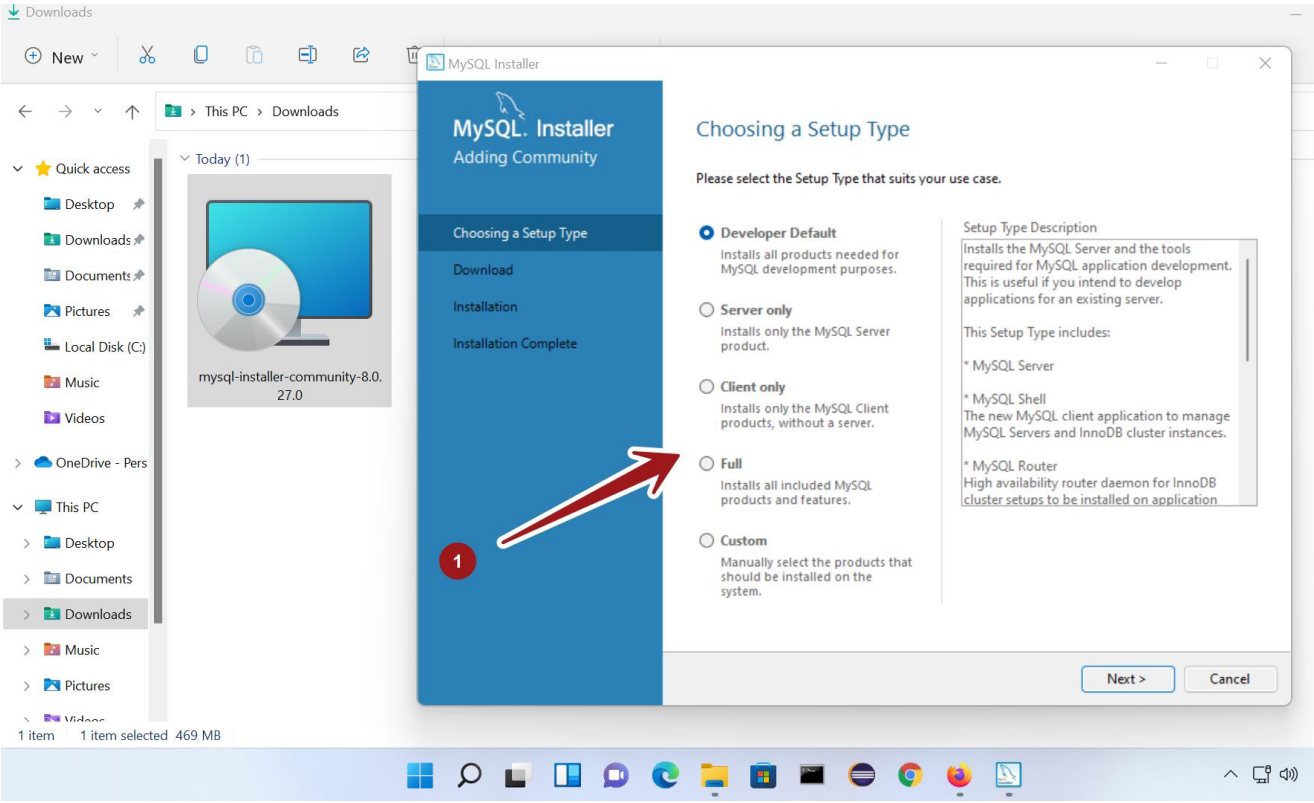


Setup Type

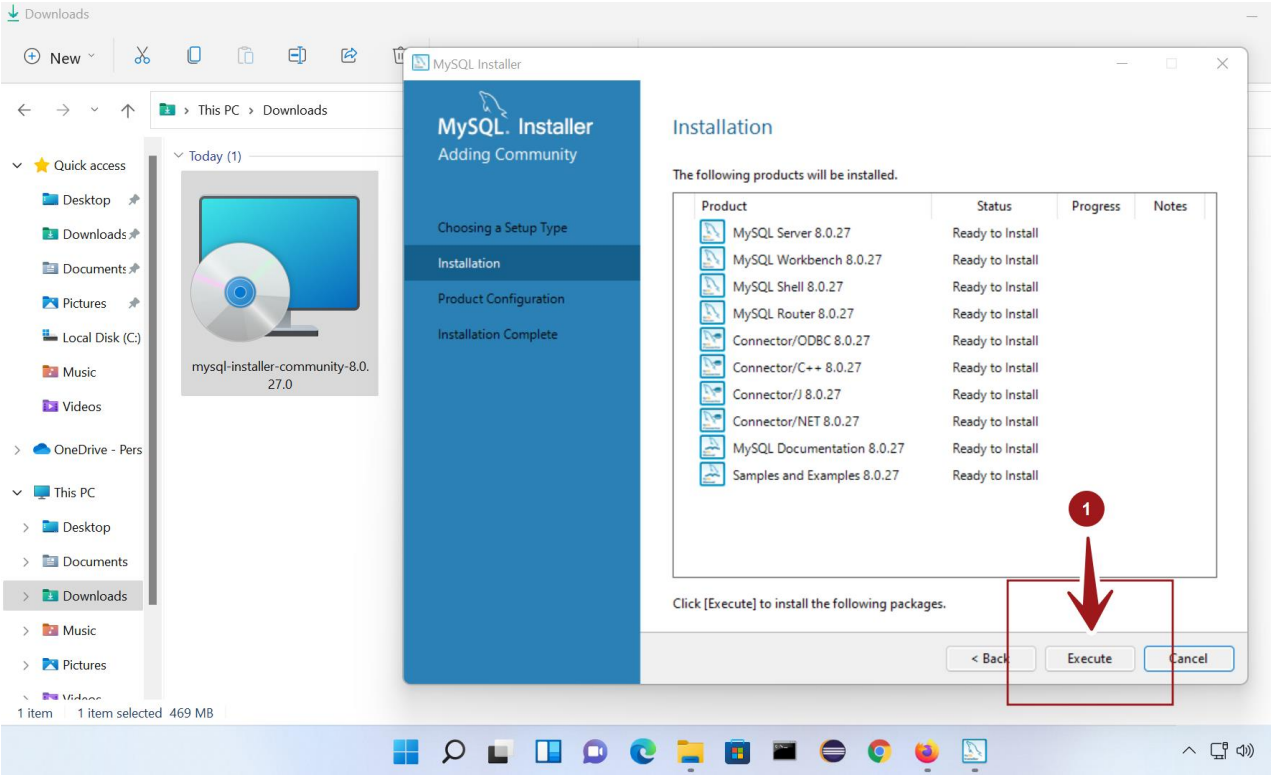
Choose the Setup type and click on the *Next* > button. MySQL supports many install types like:

- **Developer**
- **Server only**
- **Client only**
- **Full**
- **Custom**

For example, to install all the MySQL components select *Full* install type.



Click on the *Execute* button to install the MySQL components.



Configuration

Type and Networking

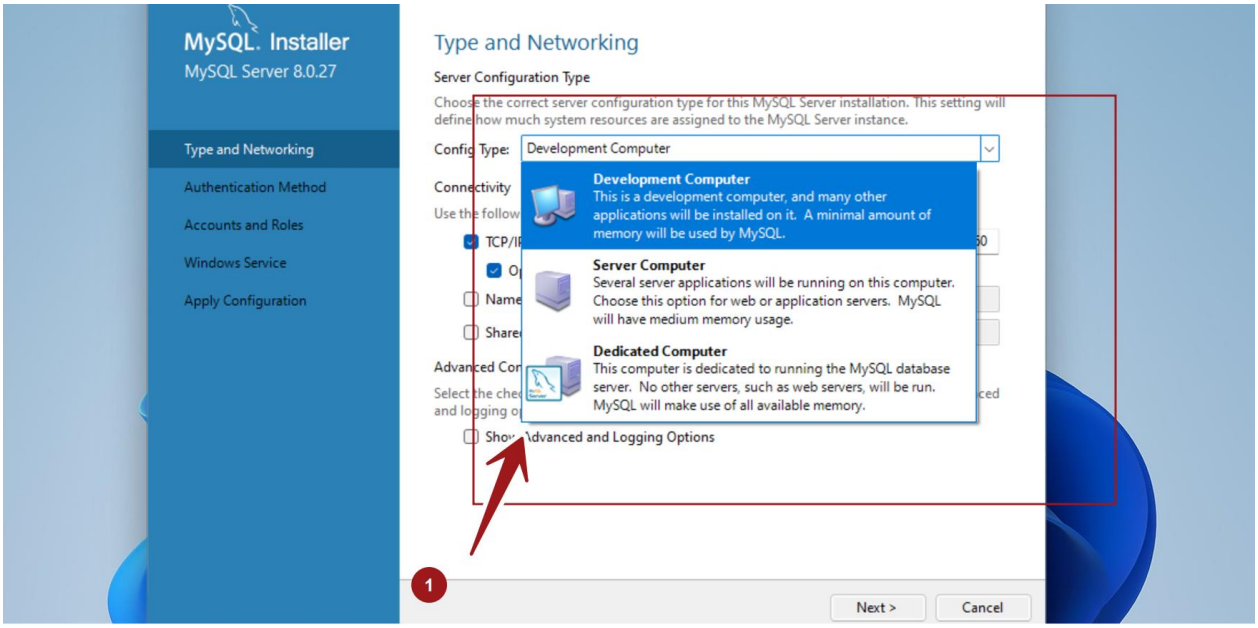
Choose the Config Type. We have three options

- **Development Computer**
- **Server Computer**
- **Dedicated Computer**

Development Computer: Choose this if several applications and servers run on the installed computer. MySQL will use a minimum memory footprint on this configuration.

Server Computer: Choose this configuration if other servers run on the computer.

Dedicated Computer: Choose this configuration if only the MySQL server runs on the computer. MySQL would use all the available memory.



TCP/IP Ports

We can configure TCP/IP ports for server communication. It's recommended to leave the default ports. Click on the *Next* > button.

Authentication Method

Use Strong Password Encryption for Authentication. MySQL 8 supports new authentication based on *SHA256* encryption.

Click on the *Next* > button.

Accounts and Roles

We can set the root password on this screen. Set a strong password and store this password in a secure place.

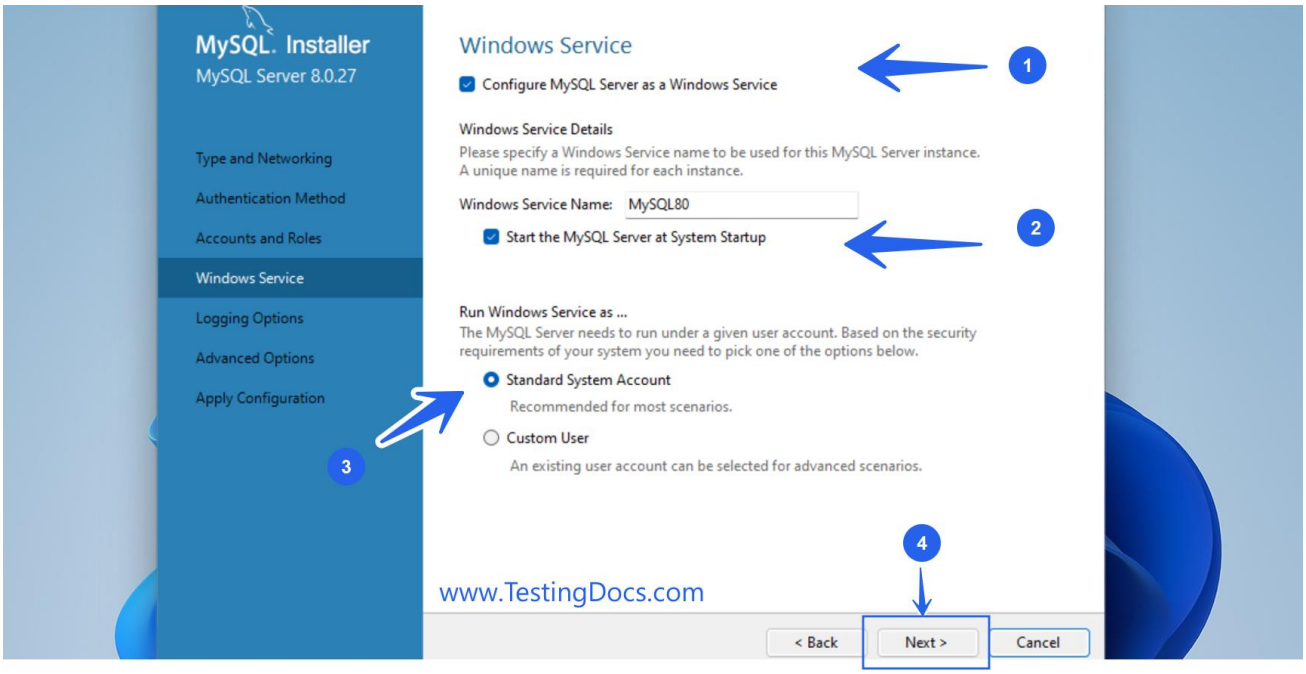
To add user accounts click on the *Add User* button.

The use of an anonymous user account is not recommended due to the lack of security.

Windows Service

Configure MySQL Server as a Windows service. To start the MySQL server at system startup

Check the option *Start the MySQL Server at System Startup*. This option will start the MYSQL Server Service during the Windows 11 boot/ restart process. We can also configure the user account for the MySQL Windows Service.



Apply Configuration

Click on the *Execute* button to apply the changes. Once complete click on the *Finish* button. Depending on the machine and network configuration the MySQL install might take a while to get complete.

That's it. We are done with the MySQL installation on Windows 11 operating system. MySQL Server would be started by default.

Start MySQL Service

Steps to start MySQL Server Service manually on Windows:

Start Mysql Service on Windows 11

Start MySQL Client

Connect MySQL client and connect to MySQL server to issue SQL commands:

Start Mysql Client on Windows 11

How to Login to Ubuntu/ Fedora and open terminal:

1. Login to Ubuntu using UN and Pwd
2. Open terminal using tab or right click and select terminal

How to use MySQL Command Line Client

1. Open Command Prompt.
2. Navigate to the bin folder. For example: cd C:\Program Files\MySQL\MySQL Server 8.0\bin.
3. Run the mysql -u root -p command.
4. Enter the password twice.

How to create and select a database:

In MySQL prompt type “CREATE DATABASE <database name>; (USN)

To view the created databases type ” SHOW DATABASES”

To start SQL query, in mysql prompt type “ USE <databasename>;

LIBRARY DATABASE

Consider the following schema for a Library Database:

BOOK (BOOK_ID, TITLE, PUBLISHER_NAME,PUB_YEAR)
BOOKAUTHORS (BOOK_ID, AUTHOR_NAME)
PUBLISHER (NAME, ADDRESS,PHONE)
BOOK_COPIES (BOOK_ID, PROGRAMME _ID,NO_OF_COPIES)
BOOK_LENDING (BOOK_ID, PROGRAMME _ID, CARD_NO, DATE_OUT, DUE_DATE)
LIBRARY_PROGRAMME (PROGRAMME_ID, PROGRAMME_NAME, ADDRESS)

Write SQL queries to

1. Retrieve details of all books in the library – id, title, name of publisher, authors, number of copies in each branch,etc.
2. Get the particulars of borrowers who have borrowed more than 3 books, but from Jan 2017 to Jun2017.
3. Delete a book in BOOK table. Update the contents of other tables to reflect this data manipulation operation.
4. Partition the BOOK table based on year of publication. Demonstrate its working with a simple query.
5. Create a view of all books and its number of copies that are currently available in the Library.

CREATE TABLE COMMANDS

1.
```
CREATE TABLE PUBLISHER
(
        NAME VARCHAR2(15),
        ADDRESS VARCHAR2(25),
        PHONE NUMBER(10),
CONSTRAINT PK_PN PRIMARY KEY(NAME)
);
```

2.
```
CREATE TABLE LIBRARY_PROGRAMME
(
        PROGRAMME _ID NUMBER(5),
          PROGRAMME _NAME VARCHAR(15),
        ADDRESS VARCHAR2(15),
          CONSTRAINT PK_ID PRIMARY KEY(PROGRAMME_ID)
);
```

3.
```
CREATE TABLE BOOK
(
BOOK_IDNUMBER(5),
```

```
        TITLE VARCHAR2(25),
        PUBLISHER_NAME VARCHAR2(15),
        PUB_YEAR NUMBER(4),
        CONSTRAINT PK_BID PRIMARY KEY(BOOK_ID),
        CONSTRAINT FK_N FOREIGN KEY (PUBLISHER_NAME) REFERENCES
        PUBLISHER (NAME) ON DELETE CASCADE
);
```

4. CREATE TABLE BOOK_AUTHORS
```
(
        BOOK_ID NUMBER(5),
        AUTHOR_NAME VARCHAR2(25),
        CONSTRAINT FK_B FOREIGN KEY(BOOK_ID) REFERENCES BOOK(BOOK_ID)
        ON DELETECASCADE
);
```

5. CREATE TABLEBOOK_COPIES
```
(       BOOK_IDNUMBER(5),
        PROGRAMME _ID NUMBER (5),
        NO_OF_COPIES NUMBER (2),
        CONSTRAINT CPK_BBI PRIMARY KEY(BOOK_ID,PROGRAMME_ID),
        CONSTRAINT FK_BI FOREIGN KEY(BOOK_ID) REFERENCES BOOK(BOOK_ID)
        ON DELETE CASCADE,
        CONSTRAINT FK_I FOREIGN KEY(PROGRAMME _ID) REFERENCES
        LIBRARY_ PROGRAMME (PROGRAMME _ID) ON DELETE CASCADE
);
```

6. CREATE TABLEBOOK_LENDING
```
(BOOK_ID NUMBER(5),
        PROGRAMME_ID
        NUMBER(5), CARD_NO
        NUMBER(3), DATE_OUT
        DATE, DUE_DATE DATE,
    CONSTRAINT CPK_BBC PRIMARY KEY(BOOK_ID,PROGRAMME_ID,CARD_NO),
        CONSTRAINTFK_A FOREIGN KEY(BOOK_ID) REFERENCES BOOK(BOOK_ID)
        ON DELETE CASCADE,
        CONSTRAINT FK_C FOREIGN KEY (PROGRAMME_ID) REFERENCES
        LIBRARY_ PROGRAMME (PROGRAMME _ID) ON DELETE CASCADE
);
```

INSERTION COMMANDS

```
SQL> INSERT INTO PUBLISHER VALUES('&NAME','&ADDRESS',&PHONE);
Enter value for name: PHI
Enter value for address: PUNE,INDIA
Enter value for phone: 9080706050
```

```
old 1: INSERT INTO PUBLISHER VALUES('&NAME','&ADDRESS',&PHONE)

new 1: INSERT INTO PUBLISHER VALUES('PHI','PUNE,INDIA',908070)
1 row created.

SQL> SELECT * FROM PUBLISHER;

NAME                    ADDRESS                 PHONE
------------------------------------------------------------------------------
PHI                     PUNE,INDIA              9080706050
PEARSON                 MUMBAI,INDIA            8080707060
MCGRAWHILL              HOUSTIN,USA             1020707070
WILEY                   CALIFORNIA,USA          1108080808
SSI                     FLORIDA                 1208080909
SP                      BANGALORE,INDIA         9090909080
6 rows selected.


SQL> INSERT INTO LIBRARY_ PROGRAMME
VALUES(&PROGRAMME_ID,'&PROGRAMME_NAME','&A
DDRESS');
Enter value for PROGRAMME _id: 1000
Enter value for PROGRAMME _name: SMVIT
Enter value for address: HUNASEMARANAHALLI
old 1: INSERT INTO LIBRARY_ PROGRAMME VALUES(&PROGRAMME _ID,'& PROGRAMME
_NAME','&ADDRESS')
new 1: INSERT INTO LIBRARY_ PROGRAMME
VALUES(1000,'SMVIT','HUNASEMARANAHALLI')
1 row created.

SQL> SELECT * FROM LIBRARY_ PROGRAMME;

 PROGRAMME_ID            PROGRAMME_NAME                ADDRESS
------------------------------------------------------------------------------
      1000                    MVIT          HUNASEMARANAHALLI
      2000                    SVIT          DODDABALLAPUR
      3000                    BMSIT         AVANAHALLI
      4000                    SVCE          VIDYANAGAR
      5000                    MSCE          CHIKKAJALA
      6000                    NMIT          YELAHANKA

6 rows selected.


SQL> INSERT INTO BOOK
VALUES(&BOOK_ID,'&TITLE','&PUBLISHER_NAME',&PUB_YEAR);
```

```
Enter value for book_id: 1111
Enter value for title: FUNNDAMENTALS OF DATABASE
Enter value for publisher_name: PHI
Enter value for pub_year: 2009
old 1: INSERT INTO BOOK
VALUES(&BOOK_ID,'&TITLE','&PUBLISHER_NAME',&PUB_YEAR)
new 1: INSERT INTO BOOK VALUES(1111,'FUNNDAMENTALS OF DATABASE','PHI',2009)
1 row created.
SQL> SELECT * FROM BOOK;

  BOOK_ID  TITLE                          PUBLISHER_NAME          PUB_YEAR
----------- ------------------------------ ------------------------------ ----------
     1111FUNNDAMENTALS OF DATABASE         PHI                          2009
     2222 BASICS OF LOGIC DESIGN           PEARSON                      2009
     3333DATASTRUCTURES                    MCGRAWHILL                   2015
     4444 ARTIFICIAL INTELLIGENCE          WILEY                        2017
     5555PROGRAMMING SKILLS                SSI                          2014
     6666 DESIGN OF ALGORITHMS             SP                           2013
6 rows selected.

SQL> INSERT INTO BOOK_AUTHORS VALUES(&BOOK_ID,'&AUTHOR_NAME');
Enter value for book_id: 1111
Enter value for author_name: NAVATHE
old 1: INSERT INTO BOOK_AUTHORS VALUES(&BOOK_ID,'&AUTHOR_NAME')
new 1: INSERT INTO BOOK_AUTHORS VALUES(1111,'NAVATHE')
1 row created.

SQL> SELECT * FROM BOOK_AUTHORS;

  BOOK_ID AUTHOR_NAME
----------- ------------------------------
     1111 NAVATHE
     2222 GODSE
     3333 SAHANI
     4444 RITCHIE KNIGHT
     5555 BALAGURUSWAMY
     6666 COREMEN

6 rows selected.

SQL> INSERT INTO BOOK_COPIES VALUES(&BOOK_ID,&PROGRAMME_ID,&NO_OF_COPIES);
Enter value for book_id: 1111
Enter value for
PROGRAMME_ID: 1000 Enter
value for no_of_copies: 10
old 1: INSERT INTO BOOK_COPIES
```

```
VALUES(&BOOK_ID,&PROGRAMME_ID,&NO_OF_COPIES) new 1: INSERT INTO
BOOK_COPIES VALUES(1111,1000,10)

1 row created.

SQL> SELECT * FROM BOOK_COPIES;

  BOOK_ID PROGRAMME_ID NO_OF_COPIES
---------- ------------ -------------
     1111       1000          10
     2222       2000           5
     3333       3000           7
     4444       4000           9
     5555       5000           6
     6666       6000          12
     2222       1000          15

7 rows selected.

SQL> INSERT INTO BOOK_LENDING
VALUES(&BOOK_ID,&PROGRAMME_ID,&CARD_NO,'&DATE_OUT','&DUE_DATE');
Enter value for book_id: 1111
Enter value for
PROGRAMME_ID: 1000 Enter
value for card_no: 10
Enter value for date_out: 15-FEB-17
Enter value for due_date: 15-JUN-17
old 1: INSERT INTO BOOK_LENDING
VALUES(&BOOK_ID,&PROGRAMME_ID,&CARD_NO,'&DATE_OUT','&DUE_DATE')
new 1: INSERT INTO BOOK_LENDING VALUES(1111,1000,10,'15-FEB-17','15-JUN-17')
1 row created.



SQL> SELECT * FROM BOOK_LENDING;

  BOOK_IDPROGRAMME_ID      CARD_NO DATE_OUTDUE_DATE
---------- ------------ ---------- --------- ---------
     1111         1000         10 15-FEB-17 15-JUN-17
     2222         2000         10 10-MAR-17 15-AUG-17
     3333         3000         10 15-APR-17 15-SEP-17
     4444         4000         10 10-JUN-17 15-NOV-17
     5555         5000         20 15-FEB-17 15-JUN-17
     6666         6000         30 10-MAR-17 15-AUG-17
     5555         5000         10 15-JAN-16 15-JUN-16
7 rows selected.
```

QUERIES

1.Retrieve details of all books in the library i.e ID ,Title, name of publisher, authors, no. of copies etc in each branch.

```
SELECTC.PROGRAMME_ID,L.PROGRAMME_NAME,B.BOOK_ID,B.TITLE,B.PUBLISHER_NAME,B.PUB_
YEAR,A.AUTHOR_NAME,C.NO_OF_COPIES
  FROM BOOK B,BOOK_AUTHORS A,LIBRARY_BRANCH L,BOOK_COPIES C
WHERE B.BOOK_ID=A.BOOK_ID AND
 B.BOOK_ID=C.BOOK_ID AND
L.PROGRAMME_ID=C.PROGRAMM
E_ID AND
(C.PROGRAMME_ID,C.BOOK_ID) IN
(SELECT
PROGRAMME_ID,BOOK_ID FROM
BOOK_COPIES
 GROUP BY PROGRAMME_ID,BOOK_ID);
```

Output:

PROGRAMME_ID	PROGRAMME_NAME	BOOK_ID	TITLE	PUBLISHER_NAME
5000	MSCE	5555	PROGRAMMINGSKILLS	SSI
1000	SMVIT	1111	FUNNDAMENTALS OF DATABASE	PHI
1000	SMVIT	2222	BASICS OFLOGICDESIGN	PEARSON
2000	SVIT	2222	BASICS OFLOGICDESIGN	PEARSON
3000	BMSIT	3333	DATASTRUCTURES	MCGRAWHILL
4000	SVCE	4444	ARTIFICIALINTELLIGENCE	WILEY
6000	NMIT	6666	DESIGNOFALGORITHMS	SP

PUB_YEAR	AUTHOR_NAME	NO_OF_COPIES
2014	BALAGURUSWAMY	6
2009	NAVATHE	10
2009	GODSE	15
2009	GODSE	5
2015	SAHANI	7
2017	RITCHIE KNIGHT	9
2013	COREMEN	12

7 rows selected.

2. Get the particulars of borrowers who have borrowed more than 3 books but from Jan 2017 to Jun 2017

```
SQL> SELECT * FROM BOOK_LENDING
  WHERE DATE_OUT BETWEEN '01-JAN-17' AND '30-JUN-17' AND CARD_NO
IN
 (SELECT CARD_NO
FROM BOOK_LENDING
GROUP BY CARD_NO
HAVING COUNT(CARD_NO)>3);
```

Output:

```
   BOOK_IDPROGRAMME_ID      CARD_NO DATE_OUTDUE_DATE
----------- ----------- ---------- --------- ---------
      4444        4000         10 10-JUN-17 15-NOV-17
      3333        3000         10 15-APR-17 15-SEP-17
      2222        2000         10 10-MAR-17 15-AUG-17
      1111        1000         10 15-FEB-17 15-JUN-17
```

3.Delete a book in book table.Update the contents of other tables to reflect this data manipulation operation.

```
 DELETE FROM BOOK WHERE BOOK_ID=&BOOK_ID;
Enter value for book_id: 5555
old 1: DELETE FROM BOOK WHERE BOOK_ID=&BOOK_ID
new 1: DELETE FROM BOOK WHERE BOOK_ID=5555
1 row deleted.
SQL> SELECT * FROM BOOK;

   BOOK_IDTITLE                        PUBLISHER_NAME          PUB_YEAR
---------- ------------------------------ ------------------------------ ----------
      1111 FUNNDAMENTALS OF DATABASE      PHI                     2009
      2222 BASICS OF LOGIC DESIGN         PEARSON                 2009
      3333 DATASTRUCTURES                 MCGRAWHILL              2015
      4444 ARTIFICIAL INTELLIGENCE        WILEY                   2017
      6666 DESIGN OF ALGORITHMS           SP                      2013
```

```
   BOOK_ID PROGRAMME_ID NO_OF_COPIES
----------- ---------- -------------
     1111       1000          10
     2222       2000           5
     3333       3000           7
     4444       4000           9
     6666       6000          12
     2222       1000          15

6 rows selected.

SQL> SELECT * FROM BOOK_AUTHORS;

   BOOK_ID AUTHOR_NAME
---------- ------------------------------
     1111 NAVATHE
     2222GODSE
     3333SAHANI
     4444RITCHIE KNIGHT
     6666COREMEN

SQL> SELECT * FROM BOOK_LENDING;
   BOOK_ID PROGRAMME_ID                                CARD_NO  DATE_OUT         DUE_DATE
----------- ------------ ------------ ------------ ------------
     1111       1000          10          15-FEB-17     15-JUN-17
     2222       2000          10          10-MAR-17     15-AUG-17
     3333       3000          10          15-APR-17     15-SEP-17
     4444       4000          10          10-JUN-17     15-NOV-17
     6666       6000          30          10-MAR-17     15-AUG-17
```

4. Partition the book table based on year of publication.Demostrate its working with a simplequery.

```
SQL> CONNECT SYSTEM/manjunath;
Connected.
SQL> GRANT CREATE VIEW TO B2;
Grant succeeded.
SQL> CONNECT B2/B2;
Connected.
SQL> CREATE VIEW YEAR AS SELECT PUB_YEAR FROM BOOK;
View created.
```

Output:

```
SQL> SELECT * FROM YEAR;

  PUB_YEAR
-----------
      2009
      2009
      2015
      2017
      2013
```

5. Create a view all books and its no. of copies that are currently available in thelibrary.

```
 CREATE VIEW ALL_BOOK AS
 SELECT
    B.BOOK_ID,B.TITLE,C.NO_OF_COPIES,L.PROGRAMME_NAME
    FROM BOOK B,BOOK_COPIES C,LIBRARY_BRANCH L
    WHERE B.BOOK_ID=C.BOOK_ID
    AND
    L.PROGRAMME_ID=C.PROGRAMME
    _ID;

View created.
```

Output:

```
SQL> SELECT * FROM ALL_BOOK;

BOOK_ID TITLE                          NO_OF_COPIESPROGRAMME_NAME
-------.------------------------------.----------------------
1111 FUNNDAMENTALS OFDATABASE          10          SMVIT
2222 BASICS OF LOGIC DESIGN             5          SVIT
2222 BASICS OF LOGIC DESIGN            15          SMVIT
3333 DATASTRUCTURES                     7          BMSIT
4444 ARTIFICIAL INTELLIGENCE            9          SVCE
6666 DESIGN OF ALGORITHMS              12          NMIT

6 rows selected.
```

ORDERS DATABASE

Consider the following schema for Order Database:

SALESMAN (SALESMAN_ID, NAME, CITY, COMISSION)
CUSTOMER(CUSTOMER_ID,CUST_NAME,CITY,GRADE,SALESMAN_ID)
ORDERS(ORD_NO,PURCHASE_AMT,ORD_DATE,CUSTOMER_ID,SALESMAN_ID)

Write SQL queries to

1. Count the customers with grades above Bangalore'saverage.
2. Find the name and numbers of all salesman who had more than onecustomer.
3. List all the salesman and indicate those who have and don't have customers in their cities (Use UNION operation.)
4. Create a view that finds the salesman who has the customer with the highest order of aday.
5. Demonstrate the DELETE operation by removing salesman with id 1000. All his orders must also bedeleted.

1. CREATE TABLESALESMAN

```
(       SALESMAN_IDNUMBER(4),
        NAME VARCHAR(15),
        CITY VARCHAR(15),
        COMISSION NUMBER(7,2),
        CONSTRAINT PK_A PRIMARY KEY(SALESMAN_ID)
  );
```

Table created.

```
DESC SALESMAN;
Name                    Null?       Type
----------------- -     -------- -  ---------------------------
SALESMAN_ID             NOT NULL    NUMBER(4)
NAME                                VARCHAR2(15)
CITY                                VARCHAR2(15)
COMISSION                           NUMBER(7,2)
```

```
CREATE TABLE CUSTOMER
(       CUSTOMER_ID NUMBER(2),
        CUST_NAMEVARCHAR(15),
  CITY VARCHAR(15),
        GRADE NUMBER(3),
        SALESMAN_ID NUMBER(4),
        CONSTRAINT PK_B PRIMARY KEY(CUSTOMER_ID),
        CONSTRAINT FK_D FOREIGN KEY(SALESMAN_ID)REFERENCES
        SALESMAN(SALESMAN_ID)ONDELETE     SETNULL
);
```

Table created.

```
DESC CUSTOMER;
Name                    Null?       Type
----------------------  ---------   ------------------------------------
CUSTOMER_ID             NOTNULL     NUMBER(2)
CUST_NAME                           VARCHAR2(15)
CITY                                VARCHAR2(15)
GRADE                               NUMBER(3)
SALESMAN_ID                         NUMBER(4)
```

```
2.CREATE TABLEORDERS
(       ORD_NO NUMBER(4),
        PURCHASE_AMT NUMBER(10,2),
        ORD_DATE DATE,
        CUSTOMER_ID NUMBER(2),
        SALESMAN_IDNUMBER(4),
        CONSTRAINT PK_E PRIMARY KEY(ORD_NO),
        CONSTRAINT FK_G FOREIGN KEY(CUSTOMER_ID) REFERENCES
        CUSTOMER(CUSTOMER_ID) ON DELETE SET NULL,
        CONSTRAINT FK_H FOREIGN KEY(SALESMAN_ID) REFERENCES
        SALESMAN(SALESMAN_ID) ON DELETE SETNULL
);
```

Table created.

```
SQL> DESC ORDERS;
Name                    Null?       Type
                        ----------- ------------------------------
ORD_NO                  NOT NULL    NUMBER(4)
PURCHASE_AMT                        NUMBER(10,2)
ORD_DATE                            DATE
CUSTOMER_ID                         NUMBER(2)
SALESMAN_ID                         NUMBER(4)
```

Insert commands:

```
INSERT INTO SALESMAN VALUES(&SALESMAN_ID,'&NAME','&CITY',&COMISSION);

Enter value for salesman_id: 1000
Enter value for name: RAMA
Enter value for city: BANGALORE
Enter value for comission: 10000.75
old 1: INSERT INTO SALESMAN VALUES(&SALESMAN_ID,'&NAME','&CITY',&COMISSION)
new 1: INSERT INTO SALESMAN VALUES(1000,'RAMA','BANGALORE',10000.75)
1 row created.

SELECT * FROM SALESMAN;

SALESMAN_ID   NAME        CITY        COMISSION
-----------   ----------  ----------  ----------
       1000   RAMA        BANGALORE   10000.75
       2000   KRISHNA     MATHURA     20000.25
       3000   SHIVA       AMARNATH    30000.3
       4000   GOVINDA     TIRUPATHI   4000.4
       5000   NARAYANA    VELLORE     50000.5

INSERT INTO CUSTOMER
VALUES(&CUSTOMER_ID,'&CUST_NAME','&CITY',&GRADE,&SALESMAN_ID);
Enter value for customer_id: 10
Enter value for cust_name:GANESH
Enter value for city: BANGALORE
Enter value for grade: 100
Enter value for salesman_id: 1000
old 1: INSERT INTO CUSTOMER
VALUES(&CUSTOMER_ID,'&CUST_NAME','&CITY',&GRADE,&SALESMAN_ID)
new 1: INSERT INTO CUSTOMER VALUES(10,'GANESH','BANGALORE',100,1000)
1 row created.

SELECT * FROM CUSTOMER;

CUSTOMER_ID   CUST_NAME   CITY        GRADE   SALESMAN_ID
-----------   ----------  ----------  ------  -----------
         10   GANESH      BANGALORE   100     1000
         20   SUDHEEP     BANGALORE   200     2000
         30   PRABHAS     HYDERABAD   300     3000
         40   ARVIND      CHENNAI     400     4000
         50   DARSHAN     BANGALORE   500     2000
         60   YASH        BANGALORE   600     1000
6 rows selected.
```

```
INSERT INTO ORDERS
VALUES(&ORD_NO,&PURCHASE_AMT,'&ORD_DATE',&CUSTOMER_ID,&SALESMAN_ID);

Enter value for ord_no: 1111
Enter value for purchase_amt: 100000.00
Enter value for ord_date: 01-JAN-17
Enter value for customer_id: 10
Enter value for salesman_id: 2000
old 1: INSERT INTO ORDERS
VALUES(&ORD_NO,&PURCHASE_AMT,'&ORD_DATE',&CUSTOMER_ID,&SALESMAN_ID)
new 1: INSERT INTO ORDERS VALUES(1111,100000.00,'01-JAN-17',10,2000)
1 row created.


 SELECT * FROM ORDERS;

   ORD_NO     PURCHASE_AMT      ORD_DATE CUSTOMER_IDSALESMAN_ID
----------- --------------- --------- ---------- --------------
     1111      100000          01-JAN-17        10              2000
     2222      200000          21-FEB-17        20              3000
     3333      300000          15-MAR-17        30              4000
     4444      400000          18-APR-17        40              5000
     5555      500000          12-MAY-17        10              1000
     6666      600000          12-MAY-17        10              1000
```

QUERIES:

1. Count the customers with grades above Banglore'saverage

```
   SELECT COUNT(CUSTOMER_ID)
    FROM CUSTOMER
    WHERE GRADE>( SELECT AVG(GRADE)
                  FROM CUSTOMER
                   WHERE CITY='BANGALORE' );
```

Output:

```
COUNT(CUSTOMER_ID)
 ----------
          3
```

2. Find the names and numbers of all salesman who had more than onecustomer.

```
 SELECT S.NAME,S.SALESMAN_ID
 FROM SALESMAN S,CUSTOMER C
 WHERE S.SALESMAN_ID=C.SALESMAN_ID
 GROUP BY S.NAME,S.SALESMAN_ID
 HAVING COUNT(C.CUSTOMER_ID)>1;
```

Output:

```
NAME          SALESMAN_ID
--------------- -------------
RAMA              1000
KRISHNA           2000
```

3. List all salesman and indicate those who have and don't have customers in their cities. Use union operation.

```
(SELECT S.SALESMAN_ID,S.NAME,C.CUST_NAME
  FROM SALESMAN S,CUSTOMER C
  WHERE S.CITY=C.CITY AND S.SALESMAN_ID=C.SALESMAN_ID)
  UNION
  (SELECT S1.SALESMAN_ID,S1.NAME,'NO CUSTOMER'
  FROM SALESMAN S1,CUSTOMER C1
  WHERE S1.CITY!=C1.CITY AND S1.SALESMAN_ID=C1.SALESMAN_ID );
```

Output:

```
SALESMAN_ID     NAME        CUST_NAME
-----------     ----------- -----------
     1000          RAMA       GANESH
     1000          RAMA       YASH
     2000         KRISHNA     NO CUSTOMER
     3000          SHIVA      NO CUSTOMER
     4000         GOVINDA     NO CUSTOMER
```

4. Create a view that finds the salesman who have the customer with the highest order of aday.

```
CONNECT SYSTEM/MANJUNATH;
Connected.
 GRANT CREATE VIEW TO B2;
Grant succeeded.
CONNECT B2;
Enter password: **
Connected.
 CREATE VIEW HIGH_ORDER_DAY AS
SELECT O.ORD_DATE,S.SALESMAN_ID,S.NAME,C.CUST_NAME,O.PURCHASE_AMT
FROM ORDERS O,SALESMAN S,CUSTOMER C
WHERE O.SALESMAN_ID=S.SALESMAN_ID AND C.CUSTOMER_ID=O.CUSTOMER_ID;

View created.

SELECT * FROM HIGH_ORDER_DAY H
WHERE H.PURCHASE_AMT=(SELECT MAX(H1.PURCHASE_AMT)
                      FROM HIGH_ORDER_DAY H1
```

WHERE H1.ORD_DATE=H.ORDER_DATE);

Output:

PURCHASE_AMT	ORD_DATE	CUST_NAME	SALESMAN_ID	NAME
100000	01-JAN-17	GANESH	2000	KRISHNA
200000	21-FEB-17	SUDHEEP	3000	SHIVA
300000	15-MAR-17	PRABHAS	4000	GOVINDA
400000	18-APR-17	ARVIND	5000	NARAYANA
600000	12-MAY-17	GANESH	1000	RAMA

5. Demonstrate the delete operation by removing salesman with ID 1000, all their orders must also be deleted.

DELETE FROM SALESMAN WHERE SALESMAN_ID=1000;

1 row deleted.

Output:

SQL> SELECT * FROM CUSTOMER;

CUSTOMER_ID	CUST_NAME	CITY	GRADE	SALESMAN_ID
20	SUDHEEP	BANGALORE	200	2000
30	PRABHAS	HYDERABAD	300	3000
40	ARVIND	CHENNAI	400	4000
50	DARSHAN	BANGALORE	500	2000

4 rows selected.

SQL> SELECT * FROM ORDERS;

ORD_NO	PURCHASE_AMT	ORD_DATE	CUSTOMER_ID	SALESMAN_ID
1111	100000	01-JAN-17	10	2000
2222	200000	21-FEB-17	20	3000
3333	300000	15-MAR-17	30	4000
4444	400000	18-APR-17	40	5000
5555	500000	12-MAY-17	10	

SQL> SELECT * FROM SALESMAN;

SALESMAN_ID	NAME	CITY	COMMISSION
2000	SUDHEEP	BANGALORE	20000.25
3000	ARJUN	HYDERABAD	30000.3
4000	ARVIND	CHENNAI	40000.4
5000	YASH	BANGALORE	50000.5

MOVIE DATABASE

Consider the schema for Movie Database:

ACTOR(ACT_ID,ACT_NAME,ACT_GENDER)
DIRECTOR(DIR_ID,DIR_NAME,DIR_PHONE)
MOVIES(MOV_ID,MOV_TITLE,MOV_YEAR,MOV_LANG,DIR_ID)
MOVIE_CAST(ACT_ID,MOV_ID,ROLE)
RATING(MOV_ID,REV_STARS)

Write SQL queries to

1. List the titles of all movies directed by'Hitchcock'.
2. Find the movie names where one or more actors acted in two or more movies.
3. List all actors who acted in a movie before 2000 and also in a movie after 2015 (use JOIN operation).
4. Find the title of movies and number of stars for each movie that has at least one rating and find the highest number of stars that movie received. Sort the result by movie title. Update rating of all movies directed by 'Steven Spielberg' to 5.

```
1. CREATE TABLEACTOR
(       ACT_ID NUMBER(2),
        ACT_NAME VARCHAR(15),
        ACT_GENDERCHAR,
        CONSTRAINT PK_AID PRIMARY KEY(ACT_ID)
  );
```

Table created.

```
SQL> DESC ACTOR;
 Name                      Null?      Type
 ----------------------    --------   ------------
 ACT_ID                    NOTNULL    NUMBER(2)
 ACT_NAME                             VARCHAR2(15)
 ACT_GENDER                           CHAR(1)
```

```
2.  CREATE TABLE DIRECTOR
(   DIR_ID NUMBER(2),
        DIR_NAME VARCHAR(20),
        DIR_PHONE NUMBER(10),
        CONSTRAINT PK_DID PRIMARY KEY(DIR_ID)
   );
```

Table created.

```
SQL> DESC DIRECTOR;
 Name                          Null?        Type
 ---------------------------------------------------------------------
 DIR_ID                        NOTNULL      NUMBER(2)
 DIR_NAME                                   VARCHAR2(20)
 DIR_PHONE                                  NUMBER(10)
```

3. CREATE TABLEMOVIES

```
(       MOV_ID NUMBER(3),
        MOV_TITLE VARCHAR(25),
        MOV_YEAR NUMBER(4),
        MOV_LANG VARCHAR(15),
        DIR_ID NUMBER(2),
        CONSTRAINT PK_MID PRIMARY KEY(MOV_ID),
        CONSTRAINT FK_DIR FOREIGN KEY(DIR_ID) REFERENCES DIRECTOR(DIR_ID) ON
        DELETE CASCADE
   );
```

Table created.

```
SQL> DESC MOVIES;
 Name                          Null?        Type
 ---------------------------------------------------------------------
 MOV_ID                        NOT NULL     NUMBER(3)
 MOV_TITLE                                  VARCHAR2(25)
 MOV_YEAR                                   NUMBER(4)
 MOV_LANG                                   VARCHAR2(15)
 DIR_ID                                     NUMBER(2)
```

4. CREATE TABLE MOVIE_CAST

```
(   ACT_ID NUMBER(2),
        MOV_ID NUMBER(3),
        ROLE VARCHAR(20),
        CONSTRAINT CPK_AM PRIMARY KEY(ACT_ID,MOV_ID),
        CONSTRAINT FK_MA FOREIGN KEY(ACT_ID) REFERENCES ACTOR(ACT_ID) ON
        DELETE CASCADE,
        CONSTRAINT FK_MD FOREIGN KEY(MOV_ID) REFERENCES MOVIES(MOV_ID) ON
        DELETE CASCADE
   );
```

Table created

```
SQL> DESC MOVIE_CAST;
 Name                    Null?          Type
 ---------------------------------------------------------------
 ACT_ID                  NOT NULL       NUMBER(2)
 MOV_ID                  NOTNULL        NUMBER(3)
 ROLE                                   VARCHAR2(20)
```

```
5.  CREATE TABLE RATING
(   MOV_IDNUMBER(3),
        REV_STARS NUMBER(1),
        CONSTRAINT CPK_MCT PRIMARY KEY(MOV_ID,REV_STARS),
        CONSTRAINT FK_MCD FOREIGN KEY(MOV_ID) REFERENCES MOVIES(MOV_ID) ON
        DELETE CASCADE
 );

Table created.

SQL> DESC RATING;
 Name                    Null?          Type
 ---------------------------------------------------------------
 MOV_ID                  NOTNULL        NUMBER(3)
 REV_STARS               NOTNULL        NUMBER(1)
```

INSERTIONCOMMANDS:

```
 INSERT INTO ACTOR VALUES(&ACT_ID,'&ACT_NAME','&ACT_GENDER');
Enter value for act_id: 10
Enter value for act_name: AAYUSHMAN
Enter value for act_gender: M
old 1: INSERT INTO ACTOR VALUES(&ACT_ID,'&ACT_NAME','&ACT_GENDER')
new 1: INSERT INTO ACTOR VALUES(10,'AAYUSHMAN','M')
1 row created.

SQL> SELECT * FROM ACTOR;

   ACT_ID        ACT_NAME              ACT_GENDER
---------      ----------          ----------
      10          AAYUSHMAN              M
      20          VARUNDHAWAN            M
      30          DEEPIKA                F
      40          CHRISPRATT             M
      50          ANTHONYPERKINS         M
      60          SHRADDHA               F
6 rowsselected.
INSERT INTO DIRECTOR VALUES(&DIR_ID,'&ADIR_NAME',&DIR_PHONE);
```

```
Enter value for dir_id: 11
Enter value for adir_name: SOOJITH
Enter value for dir_phone: 1020304050
old 1: INSERT INTO DIRECTOR VALUES(&DIR_ID,'&ADIR_NAME',&DIR_PHONE)
new 1: INSERT INTO DIRECTOR VALUES(11,'SOOJITH',1020304050)
1 row created.

SQL> SELECT * FROM DIRECTOR;

    DIR_ID          DIR_NAME            DIR_PHONE
-----------     ------------------------     ------------
        11              SOOJITH              1020304050
        22        STEVEN SPIELBERG  1122334455
        33               DAVID               9966443322
        44             HITCH COCK          1002003000
        55              BANSALI             9080706050
        66               REMO               9988776655

6 rows selected.

 INSERT INTO MOVIES
VALUES(&MOV_ID,'&MOV_TITLE',&MOV_YEAR,'&MOV_LANG',&DIR_ID);
Enter value for mov_id: 111
Enter value for mov_title: ABCD2
Enter value for mov_year: 1999
Enter value for mov_lang: HINDI
Enter value for dir_id: 66
old 1: INSERT INTO MOVIES
VALUES(&MOV_ID,'&MOV_TITLE',&MOV_YEAR,'&MOV_LANG',&DIR_ID)
new 1: INSERT INTO MOVIES VALUES(111,'ABCD2',1999,'HINDI',66)
1 row created.

SQL> SELECT * FROM MOVIES;

    MOV_ID          MOV_TITLE                MOV_YEAR        MOV_LANG        DIR_ID
-----------------------------------------------------------------------------
       111                ABCD2                  1999                HINDI           66
       222               PSYCHO                  1995                ENGLISH         44
       333               BAREILLI KI BURFI       2017                HINDI           11
       444               RAMLEELA                2015                HINDI           55
       555               MEIN TERA HERO          2014                HINDI           33
       666               JURASSIC PARK           2000                ENGLISH         22
       777               VICKY DONOR             2011                HINDI           11
```

```
7 rows selected.
 INSERT INTO MOVIE_CAST VALUES(&ACT_ID,&MOV_ID,'&ROLE');
Enter value for act_id: 10
Enter value for mov_id: 333
Enter value for role: HERO
old 1: INSERT INTO MOVIE_CAST VALUES(&ACT_ID,&MOV_ID,'&ROLE')
new 1: INSERT INTO MOVIE_CAST VALUES(10,333,'HERO')
1 row created.

SQL> SELECT * FROM MOVIE_CAST;

   ACT_ID     MOV_ID     ROLE
---------- ---------- ----------
       10        333        HERO
       20        555        HERO
       30        444    HEROINE
       40        666        HERO
       50        222     VILLAIN
       60        111    HEROINE
       20        111        HERO
       10        777        HERO
       60        333    HEROINE

SQL> INSERT INTO RATING VALUES(&MOV_ID,&REV_STARS);
Enter value for mov_id: 111
Enter value for rev_stars: 3
old 1: INSERT INTO RATING VALUES(&MOV_ID,&REV_STARS)
new 1: INSERT INTO RATING VALUES(111,3)
1 row created.
SQL> SELECT * FROM RATING;

  MOV_ID     REV_STARS
---------- ----------
     111          3
     111          5
     222          2
     222          3
     333          0
     444          4
     444          5
     555          3
     666          2
     777          4
```

10 rows selected.

QUERIES:

1.List the titles of all movies directed by hitchcock.

```
SELECT M.MOV_TITLE
FROM MOVIES M,DIRECTOR D
WHERE D.DIR_ID=M.DIR_ID AND D.DIR_NAME='HITCH COCK';
```

Output:

```
MOV_TITLE
----------------------------
PSYCHO
```

2.Find the movie names where one or more actors acted in two or more movies.

```
SELECTM.MOV_TITLE,A.ACT_NAME
FROM MOVIES M,ACTOR A,MOVIE_CASTM1
WHERE M.MOV_ID=M1.MOV_IDAND
        M1.ACT_ID=A.ACT_ID AND
        M1.ACT_ID IN( SELECT ACT_ID
                      FROM MOVIE_CAST
                      GROUP BY ACT_ID
                      HAVING COUNT(MOV_ID)>1);
```

Output:

```
MOV_TITLE                 ACT_NAME
----------------------------------------
MEINTERAHERO              VARUN DHAWAN
ABCD2                     VARUN DHAWAN
VICKYDONOR                AAYUSHMAN
BAREILLIKIBURFI           AAYUSHMAN
BAREILLIKIBURFI           SHRADDHA
ABCD2                     SHRADDHA
```

6 rows selected.

3.List all actors who acted in a movie before 2000 and also in a movie after 2015.(Use JOIN operator).

```
(SELECT A.ACT_NAME
 FROM ACTOR A JOIN MOVIE_CAST M ON A.ACT_ID=M.ACT_ID
        JOIN MOVIES M1 ON M.MOV_ID=M1.MOV_ID
WHERE M1.MOV_YEAR<2000 )
INTERSECT
 (SELECT A.ACT_NAME
```

```
        FROM ACTOR A JOIN MOVIE_CAST M ON A.ACT_ID=M.ACT_ID
                JOIN MOVIES M1 ON M.MOV_ID=M1.MOV_ID
      WHERE M1.MOV_YEAR>2015 );
```

Output:

```
ACT_NAME
----------
SHRADDHA
```

4. Find the title of movies and no. of stars for each movie that has atleast one rating and find the highest no. of stars that movie received. Sort the result by movietitkle.

```
  SELECT M.MOV_TITLE,MAX(R.REV_STARS)
  FROM MOVIES M,RATING R
  WHERE M.MOV_ID=R.MOV_ID AND
  M.MOV_ID IN(SELECT MOV_ID
                FROM RATING
                GROUP BY MOV_ID,REV_STARS
                HAVING REV_STARS>0 )
    GROUP BY M.MOV_TITLE
    ORDER BY M.MOV_TITLE;
```

Output:

```
MOV_TITLE              MAX(R.REV_STARS)
--------------------   --------------------
ABCD2                        5
JURASSIC PARK                2
MEIN TERA HERO               3
PSYCHO                       3
RAMLEELA                     5
VICKY DONOR                  4

6 rows selected.
```

5. Update ratings of all movies directed by STEVENSPIELBERG.

```
UPDATE RATING SET REV_STARS=5
 WHERE MOV_ID IN ( SELECT MOV_ID
                   FROM MOVIES
                   WHERE DIR_ID=(SELECT DIR_ID
                                 FROM DIRECTOR
                                 WHERE DIR_NAME='STEVEN SPIELBERG' )
  );
```

1 row updated.

Output:

```
SQL> SELECT * FROM RATING;

   MOV_ID REV_STARS
---------- ----------
       111          3
       111          5
       222          2
       222          3
       333          0
       444          4
       444          5
       555          3
       666          5
       777          4

10 rows selected.
```

COLLEGE DATABASE

Consider the schema for College Database:

STUDENT (USN,SNAME,ADDRESS,PHONE,GENDER)
SEMSEC (SSID,SEM,SEC)
CLASS (USN,SSID)
SUBJECT (SUBCODE,TITLE,SEM,CREDITS)
IAMARKS (USN,SUBCODE,SSID,TEST1,TEST2,TEST3,FINALIA)

Write SQL queries to

1. List all the student details studying in fourth semester 'C'section.
2. Compute the total number of male and female students in each semester and in eachsection.
3. Create a view of Test1 marks of student USN '1BI15CS101' in allsubjects.
4. Calculate the FinalIA (average of best two test marks) and update the corresponding table for all students.
5. Categorize students based on the followingcriterion:

   If FinalIA = 17 to 20 then CAT ='Outstanding'

   If FinalIA = 12 to 16 then CAT = 'Average'

   If FinalIA< 12 then CAT = 'Weak'

Give these details only for 8th semester A, B, and C section students.

CREATE TABLE COMMANDS:

```
1. CREATE TABLE STUDENT
(   USN CHAR(10),
        SNAME VARCHAR(20),
        ADDRESS VARCHAR(25),
        PHONE NUMBER(10),
        GENDER CHAR,
        CONSTRAINT A PRIMARY KEY(USN)
    );
```

Table created.

```
SQL> DESC STUDENT;
 Name                    Null?                 Type
 ----------------------------------------------------------------------
 USN                     NOT NULL              CHAR(10)
 SNAME                                         VARCHAR2(20)
 ADDRESS                                       VARCHAR2(25)
```

```
 PHONE                          NUMBER(10)
 GENDER                          CHAR
2. CREATE TABLE SEMSEC
(   SSIDCHAR(2),
        SEM NUMBER(1),
        SEC CHAR,
        CONSTRAINT B PRIMARY KEY(SSID),
        CONSTRAINT C CHECK(SEM BETWEEN 1 AND 8)
  );
Table created.

SQL> DESC SEMSEC;
 Name                Null?            Type
 ---------------------------------------------------------------------------
 SSID                NOT NULL          CHAR(2)
 SEM                                   NUMBER(1)
 SEC                                   CHAR(1)

3. CREATE TABLE CLASS
(   USN CHAR(10),
        SSID CHAR(2),
        CONSTRAINT D PRIMARY KEY(USN,SSID),
        CONSTRAINT E FOREIGN KEY(USN) REFERENCES STUDENT(USN) ON DELETE
        CASCADE,
        CONSTRAINT F FOREIGN KEY(SSID) REFERENCES SEMSEC(SSID) ON DELETE
        CASCADE
 );
Table created.

SQL> DESC CLASS;
 Name                          Null?          Type
 ---------------------------------------------------------------------------
 USN                           NOT NULL       CHAR(10)
 SSID                          NOT NULL        CHAR(2)

4. CREATE TABLESUBJECT
(       SUBCODEVARCHAR(7),
        TITLE VARCHAR(20),
        SEM NUMBER(1),
        CREDITSNUMBER(1),
        CONSTRAINT G PRIMARY KEY(SUBCODE)
  );
Table created.
```

```
SQL> DESC SUBJECT;
 Name                          Null?          Type
 ---------------------------------------------------------------------
 SUBCODE                    NOT NULL       VARCHAR2(7)
 TITLE                                     VARCHAR2(20)
 SEM                                       NUMBER(1)
CREDITS                                     NUMBER(1)
```

5. CREATE TABLE IAMARKS

```
(   USN CHAR(10),
        SUBCODE VARCHAR(7),
        SSID CHAR(2),
         TEST1 NUMBER(2),
        TEST2 NUMBER(2),
        TEST3 NUMBER(2),
        FINALIA NUMBER(2),
        CONSTRAINT H PRIMARY KEY(USN,SUBCODE,SSID),
        CONSTRAINT I FOREIGN KEY(USN) REFERENCES STUDENT(USN) ON DELETE
        CASCADE,
        CONSTRAINT J FOREIGN KEY(SSID) REFERENCES SEMSEC(SSID) ON DELETE
        CASCADE,
        CONSTRAINT K FOREIGN KEY(SUBCODE) REFERENCES SUBJECT(SUBCODE) ON
        DELETE CASCADE
  );

Table created.

SQL> DESC IAMARKS;
 Name                    Null?                  Type
 ---------------------------------------------------------------------
 USN                  NOT NULL              CHAR(10)
 SUBCODE               NOT NULL             VARCHAR2(7)
 SSID                 NOT NULL               CHAR(2)
 TEST1                                      NUMBER(2)
 TEST2                                      NUMBER(2)
 TEST3                                      NUMBER(2)
 FINALIA                                    NUMBER(2)
```

INSERTION COMMANDS:

```
SQL> INSERT INTO STUDENT
VALUES('&USN','&SNAME','&ADDRESS',&PHONE,'&GENDER');
Enter value for usn: 1MV17CS001
Enter value for sname: AASHISH
Enter value for address:BANGALORE
Enter value for phone: 1020304050
Enter value for gender:M
old 1: INSERT INTO STUDENT
VALUES('&USN','&SNAME','&ADDRESS',&PHONE,'&GENDER')
new 1: INSERT INTO STUDENT
VALUES('1MV17CS001','AASHISH','BANGALORE',1020304050,'M')
1 row created.

SQL> SELECT * FROM STUDENT;

USN          SNAME        ADDRESS          PHONE        GENDER
-----------  ------------ ---------------- ------------ ------
1MV17CS001   AASHISH      BANGALORE        1020304050   M
1MV17CS060   NAELA        MYSORE           1122334455   F
1MV17CS130   MILIND       JAMMU            5060708090   M
1MV16CS001   ABHIJITH     PUNE             9988776655   M
1MV16CS060   NIKITHA      HYDERABAD        9080706050   F
1MV16CS130   SANJANA      GUWAHATTI        1234567890   F
1MV15CS001   ANSHUMAN     PANAJI           1112223334   M
1MV15CS060   AMRUTHA      BANGALORE        1002003004   F
1MV15CS130   BHUVANESH    JAIPUR           9008007006   M
1MV14CS001   DEVAYANI     BANGALORE        1000200030   F
1MV14CS060   DAVID        KOCHI            9000800070   M
1MV14CS130   AISHWARYA    MUMBAI           1000020000   F

12 rows selected.

SQL> INSERT INTO SEMSEC VALUES('&SSID',&SEM,'&SEC');
Enter value for ssid: 2A
Enter value for sem: 2
Enter value for sec: A
old 1: INSERT INTO SEMSEC VALUES('&SSID',&SEM,'&SEC')
new 1: INSERT INTO SEMSEC VALUES('2A',2,'A')
1 row created.
```

```
SQL> SELECT * FROM SEMSEC;

SSID       SEM     SEC
--         ---     -------
2A          2       A
2B          2       B
2C          2       C
4A          4       A
4B          4       B
4C          4       C
6A          6       A
6B          6       B
6C          6       C
8A          8       A
8B          8       B
8C          8       C
12 rows selected.

SQL> INSERT INTO CLASS VALUES('&USN','&SSID');
Enter value for usn: 1MV17CS001
Enter value for ssid: 2A
old   1: INSERT INTO CLASS VALUES('&USN','&SSID')
new 1: INSERT INTO CLASSVALUES('1MV17CS001','2A')
1 row created.
SQL> SELECT * FROM CLASS;

USN              SSID
-------          -----
1MV14CS001  8A
1MV14CS060  8B
1MV14CS130  8C
1MV15CS001  6A
1MV15CS060  6B
1MV15CS130  6C
1MV16CS001  4A
1MV16CS060  4B
1MV16CS130  4C
1MV17CS001  2A
1MV17CS060  2B
1MV17CS130  2C
12 rows selected.
```

```
SQL> INSERT INTO SUBJECT VALUES('&SUBCODE','&TITLE',&SEM,&CREDITS);
Enter value for subcode: 15CS21
Enter value for title:M2
Enter value for sem: 2
Enter value for credits:4
old 1: INSERT INTO SUBJECT VALUES('&SUBCODE','&TITLE',&SEM,&CREDITS)
new 1: INSERT INTO SUBJECT VALUES('15CS21','M2',2,4)
1 row created.

SQL> SELECT * FROM SUBJECT;

SUBCODE     TITLE          SEM    CREDITS
--------------------------------------------------------------
15CS21      M2             2      4
15PCD23     PCD            2      4
15CS42      SE             4      4
15CS44      MP             4      4
15CS64      CG             6      4
15CS62      USP            6      4
10CS81      SA             8      4
10CS842     ST             8      3

8 rows selected.

SQL> INSERT
INTOIAMARKS(USN,SUBCODE,SSID,TEST1,TEST2,TEST3)VALUES('&USN','&SUBCODE','&SSI
D',&TEST1,&TEST2,&TEST3);
Enter value for usn: 1MV17CS001
Enter value for subcode: 15CS21
Enter value for ssid: 2A
Enter value for test1: 15
Enter value for test2: 14
Enter value for test3: 13
old 1: INSERT INTO
IAMARKS(USN,SUBCODE,SSID,TEST1,TEST2,TEST3)VALUES('&USN','&SUBCODE','&SSID',&
TEST1,&TEST2,&TEST3);
new 1: INSERT INTO
IAMARKS(USN,SUBCODE,SSID,TEST1,TEST2,TEST3)VALUES('1MV17CS001','15CS21','2A',15,1
4,13);
1 row created.
```

```
SQL> SELECT * FROM IAMARKS;
USN          SUBCODE SSID    TEST1    TEST2          TEST3    FINALIA
-----------  -------- ----   -------  ----------     -------
1MV17CS001 15CS21     2A     15       14             13
1MV17CS060  15PCD23 2B       15       15             14

1MV17CS130  15CS21    2C     11       12             13
1MV16CS001  15CS42    4A     19       19             18
1MV16CS060  15CS44    4B      5        8              5
1MV16CS130  15CS42    4C     20       20             20
1MV15CS001  15CS64    6A     12       12             12
1MV15CS060  15CS62    6B     18       19             20
1MV15CS130  15CS64    6C      8       12             11
1MV14CS001  10CS81    8A      3       11             12
1MV14CS060  10CS842   8B      0        0              7
1MV14CS130   10CS81   8C      0        0             20
12 rows selected.
```

QUERIES:

1. Make a list of all students details studying in 4th sem c-sec.

```
SELECT S.USN,S.SNAME,S.ADDRESS,S.PHONE,S.GENDER
FROM STUDENT S,CLASS C,SEM_SECSS
WHERE S.USN=C.USN AND
      SS.SSID=C.SSID AND
      SS.SEM=4 AND
      SS.SEC='C';
```

Output:

```
USN          SNAME      ADDRESS     PHONE       GENDER
1MV16CS130   SANJANA   GUWATHI     1234567890    F
```

2. Compute the total no.of male and female students in each semester and in eachsec.

```
SELECT SS.SEM,SS.SEC,S.GENDER,COUNT(S.GENDER)
FROM STUDENT S,SEMSEC SS,CLASS C
WHERE S.USN=C.USN AND SS.SSID=C.SSID
GROUP BY SS.SEM,SS.SEC,S.GENDER;
```

Output:

```
SEM    SEC GENDER   COUNT(S.GENDER)
------ ---- -------  ------------
  4     B     F          1
  2     C     M          1
  6     A     M          1
  8     B     M          1
  2     A     M          1
  8     A     F          1
  2     B     F          1
  4     A     M          1
  4     C     F          1
  6     B     F          1
  6     C     M          1
  8     C     F          1
12 rows selected.
```

3. Create view of test1 marks of student 1MV15CS060 in all subjects.

```
CREATE VIEW TEST1_MARKS AS
SELECT USN,SUBCODE,TEST1
FROM IAMARKS
WHERE USN='1MV15CS060';
View created.
```

Output:

```
SQL> SELECT * FROM TEST1_MARKS;
USN          SUBCODE    TEST1
----------   -------    ----------
1MV15CS060   15CS62      18

SQL> UPDATE IAMARKS SET TEST1=19,TEST2=18,TEST3=17 WHERE USN='1MV14CS001';
1 row updated.

SQL> UPDATE IAMARKS SET TEST1=11,TEST2=0,TEST3=14 WHERE USN='1MV14CS060';
1 row updated.

SQL> UPDATE IAMARKS SET TEST1=10,TEST2=0,TEST3=7 WHERE USN='1MV14CS130';
1 row updated.

SQL> SELECT * FROM IAMARKS;
```

USN	SUBCODE	SSID	TEST1	TEST2	TEST3	FINALIA
1MV17CS001	15CS21	2A	15	14	13	
1MV17CS060	15PCD23	2B	15	15	14	
1MV17CS130	15CS21	2C	11	12	13	
1MV16CS001	15CS42	4A	19	19	18	
1MV16CS060	15CS44	4B	5	8	5	
1MV16CS130	15CS42	4C	20	20	20	
1MV15CS001	15CS64	6A	12	12	12	
1MV15CS060	15CS62	6B	18	19	20	
1MV15CS130	15CS64	6C	8	12	11	
1MV14CS001	10CS81	8A	19	18	17	
1MV14CS060	10CS842	8B	11	0	14	
1MV14CS130	10CS81	8C	10	0	7	

12 rows selected.

4. Calculate the final IA marks and update the corresponding table for all students.

```
CREATE OR REPLACE PROCEDURE AVGMARKS
 IS
CURSOR C_IAMARKS IS
SELECT GREATEST(TEST1,TEST2) AS A,GREATEST(TEST1,TEST3) AS
B,GREATEST(TEST2,TEST3) AS C
FROM IAMARKS
WHERE FINALIA IS NULL
 FOR UPDATE;
 C_A NUMBER;
C_B NUMBER;
C_C NUMBER;
C_SUM NUMBER;
C_AVG NUMBER;
BEGIN
OPEN C_IAMARKS;
LOOP
FETCH C_IAMARKS INTO C_A,C_B,C_C;
EXIT WHEN C_IAMARKS%NOTFOUND;
DBMS_OUTPUT.PUT_LINE(C_A||''||C_B||''||C_C);
IF(C_A!=C_B) THEN
C_SUM:=C_A+C_B;
ELSE
C_SUM:=C_A+C_C;
END IF;
```

```
C_AVG:=C_SUM/2;
DBMS_OUTPUT.PUT_LINE('SUM='||C_SUM);
DBMS_OUTPUT.PUT_LINE('AVERAGE='||C_AVG);
UPDATE IAMARKS SET FINALIA=C_AVG
WHERE CURRENT OFC_IAMARKS;
END LOOP;
CLOSE C_IAMARKS;
END;
/
```

Procedure created.

```
SQL> BEGIN AVGMARKS;
  END;
  /
```

PL/SQL procedure successfully completed.

Output:

```
SQL> SELECT * FROM IAMARKS;

USN          SUBCODE SSID   TEST1  TEST2 TEST3   FINALIA
-----------  -------- --   -------  ----  -----  -------
1MV17CS001 15CS21      2A    15     14    13     15
1MV17CS060 15PCD23     2B    15     15    14     15
1MV17CS130 15CS21      2C    11     12    13     13
1MV16CS001 15CS42      4A    19     19    18     19
1MV16CS060 15CS44      4B    5      8     5      7
1MV16CS130 15CS42      4C    20     20    20     20
1MV15CS001 15CS64      6A    12     12    12     12
1MV15CS060 15CS62      6B    18     19    20     20
1MV15CS130 15CS64      6C    8      12    11     12
1MV14CS001 10CS81      8A    19     18    17     19
1MV14CS060 10CS842     8B    11     0     14     13
1MV14CS130 10CS81      8C    10     0     7      9

12 rows selected.

SQL> UPDATE IAMARKS SET FINALIA=NULL;
12 rows updated.
```

```
SQL> SELECT * FROM IAMARKS;
USN          SUBCODE   SSID   TEST1   TEST2   TEST3   FINALIA
-----------  ---------  ------  -------  -------  -------  -------
1MV17CS001   15CS21    2A      15      14      13
1MV17CS060   15PCD23   2B      15      15      14
1MV17CS130   15CS21    2C      11      12      13
1MV16CS001   15CS42    4A      19      19      18
1MV16CS060   15CS44    4B      5       8       5
1MV16CS130   15CS42    4C      20      20      20
1MV15CS001   15CS64    6A      12      12      12
1MV15CS060   15CS62    6B      18      19      20
1MV15CS130   15CS64    6C      8       12      11
1MV14CS001   10CS81    8A      19      18      17
1MV14CS060   10CS842   8B      11      0       14
1MV14CS130   10CS81    8C      10      0       7

12 rows selected.
```

Alternate:

```
SQL> DECLARE
  CURSOR C_IA_MARKS
  IS
   SELECT TEST1,TEST2,TEST3 FROM IA_MARKS
   WHERE FINAL_IA IS NULL
   FOR UPDATE;
  C_T1 NUMBER;
  C_T2 NUMBER;
  C_T3 NUMBER;
  C_SUM NUMBER;
  C_AVG NUMBER;
  C_MIN NUMBER;
  BEGIN
  OPEN C_IA_MARKS;
  LOOP
  FETCH C_IA_MARKS INTO C_T1,C_T2,C_T3;
  EXIT WHEN C_IA_MARKS%NOTFOUND;
  C_SUM:=C_T1+C_T2+C_T3;
  DBMS_OUTPUT.PUT_LINE('SUM='||C_SUM);
  IF((C_T1<=C_T2)AND(C_T1<=C_T3))
  THEN
  C_MIN:=C_T1;
  ELSE IF((C_T2<=C_T1)
  AND(C_T2<=C_T3))THEN
  C_MIN:=C_T2;
  ELSE
```

```
C_MIN:=C_T3;
END IF;
END IF;
DBMS_OUTPUT.PUT_LINE('MIN='||C_MIN);
C_AVG:=(C_SUM-C_MIN)/2;
DBMS_OUTPUT.PUT_LINE('AVERAGE='||C_AVG);
UPDATE IA_MARKS SET FINAL_IA=C_AVG WHERE
CURRENT OF C_IA_MARKS;
END LOOP;
CLOSE C_IA_MARKS;
END;
/
```

PL/SQL procedure successfully completed.

Output:

SQL> SELECT * FROM IAMARKS;

USN	SUBCODE	SSID	TEST1	TEST2	TEST3	FINALIA
1MV17CS001	15CS21	2A	15	14	13	15
1MV17CS060	15PCD23	2B	15	15	14	15
1MV17CS130	15CS21	2C	11	12	13	13
1MV16CS001	15CS42	4A	19	19	18	19
1MV16CS060	15CS44	4B	5	8	5	7
1MV16CS130	15CS42	4C	20	20	20	20
1MV15CS001	15CS64	6A	12	12	12	12
1MV15CS060	15CS62	6B	18	19	20	20
1MV15CS130	15CS64	6C	8	12	11	12
1MV14CS001	10CS81	8A	19	18	17	19
1MV14CS060	10CS842	8B	11	0	14	13
1MV14CS130	10CS81	8C	10	0	7	9

12 rows selected.

5. Categorise students based on the following criteria
If FINALIA=17 to 20 then
CAT=’OUTSTANDING’ If FINALIA=12 to 16
then CAT=’AVERAGE’
If FINALIA=00 to 11 then CAT=’WEAK’
Give these details only for 8th semester A, B and C section students.

```
SELECT S.USN,S.SNAME,S.ADDRESS,S.PHONE,S.GENDER,
 ( CASE
 WHEN IA.FINALIA BETWEEN 17 AND 20
 THEN 'OUTSTANDING'
 WHEN IA.FINALIA BETWEEN 12 AND 16
```

```
THEN 'AVERAGE'
ELSE 'WEAK'
END
) AS CAT
FROM STUDENT S,SEMSEC SS,IAMARKS IA
WHERE S.USN=IA.USN AND
SS.SSID=IA.SSID AND
SS.SEM=8;
```

USN	SNAME	ADDRESS	PHONE	GENDER	CAT
1MV14CS001	DEVAYANI	BANGALORE	10020030	F	OUTSTANDING
1MV14CS060	DAVID	KOCHI	90080070	M	AVERAGE
1MV14CS130	AISHWARYA	MUMBAI	1000020000	F	WEAK

COMPANY DATABASE

EMPLOYEE(SSN,NAME,ADDRESS,SEX,SALARY,SUPERSSN,DNO)
DEPARTMENT(DNO,DNAME,MGRSSN,MGRSTARTDATE)
DLOCATION(DNO,DLOC)
PROJECT(PNO,PNAME,PLOCATION,DNO)
WORKS_ON(SSN,PNO,HOURS)

```
1.  CREATE TABLE EMPLOYEE
(   SSN NUMBER(9), NAME
VARCHAR(20),
        ADDRESS VARCHAR(25),
        SEX CHAR,
        SALARY NUMBER(10,2),
        SUPERSSN NUMBER(9),
        DNO NUMBER(2),
        CONSTRAINT PA PRIMARY KEY(SSN),
        CONSTRAINT PB FOREIGN KEY(SUPERSSN) REFERENCES EMPLOYEE(SSN) ON
        DELETE CASCADE
);
Table created.

SQL> DESC EMPLOYEE;
 Name                    Null?        Type
-----------------                ---------------------------------------
 SSN                     NOT NULL     NUMBER(9)
 NAME                                 VARCHAR2(20)
 ADDRESS                              VARCHAR2(25)
 SEX                                  CHAR(1)
 SALARY                               NUMBER(10,2)
 SUPERSSN                             NUMBER(9)
 DNO                                  NUMBER(2)

2. CREATE TABLE DEPARTMENT
 ( DNO NUMBER(2),
        DNAME VARCHAR(15),
        MGRSSN NUMBER(9),
        MGRSTARTDATE DATE,
        CONSTRAINT PC PRIMARY KEY(DNO),
        CONSTRAINT PKD FOREIGN KEY(MGRSSN) REFERENCES EMPLOYEE(SSN) ON
        DELETE CASCADE

);

Table created.
```

```
SQL> DESC DEPARTMENT;
 Name                      Null?        Type
 ---------------------------------------------------------------------
 DNO                       NOTNULL      NUMBER(2)
 DNAME                                  VARCHAR2(15)
 MGRSSN                                 NUMBER(9)
 MGRSTARTDATE                           DATE
SQL> ALTER TABLE EMPLOYEE ADD CONSTRAINT PE FOREIGN KEY (DNO) REFERENCES
DEPARTMENT (DNO) ON DELETECASCADE;

Table altered.
```

3. CREATE TABLE DLOCATION

```
 ( DNO NUMBER(2),
        DLOC VARCHAR(15),
        CONSTRAINT CPK PRIMARY KEY (DNO,DLOC),
        CONSTRAINT FK_D1 FOREIGN KEY (DNO) REFERENCES DEPARTMENT (DNO) ON
        DELETE CASCADE
);

Table created.

SQL> DESC DLOCATION;
 Name                      Null?        Type
 ---------------------------------------------------------------------
 DNO                       NOT NULL     NUMBER(2)
 DLOC                      NOT NULL     VARCHAR2(15)
```

4. CREATE TABLE PROJECT

```
 (
        PNO NUMBER(2),
        PNAME VARCHAR(15),
        PLOCATION VARCHAR(15),
        DNO NUMBER(2),
        CONSTRAINT PKP PRIMARY KEY(PNO),
        CONSTRAINT FKD FOREIGN KEY (DNO) REFERENCES DEPARTMENT (DNO) ON
        DELETE CASCADE
);
Table created.
```

```
SQL> DESC PROJECT;

 Name                      Null?              Type
 ---------------------------------------------------------------
 PNO                   NOT NULL           NUMBER(2)
 PNAME                                    VARCHAR2(15)
 PLOCATION                                VARCHAR2(15)
 DNO                                      NUMBER(2)
```

5. CREATE TABLE WORKS_ON

```
 (
        SSN NUMBER(9),
        PNO NUMBER(2),
        HOURS NUMBER(3),
        CONSTRAINT SP PRIMARY KEY (SSN,PNO),
        CONSTRAINT FKS FOREIGN KEY (SSN) REFERENCES EMPLOYEE (SSN) ON DELETE
        CASCADE,
        CONSTRAINT FKP FOREIGN KEY (PNO) REFERENCES PROJECT (PNO) ON DELETE
        CASCADE
);

Table created

SQL> DESC WORKS_ON;
 Name                      Null?              Type
 ---------------------------------------------------------------
 SSN                   NOT NULL           NUMBER(9)
 PNO                   NOT NULL           NUMBER(2)
 HOURS                                    NUMBER(3)
```

INSERTION:

```
SQL> INSERT INTO EMPLOYEE(SSN,NAME,ADDRESS,SEX,SALARY)
VALUES(&SSN,'&NAME','&ADDRESS','&SEX',&SALARY
);
Enter value for ssn: 123456789
Enter value for name: ASHA
Enter value for address: YELAHANKA
Enter value for sex: F
Enter value for salary: 500000
old 1: INSERT INTO EMPLOYEE(SSN,NAME,ADDRESS,SEX,SALARY)
VALUES(&SSN,'&NAME','&ADDRESS','&SEX',&SA
new 1: INSERT INTO EMPLOYEE(SSN,NAME,ADDRESS,SEX,SALARY)
VALUES(123456789,'ASHA','YELAHANKA','F',5

1 row created.
```

```
SQL> SELECT * FROM DEPARTMENT;

      DNODNAME          MGRSSN     MGRSTARTDATE
----------- ------------------- ----------- -----------
         1  CSE             234567891   01-JAN-10
         2  ISE             345678912   15-FEB-11
         3  ECE             456789123   01-MAR-12
         4  ACCOUNTS        678912345   15-APR-13
         5  TCE             123456789   02-MAY-14
         6  ECE             567891234   15-JUN-15

6 rows selected.

SQL> UPDATE EMPLOYEE SET SUPERSSN=&SUPERSSN,DNO=&DNO WHERE SSN=&SSN;
Enter value for superssn: 234567891
Enter value for dno: 1
Enter value for ssn: 123456789
old 1: UPDATE EMPLOYEE SET SUPERSSN=&SUPERSSN,DNO=&DNO WHERE SSN=&SSN
new 1: UPDATE EMPLOYEE SET SUPERSSN=234567891,DNO=1 WHERE SSN=123456789
1 row updated.
SQL> /

Enter value for superssn: 678912345
Enter value for dno: 2
Enter value for ssn: 234567891
old 1: UPDATE EMPLOYEE SET SUPERSSN=&SUPERSSN,DNO=&DNO WHERE SSN=&SSN
new 1: UPDATE EMPLOYEE SET SUPERSSN=678912345,DNO=2 WHERE SSN=234567891

SQL> SELECT *FROM EMPLOYEE;

 SSN         NAME          ADDRESS          SEX  SALARY     SUPERSSN     DNO
---------- ---------- ----------- ---------- ---------- ---------- ----------
123456789   ASHA        YELAHANKA          F        500000   234567891      1
 234567891   SHEELA     JAKPUR             F        700000   678912345      2
 345678912   PALLAVI    NEWTOWN            F        700000   234567891      3
 456789123SHREYAS       BASAWESWAR         M        750000   234567891      4
 567891234   MOHAN      TUMKUR             M        350000   678912345      5
 678912345   SCOTT      NEWYORK            M        1000000                 6
 789123456   DIVYA      HUNSMARENHALLI     F        350000   223344556      2
 891234567   SAPNA      VIDYARANYAPURAF     350000   112233445      2
 912345678   REVAN      MADIWALA           M        450000   123456789      2
112233445   SAVITHA     DBSANDRA           F        700000   234567891      2
 223344556   VIJAY      VIJAYPURA          M        600000   345678912      2
 334455667  MANDHAR     JAYMAHAL           M        500000   456789123      4
445566778 RAGHAV        YELAHANKA          M        600000   345678912      4
```

13 rows selected

```
SQL> INSERT INTO DLOCATION VALUES(&DNO,'&DLOC');
Enter value for dno: 1
Enter value for dloc: NB1
old 1: INSERT INTO DLOCATION VALUES(&DNO,'&DLOC')
new 1: INSERT INTO DLOCATION VALUES(1,'NB1')
1 row created.

SQL> SELECT * FROM DLOCATION;

      DNO    DLOC
----------  ----------
         1        NB1
         2        NB0
         3        NB2
         4        NB1
         5        NB3
         6       EEE2

6 rows selected.

SQL> INSERT INTO PROJECT VALUES(&PNO,'&PNAME','&PLOCATION',&DNO);
Enter value for pno: 11
Enter value for pname: JAVA
Enter value for plocation: MARATHAHALLI

Enter value for dno: 1
old 1: INSERT INTO PROJECT VALUES(&PNO,'&PNAME','&PLOCATION',&DNO)
new 1: INSERT INTO PROJECT VALUES(11,'JAVA','MARATHAHALLI',1)
1 row created.

SQL> SELECT * FROM PROJECT;

      PNO    PNAME          PLOCATION             DNO
----------  ----------  ----------  ----------
        11        JAVA          MARATHAHALLI           1
        22     DOTNET         HEBBAL                 2
        33         IOT          MANYATA                3
        44     ANDROID        YELAHANKA              4
        55     BIGDATA        KR PURAM                5
        66       WEB            ELECTRONIC CITY        6

6 rows selected.

SQL> INSERT INTO WORKS_ON VALUES(&SSN,&PNO,&HOURS);
```

```
Enter value for ssn: 678912345
Enter value for pno: 11
Enter value for hours: 25
old 1: INSERT INTO WORKS_ON VALUES(&SSN,&PNO,&HOURS)
new 1: INSERT INTO WORKS_ON VALUES(678912345,11,25)
1 row created.

SQL> SELECT * FROM WORKS_ON;

      SSN        PNO
-----------  -----------
 678912345
 123456789
 234567891
 678912345
 345678912
 456789123

6 rows selected.
```

QUERIES:

1. Make a list of all project members for projects that involve an employee whose name is SCOTT either as a worker or as a manager of the department that controls the project.

```
 (SELECT DISTINCT P.PNO
   FROM PROJECT P,DEPARTMENT D,EMPLOYEE E
   WHERE P.DNO=D.DNO AND D.MGRSSN=E.SSN AND E.NAME='SCOTT')
 UNION
 (SELECT DISTINCT P.PNO
 FROM PROJECT P,WORKS_ON W,EMPLOYEE E
 WHERE P.PNO=W.PNO AND W.SSN=E.SSN AND E.NAME='SCOTT');
```

Output:

```
      PNO
-----------
       11
       44
```

2. Show the resulting salary for employee working on IOT project is given a 10% raise.

```
 SELECT E.NAME,1.1*E.SALARY AS HIKE_SALARY
 FROM EMPLOYEE E,WORKS_ON W,PROJECT P
 WHERE E.SSN=W.SSN AND P.PNO=W.PNO AND P.PNAME='IOT';
```

Output:

```
NAME            HIKE_SALARY
--------------------------------------
SHEELA              770000
```

3. Find the sum of salaries of all employees of 'ACCOUNTS' department as well as the MAX(SAL),MIN(SAL),AVG(SAL) in this department.

```
SELECT SUM(E.SALARY) AS SUM_SAL,MAX(E.SALARY) AS MAX_SAL,MIN(E.SALARY) AS
MIN_SAL,AVG(E.SALARY) AS AVG_SAL
FROM EMPLOYEE E,DEPARTMENT D
WHERE E.DNO=D.DNO AND D.DNAME='ACCOUNTS';
```

```
  SUM_SAL    MAX_SAL    MIN_SAL    AVG_SAL
-----------  -------------  -------------  -------------
  1850000      750000       500000       616666.667
```

4. Retrieve the name of each employee who works on all the projects controlled by department no. 5.(use NOT EXISTS)operator.

```
SQL>SELECT E.NAME FROM EMPLOYEE E
      WHERE NOT EXISTS((SELECT P.PNO FROM PROJECT P WHERE P.DNO=5)
      MINUS (SELECT W.PNO FROM WORKS_ON W WHERE E.SSN=W.SSN));
```

```
NAME
----------------------
PALLAVI
```

Alternate:

```
 SELECT E.NAME    FROM EMPLOYEE E
  WHERE NOT EXISTS(SELECT*
                          FROM WORKS_ON W1
                         WHERE (W1.PNO IN(SELECTP.PNO
                                                FROM PROJECT P
                                            WHEREP.DNO=5)
 2  AND
 3  NOT EXISTS(SELECT*
                 FROM WORKS_ON W2
                 WHERE W2.SSN=E.SSN AND W2.PNO=W1.PNO)));
```

Output:

```
NAME
----------------------
PALLAVI
```

5.For each department that has more than 5 employees retrieve the dno and no. of its employees who are making more than 6,00,000.

```
SELECT DNO,COUNT(*) AS NO_OF_EMP
FROM EMPLOYEE
WHERE SALARY>600000 AND DNO IN(SELECT DNO
                                FROM EMPLOYEE
                               GROUP BY(DNO)
                               HAVING COUNT(*)>5)
GROUP BY DNO;
```

Output:

```
DNO      NO_OF_EMP
-----------  -------------
     2          2
```