**Practical file submitted in partial fulfillment for the evaluation of**

**“Database Management System Lab (AIDS-254)”**



**Submitted By:**

**Student Name:** Kirty Gupta

**Enrolment no:** 03317711922

**Branch & Section:** AI-DS (A)

**Submitted To:**

* Dr. Deepika Bhatia

**Index**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S.No | Experiment Title (GGSIPU) | Page No. | Date | Grade/  Evaluation | Sign |
| 1 | Study and practice various database management systems like MySQL/Oracle/PostgreSQL/SQL Server and others. |  |  |  |  |
| 2 | Implement simple queries of DDL and DML. |  |  |  |  |
| 3 | Implement basic queries to Create, Insert, Update, Delete and Select Statements for two different scenarios (For instance: Bank, College etc.) |  |  |  |  |
| 4 | Implement queries including various functions- mathematical, string, date etc. |  |  |  |  |
| 5 | Implement queries including Sorting, Grouping and Subqueries- like any, all, exists, not exists. |  |  |  |  |
| 6 | Implement queries including various Set operations (Union, Intersection, Except etc.). |  |  |  |  |
| 7 | Implement various JOIN operations- (Inner, Outer). |  |  |  |  |
| 8 | Write a PL/SQL program using FOR loop to insert ten rows into a database table. |  |  |  |  |
| 9 | Given the table EMPLOYEE (Emp No, Name, Salary, Designation, DeptID), write a cursor to select the five highest-paid employees from the table. |  |  |  |  |
| 10 | Illustrate how you can embed PL/SQL in a high-level host language such as C/Java And demonstrates how a banking debit transaction might be done. |  |  |  |  |

**Index**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S.No | Experiment Title (Beyond Curriculum) | Page No. | Date | Grade/  Evaluation | Sign |
| 1 | Write the steps to install and implement NOSQL databases-MongoDB |  |  |  |  |
| 2 | Study and implement basic commands of MongoDB |  |  |  |  |
| 3 | Implement any one real-time project using MySQL/MongoDB such as Library Database Management System etc.. |  |  |  |  |

# SECTION 1

**Database Management System Lab**

**GGSIPU**

**EXPERIMENT - 1**

**AIM: -** STUDY AND PRACTICE VARIOUS DATABASE MANAGEMENT SYSTEMS LIKE MYSQL / ORACLE / POSTGRE SQL/ SQL SERVER AND MONGO DB.

**THEORY: -**

A **Database Management System (DBMS)** is a software system that allows users to create, maintain, and manage databases. It provides an interface between the end-user and the database, enabling users to create, read, update, and delete data in the database. Some popular databases are MySQL, Oracle, MongoDB, etc. DBMS provides many operations such as creating a database, storing in the database, updating an existing database, and deleting from the database. DBMS is a system that enables you to store, modify, and retrieve data in an organized way. It also provides security to the database. In this Database Management System tutorial, you’ll learn basic to advanced topics like ER model, Relational Model, Relation Algebra, Normalization, File Organization, etc.

**MySQL:**

MySQL is an open-source relational database management system (RDBMS) that enables users to store, manage, and retrieve structured data efficiently. It is widely used for various applications, from small-scale projects to large-scale websites and enterprise-level solutions. MySQL offers various features and solutions for different scenarios, such as data warehousing, data lakes, machine learning, and cloud applications. MySQL is available in different editions, including MySQL Heat Wave, Enterprise Edition, Cluster CGE, and other products and services.

**FEATURES OF MySQL :**

**Open Source -** MySQL is freely available and open-source, allowing users to view, modify, and distribute the source code as per the terms of the GNU General Public License (GPL).

**Cross-Platform Compatibility -** MySQL is designed to run on various operating systems, including Linux, Windows, macOS, and others, providing flexibility and ease of deployment.

**Ease of Use -** MySQL is known for its user-friendly interface and ease of use. Tools like MySQL Workbench provide a graphical environment for managing databases.

**Scalability -** MySQL supports both vertical and horizontal scaling, making it suitable for small projects as well as large-scale applications with high traffic.

**ADVANTAGES :**

**Faster Query Processing –** Large amount of data is retrieved quickly and efficiently. Operations like Insertion, deletion, manipulation of data is also done in almost no time.

**No Coding Skills –** For data retrieval, large number of lines of code is not required. All basic keywords such as SELECT, INSERT INTO, UPDATE are used and also the syntactical rules are not complex in SQL, which makes it a user-friendly language.

**Standardized Language –** Due to documentation and long establishment over years, it provides a uniform platform worldwide to all its users.

**Portable –** It can be used in programs in PCs, server, laptops independent of any platform (Operating System,). Also, it can be embedded with other applications as per need/requirement/use.

**DISADVANTAGES :**

**Limited Support for Complex Transactions -** MySQL may not be the best choice for applications that require complex transactions and full ACID compliance, as it may face limitations compared to some other relational database management systems (RDBMS).

**Lack of Features in Earlier Versions -** Older versions of MySQL lacked some advanced features compared to other database systems. However, many of these limitations have been addressed in recent versions.

**Default Configuration Can Be Insecure -** The default configuration of MySQL may not be as secure as needed for production environments. Proper configuration and security measures must be implemented to safeguard against potential vulnerabilities.

**Limited Support for NoSQL Features-** While MySQL is primarily a relational database, it has introduced some support for NoSQL features in recent versions. However, it may not be as feature-rich as dedicated NoSQL databases for certain use cases.

**ORACLE**

Oracle Database Management System (DBMS) is a software system that allows users to create, maintain, and manage databases. It provides an interface between the end-user and the database, enabling users to create, read, update, and delete data in the database. Oracle DBMS is known for its portability as it can be easily ported to different platforms than its competition. It runs on almost 20 networking protocols and more than 100 hardware platforms. Oracle DBMS is also known for its backup and recovery features.

**FEATURES OF ORACLE :**

**High Availability -** Oracle provides features like Oracle Real Application Clusters (RAC) and Data Guard for ensuring high availability, fault tolerance, and disaster recovery.

**Scalability -** Oracle Database is designed to scale vertically and horizontally, making it suitable for both small-scale applications and large enterprise systems.

**ACID Compliance -** Oracle follows the ACID properties (Atomicity, Consistency, Isolation, Durability), ensuring reliable and transactionally consistent data.

**Multitenancy Support -** Oracle supports multitenant architecture, allowing multiple pluggable databases (PDBs) to be managed within a single container database (CDB).

**ADVANTAGES :**

**Robust and scalable -**Oracle is known for its robustness and scalability, making it suitable for large-scale enterprise applications with high transaction volumes.

**Acid compliance-**Oracle ensures full ACID (Atomicity, Consistency, Isolation, Durability) compliance, providing a high level of data integrity and reliability.

**Comprehensive feature set**-Oracle offers a wide range of features, including partitioning, clustering, online analytical processing (OLAP), and advanced security features, making it versatile for various application requirements

**high performance**-Oracle Database is optimized for high performance, and it includes features like caching, indexing, and query optimization to enhance data retrieval speed.

**DISADVANTAGES :**

**Cost-** Oracle Database is known for its high licensing and maintenance costs, which can be a significant factor for smaller businesses or projects with budget constraints.

**Complexity**- The advanced feature set of Oracle can lead to a steeper learning curve and increased complexity, especially for users new to the system.

**Resource**- Oracle databases may consume more system resources compared to some other database systems, potentially impacting the performance of less powerful hardware.

**Propriety software**- Oracle is proprietary software, and its source code is not open to the public. This limits customization options and may pose challenges for organizations seeking complete control over their database system.

**POSTGRE SQL**

PostgreSQL is an open-source object-relational database system that has been under active development for over 35 years 1. It is known for its reliability, feature robustness, and performance. PostgreSQL is available for download on various platforms, including Linux, mac OS, Windows, BSD, and Solaris. PostgreSQL is a powerful database management system that supports a wide range of features, including ACID transactions, multi-version concurrency control, table inheritance, foreign keys, triggers, and stored procedures.

**FEATURES OF POSTGRE SQL:**

**Open source-** PostgreSQL is open-source and free to use, allowing users to view, modify, and distribute the source code.

**Acid compliance-** PostgreSQL follows ACID properties (Atomicity, Consistency, Isolation, Durability), ensuring data integrity and reliability.

**Extensibility-** PostgreSQL allows users to define their data types, operators, functions, and aggregates, making it highly extensible.

**Support for complex data types-** Besides standard SQL types, PostgreSQL supports complex data types like arrays, store (key-value pairs), JSON, and XML.

**ADVANTAGES :**

**Open source licensing -** Being open-source, PostgreSQL offers cost savings as there are no licensing fees.

**Extensive document -** PostgreSQL has comprehensive and well-maintained documentation, making it easy for users to learn and troubleshoot.

**Cross-platform compatibility -** PostgreSQL is available for various operating systems, including Linux, Windows, and macOS, ensuring cross-platform compatibility.

**Scalablity -** It is designed to scale horizontally and vertically, making it suitable for both small projects and large, high-traffic applications.

**DISADVANTAGES :**

**Performance**- In certain scenarios, PostgreSQL may have slightly lower performance compared to some other databases, particularly in high-write scenarios.

**Complexity for novice users**- The extensive feature set and configurability can make PostgreSQL seem complex, especially for users new to database administration.

**Learning curve-** Due to its advanced features, there might be a steeper learning curve for users transitioning from simpler database systems.

**Enterprise features in commercial versions-** Some advanced enterprise features are available in the commercial version, and organizations may need to purchase additional support.

**SQL SERVER**



SQL Server, also known as MS SQL Server or Microsoft SQL Server, is a relational database management system (RDBMS). It’s a program that saves database information and runs SQL commands and queries to alter a relational database. Furthermore, it manages and executes all database activities. Microsoft created SQL Server in 1989 for business purposes. It is proprietary software written in the C and C++ programming languages. T-SQL (Transact Structured Query Language) is a different type of SQL that is nearly identical to SQL, with minor differences in query syntax.

**FEATURES OF SQL SERVER :**

**Comprehensive Feature Set -** SQL Server offers a wide range of features, including data warehousing, business intelligence, data analysis, and support for advanced analytics.

**Integration Services (SSIS) -** SQL Server Integration Services (SSIS) allows the creation of data integration and ETL (Extract, Transform, Load) processes.

**Reporting Services (SSRS) -** SQL Server Reporting Services (SSRS) provides tools for creating, managing, and delivering various types of reports.

**Analysis Services (SSAS) -** SQL Server Analysis Services (SSAS) supports online analytical processing (OLAP) and data mining functionalities.

**ADVANTAGES :**

**Integration with Microsoft Ecosystem -** SQL Server seamlessly integrates with other Microsoft products and services, providing a unified environment for development and deployment.

**Ease of Use -** SQL Server Management Studio (SSMS) provides a user-friendly interface for database management, making it relatively easy for administrators and developers to work with.

**Strong Community and Support -** SQL Server has a strong community, and Microsoft provides extensive documentation, support forums, and regular updates.

**Robust Development Tools -** SQL Server comes with powerful development tools, including SQL Server Data Tools (SSDT) for database development.

**DISADVANTAGES :**

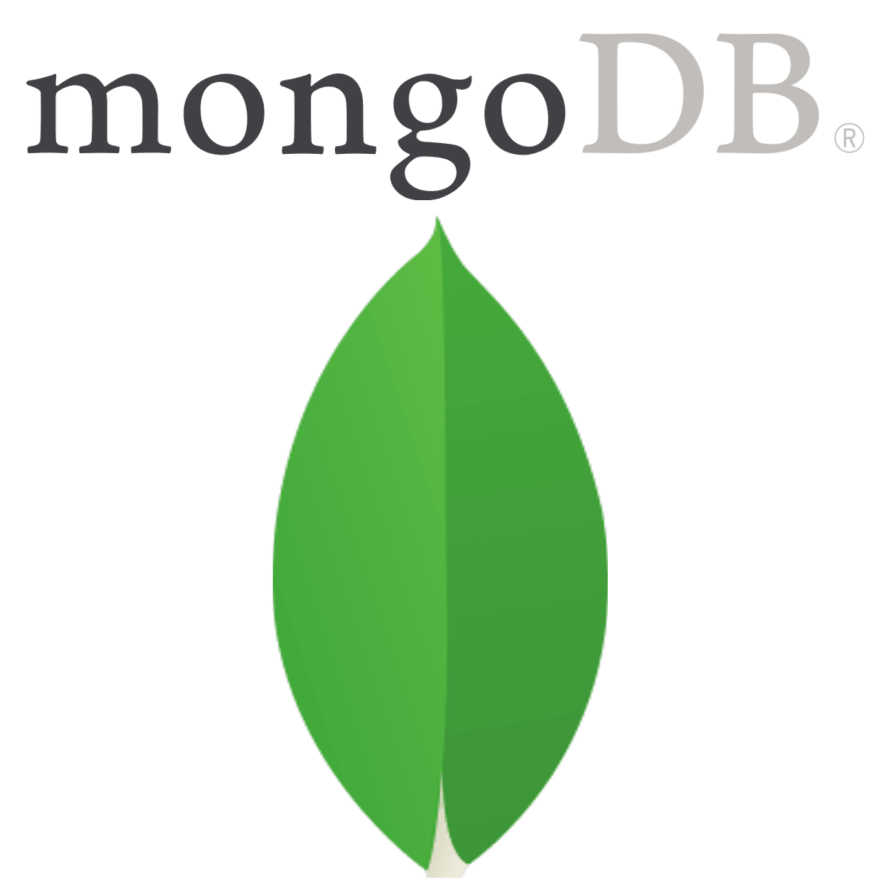
**Licensing Costs -** SQL Server can be relatively expensive, especially for larger enterprises, as it is not open source, and licensing costs can add up.

**Limited Cross-Platform Compatibility -** While there is some level of cross-platform compatibility, SQL Server is primarily designed for the Windows operating system, limiting its use in mixed-platform environments.

**Resource Intensive -** SQL Server can be resource-intensive, especially in terms of memory and disk space, which may impact performance on systems with limited resources.

**Learning Curve for Advanced Features -** Advanced features like Always on Availability Groups and certain business intelligence tools may have a steeper learning curve.

**MONGO DB**

MongoDB is a source-available, cross-platform, document-oriented database program. Classified as a NoSQL database product, MongoDB utilizes JSON-like documents with optional schemas. MongoDB is developed by MongoDB Inc. and current versions are licensed under the Server Side Public License (SSPL). MongoDB is a member of the MACH Alliance. MongoDB handles both organized and unstructured data across diverse applications. It seamlessly integrates with other popular programming languages and adapts to different tech environments without much effort.

**FEATURES OF MongoDB :**

**NoSQL Database -** MongoDB is a NoSQL database, which means it is designed to handle unstructured or semi-structured data, providing greater flexibility than traditional relational databases.

**Document-Oriented -** MongoDB stores data in a flexible, JSON-like format called BSON (Binary JSON), allowing for easy storage of nested structures and arrays.

**Schema-less -** MongoDB is schema-less, meaning there is no need to define a rigid schema beforehand. This allows for dynamic changes to the data structure.

**Horizontal Scalability -** MongoDB can scale horizontally by sharding data across multiple servers, making it suitable for large and distributed systems.

**ADVANTAGES :**

**Flexible Schema -** The schema-less nature of MongoDB allows for easy adaptation to evolving application requirements without the need for a predefined schema.

**Scalability -** MongoDB's horizontal scaling and automatic sharding make it highly scalable, suitable for handling large amounts of data and high traffic.

**Document-Oriented Model -** The document-oriented data model is well-suited for representing complex relationships and nested structures, making it easy to work with real-world data.

**High Performance -** MongoDB is designed for high performance, with features like indexing, query optimization, and horizontal scaling contributing to its efficiency.

**DISADVANTAGES :**

**Lack of Transactions Across Documents -** MongoDB does not support transactions across multiple documents, which may be a limitation in certain scenarios requiring strict transactional consistency.

**Memory Usage -** MongoDB can be memory-intensive, especially when dealing with large datasets, and may require sufficient system resources for optimal performance.

**Learning Curve for Developers New to NoSQL -** Developers accustomed to relational databases may experience a learning curve when adapting to the NoSQL and document-oriented paradigm.

**Not Suitable for All Use Cases -** While MongoDB is well-suited for certain use cases, it may not be the best choice for applications with complex relational data and heavy transactional requirements.

**LEARNING OUTCOMES : -**

* In this experiment , we learnt about different database management systems.
* Learnt about the features and advantages and disadvantages about different DBMS like MySQL , Oracle, SQL server , MongoDB and PostgreSQL.

**Experiment 2**

**Experiment 2:** Implement simple queries of DDL and DML.

**Explanation of the Experiment & Concept:**

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

**SQL Commands:**

* **DDL Commands:**

1. **CREATE:** (USED TO CREATE A NEW TABLE IN THE DATABASE.)

**SQL> CREATE TABLE employee (**

**2  s\_no INT,**

**3  name CHAR(30),**

**4  designation CHAR(30),**

**5  branch CHAR(30) );**

Table created.

**SQL> DESC employee;**

 Name                                Null?    Type

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

 S\_NO                                         NUMBER(38)

 NAME                                          CHAR(30)

 DESIGNATION                                   CHAR(30)

 BRANCH                                        CHAR(30)

**SQL> CREATE TABLE emp AS SELECT \* FROM employee;**

Table created.

**SQL> DESC emp;**

 Name                                Null?    Type

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

 S\_NO                                         NUMBER(38)

 NAME                                          CHAR(30)

 DESIGNATION                                   CHAR(30)

 BRANCH                                        CHAR(30)

1. **DROP:** (USED TO DELETE BOTH THE STRUCTURE AND RECORD STORED IN THE TABLE.)

**SQL> DROP TABLE emp;**

Table dropped.

**SQL> DESC emp;**

ERROR:

ORA-04043: object emp does not exist

1. **ALTER:** (USED TO ALTER THE STRUCTURE OF THE DATABASE.)

**SQL> ALTER TABLE employee ADD salary INT;**

Table altered.

**SQL> DESC employee;**

 Name                                Null?    Type

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

 S\_NO                                         NUMBER(38)

 NAME                                          CHAR(30)

 DESIGNATION                                   CHAR(30)

 BRANCH                                        CHAR(30)

 SALARY                                        NUMBER(38)

**SQL> ALTER TABLE employee RENAME COLUMN branch TO branch\_city;**

Table altered.

**SQL> DESC employee;**

 Name                                 Null?    Type

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

 S\_NO                                          NUMBER(38)

 NAME                                         CHAR(30)

 DESIGNATION                                  CHAR(30)

 BRANCH\_CITY                                   CHAR(30)

 SALARY                                        NUMBER(38)

1. **TRUNCATE:** (USED TO DELETE ALL THE ROWS FROM THE TABLE.)

**SQL> SELECT \* FROM employee;**

  S\_NO NAME              DESIGNATION          BRANCH\_CITY

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

     1 ABC               Manager              Chennai

     2 ABD               Designer            Chennai

     3 BCG               Analyst              Delhi

     4 HEF               Assistant            Delhi

     5 GHI               Supervisor           Madurai

**SQL> TRUNCATE TABLE employee;**

Table truncated.

**SQL> SELECT \* FROM employee;**

no rows selected

* **DML Commands:**

1. **INSERT:** (USED TO INSERT DATA INTO THE ROW OF A TABLE.)

**SQL> INSERT INTO employee VALUES(1, 'ABC', 'Manager', 'Chennai');**

1 row created.

**SQL> INSERT INTO employee VALUES(2, 'ABD', 'Designer', 'Chennai');**

1 row created.

**SQL> INSERT INTO employee VALUES(3, 'BCG', 'Analyst', 'Delhi');**

1 row created.

**SQL> INSERT INTO employee VALUES(4, 'HEF', 'Assistant', 'Delhi');**

1 row created.

**SQL> INSERT INTO employee VALUES(5, 'GHI', 'Supervisor', 'Madurai');**

1 row created.

**SQL> SELECT \* FROM employee;**

  S\_NO NAME              DESIGNATION          BRANCH\_CITY

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

     1 ABC               Manager              Chennai

     2 ABD               Designer            Chennai

     3 BCG               Analyst              Delhi

     4 HEF               Assistant            Delhi

     5 GHI               Supervisor           Madurai

1. **UPDATE:** (USED TO UPDATE OR MODIFY THE VALUE OF A COLUMN IN THE TABLE.)

**SQL> SELECT \* FROM employee;**

  S\_NO NAME              DESIGNATION          BRANCH\_CITY

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

     1 ABC               Manager              Chennai

     2 ABD               Designer            Chennai

     3 BCG               Analyst              Delhi

     4 HEF               Assistant            Delhi

     5 GHI               Supervisor           Madurai

**SQL> UPDATE employee**

**2  SET designation = 'Senior Manager'**

**3  WHERE s\_no = 1;**

1 row updated.

**SQL> SELECT \* FROM employee;**

  S\_NO NAME              DESIGNATION          BRANCH\_CITY

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

     1 ABC               Senior Manager       Chennai

     2 ABD               Designer            Chennai

     3 BCG               Analyst              Delhi

     4 HEF               Assistant            Delhi

     5 GHI               Supervisor           Madurai

1. **DELETE:** (USED TO REMOVE ONE OR MORE ROW FROM A TABLE.)

**SQL> DELETE FROM employee WHERE name = 'ABD';**

1 row deleted.

**SQL> SELECT \* FROM employee;**

  S\_NO NAME              DESIGNATION          BRANCH\_CITY

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

     1 ABC               Senior Manager       Chennai

     3 BCG               Analyst              Delhi

     4 HEF               Assistant            Delhi

     5 GHI               Supervisor           Madurai

1. **SELECT:** (USED TO RETRIEVE DATA FROM THE DATABASE.)

**SQL> SELECT \* FROM employee;**

  S\_NO NAME              DESIGNATION          BRANCH\_CITY

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

     1 ABC               Senior Manager       Chennai

     3 BCG               Analyst              Delhi

     4 HEF               Assistant            Delhi

     5 GHI               Supervisor           Madurai

**SQL> SELECT s\_no, name, designation FROM EMPLOYEE;**

  S\_NO NAME              DESIGNATION

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

     1 ABC               Senior Manager

     3 BCG               Analyst

     4 HEF               Assistant

     5 GHI               Supervisor

**Learning outcome of the Experiment:**

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |

**Experiment 3**

**Experiment 3:** Implement basic queries to Create, Insert, Update, Delete and Select Statements for two different scenarios (For instance: College, Bank).

**Explanation of the Experiment & Concept:**

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

**SQL Commands:**

* **For College:**

1. **Creating tables with constraints:**
   1. **faculty (fid, name, *dno*, sal, address, phone, dob, exp)**
   2. **dept (dno, dname, budget)**
   3. **student (sid, name, *dno*, phone, dob, address, sem)**
   4. **society (sc\_id, name)**
   5. **std\_soc(*sid*, *sc\_id*)**
   6. **course (cid, cname)**
   7. **fac\_course (*fid*, *cid*)**

**SQL> CREATE TABLE faculty (**

**2  fid INT NOT NULL PRIMARY KEY,**

**3  name VARCHAR(20) NOT NULL,**

**4  dno INT,**

**5  sal INT,**

**6  address VARCHAR(50),**

**7  phone INT,**

**8  dob DATE,**

**9  exp INT);**

Table created.

**SQL> CREATE TABLE dept(**

**2  dno INT NOT NULL PRIMARY KEY,**

**3  dname VARCHAR(20) NOT NULl,**

**4  budget INT);**

Table created.

**SQL> ALTER TABLE faculty ADD FOREIGN KEY (dno) REFERENCES dept(dno);**

Table altered.

**SQL> CREATE TABLE student (**

**2  sid INT NOT NULL PRIMARY KEY,**

**3  name VARCHAR(20),**

**4  dno INT NOT NULL,**

**5  phone INT,**

**6  dob DATE,**

**7  address VARCHAR(50),**

**8  sem INT);**

Table created.

**SQL> ALTER TABLE STUDENT ADD FOREIGN KEY (dno) REFERENCES dept(dno);**

Table altered.

**SQL> CREATE TABLE society (**

**2  sc\_id INT NOT NULL PRIMARY KEY,**

**3  name VARCHAR(20) NOT NULL);**

Table created.

**SQL> CREATE TABLE course (**

**2  cid INT NOT NULL PRIMARY KEY,**

**3  cname VARCHAR(20) NOT NULL);**

Table created.

**SQL> CREATE TABLE std\_soc (**

**2  sid INT NOT NULL,**

**3  sc\_id INT NOT NULL,**

**4  PRIMARY KEY (sid, sc\_id),**

**5  FOREIGN KEY (sid) REFERENCES student (sid),**

**6  FOREIGN KEY (sc\_id) REFERENCES society (sc\_id) );**

Table created.

**SQL> CREATE TABLE fac\_course (**

**2  fid INT NOT NULL,**

**3  cid INT NOT NULL,**

**4  PRIMARY KEY (fid, cid),**

**5  FOREIGN KEY (fid) REFERENCES faculty (fid),**

**6  FOREIGN KEY (cid) REFERENCES course (cid) );**

Table created.

**SQL> DESC faculty;**

 Name                              Null?    Type

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

 FID                               NOT NULL NUMBER(38)

 NAME                              NOT NULL VARCHAR2(20)

 DNO                                        NUMBER(38)

 SAL                                        NUMBER(38)

 ADDRESS                                    VARCHAR2(50)

 PHONE                                      NUMBER(38)

 DOB                                        DATE

 EXP                                        NUMBER(38)

**SQL> DESC dept;**

 Name                              Null?    Type

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

 DNO                               NOT NULL NUMBER(38)

 DNAME                             NOT NULL VARCHAR2(20)

 BUDGET                                     NUMBER(38)

**SQL> DESC student;**

 Name                              Null?    Type

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

 SID                               NOT NULL NUMBER(38)

 NAME                                       VARCHAR2(20)

 DNO                               NOT NULL NUMBER(38)

 PHONE                                      NUMBER(38)

 DOB                                        DATE

 ADDRESS                                    VARCHAR2(50)

 SEM                                        NUMBER(38)

**SQL> DESC society;**

 Name                              Null?    Type

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

 SC\_ID                             NOT NULL NUMBER(38)

 NAME                              NOT NULL VARCHAR2(20)

**SQL> DESC std\_soc;**

 Name                              Null?    Type

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

 SID                               NOT NULL NUMBER(38)

 SC\_ID                             NOT NULL NUMBER(38)

**SQL> DESC course;**

 Name                              Null?    Type

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

 CID                               NOT NULL NUMBER(38)

 CNAME                             NOT NULL VARCHAR2(20)

**SQL> DESC fac\_course;**

 Name                              Null?    Type

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

 FID                               NOT NULL NUMBER(38)

 CID                               NOT NULL NUMBER(38)

1. **Add Column no\_of\_awards to faculty table and update values in column:**

**SQL> ALTER TABLE faculty ADD no\_of\_awards INT;**

Table altered.

**SQL> DESC faculty;**

 Name                              Null?    Type

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

 FID                               NOT NULL NUMBER(38)

 NAME                              NOT NULL VARCHAR2(20)

 DNO                                        NUMBER(38)

 SAL                                        NUMBER(38)

 ADDRESS                                    VARCHAR2(50)

 PHONE                                      NUMBER(38)

 DOB                                        DATE

 EXP                                        NUMBER(38)

 NO\_OF\_AWARDS                               NUMBER(38)

1. **Insert rows in each table:**

**SQL> INSERT INTO dept VALUES (1, 'CSE', 10000);**

1 row created.

**SQL> INSERT INTO dept VALUES (2, 'AIDS', 12000);**

1 row created.

**SQL> INSERT INTO dept VALUES (3, 'AIML', 9000);**

1 row created.

**SQL> INSERT INTO dept VALUES (4, 'IIOT', 8000);**

1 row created.

**SQL> INSERT INTO dept VALUES (5, 'VSLI', 6000);**

1 row created.

**SQL> INSERT INTO society VALUES (1, 'Kalakriti');**

1 row created.

**SQL> INSERT INTO society VALUES (2, 'Srijan');**

1 row created.

**SQL> INSERT INTO society VALUES (3, 'EDC Cell');**

1 row created.

**SQL> INSERT INTO society VALUES (4, 'NSS');**

1 row created.

**SQL> INSERT INTO society VALUES (5, 'NCC');**

1 row created.

**SQL> INSERT INTO course VALUES (1, 'Computational Method');**

1 row created.

**SQL> INSERT INTO course VALUES (2, 'Software Engineering');**

1 row created.

**SQL> INSERT INTO course VALUES (3, 'Fundamentals of ML');**

1 row created.

**SQL> INSERT INTO course VALUES (4, 'OOPs');**

1 row created.

**SQL> INSERT INTO course VALUES (5, 'DBMS');**

1 row created.

**SQL> INSERT INTO faculty VALUES (1, 'Faculty1', 1, 11000, 'Block 1 Delhi', 9876543210, '1990-01-05', 3, 4);**

1 row created.

**SQL> INSERT INTO faculty VALUES (2, 'Faculty2', 1, 9000, 'Block 1 Sector 3 Delhi', 9876546330, '1992-10-15', 2, 2);**

1 row created.

**SQL> INSERT INTO faculty VALUES (3, 'Faculty3', 2, 12000, 'Block 2 Sector 1 Delhi', 8446546330, '1995-06-11', 0, 1);**

1 row created.

**SQL> INSERT INTO faculty VALUES (4, 'Faculty4', 2, 16000, 'Block 2 Sector 2 Delhi', 7925246330, '1990-08-11', 3,5);**

1 row created.

**SQL> INSERT INTO faculty VALUES (5, 'Faculty5', 3, 8000, 'Block 1 Sector 2 Delhi', 8435246362, '1993-11-21', 1,3);**

1 row created.

**SQL> INSERT INTO student VALUES (1, 'Student1', 2, 9876543212, '2004-01-05', 'Block 5 Delhi', 4);**

1 row created.

**SQL> INSERT INTO student VALUES (2, 'Student2', 3, 8765432167, '2002-10-25', 'Block 2 Sector 1 Delhi', 2);**

1 row created.

**SQL> INSERT INTO student VALUES (3, 'Student3', 2, 8765787655, '2002-05-11', 'Block 4 Sector 1 Delhi', 2);**

1 row created.

**SQL> INSERT INTO student VALUES (4, 'Student4', 4, 7898765456, '2004-07-19', 'Block 2 Sector 4 Delhi', 6);**

1 row created.

**SQL> INSERT INTO student VALUES (5, 'Student5', 1, NULL, '2004-02-29', 'Block 1 Sector 2 Delhi', 4);**

1 row created.

**SQL> INSERT INTO std\_soc VALUES (1,2);**

1 row created.

**SQL> INSERT INTO std\_soc VALUES (3,2);**

1 row created.

**SQL> INSERT INTO std\_soc VALUES (4,1);**

1 row created.

**SQL> INSERT INTO std\_soc VALUES (2,1);**

1 row created.

**SQL> INSERT INTO std\_soc VALUES (2,4);**

1 row created.

**SQL> INSERT INTO fac\_course VALUES (1,2);**

1 row created.

**SQL> INSERT INTO fac\_course VALUES (2,2);**

1 row created.

**SQL> INSERT INTO fac\_course VALUES (3,1);**

1 row created.

**SQL> INSERT INTO fac\_course VALUES (4,4);**

1 row created.

**SQL> INSERT INTO fac\_course VALUES (5,3);**

1 row created.

1. **Showing the content of all the table:**

**SQL> SELECT \* FROM faculty;**

 FID NAME       DNO    SAL ADDRESS                       PHONE DOB         EXP NO\_OF\_AWARDS

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

   1 Faculty1     1  11000 Block 1 Delhi            9876543210 1990-01-05    3            4

   2 Faculty2     1   9000 Block 1 Sector 3 Delhi   9876546330 1992-10-15    2            2

   3 Faculty3     2  12000 Block 2 Sector 1 Delhi   8446546330 1995-06-11    0            1

   4 Faculty4     2  16000 Block 2 Sector 2 Delhi   7925246330 1990-08-11    3            5

   5 Faculty5     3   8000 Block 1 Sector 2 Delhi   8435246362 1993-11-21    1            3

**SQL> SELECT \* FROM dept;**

 DNO DNAME       BUDGET

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

   1 CSE          10000

   2 AIDS         12000

   3 AIML          9000

   4 IIOT          8000

   5 VSLI          6000

**SQL> SELECT \* FROM student;**

 SID NAME        DNO      PHONE DOB        ADDRESS                    SEM

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

   1 Student1      2 9876543212 2004-01-05 Block 5 Delhi                4

   2 Student2      3 8765432167 2002-10-25 Block 2 Sector 1 Delhi       2

   3 Student3      2 8765787655 2002-05-11 Block 4 Sector 1 Delhi       2

   4 Student4      4 7898765456 2004-07-19 Block 2 Sector 4 Delhi       6

   5 Student5      1            2004-02-29 Block 1 Sector 2 Delhi       4

**SQL> SELECT \* FROM society;**

     SC\_ID NAME

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

         1 Kalakriti

         2 Srijan

         3 EDC Cell

         4 NSS

         5 NCC

**SQL> SELECT \* FROM std\_soc;**

       SID      SC\_ID

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

         1          2

         3          2

         4          1

         2          1

         2          4

**SQL> SELECT \* FROM course;**

       CID CNAME

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

         1 Computational Method

         2 Software Engineering

         3 Fundamentals of ML

         4 OOPs

         5 DBMS

**SQL> SELECT \* FROM fac\_course;**

       FID        CID

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

         1          2

         2          2

         3          1

         4          4

         5          3

1. **Retrieve the faculties with experience greater than 1 year and salary more than 10000:**

**SQL> SELECT \* FROM faculty WHERE exp > 1 AND sal > 10000;**

 FID NAME       DNO    SAL ADDRESS                       PHONE DOB         EXP NO\_OF\_AWARDS

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

   1 Faculty1     1  11000 Block 1 Delhi            9876543210 1990-01-05    3            4

   4 Faculty4     2  16000 Block 2 Sector 2 Delhi   7925246330 1990-08-11    3            5

1. **Retrieve the students enrolled in department number 2 or semester 4:**

**SQL> SELECT \* FROM student WHERE dno = 2 OR sem = 4;**

 SID NAME        DNO      PHONE DOB        ADDRESS                    SEM

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

   1 Student1      2 9876543212 2004-01-05 Block 5 Delhi                4

   3 Student3      2 8765787655 2002-05-11 Block 4 Sector 1 Delhi       2

1. **Increase the faculty salary whose department number is 1:**

**SQL> -- Salary of faculties before updating:**

**SQL> SELECT name, sal FROM faculty WHERE dno = 1;**

NAME                        SAL

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

Faculty1                  11000

Faculty2                   9000

**SQL> -- Increasing the salary by 2000 of faculties in department no. 1**

**SQL> UPDATE faculty**

**2  SET sal = sal + 2000**

**3  WHERE dno = 1;**

**SQL> -- Salary of faculties after updating:**

**SQL> SELECT name, sal FROM faculty WHERE dno = 1;**

NAME                        SAL

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

Faculty1                  13000

Faculty2                  11000

**SQL> -- Budget of CSE department before updating:**

**SQL> SELECT dname, budget FROM dept WHERE dname = 'CSE';**

DNAME                    BUDGET

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

CSE                       10000

1. **Increase the budget of CSE department by 1000:**

**SQL> -- Increasing the budget by 1000 of CSE department:**

**SQL> UPDATE dept**

**2  SET budget = budget + 1000**

**3  WHERE dname = 'CSE';**

**SQL> -- budget of CSE department after updating:**

**SQL> SELECT dname, budget FROM dept WHERE dname = 'CSE';**

DNAME                    BUDGET

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

CSE                       11000

**SQL> -- Phone no. of student with id no. 2 before updating:**

**SQL> SELECT sid, name, phone FROM student WHERE sid = 2;**

       SID NAME                      PHONE

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

         2 Student2             8765432167

1. **Update the phone number of student ID 2:**

**SQL> -- Updating the phone number:**

**SQL> UPDATE student**

**2  SET phone = 7654568654**

**3  WHERE sid = 2;**

**SQL> -- Phone no. of student with id no. 2 after updating:**

**SQL> SELECT sid, name, phone FROM student WHERE sid = 2;**

       SID NAME                      PHONE

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

         2 Student2             7654568654

1. **Delete the society name Kalakriti :  
   (***See if it allows provided there are students enrolled***) :**

**SQL> DELETE FROM society WHERE name = 'Kalakriti';**DELETE FROM society WHERE name = 'Kalakriti'

\*

ERROR at line 1:

ORA-02292: integrity constraint (ROHIT.SYS\_C008373) violated - child record found

**SQL> SELECT \* FROM society;**

     SC\_ID NAME

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

         1 Kalakriti

         2 Srijan

         3 EDC Cell

         4 NSS

         5 NCC

**SQL> -- SQL doesnt allow deleting as Students are enrolled in the given society:**

**SQL> SELECT \* FROM std\_soc;**

       SID      SC\_ID

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

         1          2

         3          2

         4          1

         2          1

         2          4

1. **Delete the student with no phone number:**

**SQL> DELETE FROM student WHERE phone IS NULL;**

**SQL> SELECT \* FROM student;**

 SID NAME       DNO      PHONE DOB        ADDRESS                    SEM

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

   1 Student1     2 9876543212 2004-01-05 Block 5 Delhi                4

   2 Student2     3 7654568654 2002-10-25 Block 2 Sector 1 Delhi       2

   3 Student3     2 8765787655 2002-05-11 Block 4 Sector 1 Delhi       2

   4 Student4     4 7898765456 2004-07-19 Block 2 Sector 4 Delhi       6

* **For Bank:**

1. **Creating tables with constraints:**
   1. **acc (acc\_no, name, phone)**
   2. **trans (*acc\_no*, c\_d, amount)**
   3. **loan (*acc\_no*, interest, time, type)**
   4. **fd (*acc\_no,* amount, interest, time)**
   5. **locker (*acc\_no*, l\_no)**

**SQL> CREATE TABLE acc (**

**2  acc\_no INT NOT NULL,**

**3  name VARCHAR(20) NOT NULL,**

**4  phone INT,**

**5  PRIMARY KEY(acc\_no));**

Table created.

**SQL> CREATE TABLE trans (**

**2  acc\_no INT NOT NULL,**

**3  c\_d CHAR(1) CHECK ( c\_d='C' OR c\_d='D'),**

**4  amount INT,**

**5  FOREIGN KEY (acc\_no) REFERENCES acc(acc\_no));**

Table created.

**SQL> CREATE TABLE loan (**

**2  acc\_no INT NOT NULL,**

**3  interest FLOAT,**

**4  time INT,**

**5  type VARCHAR(20),**

**6  FOREIGN KEY (acc\_no) REFERENCES acc(acc\_no));**

Table created.

**SQL> CREATE TABLE fd (**

**2  acc\_no INT NOT NULL,**

**3  amount INT,**

**4  interest FLOAT,**

**5  time INT,**

**6  FOREIGN KEY (acc\_no) REFERENCES acc(acc\_no));**

Table created.

**SQL> CREATE TABLE locker (**

**2  acc\_no INT NOT NULL,**

**3  l\_no INT,**

**4  FOREIGN KEY (acc\_no) REFERENCES acc(acc\_no));**

Table created.

**SQL> DESC acc;**

 Name                               Null?    Type

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

 ACC\_NO                             NOT NULL NUMBER(38)

 NAME                              NOT NULL VARCHAR2(20)

 PHONE                                      NUMBER(38)

**SQL> DESC trans;**

 Name                               Null?    Type

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

 ACC\_NO                             NOT NULL NUMBER(38)

 C\_D                                         CHAR(1)

 AMOUNT                                      NUMBER(38)

**SQL> DESC loan;**

 Name                              Null?    Type

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

 ACC\_NO                             NOT NULL NUMBER(38)

 INTEREST                                   FLOAT(126)

 TIME                                        NUMBER(38)

 TYPE                                       VARCHAR2(20)

**SQL> DESC fd;**

 Name                              Null?    Type

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

 ACC\_NO                            NOT NULL NUMBER(38)

 AMOUNT                                     NUMBER(38)

 INTEREST                                   FLOAT(126)

 TIME                                       NUMBER(38)

**SQL> DESC locker;**

 Name                               Null?    Type

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

 ACC\_NO                             NOT NULL NUMBER(38)

 L\_NO                                       NUMBER(38)

1. **Insert values in each table & show contents:**

**SQL> INSERT INTO acc VALUES (1, 'name1', 9876543223);**

1 row created.

**SQL> INSERT INTO acc VALUES (2, 'name2', 8362543223);**

1 row created.

**SQL> INSERT INTO acc VALUES (3, 'name3', 7834243647);**

1 row created.

**SQL> INSERT INTO acc VALUES (4, 'name4', 9544435647);**

1 row created.

**SQL> INSERT INTO acc VALUES (5, 'name5', 7989239878);**

1 row created.

**SQL> INSERT INTO trans VALUES (2, 'C', 2000);**

1 row created.

**SQL> INSERT INTO trans VALUES (5, 'D', 100);**

1 row created.

**SQL> INSERT INTO trans VALUES (2, 'C', 8000);**

1 row created.

**SQL> INSERT INTO trans VALUES (3, 'D', 1500);**

1 row created.

**SQL> INSERT INTO trans VALUES (2, 'D', 10000);**

1 row created.

**SQL> INSERT INTO trans VALUES (1, 'C', 400);**

1 row created.

**SQL> INSERT INTO trans VALUES (1, 'C', 3000);**

1 row created.

**SQL> INSERT INTO trans VALUES (4, 'D', 55000);**

1 row created.

**SQL> INSERT INTO loan VALUES (4, 3.5, 18, 'education');**

1 row created.

**SQL> INSERT INTO loan VALUES (2, 10.0, 120, 'personal');**

1 row created.

**SQL> INSERT INTO loan VALUES (3, 6.5, 36, 'car');**

1 row created.

**SQL> INSERT INTO fd VALUES (1, 10000, 6.5, 36);**

1 row created.

**SQL> INSERT INTO fd VALUES (2, 12000, 7.5, 30);**

1 row created.

**SQL> INSERT INTO fd VALUES (5, 20000, 7.5, 48);**

1 row created.

**SQL> INSERT INTO locker VALUES (1, 101);**

1 row created.

**SQL> INSERT INTO locker VALUES (2, 102);**

1 row created.

**SQL> INSERT INTO locker VALUES (4, 203);**

1 row created.

**SQL> INSERT INTO locker VALUES (5, 201);**

1 row created.

**SQL> SELECT \* FROM acc;**

    ACC\_NO NAME                      PHONE

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

         1 name1                9876543223

         2 name2                8362543223

         3 name3                7834243647

         4 name4                9544435647

         5 name5                7989239878

6 rows selected.

**SQL> SELECT \* FROM trans;**

    ACC\_NO C     AMOUNT

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

         2 C       2000

         5 D        100

         2 C       8000

         3 D       1500

         2 D      10000

         1 C        400

         1 C       3000

         4 D      55000

8 rows selected.

**SQL> SELECT \* FROM loan;**

    ACC\_NO   INTEREST       TIME TYPE

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

         4        3.5         18 education

         2         10        120 personal

         3        6.5         36 car

**SQL> SELECT \* FROM fd;**

    ACC\_NO     AMOUNT   INTEREST       TIME

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

         1      10000        6.5         36

         2      12000        7.5         30

         5      20000        7.5         48

**SQL> SELECT \* FROM locker;**

    ACC\_NO       L\_NO

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

         1        101

         2        102

         4        203

         5        201

1. **Add a column ‘loan\_no’ in loan table and update values for column:**

**SQL> ALTER TABLE loan ADD loan\_no INT;**

Table altered.

**SQL> DESC loan;**

 Name                              Null?    Type

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

 ACC\_NO                            NOT NULL NUMBER(38)

 INTEREST                                    FLOAT(126)

 TIME                                        NUMBER(38)

 TYPE                                        VARCHAR2(20)

 LOAN\_NO                                    NUMBER(38)

**SQL> UPDATE loan**

**2  SET loan\_no = 1**

**3  WHERE acc\_no = 4;**

1 row updated.

**SQL> UPDATE loan**

**2  SET loan\_no = 2**

**3  WHERE acc\_no = 2;**

1 row updated.

**SQL> UPDATE loan**

**2  SET loan\_no = 3**

**3  WHERE acc\_no = 3;**

1 row updated.

**SQL> SELECT \* FROM loan;**

    ACC\_NO   INTEREST       TIME TYPE                    LOAN\_NO

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

         4        3.5         18 education                     1

         2         10        120 personal                      2

         3        6.5         36 car                           3

1. **Insert a row in acc table without phone number:**

**SQL> INSERT INTO acc VALUES (6, 'name6', NULL);**

1 row created.

**SQL> SELECT \* FROM acc;**

    ACC\_NO NAME                      PHONE

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

         1 name1                9876543223

         2 name2                8362543223

         3 name3                7834243647

         4 name4                9544435647

         5 name5                7989239878

         6 name6

1. **Delete account with no phone number:**

**SQL> DELETE FROM acc WHERE phone IS NULL;**

1 row deleted.

**SQL> SELECT \* FROM acc;**

    ACC\_NO NAME                      PHONE

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

         1 name1                9876543223

         2 name2                8362543223

         3 name3                7834243647

         4 name4                9544435647

         5 name5                7989239878

**Learning outcome of the Experiment:**

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |

**Experiment 4**

**Experiment 4:** Implement queries including various functions- mathematical, string, date etc.

**Explanation of the Experiment & Concept:**

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

**SQL Commands:**

1. **Creating table and adding values with constraints:**
   1. **faculty(f\_id, name, dob, exp, no\_of\_awards, d\_no, sal, address, phone)**

**SQL> CREATE TABLE faculty (**

**2  f\_id INT PRIMARY KEY,**

**3  name VARCHAR(20),**

**4  dob DATE,**

**5  exp INT,**

**6  no\_of\_awards INT,**

**7  d\_no INT,**

**8  sal INT,**

**9  address VARCHAR(30),**

**10  phone INT);**

Table created.

**SQL> DESC faculty;**

 Name              Null?    Type

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

 F\_ID              NOT NULL NUMBER(38)

 NAME                       VARCHAR2(20)

 DOB                        DATE

 EXP                        NUMBER(38)

 NO\_OF\_AWARDS               NUMBER(38)

 D\_NO                       NUMBER(38)

 SAL                        NUMBER(38)

 ADDRESS                    VARCHAR2(30)

 PHONE                      NUMBER(38)

**SQL> INSERT INTO faculty VALUES (1, 'fac1', '12-MAR-1992', 6, 3, 1, 60000, 'dwarka', 9876567871);**

1 row created.

**SQL> INSERT INTO faculty VALUES (2, 'fac2', '23-MAR-1995', 3, 2, 1, 50000, 'sector 1 delhi', 9327859812);**

1 row created.

**SQL> INSERT INTO faculty VALUES (3, 'fac3', '25-FEB-1990', 7, 5, 1, 65000, 'sector 2 delhi', 9327842812);**

1 row created.

**SQL> INSERT INTO faculty VALUES (4, 'fac4', '14-AUG-1997', 2, 2, 2, 40000, 'sector 1 delhi', 9352845312);**

1 row created.

**SQL> INSERT INTO faculty VALUES (5, 'fac5', '12-SEP-1996', 2, 1, 2, 35000, 'sector 4 delhi', 8562335415);**

1 row created.

**SQL> INSERT INTO faculty VALUES (6, 'fac6', '10-OCT-1989', 6, 3, 2, 60000, 'sector 2 delhi', 8542345215);**

1 row created.

**SQL> INSERT INTO faculty VALUES (7, 'fac7', '17-MAR-1992', 4, 2, 3, 25000, 'sector 1 delhi', 7923345235);**

1 row created.

**SQL> INSERT INTO faculty VALUES (8, 'fac8', '16-JAN-1994', 2, 1, 3, 28000, 'sector 4 delhi', 7923345433);**

1 row created.

**SQL> INSERT INTO faculty VALUES (9, 'fac9', '06-DEC-1990', 4, 2, 3, 24000, 'sector 3 delhi', 8564345733);**

1 row created.

**SQL> INSERT INTO faculty VALUES (10, 'fac10', '19-DEC-1990', 4, 4, 4, 87000, 'sector 2 delhi', 9939932231);**

1 row created.

**SQL> INSERT INTO faculty VALUES (11, 'fac11', '31-DEC-1991', 3, 1, 4, 85000, 'sector 5 delhi', 9559935535);**

1 row created.

**SQL> INSERT INTO faculty VALUES (12, 'fac12', '11-FEB-1989', 5, 3, 4, 90000, 'sector 1 delhi', 8845365535);**

1 row created.

**SQL> SELECT \* FROM faculty;**

 F\_ID NAME   DOB        EXP NO\_OF\_AWARDS  D\_NO    SAL ADDRESS              PHONE

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

    1 fac1   12-MAR-92    6            3     1  60000 dwarka          9876567871

    2 fac2   23-MAR-95    3            2     1  50000 sector 1 delhi  9327859812

    3 fac3   25-FEB-90    7            5     1  65000 sector 2 delhi  9327842812

    4 fac4   14-AUG-97    2            2     2  40000 sector 1 delhi  9352845312

    5 fac5   12-SEP-96    2            1     2  35000 sector 4 delhi  8562335415

    6 fac6   10-OCT-89    6            3     2  60000 sector 2 delhi  8542345215

    7 fac7   17-MAR-92    4            2     3  25000 sector 1 delhi  7923345235

    8 fac8   16-JAN-94    2            1     3  28000 sector 4 delhi  7923345433

    9 fac9   06-DEC-90    4            2     3  24000 sector 3 delhi  8564345733

   10 fac10  19-DEC-90    4            4     4  87000 sector 2 delhi  9939932231

   11 fac11  31-DEC-91    3            1     4  85000 sector 5 delhi  9559935535

   12 fac12  11-FEB-89    5            3     4  90000 sector 1 delhi  8845365535

12 rows selected.

1. **Retrieve the sum of salary of all faculties**

**SQL> SELECT SUM(sal) FROM faculty;**

  SUM(SAL)

----------

    649000

1. **Retrieve the sum of salary of all faculties of department number 1**

**SQL> SELECT SUM(sal) FROM faculty WHERE d\_no=1;**

  SUM(SAL)

----------

    175000

1. **Retrieve avg, min, max, salary of all faculties:**

**SQL> SELECT AVG(sal), MIN(sal), MAX(sal) FROM faculty;**

  AVG(SAL)   MIN(SAL)   MAX(SAL)

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

54083.3333      24000      90000

1. **Retrieve avg, min, max, salary of department number 4:**

**SQL> SELECT AVG(sal), MIN(sal), MAX(sal) FROM faculty WHERE d\_no=4;**

  AVG(SAL)   MIN(SAL)   MAX(SAL)

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

87333.3333      85000      90000

1. **Retrieve the average experience of faculties of department number 1**

**SQL> SELECT AVG(exp) FROM faculty WHERE d\_no=1;**

  AVG(EXP)

----------

5.33333333

1. **Retrieve the number of faculties in college**

**SQL> SELECT COUNT(f\_id) FROM faculty;**

COUNT(F\_ID)

-----------

         12

1. **Retrieve the number of faculties in dept no 2**

**SQL> SELECT COUNT(f\_id) FROM faculty WHERE d\_no=2;**

COUNT(F\_ID)

-----------

          3

1. **Demonstrate the use of : ASCII, abs, concat, subs, trim, upper, lower, cos, sin, tan, log, power, round, floor, ceil, sqrt.**

**SQL> SELECT ASCII(name) FROM faculty;**

ASCII(NAME)

-----------

        102

        102

        102

        102

        102

        102

        102

        102

        102

        102

        102

        102

12 rows selected.

**SQL> SELECT abs(sal) FROM faculty;**

  ABS(SAL)

----------

     60000

     50000

     65000

     40000

     35000

     60000

     25000

     28000

     24000

     87000

     85000

     90000

12 rows selected.

**SQL> SELECT name || ' with ' || exp || ' years of experience, is in department ' || d\_no AS faculty\_info FROM faculty;**

FACULTY\_INFO

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

fac1 with 6 years of experience, is in department 1

fac2 with 3 years of experience, is in department 1

fac3 with 7 years of experience, is in department 1

fac4 with 2 years of experience, is in department 2

fac5 with 2 years of experience, is in department 2

fac6 with 6 years of experience, is in department 2

fac7 with 4 years of experience, is in department 3

fac8 with 2 years of experience, is in department 3

fac9 with 4 years of experience, is in department 3

fac10 with 4 years of experience, is in department 4

fac11 with 3 years of experience, is in department 4

fac12 with 5 years of experience, is in department 4

12 rows selected.

**SQL> SELECT SUBSTR(address, 1, 8) AS sector FROM faculty;**

SECTOR

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

dwarka

sector 1

sector 2

sector 1

sector 4

sector 2

sector 1

sector 4

sector 3

sector 2

sector 5

sector 1

12 rows selected.

**SQL> SELECT TRIM(address) FROM faculty;**

TRIM(ADDRESS)

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

dwarka

sector 1 delhi

sector 2 delhi

sector 1 delhi

sector 4 delhi

sector 2 delhi

sector 1 delhi

sector 4 delhi

sector 3 delhi

sector 2 delhi

sector 5 delhi

sector 1 delhi

12 rows selected.

**SQL> SELECT UPPER(name) FROM faculty;**

UPPER(NAME)

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

FAC1

FAC2

FAC3

FAC4

FAC5

FAC6

FAC7

FAC8

FAC9

FAC10

FAC11

FAC12

12 rows selected.

**SQL> SELECT LOWER(name) FROM faculty;**

LOWER(NAME)

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

fac1

fac2

fac3

fac4

fac5

fac6

fac7

fac8

fac9

fac10

fac11

fac12

12 rows selected.

**SQL> SELECT COS(num) FROM numbers;**

  COS(NUM)

----------

         1

.540302306

-.41614684

 -.9899925

-.65364362

.283662185

.960170287

.753902254

8 rows selected.

**SQL> SELECT SIN(num) FROM numbers;**

  SIN(NUM)

----------

         0

.841470985

.909297427

.141120008

 -.7568025

-.95892427

 -.2794155

.656986599

8 rows selected.

**SQL> SELECT TAN(num) FROM numbers;**

  TAN(NUM)

----------

         0

1.55740772

-2.1850399

-.14254654

1.15782128

 -3.380515

-.29100619

.871447983

8 rows selected.

**SQL> SELECT LOG(num, 2) FROM numbers WHERE num > 2;**

LOG(NUM,2)

----------

.630929754

        .5

.430676558

.386852807

.356207187

**SQL> SELECT POWER(num, 2) FROM numbers;**

POWER(NUM,2)

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

           0

           1

           4

           9

          16

          25

          36

          49

8 rows selected.

**SQL> SELECT ROUND(num, -1) FROM numbers;**

ROUND(NUM,-1)

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

            0

            0

            0

            0

            0

           10

           10

           10

8 rows selected.

**SQL> SELECT FLOOR(num) FROM numbers;**

FLOOR(NUM)

----------

         0

         1

         2

         3

         4

         5

         6

         7

8 rows selected.

**SQL> SELECT CEIL(num) FROM numbers;**

 CEIL(NUM)

----------

         0

         1

         2

         3

         4

         5

         6

         7

8 rows selected.

**SQL> SELECT SQRT(num) FROM numbers;**

 SQRT(NUM)

----------

         0

         1

1.41421356

1.73205081

         2

2.23606798

2.44948974

2.64575131

8 rows selected.

1. **Demonstrate the use of any 5 date functions**

**SQL> SELECT SYSDATE FROM DUAL;**

SYSDATE

---------

04-MAR-24

**SQL> SELECT TO\_DATE('2022-01-10', 'YYYY-MM-DD') FROM DUAL;**

TO\_DATE('

---------

10-JAN-22

**SQL> SELECT MONTHS\_BETWEEN('01-JAN-2022', '01-JAN-2021') FROM DUAL;**

MONTHS\_BETWEEN('01-JAN-2022','01-JAN-2021')

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

                                         12

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

ADD\_MONTH

---------

04-JUN-24

**SQL> SELECT LAST\_DAY('01-JAN-2022') FROM DUAL;**

LAST\_DAY(

---------

31-JAN-22

**Learning outcome of the Experiment:**

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |

**Experiment 5**

**Experiment 5:** Implement queries including sorting, grouping and subqueries – LIKE, ANY, ALL, EXISTS, NOT EXISTS.

**Explanation of the Experiment & Concept:**

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

**SQL Commands:**

1. **Creating table and adding values with constraints:**
   1. **faculty(fid, name, *dno*, sal, address, phone dob, exp)**
   2. **dept(dno, dname, budget)**

**SQL> CREATE TABLE dept (**

**2  dno INT NOT NULL PRIMARY KEY,**

**3  dname VARCHAR(5),**

**4  budget INT);**

Table created.

**SQL> CREATE TABLE faculty (**

**2  fid INT NOT NULL PRIMARY KEY,**

**3  name VARCHAR(10),**

**4  dno INT,**

**5  sal INT,**

**6  address VARCHAR(15),**

**7  phone INT,**

**8  dob DATE,**

**9  exp INT,**

**10  FOREIGN KEY (dno) REFERENCES dept(dno));**

Table created.

**SQL> DESC dept;**

 Name                              Null?    Type

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

 DNO                               NOT NULL NUMBER(38)

 DNAME                                      VARCHAR2(5)

 BUDGET                                     NUMBER(38)

**SQL> DESC faculty;**

 Name                                  Null?    Type

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

 FID                                   NOT NULL NUMBER(38)

 NAME                                           VARCHAR2(10)

 DNO                                            NUMBER(38)

 SAL                                            NUMBER(38)

 ADDRESS                                        VARCHAR2(15)

 PHONE                                          NUMBER(38)

 DOB                                            DATE

 EXP                                            NUMBER(38)

**SQL> INSERT INTO dept VALUES (10, 'AIML', 5000);**

1 row created.

**SQL> INSERT INTO dept VALUES (20, 'AIDS', 7000);**

1 row created.

**SQL> INSERT INTO dept VALUES (30, 'IIOT', 8000);**

1 row created.

**SQL> INSERT INTO dept VALUES (40, 'CSE', 9000);**

1 row created.

**SQL> INSERT INTO dept VALUES (50, 'IT', 12000);**

1 row created.

**SQL> INSERT INTO faculty VALUES (1, 'Puneet', 10, 5000, 'Paschim Vihar', 9988203942, '01-mar-85', 14);**

1 row created.

**SQL> INSERT INTO faculty VALUES (2, 'Ashish', 10, 5000, 'Tilak Nagar', 9648203942, '07-nov-85', 13);**

1 row created.

**SQL> INSERT INTO faculty VALUES (3, 'Ishleen', 20, 6000, 'Rajouri', 889203942, '08-dec-85', 5);**

1 row created.

**SQL> INSERT INTO faculty VALUES (4, 'Sonakshi', 30, 2000, 'Pitampura', NULL, '23-jun-85', 8);**

1 row created.

**SQL> INSERT INTO faculty VALUES (5, 'Suman', 20, 1500, 'ITO', 778203532, '19-apr-85', 19);**

1 row created.

**SQL> INSERT INTO faculty VALUES (6, 'Gopal', 40, 9000, 'Vikas Puri', 9013203216, '10-mar-85', 12);**

1 row created.

**SQL> INSERT INTO faculty VALUES (7, 'Archa', 40, 6000, 'Hari Nagar', NULL, '07-mar-85', 9);**

1 row created.

**SQL> SELECT \* FROM dept;**

       DNO DNAME     BUDGET

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

        10 AIML        5000

        20 AIDS        7000

        30 IIOT        8000

        40 CSE         9000

        50 IT         12000

**SQL> SELECT \* FROM faculty;**

 FID NAME        DNO   SAL ADDRESS              PHONE DOB        EXP

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

   1 Puneet       10  5000 Paschim Vihar   9988203942 01-MAR-85   14

   2 Ashish       10  5000 Tilak Nagar     9648203942 07-NOV-85   13

   3 Ishleen      20  6000 Rajouri          889203942 08-DEC-85    5

   4 Sonakshi     30  2000 Pitampura                  23-JUN-85    8

   5 Suman        20  1500 ITO              778203532 19-APR-85   19

   6 Gopal        40  9000 Vikas Puri      9013203216 10-MAR-85   12

   7 Archa        40  6000 Hari Nagar                 07-MAR-85    9

1. rows selected.
2. **Retrieve the id, name and salary of faculty in increasing order of salary.**

**SQL> SELECT fid, name, sal FROM faculty ORDER BY sal ASC;**

       FID NAME              SAL

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

         5 Suman            1500

         4 Sonakshi         2000

         1 Puneet           5000

         2 Ashish           5000

         3 Ishleen          6000

         7 Archa            6000

         6 Gopal            9000

7 rows selected.

1. **Retrieve the id, name and salary of faculty in decreasing order of experience**

**SQL> SELECT fid, name, sal FROM faculty ORDER BY exp DESC;**

       FID NAME              SAL

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

         5 Suman            1500

         1 Puneet           5000

         2 Ashish           5000

         6 Gopal            9000

         7 Archa            6000

         4 Sonakshi         2000

         3 Ishleen          6000

7 rows selected.

1. **Retrieve the id, name and salary of faculty in increasing order of salary and decreasing order of experience:**

**SQL> SELECT fid, name, sal FROM faculty ORDER BY sal ASC, exp DESC;**

       FID NAME              SAL

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

         5 Suman            1500

         4 Sonakshi         2000

         1 Puneet           5000

         2 Ashish           5000

         7 Archa            6000

         3 Ishleen          6000

         6 Gopal            9000

7 rows selected.

1. **Show the average experience department wise**

**SQL> SELECT dno, AVG(exp) FROM faculty GROUP BY dno;**

       DNO   AVG(EXP)

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

        10       13.5

        20         12

        30          8

        40       10.5

1. **Retrieve the maximum salary department wise**

**SQL> SELECT dno, MAX(exp) FROM faculty GROUP BY dno;**

       DNO   MAX(EXP)

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

        10         14

        20         19

        30          8

        40         12

1. **Retrieve the average salary department wise having average salary greater than 5000**

**SQL> SELECT dno, AVG(sal) FROM faculty HAVING AVG(sal)>5000 GROUP BY dno;**

       DNO   AVG(SAL)

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

        40       7500

1. **Retrieve faculties details where name (table should have names accordingly)**
   1. **Starting with ‘s’**
   2. **Ending with ‘n’**
   3. **Starting with ‘a’ and ending with ‘h’**
   4. **Must contain ‘a’**
   5. **Should have 5 letter only starting with ‘g’**

**SQL> SELECT \* FROM faculty WHERE LOWER(name) LIKE LOWER('s%');**

 FID NAME        DNO   SAL ADDRESS         PHONE DOB        EXP

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

   4 Sonakshi     30  2000 Pitampura             23-JUN-85    8

   5 Suman        20  1500 ITO         778203532 19-APR-85   19

**SQL> SELECT \* FROM faculty WHERE LOWER(name) LIKE LOWER('%n');**

 FID NAME      DNO   SAL ADDRESS        PHONE DOB         EXP

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

   3 Ishleen    20  6000 Rajouri    889203942 08-DEC-85     5

   5 Suman      20  1500 ITO        778203532 19-APR-85    19

**SQL> SELECT \* FROM faculty WHERE LOWER(name) LIKE LOWER('a%h');**

 FID NAME       DNO    SAL ADDRESS           PHONE DOB        EXP

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

   2 Ashish      10   5000 Tilak Nagar  9648203942 07-NOV-85   13

**SQL> SELECT \* FROM faculty WHERE LOWER(name) LIKE LOWER('%a%');**

 FID NAME        DNO   SAL ADDRESS            PHONE DOB        EXP

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

   2 Ashish       10  5000 Tilak Nagar   9648203942 07-NOV-85   13

   4 Sonakshi     30  2000 Pitampura                23-JUN-85    8

   5 Suman        20  1500 ITO            778203532 19-APR-85   19

   6 Gopal        40  9000 Vikas Puri    9013203216 10-MAR-85   12

   7 Archa        40  6000 Hari Nagar               07-MAR-85    9

**SQL> SELECT \* FROM faculty WHERE LOWER(name) LIKE LOWER('g\_\_\_\_');**

 FID NAME    DNO   SAL ADDRESS          PHONE DOB        EXP

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

   6 Gopal    40  9000 Vikas Puri  9013203216 10-MAR-85   12

1. **Show the use of any/all clause: list the faculties of department whose budget is less than 10000**

**SQL> SELECT name, dno FROM faculty WHERE dno<ANY (SELECT dno FROM dept WHERE** budget<10000);

NAME              DNO

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

Puneet             10

Ashish             10

Ishleen            20

Suman              20

Sonakshi           30

**SQL> SELECT name, dno FROM faculty WHERE dno>ALL (SELECT dno FROM dept WHERE** budget<8000);

NAME              DNO

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

Sonakshi           30

Gopal              40

Archa              40

1. **Show the use of exists/not exists clause.**

**SQL> SELECT name, dno FROM faculty WHERE EXISTS ( SELECT dno FROM dept WHERE faculty.dno=dept.dno AND budget>7000);**

NAME              DNO

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

Sonakshi           30

Gopal              40

Archa              40

**SQL> SELECT name, dno FROM faculty WHERE NOT EXISTS ( SELECT dno FROM dept WHERE faculty.dno=dept.dno AND budget<7000);**

NAME              DNO

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

Gopal              40

Archa              40

Sonakshi           30

Ishleen            20

Suman              20

1. **Find the names of faculty who work in department aiml ( using subquery)**

**SQL> SELECT name FROM faculty WHERE dno = (SELECT dno FROM dept WHERE dname = 'AIML');**

NAME

----------

Puneet

Ashish

**Learning outcome of the Experiment:**

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |

**Experiment 6**

**Experiment 6:** Implement queries using various set operations (UNION, INTERACTION, EXCEPT etc.)

**Explanation of the Experiment & Concept:**

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

**SQL Commands:**

1. **Creating table and adding values with constraints:**
   1. **section(course\_id, sec\_id, semester, year, building, room\_number, time\_slot\_id)**

**SQL> CREATE TABLE section (**

**2  course\_id VARCHAR(10),**

**3  sec\_id INT,**

**4  semester VARCHAR(10),**

**5  year INT,**

**6  building VARCHAR(10),**

**7  room\_number INT,**

**8  time\_slot\_id CHAR(1));**

Table created.

**SQL> INSERT INTO section VALUES ('BIO-101', 1, 'Summer', 2017, 'Painter', 514, 'B');**

1 row created.

**SQL> INSERT INTO section VALUES ('BIO-301', 1, 'Summer', 2018, 'Painter', 514, 'A');**

1 row created.

**SQL> INSERT INTO section VALUES ('CS-101', 1, 'Fall', 2017, 'Packard', 101, 'H');**

1 row created.

**SQL> INSERT INTO section VALUES ('CS-101', 1, 'Spring', 2018, 'Packard', 101, 'F');**

1 row created.

**SQL> INSERT INTO section VALUES ('CS-190', 1, 'Spring', 2017, 'Taylor', 3128, 'E');**

1 row created.

**SQL> INSERT INTO section VALUES ('CS-190', 2, 'Spring', 2017, 'Taylor', 3128, 'A');**

1 row created.

**SQL> INSERT INTO section VALUES ('CS-315', 1, 'Spring', 2018, 'Watson', 120, 'D');**

1 row created.

**SQL> INSERT INTO section VALUES ('CS-319', 1, 'Spring', 2018, 'Watson', 100, 'B');**

1 row created.

**SQL> INSERT INTO section VALUES ('CS-319', 2, 'Spring', 2018, 'Taylor', 3128,** **'C');**

1 row created.

**SQL> INSERT INTO section VALUES ('CS-347', 1, 'Fall', 2017, 'Taylor', 3128, 'A');**

1 row created.

**SQL> INSERT INTO section VALUES ('EE-181', 1, 'Spring', 2017, 'Taylor', 3128,** **'C');**

1 row created.

**SQL> INSERT INTO section VALUES ('FIN-201', 1, 'Spring', 2018, 'Packard', 101,** **'B');**

1 row created.

**SQL> INSERT INTO section VALUES ('HIS-351', 1, 'Spring', 2018, 'Painter', 514, 'C');**

1 row created.

**SQL> INSERT INTO section VALUES ('MU-199', 1, 'Spring', 2018, 'Packard', 101, 'D');**

1 row created.

**SQL> INSERT INTO section VALUES ('PHY-101', 1, 'Fall', 2017, 'Watson', 100, 'A');**

1 row created.

**SQL> DESC section;**

 Name                       Null?    Type

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

 COURSE\_ID                           VARCHAR2(10)

 SEC\_ID                              NUMBER(38)

 SEMESTER                            VARCHAR2(10)

 YEAR                                NUMBER(38)

 BUILDING                            VARCHAR2(10)

 ROOM\_NUMBER                         NUMBER(38)

 TIME\_SLOT\_ID                        CHAR(1)

**SQL> SELECT \* FROM section;**

COURSE\_ID      SEC\_ID SEMESTER         YEAR BUILDING   ROOM\_NUMBER T

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

BIO-101             1 Summer           2017 Painter            514 B

BIO-301             1 Summer           2018 Painter            514 A

CS-101              1 Fall             2017 Packard            101 H

CS-101              1 Spring           2018 Packard            101 F

CS-190              1 Spring           2017 Taylor            3128 E

CS-190              2 Spring           2017 Taylor            3128 A

CS-315              1 Spring           2018 Watson             120 D

CS-319              1 Spring           2018 Watson             100 B

CS-319              2 Spring           2018 Taylor            3128 C

CS-347              1 Fall             2017 Taylor            3128 A

EE-181              1 Spring           2017 Taylor            3128 C

FIN-201             1 Spring           2018 Packard            101 B

HIS-351             1 Spring           2018 Painter            514 C

MU-199              1 Spring           2018 Packard            101 D

PHY-101             1 Fall             2017 Watson             100 A

15 rows selected.

1. **Find courses that ran in Fall 2017.**

**SQL> SELECT course\_id FROM section WHERE semester = 'Fall' AND year = 2017;**

COURSE\_ID

----------

CS-101

CS-347

PHY-101

1. **Find courses that ran in Spring 2018**

**SQL> SELECT course\_id FROM section WHERE semester = 'Spring' AND year = 2018;**

COURSE\_ID

----------

CS-101

CS-315

CS-319

CS-319

FIN-201

HIS-351

MU-199

7 rows selected.

1. **Find courses that ran in Fall 2017 or in Spring 2018:**

**SQL> (SELECT course\_id FROM section WHERE semester = 'Fall' AND year = 2017) UNION (SELECT course\_id FROM section WHERE semester =**

**'Spring' AND year = 2018);**

COURSE\_ID

----------

CS-101

CS-347

PHY-101

CS-315

CS-319

FIN-201

HIS-351

MU-199

8 rows selected.

1. **Find courses that ran in Fall 2017 and in Spring 2018**

**SQL> (SELECT course\_id FROM section WHERE semester = 'Fall' AND year = 2017) INTERSECT (SELECT course\_id FROM section WHERE semester = 'Spring' AND year = 2018);**

COURSE\_ID

----------

CS-101

1. **Find courses that ran in Fall 2017 but not in Spring 2018**

**SQL> (SELECT course\_id FROM section WHERE semester = 'Fall' AND year = 2017) MINUS (SELECT course\_id FROM section WHERE semester = 'Spring' AND year = 2018);**

COURSE\_ID

----------

CS-347

PHY-101

1. **Find courses that ran in Spring 2018 but not in Fall 2017**

**SQL> (SELECT course\_id FROM section WHERE semester = 'Spring' AND year = 2018) MINUS (SELECT course\_id FROM section WHERE semester = 'Fall' AND year = 2017);**

COURSE\_ID

----------

CS-315

CS-319

FIN-201

HIS-351

MU-199

1. **Create 2 different tables with same no. of columns and make use of union, union all, intersect and except**
   1. **Creating tables and inserting values**

**SQL> CREATE TABLE employees\_dept1 (**

**2  employee\_id INT,**

**3  employee\_name VARCHAR(10),**

**4  department VARCHAR(10));**

Table created.

**SQL> CREATE TABLE employees\_dept2 (**

**2  employee\_id INT,**

**3  employee\_name VARCHAR(10),**

**4  department VARCHAR(10));**

Table created.

**SQL> DESC employees\_dept1;**

 Name                              Null?    Type

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

 EMPLOYEE\_ID                                NUMBER(38)

 EMPLOYEE\_NAME                              VARCHAR2(10)

 DEPARTMENT                                 VARCHAR2(10)

**SQL> DESC employees\_dept2;**

 Name                             Null?    Type

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

 EMPLOYEE\_ID                               NUMBER(38)

 EMPLOYEE\_NAME                             VARCHAR2(10)

 DEPARTMENT                                VARCHAR2(10)

**SQL> INSERT INTO employees\_dept1 VALUES (1, 'Alice', 'Finance');**

1 row created.

**SQL> INSERT INTO employees\_dept1 VALUES (2, 'Bob', 'Finance');**

1 row created.

**SQL> INSERT INTO employees\_dept1 VALUES (3, 'Charlie', 'Finance');**

1 row created.

**SQL> INSERT INTO employees\_dept2 VALUES (4, 'David', 'HR');**

1 row created.

**SQL> INSERT INTO employees\_dept2 VALUES (5, 'Emma', 'HR');**

1 row created.

**SQL> INSERT INTO employees\_dept2 VALUES (6, 'Frank', 'HR');**

1 row created.

**SQL> SELECT \* from employees\_dept1;**

EMPLOYEE\_ID EMPLOYEE\_N DEPARTMENT

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

          1 Alice      Finance

          2 Bob        Finance

          3 Charlie    Finance

**SQL> SELECT \* from employees\_dept2;**

EMPLOYEE\_ID EMPLOYEE\_N DEPARTMENT

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

          4 David      HR

          5 Emma       HR

          6 Frank      HR

* 1. **Using UNION**

**SQL> SELECT \* FROM employees\_dept1**

**2  UNION**

**3  SELECT \* FROM employees\_dept2;**

EMPLOYEE\_ID EMPLOYEE\_N DEPARTMENT

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

          1 Alice      Finance

          2 Bob        Finance

          3 Charlie    Finance

          4 David      HR

          5 Emma       HR

          6 Frank      HR

6 rows selected.

* 1. **Using UNION ALL**

**SQL> SELECT \* FROM employees\_dept1**

**2  UNION ALL**

**3  SELECT \* FROM employees\_dept2;**

EMPLOYEE\_ID EMPLOYEE\_N DEPARTMENT

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

          1 Alice      Finance

          2 Bob        Finance

          3 Charlie    Finance

          4 David      HR

          5 Emma       HR

          6 Frank      HR

6 rows selected.

* 1. **Using INTERSECT**

**SQL> SELECT \* FROM employees\_dept1**

**2  INTERSECT**

**3  SELECT \* FROM employees\_dept2;**

no rows selected

* 1. **Using EXCEPT**

**SQL> SELECT \* FROM employees\_dept1**

**2  EXCEPT**

**3  SELECT \* FROM employees\_dept2;**

EMPLOYEE\_ID EMPLOYEE\_N DEPARTMENT

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

          1 Alice      Finance

          2 Bob        Finance

          3 Charlie    Finance

**Learning outcome of the Experiment:**

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |

**Experiment 7**

**Experiment 7:** Implement various join operations (INNER, OUTER).

**Explanation of the Experiment & Concept:**

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

**SQL Commands:**

1. **Creating table and adding values with constraints:**
   1. **student(roll\_no, name, address, phone, age)**
   2. **studentcourse(course\_id, roll\_no)**
   3. **course(course\_id, c\_name)**

**SQL> CREATE TABLE student (**

**2  roll\_no INT PRIMARY KEY,**

**3  name VARCHAR(10) UNIQUE,**

**4  address VARCHAR(10),**

**5  phone INT,**

**6  age INT NOT NULL);**

Table created.

**SQL> CREATE TABLE course (**

**2  course\_id INT PRIMARY KEY,**

**3  c\_name VARCHAR(5));**

Table created.

**SQL> CREATE TABLE studentcourse (**

**2  course\_id INT,**

**3  roll\_no INT);**

Table created.

**SQL> DESC student;**

 Name                          Null?    Type

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

 ROLL\_NO                       NOT NULL NUMBER(38)

 NAME                                   VARCHAR2(10)

 ADDRESS                                VARCHAR2(10)

 PHONE                                  NUMBER(38)

 AGE                           NOT NULL NUMBER(38)

**SQL> DESC course;**

 Name                          Null?    Type

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

 COURSE\_ID                     NOT NULL NUMBER(38)

 C\_NAME                                 VARCHAR2(5)

**SQL> DESC studentcourse;**

 Name                          Null?    Type

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

 COURSE\_ID                              NUMBER(38)

 ROLL\_NO                                NUMBER(38)

**SQL> INSERT INTO student VALUES(1, 'Harsh', 'Delhi', 9876543210, 18);**

1 row created.

**SQL> INSERT INTO student VALUES(2, 'Pratik', 'Bihar', 9422543210, 19);**

1 row created.

**SQL> INSERT INTO student VALUES(3, 'Riyanka', 'Siliguri', 7568184452, 20);**

1 row created.

**SQL> INSERT INTO student VALUES(4, 'Deep', 'Ramnagar', 9273874628, 18);**

1 row created.

**SQL> INSERT INTO student VALUES(5, 'Saptarhi', 'Kolkata', 8368747729, 19);**

1 row created.

**SQL> INSERT INTO student VALUES(6, 'Dhanraj', 'Barabajar', 9687477234, 20);**

1 row created.

**SQL> INSERT INTO student VALUES(7, 'Rohit', 'Balurghat', 7988924429, 18);**

1 row created.

**SQL> INSERT INTO student VALUES(8, 'Niraj', 'Alipur', 9382098430, 19);**

1 row created.

**SQL> INSERT INTO course VALUES(1, 'AIML');**

1 row created.

**SQL> INSERT INTO course VALUES(2, 'AIDS');**

1 row created.

**SQL> INSERT INTO course VALUES(3, 'IIOT');**

1 row created.

**SQL> INSERT INTO course VALUES(4, 'CSE');**

1 row created.

**SQL> INSERT INTO course VALUES(5, 'IT');**

1 row created.

**SQL> INSERT INTO studentcourse VALUES(1,1);**

1 row created.

**SQL> INSERT INTO studentcourse VALUES(2,2);**

1 row created.

**SQL> INSERT INTO studentcourse VALUES(2,3);**

1 row created.

**SQL> INSERT INTO studentcourse VALUES(3,4);**

1 row created.

**SQL> INSERT INTO studentcourse VALUES(1,5);**

1 row created.

**SQL> INSERT INTO studentcourse VALUES(4,9);**

1 row created.

**SQL> INSERT INTO studentcourse VALUES(5,10);**

1 row created.

**SQL> INSERT INTO studentcourse VALUES(4,11);**

1 row created.

**SQL> SELECT \* FROM student;**

   ROLL\_NO NAME       ADDRESS         PHONE        AGE

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

         1 Harsh      Delhi      9876543210         18

         2 Pratik     Bihar      9422543210         19

         3 Riyanka    Siliguri   7568184452         20

         4 Deep       Ramnagar   9273874628         18

         5 Saptarhi   Kolkata    8368747729         19

         6 Dhanraj    Barabajar  9687477234         20

         7 Rohit      Balurghat  7988924429         18

         8 Niraj      Alipur     9382098430         19

8 rows selected.

**SQL> SELECT \* FROM course;**

 COURSE\_ID C\_NAM

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

         1 AIML

         2 AIDS

         3 IIOT

         4 CSE

         5 IT

**SQL> SELECT \* FROM studentcourse;**

 COURSE\_ID    ROLL\_NO

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

         1          1

         2          2

         2          3

         3          4

         1          5

         4          9

         5         10

         4         11

8 rows selected.

1. **Retrieve the names and age of students enrolled in different courses.**

**SQL> SELECT studentcourse.course\_id, student.name, student.age FROM student INNER JOIN studentcourse ON student.roll\_no = studentcourse.roll\_no;**

 COURSE\_ID NAME              AGE

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

         1 Harsh              18

         2 Pratik             19

         2 Riyanka            20

         3 Deep               18

         1 Saptarhi           19

1. **Apply left outer join, right outer join and full outer join on tables**

**SQL> SELECT student.name, studentcourse.course\_id FROM student LEFT JOIN studentcourse ON studentcourse.roll\_no = student.roll\_no;**

NAME        COURSE\_ID

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

Harsh               1

Pratik              2

Riyanka             2

Deep                3

Saptarhi            1

Dhanraj

Rohit

Niraj

8 rows selected.

**SQL> SELECT student.name, studentcourse.course\_id FROM student RIGHT JOIN studentcourse ON studentcourse.roll\_no = student.roll\_no;**

NAME        COURSE\_ID

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

Harsh               1

Pratik              2

Riyanka             2

Deep                3

Saptarhi            1

                    4

                    5

                    4

8 rows selected.

**SQL> SELECT student.name, studentcourse.course\_id FROM student FULL JOIN studentcourse ON studentcourse.roll\_no = student.roll\_no;**

NAME        COURSE\_ID

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

Harsh               1

Pratik              2

Riyanka             2

Deep                3

Saptarhi            1

                    4

                    5

                    4

Dhanraj

Rohit

Niraj

11 rows selected.

1. **Retrieve the names and address of students enrolled in courses with course\_id=3:**

**SQL> SELECT a.name, a.address FROM student a, studentcourse b WHERE a.roll\_no = b.roll\_no AND b.course\_i**

**d = 3;**

NAME       ADDRESS

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

Deep       Ramnagar

1. **Retrieve the names and phone of students enrolled in courses with course\_id = 2 and age = 20**

**SQL> SELECT a.name, a.phone FROM student a, studentcourse b WHERE a.roll\_no = b.roll\_no AND b.course\_id = 2 AND a.age = 20;**

NAME            PHONE

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

Riyanka    7568184452

1. **Retrieve the name, age, course\_id and course name of student enrolled in different courses**

**SQL> SELECT s.name, s.age, c.course\_id, c.c\_name FROM student s, studentcourse sc, course c WHERE s.r**

**oll\_no = sc.roll\_no AND sc.course\_id = c.course\_id;**

NAME              AGE  COURSE\_ID C\_NAM

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

Harsh              18          1 AIML

Pratik             19          2 AIDS

Riyanka            20          2 AIDS

Deep               18          3 IIOT

Saptarhi           19          1 AIML

1. **Retrieve the names of students enrolled in AIDS**

**SQL> SELECT s.name, s.age, c.course\_id, c.c\_name FROM student s, studentcourse sc, course c where s.r**

**oll\_no = sc.roll\_no AND sc.course\_id = c.course\_id AND c.c\_name = 'AIDS';**

NAME              AGE  COURSE\_ID C\_NAM

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

Pratik             19          2 AIDS

Riyanka            20          2 AIDS

**Learning outcome of the Experiment:**

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |

**Experiment 8**

**Experiment 8:** Write a PL/SQL program using FOR loop to insert ten rows into a database table.

**Explanation of the Experiment & Concept:**

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

**SQL Commands:**

1. **Creating table with constraints:**
   1. **temp(num\_col1, num\_col2, char\_col)**

**SQL> CREATE TABLE temp (**

**2  num\_col1 INT,**

**3  num\_col2 INT,**

**4  char\_col VARCHAR(25)**

**5  );**

Table created.

**SQL> DESC temp;**

 Name                Null?    Type

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

 NUM\_COL1                     NUMBER(38)

 NUM\_COL2                     NUMBER(38)

 CHAR\_COL                     VARCHAR2(25)

1. **Inserting values in table using PL/SQL FOR loop.**

**SQL> DECLARE**

**2  x NUMBER := 100;**

**3  BEGIN**

**4  FOR i IN 1..10 LOOP**

**5  IF MOD(i,2) = 0 THEN -- i is even**

**6  INSERT INTO temp VALUES (i, x, 'i is even');**

**7  ELSE**

**8  INSERT INTO temp VALUES (i, x, 'i is odd');**

**9  END IF;**

**10  x := x + 100;**

**11  END LOOP;**

**12  COMMIT;**

**13  END;**

**14  /**

PL/SQL procedure successfully completed.

**SQL> SELECT \* FROM temp ORDER BY num\_col1;**

  NUM\_COL1   NUM\_COL2 CHAR\_COL

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

         1        100 i is odd

         2        200 i is even

         3        300 i is odd

         4        400 i is even

         5        500 i is odd

         6        600 i is even

         7        700 i is odd

         8        800 i is even

         9        900 i is odd

        10       1000 i is even

10 rows selected.

**Learning outcome of the Experiment:**

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |

**Experiment 9**

**Experiment 9:** Given the table EMPLOYEE (Emp No, Name, Salary, Designation, DeptID), write a cursor to select the five highest-paid employees from the table.

**Explanation of the Experiment & Concept:**

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

**SQL Commands:**

1. **Creating table and adding values with constraints:**
   1. **employee(empNo, name, salary, designation, deptID)**
   2. **temp\_new(salary, empNo, name)**

**SQL> CREATE TABLE employee (**

**2  empNo INT,**

**3  name VARCHAR(15),**

**4  salary INT,**

**5  designation VARCHAR(15),**

**6  deptID INT);**

Table created.

**SQL> DESC employee;**

 Name                              Null?    Type

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

 EMPNO                                      NUMBER(38)

 NAME                                       VARCHAR2(15)

 SALARY                                     NUMBER(38)

 DESIGNATION                                VARCHAR2(15)

 DEPTID                                     NUMBER(38)

**SQL> CREATE TABLE temp\_new (**

**2  salary INT,**

**3  empNo INT,**

**4  name VARCHAR(15)**

**5  );**

Table created.

**SQL> DESC temp\_new;**

 Name                      Null?    Type

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

 SALARY                             NUMBER(38)

 EMPNO                              NUMBER(38)

 NAME                               VARCHAR2(15)

**SQL> INSERT INTO employee VALUES (1, 'abc', 10000, 'jr engineer', 101);**

1 row created.

**SQL> INSERT INTO employee VALUES (2, 'def', 12000, 'database engg', 120);**

1 row created.

**SQL> INSERT INTO employee VALUES (3, 'ghi', 9000, 'personal asst', 111);**

1 row created.

**SQL> INSERT INTO employee VALUES (4, 'jkl', 20000, 'sr engineer', 105);**

1 row created.

**SQL> INSERT INTO employee VALUES (5, 'mno', 24000, 'sr engineer', 105);**

1 row created.

**SQL> INSERT INTO employee VALUES (6, 'pqr', 13000, 'jr engineer', 101);**

1 row created.

**SQL> INSERT INTO employee VALUES (7, 'stu', 30000, 'manager', 002);**

1 row created.

**SQL> INSERT INTO employee VALUES (8, 'vwx', 5000, 'helper', 111);**

1 row created.

**SQL> SELECT \* FROM employee;**

     EMPNO NAME                SALARY DESIGNATION         DEPTID

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

         1 abc                  10000 jr engineer            101

         2 def                  12000 database engg          120

         3 ghi                   9000 personal asst          111

         4 jkl                  20000 sr engineer            105

         5 mno                  24000 sr engineer            105

         6 pqr                  13000 jr engineer            101

         7 stu                  30000 manager                  2

         8 vwx                   5000 helper                 111

8 rows selected.

1. **Using cursor to get the five highest-paid employees.**

**SQL> DECLARE**

**2  CURSOR c1 IS**

**3  SELECT name, empNo, salary FROM employee**

**4  ORDER BY salary DESC;**

**5  my\_ename employee.name%type;**

**6  my\_empno employee.empNo%type;**

**7  my\_sal employee.salary%type;**

**8  BEGIN**

**9  OPEN c1;**

**10  FOR i IN 1..5 LOOP**

**11  FETCH c1 INTO my\_ename, my\_empno, my\_sal;**

**12  EXIT WHEN c1%NOTFOUND;**

**13  INSERT INTO temp\_new VALUES (my\_sal, my\_empno, my\_ename);**

**14  COMMIT;**

**15  END LOOP;**

**16  CLOSE c1;**

**17  END;**

**18  /**

PL/SQL procedure successfully completed.

**SQL> SELECT \* FROM temp\_new;**

    SALARY      EMPNO NAME

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

     30000          7 stu

     24000          5 mno

     20000          4 jkl

     13000          6 pqr

     12000          2 def

**Learning outcome of the Experiment:**

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |

# SECTION II

**Database Management System Lab**

**Beyond Curriculum**

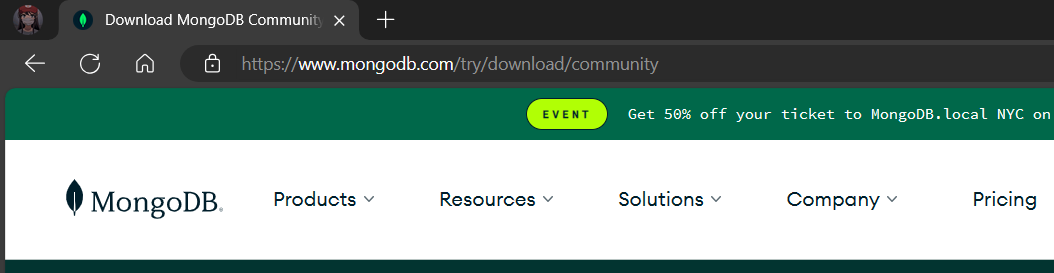
**Experiment 1**

**Experiment 1:** Write the steps to install and implement NOSQL databases MongoDB.

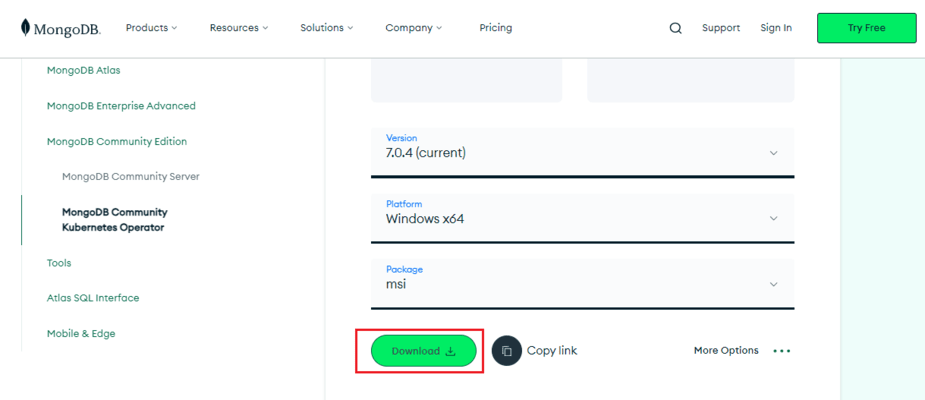
**MongoDB Download and Installation Steps:**

Step 1)**Installing MongoDB**

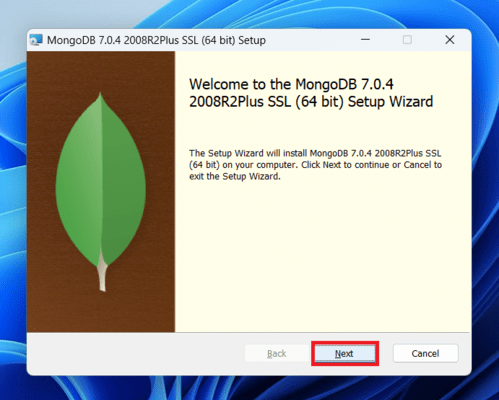
Open your browser and type <https://www.mongodb.com/try/download/community>



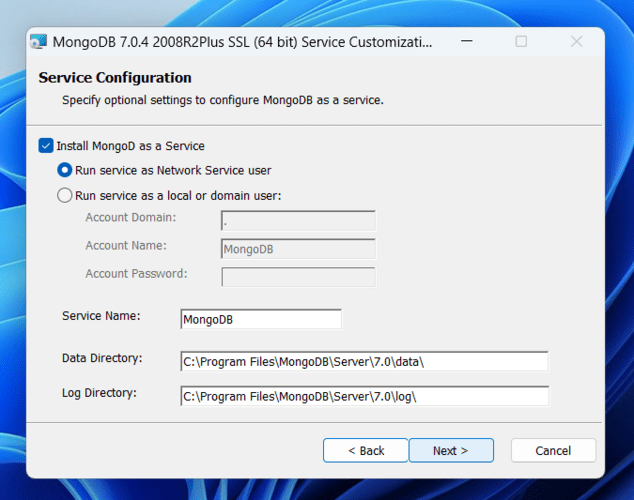
Step 2)**Click on “Download” button.**



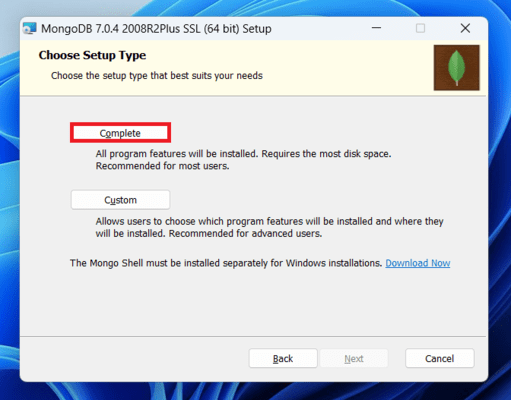
Step 3) **When the download is complete open the msi file and click the next button in the startup screen:**



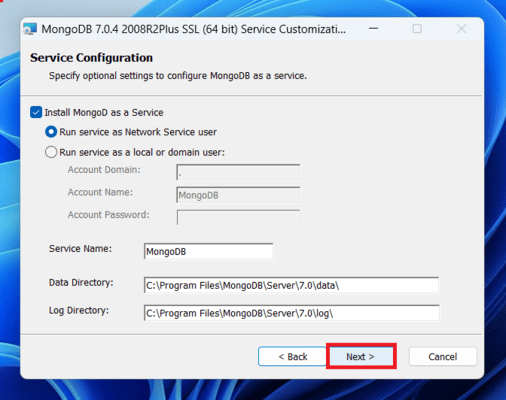
Step 4)**Now accept the End-User License Agreement and click the next button:**



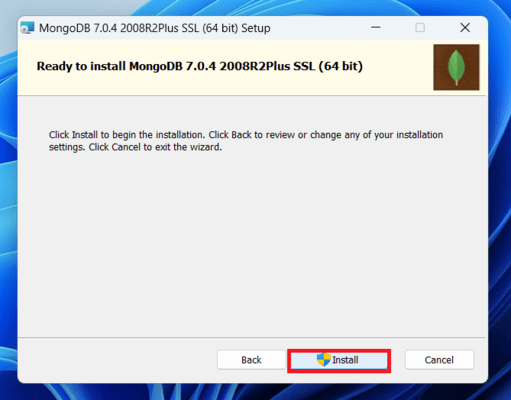
Step 5)**Now select the complete option to install all the program features.**



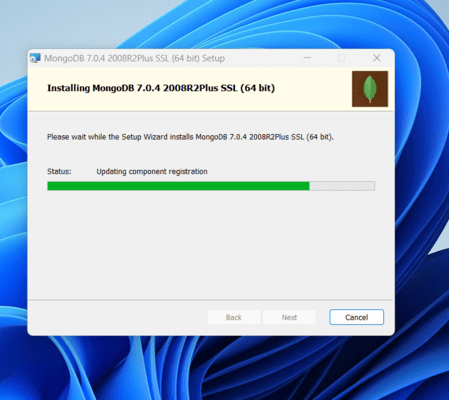
Step 6)**Select “Run service as Network Service user” and copy the path of the data directory. Click Next:**



Step 7)**Click the Install button to start the MongoDB installation process:**

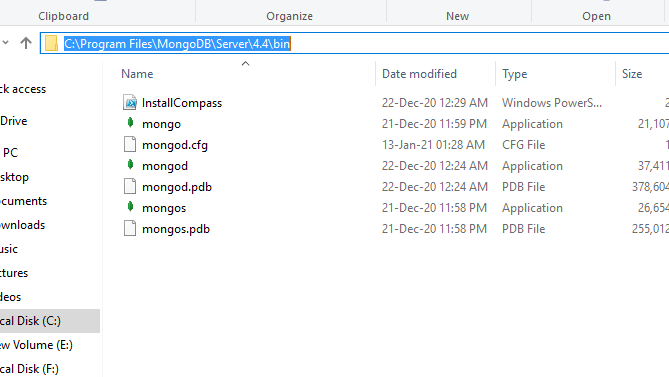


Step 8)**After clicking on the install button installation of MongoDB begins:**



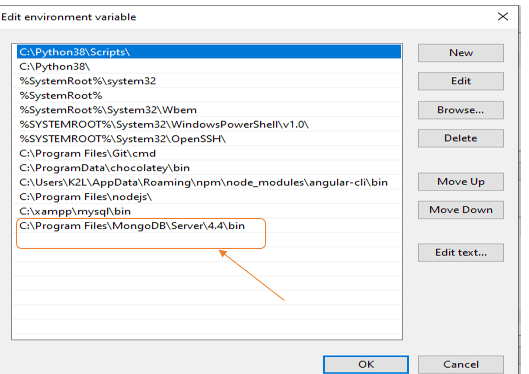
Step 9)**Now click the Finish button to complete the MongoDB installation process**

Step 10)**Now we go to the location where MongoDB installed in step 6 in your system and copy the bin path:**



Step 11)[**Create Environment Variable**](https://www.guru99.com/java-packages.html)**.**

To create an environment variable open system properties << Environment Variable << System variable << path << Edit Environment variable and paste the copied link to your environment system and click Ok:



Step 12)**Running MongoDB Server**

After setting the environment variable, we will run the MongoDB server, i.e. mongod.  So, open the command prompt and run the following command:

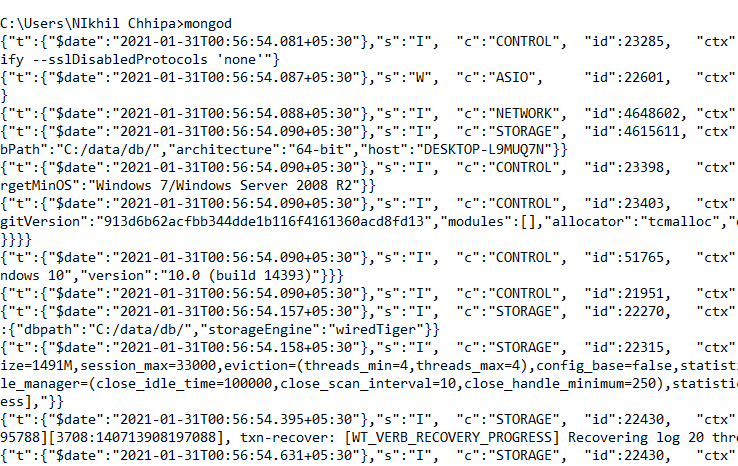
**mongod**

When you run this command you will get an error i.e. *C:/data/db/ not found*.

**Step 12:** Now, Open C drive and create a folder named “data” inside this folder create another folder named “db”. After creating these folders. Again open the command prompt and run the following command:

**mongod**

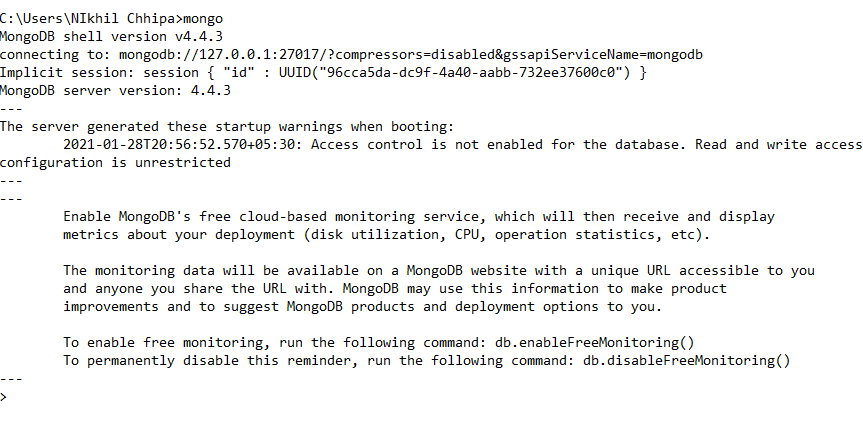
Now, this time the MongoDB server(i.e., mongod) will run successfully.



Step 13)**Running mongo Shell**

Now we are going to connect our server (mongod) with the mongo shell. So, keep that mongod window and open a new command prompt window and write **mongo.** Now, our mongo shell will successfully connect to the mongod.

**Important Point:** Please do not close the mongod window if you close this window your server will stop working and it will not able to connect with the mongo shell.



Now, you are ready to write queries in the mongo Shell.

Step 14)**Run MongoDB**

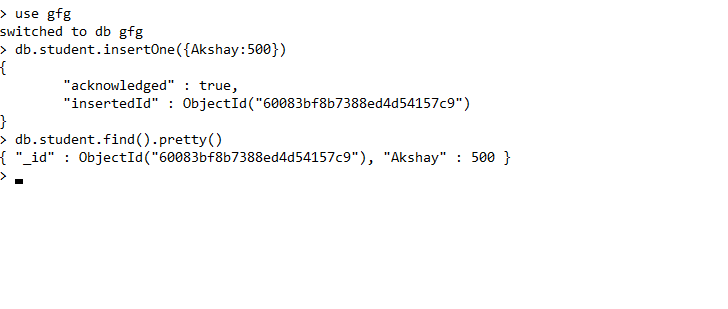
Now you can make a new database, collections, and documents in your shell. The *use Database\_name*command makes a new database in the system if it does not exist, if the database exists it uses that database:

**use gfg**

Now your database is ready of name gfg.

The db.Collection\_name command makes a new collection in the gfg database and the insertOne() method inserts the document in the student collection:

db.student.insertOne({Akshay:500})



**Learning outcome of the Program:**

|  |
| --- |
|  |
|  |
|  |
|  |

**Experiment 2**

**Experiment 2:** Study and implement basic commands of MongoDB.

**Explanation of the Experiment & Concept:**

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

**SQL Commands:**

1. **Creating and using database**

**test> use mydatabase**

switched to db mydatabase

1. **Creating Collection**

**mydatabase> db.createCollection("mycollection")**

{ ok: 1 }

1. **Getting list of all collections**

**mydatabase> show collections**

mycollection

1. **Inserting document in collection**

**mydatabase> db.mycollection.insertOne({ name: "John", age: 30 })**

{

  acknowledged: true,

  insertedId: ObjectId('662a881e677a8cd27d46b79d')

}

**mydatabase> db.mycollection.insertMany([**

**... { name: "Alice", age: 25 },**

**... { name: "Bob", age: 35 }**

**... ])**

{

  acknowledged: true,

  insertedIds: {

    '0': ObjectId('662a883b677a8cd27d46b79e'),

    '1': ObjectId('662a883b677a8cd27d46b79f')

  }

}

1. **Get collection document**

**mydatabase> db.mycollection.find()**

[

  { \_id: ObjectId('662a881e677a8cd27d46b79d'), name: 'John', age: 30 },

  { \_id: ObjectId('662a883b677a8cd27d46b79e'), name: 'Alice', age: 25 },

  { \_id: ObjectId('662a883b677a8cd27d46b79f'), name: 'Bob', age: 35 }

]

**mydatabase> db.mycollection.find({ name: "John" })**

[

  { \_id: ObjectId('662a881e677a8cd27d46b79d'), name: 'John', age: 30 }

]

1. **Updating Document**

**mydatabase> db.mycollection.updateOne({ name: "John" }, { $set: { age: 32 } })**

{

  acknowledged: true,

  insertedId: null,

  matchedCount: 1,

  modifiedCount: 1,

  upsertedCount: 0

}

**mydatabase> db.mycollection.find()**

[

  { \_id: ObjectId('662a881e677a8cd27d46b79d'), name: 'John', age: 32 },

  { \_id: ObjectId('662a883b677a8cd27d46b79e'), name: 'Alice', age: 25 },

  { \_id: ObjectId('662a883b677a8cd27d46b79f'), name: 'Bob', age: 35 }

]

**mydatabase> db.mycollection.updateMany({ age: { $gt: 30 } }, { $set: { status: "adult" } })**

{

  acknowledged: true,

  insertedId: null,

  matchedCount: 2,

  modifiedCount: 2,

  upsertedCount: 0

}

**mydatabase> db.mycollection.find()**

[

  {

    \_id: ObjectId('662a881e677a8cd27d46b79d'),

    name: 'John',

    age: 32,

    status: 'adult'

  },

  { \_id: ObjectId('662a883b677a8cd27d46b79e'), name: 'Alice', age: 25 },

  {

    \_id: ObjectId('662a883b677a8cd27d46b79f'),

    name: 'Bob',

    age: 35,

    status: 'adult'

  }

]

1. **Deleting Document**

**mydatabase> db.mycollection.deleteOne({ name: "John" })**

{ acknowledged: true, deletedCount: 1 }

**mydatabase> db.mycollection.find()**

[

  { \_id: ObjectId('662a883b677a8cd27d46b79e'), name: 'Alice', age: 25 },

  {

    \_id: ObjectId('662a883b677a8cd27d46b79f'),

    name: 'Bob',

    age: 35,

    status: 'adult'

  }

]

**mydatabase> db.mycollection.deleteMany({ age: { $lt: 30 } })**

{ acknowledged: true, deletedCount: 1 }

**mydatabase> db.mycollection.find()**

[

  {

    \_id: ObjectId('662a883b677a8cd27d46b79f'),

    name: 'Bob',

    age: 35,

    status: 'adult'

  }

]

1. **Dropping Collection**

**mydatabase> db.mycollection.drop()**

true

**mydatabase> db.mycollection.find()**

1. **Dropping Database**

**mydatabase> db.dropDatabase()**

{ ok: 1, dropped: 'mydatabase' }

**Learning outcome of the Experiment:**

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |

**Experiment 3**

**Experiment 3:** Implement any one real-time project using MySQL/MongoDB such as Library Database Management System etc..

**Explanation of the Experiment & Concept:**

**Library Management System**

The library management system is a database-driven application designed to efficiently manage the operations of a library. It allows librarians to keep track of books, users, book loans, fines, and more. Here's a brief description of the project:

**Features:**

User Management: Allows librarians to manage users, including adding new users, updating user information, and deleting users.

Book Management: Provides functionalities for managing books in the library, including adding new books, updating book information, and removing books from the collection.

Book Issuance: Enables librarians to issue books to users, record the issue date, and set return dates. It also automatically updates the availability of books in the inventory.

Fine Management: Calculates fines for late returns and records fines in the database. It also provides functionalities for viewing and managing fines.

Authentication and Authorization: Supports user authentication and authorization to ensure that only authorized users can perform certain actions, such as issuing books or managing user accounts.

**Database Schema:**

The database schema includes tables for users, books, book issues, and fines, along with appropriate relationships between them.

Triggers are implemented to enforce business rules, automate tasks, and maintain data integrity.

**Benefits:**

Efficiently manages library operations, reducing manual effort and errors.

Provides real-time access to information about books, users, and transactions.

Enhances user experience by automating fine calculations and providing timely notifications.

**Use Cases:**

Librarians can easily add, update, or remove books from the library collection.

Users can borrow books, return them on time to avoid fines, and view their borrowing history.

Administrators can monitor library activities, manage user accounts, and generate reports for analysis.

**Entity Descriptions:**

**Users**: Represents users of the library system. Each user has a unique user\_id, username, password, and role (e.g., admin, librarian, member).

**Books**: Represents books available in the library. Each book has a unique book\_id, title, author, isbn, publication\_year, and quantity\_available.

**BookIssues**: Represents the issuance of books to users. Each issue has a unique issue\_id, references a user\_id from the Users table, and a book\_id from the Books table. It also includes issue\_date, return\_date, and status (issued or returned).

**Fines**: Represents fines imposed on users for late returns. Each fine has a unique fine\_id, references a user\_id and issue\_id, and includes fine\_amount, fine\_reason, and fine\_date.

Relationship Descriptions:

**Entity Relations:**

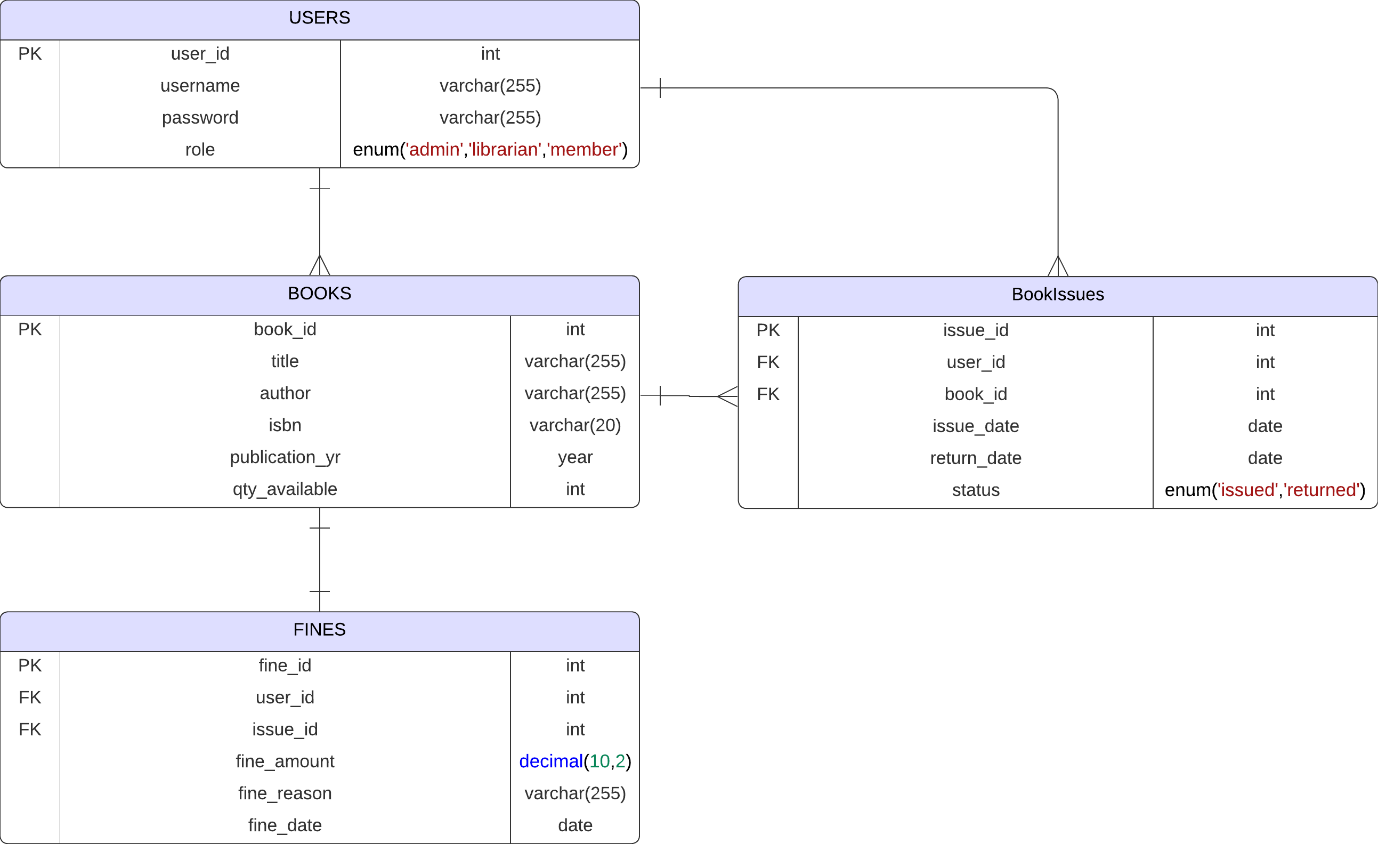
**Users - Books**: One-to-Many relationship where a user can issue multiple books, but each book can be issued to multiple users.

**Books - BookIssues**: One-to-Many relationship where each book can have multiple issues, but each issue is associated with only one book.

**Users - BookIssues**: One-to-Many relationship where each user can have multiple issued books, but each issue is associated with only one user.

**BookIssues - Fines**: One-to-One relationship where each issue can have only one fine, but each fine is associated with only one issue.

**ER Diagram:**



**Triggers:**

**Trigger 1**: This trigger updates the quantity\_available column in the Books table after a book is issued (inserted into the BookIssues table).

**Trigger 2**: This trigger updates the quantity\_available column in the Books table before a book is returned (deleted from the BookIssues table).

**Trigger 3**: This trigger calculates fines for late returns. It calculates the number of days between the issue date and return date, and if the return date is more than 15 days after the issue date, it inserts a fine record into the Fines table.

**SQL Commands:**

1. **Creating and Using Database:**

**mysql> CREATE DATABASE LibraryDB;**

Query OK, 1 row affected (0.01 sec)

**mysql> USE LibraryDB;**

Database changed

1. **Creating table with constraints:**
   1. Users:
      1. user\_id (Primary Key)
      2. username
      3. password
      4. role (e.g., admin, librarian, member)
   2. Books:
      1. book\_id (Primary Key)
      2. title
      3. author
      4. isbn
      5. publication\_year
      6. quantity\_available
   3. BookIssues:
      1. issue\_id (Primary Key)
      2. user\_id (Foreign Key referencing Users Table)
      3. book\_id (Foreign Key referencing Books Table)
      4. issue\_date
      5. return\_date
      6. status (e.g., issued, returned)
   4. Fines:
      1. fine\_id (Primary Key)
      2. user\_id (Foreign Key referencing Users Table)
      3. issue\_id (Foreign Key referencing Book Issues Table)
      4. fine\_amount
      5. fine\_reason
      6. fine\_date

**mysql> -- Create Users Table**

**mysql> CREATE TABLE Users (**

**->     user\_id INT AUTO\_INCREMENT PRIMARY KEY,**

**->     username VARCHAR(255) NOT NULL UNIQUE,**

**->     password VARCHAR(255) NOT NULL,**

**->     role ENUM('admin', 'librarian', 'member') NOT NULL**

**-> );**

Query OK, 0 rows affected (0.16 sec)

**mysql> -- Create Books Table**

**mysql> CREATE TABLE Books (**

**->     book\_id INT AUTO\_INCREMENT PRIMARY KEY,**

**->     title VARCHAR(255) NOT NULL,**

**->     author VARCHAR(255) NOT NULL,**

**->     isbn VARCHAR(20) NOT NULL UNIQUE,**

**->     publication\_year YEAR NOT NULL,**

**->     quantity\_available INT NOT NULL**

**-> );**

Query OK, 0 rows affected (0.14 sec)

**mysql> -- Create Book Issues Table**

**mysql> CREATE TABLE BookIssues (**

**->     issue\_id INT AUTO\_INCREMENT PRIMARY KEY,**

**->     user\_id INT,**

**->     book\_id INT,**

**->     issue\_date DATE NOT NULL,**

**->     return\_date DATE,**

**->     status ENUM('issued', 'returned') NOT NULL,**

**->     FOREIGN KEY (user\_id) REFERENCES Users(user\_id),**

**->     FOREIGN KEY (book\_id) REFERENCES Books(book\_id)**

**-> );**

Query OK, 0 rows affected (0.19 sec)

**mysql> -- Create Fines Table**

**mysql> CREATE TABLE Fines (**

**->     fine\_id INT AUTO\_INCREMENT PRIMARY KEY,**

**->     user\_id INT,**

**->     issue\_id INT,**

**->     fine\_amount DECIMAL(10, 2) NOT NULL,**

**->     fine\_reason VARCHAR(255) NOT NULL,**

**->     fine\_date DATE NOT NULL,**

**->     FOREIGN KEY (user\_id) REFERENCES Users(user\_id),**

**->     FOREIGN KEY (issue\_id) REFERENCES BookIssues(issue\_id)**

**-> );**

Query OK, 0 rows affected (0.19 sec)

**mysql> DESC users;**

+----------+------------------------------------+------+-----+---------+----------------+

| Field    | Type                               | Null | Key | Default | Extra          |

+----------+------------------------------------+------+-----+---------+----------------+

| user\_id  | int                                | NO   | PRI | NULL    | auto\_increment |

| username | varchar(255)                       | NO   | UNI | NULL    |                |

| password | varchar(255)                       | NO   |     | NULL    |                |

| role     | enum('admin','librarian','member') | NO   |     | NULL    |                |

+----------+------------------------------------+------+-----+---------+----------------+

4 rows in set (0.01 sec)

**mysql> DESC books;**

+--------------------+--------------+------+-----+---------+----------------+

| Field              | Type         | Null | Key | Default | Extra          |

+--------------------+--------------+------+-----+---------+----------------+

| book\_id            | int          | NO   | PRI | NULL    | auto\_increment |

| title              | varchar(255) | NO   |     | NULL    |                |

| author             | varchar(255) | NO   |     | NULL    |                |

| isbn               | varchar(20)  | NO   | UNI | NULL    |                |

| publication\_year   | year         | NO   |     | NULL    |                |

| quantity\_available | int          | NO   |     | NULL    |                |

+--------------------+--------------+------+-----+---------+----------------+

6 rows in set (0.00 sec)

**mysql> DESC BookIssues;**

+-------------+---------------------------+------+-----+---------+----------------+

| Field       | Type                      | Null | Key | Default | Extra          |

+-------------+---------------------------+------+-----+---------+----------------+

| issue\_id    | int                       | NO   | PRI | NULL    | auto\_increment |

| user\_id     | int                       | YES  | MUL | NULL    |                |

| book\_id     | int                       | YES  | MUL | NULL    |                |

| issue\_date  | date                      | NO   |     | NULL    |                |

| return\_date | date                      | YES  |     | NULL    |                |

| status      | enum('issued','returned') | NO   |     | NULL    |                |

+-------------+---------------------------+------+-----+---------+----------------+

6 rows in set (0.00 sec)

**mysql> DESC Fines;**

+-------------+---------------+------+-----+---------+----------------+

| Field       | Type          | Null | Key | Default | Extra          |

+-------------+---------------+------+-----+---------+----------------+

| fine\_id     | int           | NO   | PRI | NULL    | auto\_increment |

| user\_id     | int           | YES  | MUL | NULL    |                |

| issue\_id    | int           | YES  | MUL | NULL    |                |

| fine\_amount | decimal(10,2) | NO   |     | NULL    |                |

| fine\_reason | varchar(255)  | NO   |     | NULL    |                |

| fine\_date   | date          | NO   |     | NULL    |                |

+-------------+---------------+------+-----+---------+----------------+

6 rows in set (0.00 sec)

1. **Creating Triggers:**
   1. **Issue a Book to a User**

**mysql> CREATE TRIGGER update\_quantity\_available AFTER INSERT ON BookIssues**

**-> FOR EACH ROW**

**-> BEGIN**

**->     UPDATE Books**

**->     SET quantity\_available = quantity\_available - 1**

**->     WHERE book\_id = NEW.book\_id;**

**-> END;**

**-> //**

Query OK, 0 rows affected (0.03 sec)

* 1. **Return a Book from a User**

**mysql> CREATE TRIGGER update\_quantity\_available\_return BEFORE DELETE ON BookIssues**

**-> FOR EACH ROW**

**-> BEGIN**

**->     UPDATE Books**

**->     SET quantity\_available = quantity\_available + 1**

**->     WHERE book\_id = OLD.book\_id;**

**-> END;**

**-> //**

Query OK, 0 rows affected (0.04 sec)

* 1. **Calculate Fines for Late Returns**

**mysql> CREATE TRIGGER calculate\_fine AFTER INSERT ON BookIssues**

**-> FOR EACH ROW**

**-> BEGIN**

**->     DECLARE fine\_days INT;**

**->     DECLARE fine\_amount DECIMAL(10, 2);**

**->**

**->     IF NEW.return\_date IS NOT NULL   
 AND NEW.return\_date > NEW.issue\_date THEN**

**->         SET fine\_days = DATEDIFF(NEW.return\_date, NEW.issue\_date);**

**->         IF fine\_days > 15 THEN**

**->             SET fine\_amount = (fine\_days - 15) \* 0.5;**

**->             INSERT INTO Fines (user\_id, issue\_id, fine\_amount,   
 fine\_reason, fine\_date)**

**->             VALUES (NEW.user\_id, NEW.issue\_id, fine\_amount,   
 'Late return', NOW());**

**->         END IF;**

**->     END IF;**

**-> END;**

**-> //**

Query OK, 0 rows affected (0.13 sec)

1. **Inserting Sample Data:**

**mysql> -- Sample data for Users Table**

**mysql> INSERT INTO Users (username, password, role) VALUES**

**-> ('admin', 'admin123', 'admin'),**

**-> ('librarian', 'librarian123', 'librarian'),**

**-> ('user1', 'user123', 'member'),**

**-> ('user2', 'user456', 'member');**

Query OK, 4 rows affected (0.02 sec)

Records: 4  Duplicates: 0  Warnings: 0

**mysql> -- Sample data for Books Table**

**mysql> INSERT INTO Books (title, author, isbn, publication\_year, quantity\_available) VALUES**

**-> ('Book1', 'Author1', '978-3-16-148410-0', '2020', 6),**

**-> ('Book2', 'Author2', '978-3-16-148411-0', '2019', 4),**

**-> ('Book3', 'Author3', '978-3-16-148412-0', '2021', 7);**

Query OK, 3 rows affected (0.02 sec)

Records: 3  Duplicates: 0  Warnings: 0

1. **Accessing table data:**

**mysql> select \* from users;**

+---------+-----------+--------------+-----------+

| user\_id | username  | password     | role      |

+---------+-----------+--------------+-----------+

|       1 | admin     | admin123     | admin     |

|       2 | librarian | librarian123 | librarian |

|       3 | user1     | user123      | member    |

|       4 | user2     | user456      | member    |

+---------+-----------+--------------+-----------+

4 rows in set (0.00 sec)

**mysql> select \* from books;**

+---------+-------+---------+-------------------+------------------+--------------------+

| book\_id | title | author  | isbn              | publication\_year | quantity\_available |

+---------+-------+---------+-------------------+------------------+--------------------+

|       1 | Book1 | Author1 | 978-3-16-148410-0 |             2020 |                  6 |

|       2 | Book2 | Author2 | 978-3-16-148411-0 |             2019 |                  4 |

|       3 | Book3 | Author3 | 978-3-16-148412-0 |             2021 |                  7 |

+---------+-------+---------+-------------------+------------------+--------------------+

3 rows in set (0.00 sec)

**mysql> select \* from bookissues;**

Empty set (0.01 sec)

**mysql> select \* from fines;**

Empty set (0.00 sec)

1. **Adding data to BookIssues table and checking auto trigger for books and fines table:**

**mysql> INSERT INTO BookIssues (user\_id, book\_id, issue\_date, return\_date, status)**

**-> VALUES**

**-> (4, 2, '2024-04-05', NULL, 'issued');**

Query OK, 1 row affected (0.01 sec)

**mysql> INSERT INTO BookIssues (user\_id, book\_id, issue\_date, return\_date, status)**

**-> VALUES**

**-> (3, 3, '2024-04-01', '2024-04-11', 'returned');**

Query OK, 1 row affected (0.01 sec)

**mysql> INSERT INTO BookIssues (user\_id, book\_id, issue\_date, return\_date, status)**

**-> VALUES**

**-> (3, 1, '2024-04-02', '2024-04-20', 'returned');**

Query OK, 1 row affected (0.01 sec)

**mysql> SELECT \* FROM BookIssues;**

+----------+---------+---------+------------+-------------+----------+

| issue\_id | user\_id | book\_id | issue\_date | return\_date | status   |

+----------+---------+---------+------------+-------------+----------+

|        8 |       4 |       2 | 2024-04-05 | NULL        | issued   |

|        9 |       3 |       3 | 2024-04-01 | 2024-04-11  | returned |

|       10 |       3 |       1 | 2024-04-02 | 2024-04-20  | returned |

+----------+---------+---------+------------+-------------+----------+

3 rows in set (0.00 sec)

**mysql> SELECT book\_id, title, quantity\_available FROM books;**

+---------+-------+--------------------+

| book\_id | title | quantity\_available |

+---------+-------+--------------------+

|       1 | Book1 |                  6 |

|       2 | Book2 |                  3 |

|       3 | Book3 |                  7 |

+---------+-------+--------------------+

3 rows in set (0.00 sec)

**mysql> SELECT \* FROM fines;**

+---------+---------+----------+-------------+-------------+------------+

| fine\_id | user\_id | issue\_id | fine\_amount | fine\_reason | fine\_date  |

+---------+---------+----------+-------------+-------------+------------+

|       1 |       3 |       10 |        1.50 | Late return | 2024-04-27 |

+---------+---------+----------+-------------+-------------+------------+

1 row in set (0.00 sec)

1. **Returning a Book:**

**mysql> DELETE FROM BookIssues**

**-> WHERE issue\_id = 8;**

Query OK, 1 row affected (0.01 sec)

**mysql> SELECT \* FROM BookIssues;**

+----------+---------+---------+------------+-------------+----------+

| issue\_id | user\_id | book\_id | issue\_date | return\_date | status   |

+----------+---------+---------+------------+-------------+----------+

|        9 |       3 |       3 | 2024-04-01 | 2024-04-11  | returned |

|       10 |       3 |       1 | 2024-04-02 | 2024-04-20  | returned |

+----------+---------+---------+------------+-------------+----------+

2 rows in set (0.00 sec)

**mysql> SELECT book\_id, title, quantity\_available FROM books;**

+---------+-------+--------------------+

| book\_id | title | quantity\_available |

+---------+-------+--------------------+

|       1 | Book1 |                  6 |

|       2 | Book2 |                  4 |

|       3 | Book3 |                  7 |

+---------+-------+--------------------+

3 rows in set (0.00 sec)

**Learning outcome of the Experiment:**

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |