

# ISI-6, RIICO Institutional Area, Sitapura, Jaipur-302022, Rajasthan

**Phone/Fax: 0141-2770790-92,** [**www.pce.poornima.org**](http://www.pce.poornima.org/)

DATA BASE MANAGEMENT SYSTEM

Lab Manual

# (Lab Code: 4CS4-22)

**4th Semester, 2nd Year**



**Department of Computer Engineering**

# Session: 2018-19

**TABLE OF CONTENT**

|  |  |  |
| --- | --- | --- |
| **S. No.** | **Topic/Name of Experiment** | **Page Number** |
| **GENERAL DETAILS** | | |
| 1 | Vision & Mission of Institute and Department | iii |
| 2 | RTU Syllabus and Marking Scheme | iv |
| 3 | Lab Outcomes and its Mapping with POs and PSOs | v-vii |
| 4 | Rubrics for Lab. | viii-ix |
| 5 | Lab Conduction Plan | x |
| 6 | Lab Rotar Plan | xi |
| 7 | General Lab Instructions | xii-xiii |
| 8 | Lab Specific Safety Rules | xiii |
| **LIST OF EXEPERIMENTS (AS PER RTU SYLLABUS)** | | |
| 1 | Zero Lab | 1-3 |
| 2 | Introduction: Objective, scope and outcome of the course. | 4 |
| 3 | Design Database and create required tables Bank or College using SQL statements like CREATE, INSERT INTO | 5-7 |
| 4 | Apply the constraints like Primary key, foreign key, NOT NULL to the tables. | 8-10 |
| 5 | Write SQL statement for implementing ALTER, UPDATE and DELETE | 11-16 |
| 6 | Write SQL statements like SELECT, WHERE, ORDER BY, DESC BY | 17-19 |
| 7 | Write the query for implementing the following functions: MAX (), MIN (), AVG (), SUM () and COUNT (). | 20-24 |
| 8 | Write the query to implement Joins. | 25-27 |
| 9 | Perform the queries for triggers. | 28-30 |
| 10 | Perform the following operation for demonstration the insertion, updation and deletion | 31-32 |
| 11 | using the referential integrity constraints | 33-35 |
| 12 | Write the query for creating the users and their role | 36-38 |
| 13 | Write the query to create the views. | 39-40 |
| 14 | Beyond the syllabus Exp-1-Declare a Cursor. | 41-42 |
| 15 | Beyond the syllabus Exp-2-Creating a function | 43-44 |

**INSTITUTE MISSION & VISION**

**VISION**

To create knowledge-based society with scientific temper, team spirit and dignity of labor to face the global competitive challenges

**MISSION**

To evolve and develop skill-based systems for effective delivery of knowledge so as to equip young professionals with dedication & commitment to excellence in all spheres of life.

**DEPARTMENT MISSION & VISION**

**VISION**

Evolve as a center of excellence with wider recognition and to adapt the rapid innovation in Computer Engineering.

**MISSION**

* To provide a learning-centered environment that will enable students and faculty members to achieve their goals empowering them to compete globally for the most desirable careers in academia and industry.
* To contribute significantly to the research and the discovery of new arenas of knowledge and methods in the rapid developing field of Computer Engineering.
* To support society through participation and transfer of advanced technology from one sector to another.

# RTU SYLLABUS AND MARKING SCHEME

|  |  |  |
| --- | --- | --- |
| **4CS4-22: DATA BASE MANAGEMENT SYSTEM LAB** | | |
| **Credit: 1.5** | | **Max. Marks:100(IA:60, ETE:40)** |
| **0L+0T+3P** | | **End Term Exam: 3 Hours** |
| **S. No.** |  | |
| 1 | Introduction: Objective, scope and outcome of the course. | |
| 2 | Design Database and create required tables Bank or College using SQL statements like CREATE, INSERT INTO | |
| 3 | Apply the constraints like Primary key, foreign key, NOT NULL to the tables. | |
| 4 | Write SQL statement for implementing ALTER, UPDATE and DELETE | |
| 5 | Write SQL statements like SELECT, WHERE, ORDER BY, DESC BY | |
| 6 | Write the query for implementing the following functions: MAX (), MIN (), AVG (), SUM () and COUNT (). | |
| 7 | Write the query to implement Joins. | |
| 8 | Perform the queries for triggers. | |
| 9 | Perform the following operation for demonstration the insertion, updation and deletion | |
| 10 | using the referential integrity constraints | |
| 11 | Write the query for creating the users and their role | |
| 12 | Write the query to create the views. | |
| 13 | Declare a Cursor. | |
| 14 | Creating a function | |

## EVALUATION SCHEME

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **I+II Mid Term Examination** | | | **Attendance and performance** | | | **End Term Examination** | | | **Total Marks** |
| **Experiment** | **Viva** | **Total** | **Attendance** | **Performance** | **Total** | **Experiment** | **Viva** | **Total** |
|  |  |  |  |  |  |  |  |  | 75 |

**DISTRIBUTION OF MARKS FOR EACH EXPERIMENT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attendance** | **Record** | **Performance** | **Total** |
| 2 | 3 | 5 | 10 |

# LAB OUTCOME AND ITS MAPPING WITH PO & PSO

## LAB OUTCOMES

After completion of this course, students will be able to –

|  |  |
| --- | --- |
| **4CS4-22.1** | Select appropriate technique to design database and schemas for a given application using DDL/DML SQL commands. |
| **4CS4-22.2** | Apply the concept of Integrity Rules and Constraints to ensure accurate and error free data. |
| **4CS4-22.3** | Identify solutions for database update using pre store Procedures and Triggers |
| **4CS4-22.4** | Compare the constraints primary key and foreign key between primary table and secondary table |
| **4CS4-22.5** | Construct Views to simplify and reduce complexity of database schema |
| **4CS4-22.6** | Decision to users with different types of privileges and check user’s existence in database. |
| **4CS4-22.7** | Assemble records from multiple tables in database through Inner joins and Outer joins. |

## LO-PO-PSO MAPPING MATRIX OF COURSE

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **LO/PO/PSO** | **PO1** | **PO2** | **PO3** | **PO4** | **PO5** | **PO6** | **PO7** | **PO8** | **PO9** | **PO10** | **PO11** | **PO12** | **PSO1** | **PSO2** | **PSO3** |
| **4CS4-22.1** | - | - | 2 | - | 2 | - | - | - | - | - | - | - | 1 | 2 | - |
| **4CS4-22.2** | 3 | - | - | - | - | - | - | - | - | - | - | - | - | 2 | - |
| **4CS4-22.3** | - | - | 2 | - | - | - | - | - | - | - | - | - | 2 | - | - |
| **4CS4-22.4** | - | 2 | - | - | - | - | - | - | - | - | - | - | - | 1 | - |
| **4CS4-22.5** | - | - | 3 | - | - | - | - | - | - | - | - | - | 2 | - | - |
| **4CS4-22.6** | - | - | - | 2 | - | - | - | - | - | - | - | - | - | 2 | - |
| **4CS4-22.7** | - | - | 3 | - | - | - | - | - | - | - | - | - | 1 | 1 | - |

**PROGRAM OUTCOMES (POs)**

|  |  |
| --- | --- |
| **PO1** | **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems. |
| **PO2** | **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences. |
| **PO3** | **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations. |
| **PO4** | **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions. |
| **PO5** | **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations. |
| **PO6** | **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice. |
| **PO7** | **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development. |
| **PO8** | **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice. |
| **PO9** | **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings. |
| **PO10** | **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions. |
| **PO11** | **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments. |
| **PO12** | **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change. |

## PROGRAM SPECIFIC OUTCOMES (PSOs)

|  |  |
| --- | --- |
| **PSO1** | The ability to understand and apply knowledge of mathematics, system analysis & design, Data Modelling, Cloud Technology, and latest tools to develop  computer-based solutions in the areas of system software, Multimedia, Web Applications, Big data analytics, IOT, Business Intelligence and Networking systems |
| **PSO2** | The ability to understand the evolutionary changes in computing, apply standards and ethical practices in project development using latest tools & Technologies to solve societal problems and meet the challenges of the future. |
| **PSO3** | The ability to employ modern computing tools and platforms to be an entrepreneur, lifelong learning and higher studies |

**Laboratory Evaluation Rubrics:**

# RUBRICS FOR LAB

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **S.**  **No.** | **Crit eria** | **Sub Criteria and Marks Distribution** | | | **Outstanding (>90%)** | **Admirable (70-90%)** | **Average (40-69%)** | **Inadequate (<40%)** |
| **Mid-Term** | **End-Team** | **Continues Evaluation** |
| **A** | **PERFORMANCE (PO1, PO8, PO9)** | Procedure Followed  M.M. 50 = 3  M.M. 75 = 4  M.M. 100 = 6 | Procedure Followed  M.M. 50 = 3  M.M. 75 = 4  M.M. 100 = 6 | Procedure Followed  M.M. 50 = 1  M.M. 75 = 2  M.M. 100 = 2 | * All possible system and Input/ Output variables are taken into account * Performance measures are properly defined * Experimental scenarios are very well defined | * Most of the system and Input/ Output variables are taken into account * Most of the Performance measures are properly defined * Experimental scenarios are defined correctly | * Some of the system and Input/ Output variables are taken into account * Some of the Performance measures are properly defined * Experimental scenarios are defined but not sufficient | * System and Input/ Output variables are not defined * Performance measures are not properly defined • Experimental scenarios not defined |
| Individual/Tea m Work  M.M. 50 = 3  M.M. 75 =4  M.M. 100 = 6 | Individual/Tea m Work  M.M. 50 = 3  M.M. 75 =4  M.M. 100 = 6 | Individual/Tea m Work  M.M. 50 = 1  M.M. 75 = 2  M.M. 100 = 2 | * Coordination among the group members in performing the experiment was excellent | * Coordination among the group members in performing the experiment was good | * Coordination among the group members in performing the experiment was average | * Coordination among the group members in performing the experiment was very poor |
| Precision in data collection  M.M. 50 = 3  M.M. 75 = 4  M.M. 100 = 6 | Precision in data collection  M.M. 50 = 3  M.M. 75 = 4  M.M. 100 = 6 | Precision in data collection  M.M. 50 = 2  M.M. 75 = 2  M.M. 100 = 4 | * Data collected is correct in size and from the experiment performed | * Data collected is appropriate in size and but not from proper sources. | * Data collected is not so appropriate in size and but from proper sources. | * Data collected is neither appropriate in size and norfrom proper sources |
| **B** | **LAB RECORD/WRITTEN WORK (PO1, PO8, PO10)** | NA | NA | Timing of Evaluation of Experiment  M.M. 50 = 3  M.M. 75 = 4  M.M. 100 = 6 | * On the Same Date of Performance | * On the Next Turn from Performance | * Before Dead Line | * On the Dead Line |
| Data Analysis  M.M. 50 = 3  M.M. 75 =5  M.M. 100 = 6 | Data Analysis  M.M. 50 = 3  M.M. 75 =5  M.M. 100 = 6 | Data Analysis  M.M. 50 = 2  M.M. 75 = 3  M.M. 100 = 4 | * Data collected is exhaustively analyzed & appropriate features are selected | * Data collected is analyzed & but appropriate features are not selected | * Data collected is not analyzed properly. •Features selected are not appropriate | * Data collected is not analyzed & the features are not selected |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Results and Discussion  M.M. 50 = 3  M.M. 75 =5  M.M. 100 = 6 | Results and Discussion  M.M. 50 = 3  M.M. 75 =5  M.M. 100 = 6 | Results and Discussion  M.M. 50 = 2  M.M. 75 = 3  M.M. 100 = 4 | * All results are very well presented with all variables * Well prepared neat diagrams/plots/ tables for all performance measured * Discussed critically behavior of the system with reference to performance measures * Very well discussed pros n cons of outcome | * All results presented but not all variables mentioned * Prepared diagrams /plots/ tables for all performance measured but not so neat * Discussed behavior of the system with reference to performance measures but not critical * Discussed pros n cons of outcome in brief | * Partial results are included * Prepared diagrams /plots/ tables partially for the performance measures * Behavior of the system with reference to performance measures has been superficially presented * Discussed pros n cons of outcome but not so relevant | * Results are included but not as per experimental scenarios * No proper diagrams /plots/ tables are prepared * Behavior of the system with reference to performance measures has not been presented * Did not discuss pros n cons of outcome |
| **C** | **VIVA (PO1, PO10)** | Way of presentation  M.M. 50 = 2.5  M.M. 75 = 4  M.M. 100 = 5 | Way of presentation  M.M. 50 = 2.5  M.M. 75 = 4  M.M. 100 = 5 | Way of presentation  M.M. 50 = 2  M.M. 75 = 3  M.M. 100 = 4 | * Presentation was very good | * Presentation was good | * Presentation was satisfactory | * Presentation was poor |
| Concept Explanation  M.M. 50 = 2.5  M.M. 75 = 4  M.M. 100 = 5 | Concept Explanation  M.M. 50 = 2.5  M.M. 75 = 4  M.M. 100 = 5 | Concept Explanation  M.M. 50 = 2  M.M. 75 = 3  M.M. 100 = 4 | * Conceptual explanation was excellent | * Conceptual explanation was good | * Conceptual explanation was somewhat good | * Conceptual explanation was Poor |
| **D** | **ATTENDA NCE** | NA | NA | **A**ttendance  M.M. 50 = 5  M.M. 75 =8  M.M. 100 =10 | * Present more than 90% of lab sessions | * Present more than 75% of lab sessions | * Present more than 60% of lab sessions | * Present in less than 60% lab sessions |

**LAB CONDUCTION PLAN**

**Total number of Experiments -14**

**Total number of turns required -14**

**Number of turns required for: -**

|  |  |
| --- | --- |
| **Experiment Number** | **Scheduled Week** |
| Experiment -1 | Week 1 |
| Experiment -2 | Week 2 |
| Experiment -3 | Week 3 |
| Experiment -4 | Week 4 |
| Experiment -5 | Week 5 |
| **I Mid Term** | **Week 6** |
| Experiment -6 | Week 7 |
| Experiment -7 | Week 8 |
| Experiment -8 | Week 9 |
| Experiment-9 | Week 10 |
| Experiment-10 | Week 11 |
| Experiment-11 | Week 12 |
| Experiment-12 | Week 13 |
| Experiment-13 | Week 14 |
| Experiment-14 | Week 15 |
| **II Mid Term** | **Week 16** |

**DISTRIBUTION OF LAB HOURS**

|  |  |  |  |
| --- | --- | --- | --- |
| **S. No.** | **Activity** | **Distribution of Lab Hours** | |
| **Time (180 minute)** | **Time (120 minute)** |
| 1 | Attendance | 5 | 5 |
| 2 | Explanation of Experiment & Logic | 30 | 30 |
| 3 | Performing the Experiment | 60 | 30 |
| 4 | File Checking | 40 | 20 |
| 5 | Viva/Quiz | 30 | 20 |
| 6 | Solving of Queries | 15 | 15 |

# LAB ROTAR PLAN ROTOR-1

|  |  |
| --- | --- |
| **Ex. No.** | **NAME OF EXPERIMENTS** |
| 1 | Design a Database and create required tables. For e.g. Bank, College Database |
| 2 | Apply the constraints like Primary Key, Foreign key, NOT NULL to the tables. |
| 3 | Write a SQL statement for implementing ALTER, UPDATE and DELETE. |
| 4 | Write the queries to implement the joins. |
| 5 | Write the query for implementing the following functions: MAX (), MIN (), AVG () and COUNT (). |
| 6 | Write the query to implement the concept of Integrity constrains. |

**ROTOR-2**

|  |  |
| --- | --- |
| **Ex. No.** | **NAME OF EXPERIMENTS** |
| 7 | Write the query to create the views. |
| 8 | Perform the queries for triggers. |
| 9 | Perform the following operation for demonstrating the insertion, updation and deletion. |
| 10 | Using the referential integrity constraints. |
| 11 | Write the query for creating the users and their role. |
| 12 | Declare a Cursor. |
| 13 | Creating a function |

## GENERAL LAB INSTRUCTIONS

**DO’S**

1. Enter the lab on time and leave at proper time.
2. Wait for the previous class to leave before the next class enters.
3. Keep the bag outside in the respective racks.
4. Utilize lab hours in the corresponding.
5. Turn off the machine before leaving the lab unless a member of lab staff has specifically told you not to do so.
6. Leave the labs at least as nice as you found them.
7. If you notice a problem with a piece of equipment (e.g., a computer doesn't respond) or the room in general (e.g., cooling, heating, lighting) please report it to lab staff immediately. Do not attempt to fix the problem yourself.

## DON’TS

1. Don't abuse the equipment.
2. Do not adjust the heat or air conditioners. If you feel the temperature is not properly set, inform lab staff; we will attempt to maintain a balance that is healthy for people and machines.
3. Do not attempt to reboot a computer. Report problems to lab staff.
4. Do not remove or modify any software or file without permission.
5. Do not remove printers and machines from the network without being explicitly told to do so by lab staff.
6. Don't monopolize equipment. If you're going to be away from your machine for more than 10 or 15 minutes, log out before leaving. This is both for the security of your account, and to ensure that others are able to use the lab resources while you are not.
7. Don’t use internet, internet chat of any kind in your regular lab schedule.
8. Do not download or upload of MP3, JPG or MPEG files.
9. No games are allowed in the lab sessions.
10. No hardware including USB drives can be connected or disconnected in the labs without prior permission of the lab in-charge.
11. No food or drink is allowed in the lab or near any of the equipment. Aside from the fact that it leaves a mess and attracts pests, spilling anything on a keyboard or other piece of computer equipment could cause permanent, irreparable, and costly damage. (and in fact *has*) If you need to eat or drink, take a break and do so in the canteen.
12. Don’t bring any external material in the lab, except your lab record, copy and books.
13. Don’t bring the mobile phones in the lab. If necessary, then keep them in silence mode.
14. Please be considerate of those around you, especially in terms of noise level. While labs are a natural place for conversations of all types, kindly keep the volume turned down.
15. If you are having problems or questions, please go to either the faculty, lab in-charge or the lab supporting staff. They will help you. We need your full support and cooperation for smooth functioning of the lab.

## LAB SPECIFIC SAFETY RULES

### Before entering in the lab

1. All the students are supposed to prepare the theory regarding the next experiment/ Program.
2. Students are supposed to bring their lab records as per their lab schedule.
3. Previous experiment/program should be written in the lab record.
4. If applicable trace paper/graph paper must be pasted in lab record with proper labeling.
5. All the students must follow the instructions, failing which he/she may not be allowed in

the lab.

### While working in the lab

1. Adhere to experimental schedule as instructed by the lab in-charge/faculty.
2. Get the previously performed experiment/ program signed by the faculty/ lab in charge.
3. Get the output of current experiment/program checked by the faculty/ lab in charge in the

lab copy.

1. Each student should work on his/her assigned computer at each turn of the lab.
2. Take responsibility of valuable accessories.

**Zero Lab**

**What is MYSQL**MySQL is a relational database management system based on the Structured Query Language, which is the popular language for accessing and managing the records in the database. MySQL is open-source and free software under the GNU license. It is supported by Oracle Company.

MySQL is a Relational Database Management System (RDBMS) software that provides many things, which are as follows:

* It allows us to implement database operations on tables, rows, columns, and indexes.
* It defines the database relationship in the form of tables (collection of rows and columns), also known as relations.
* It provides the Referential Integrity between rows or columns of various tables.
* It allows us to updates the table indexes automatically.
* It uses many SQL queries and combines useful information from multiple tables for the end-users

MySQL is named after the daughter of co-founder Michael Widenius whose name is "My".

**To communicate with Oracle, mysql supports the following categories of commands:**

**1. Data Definition Language**

Create, Alter, Drop and Truncate

**2. Data Manipulation Language**

Insert, Update, Delete and Select

**3. Transaction Control Language**

Commit, Rollback and Save point

**4. Data Control Language**

Grant and Revoke

**MySQL uses many different data types broken into three categories −**

• Numeric

• Date and Time

• String Types.

**Numeric Data Types**

MySQL uses all the standard ANSI SQL numeric data types, so if you're coming to MySQL from a different database system, these definitions will look familiar to you.

The following list shows the common numeric data types and their descriptions −

**INT** − A normal-sized integer that can be signed or unsigned. If signed, the allowable range is from - 2147483648 to 2147483647. If unsigned, the allowable range is from 0 to 4294967295. You can specify a width of up to 11 digits.

**TINYINT** − A very small integer that can be signed or unsigned. If signed, the allowable range is from -128 to 127. If unsigned, the allowable range is from 0 to 255. You can specify a width of up to 4 digits.

**SMALLINT** − A small integer that can be signed or unsigned. If signed, the allowable range is from -32768 to 32767. If unsigned, the allowable range is from 0 to 65535. You can specify a width of up to 5 digits.

**MEDIUMINT** − A medium-sized integer that can be signed or unsigned. If signed, the allowable range is from -8388608 to 8388607. If unsigned, the allowable range is from 0 to 16777215. You can specify a width of up to 9 digits.

**BIGINT** − A large integer that can be signed or unsigned. If signed, the allowable range is from - 9223372036854775808 to9223372036854775807. If unsigned, the allowable range is from 0 to 18446744073709551615. You can specify a width of up to 20 digits.

• **FLOAT(M,D) −** A floating-point number that cannot be unsigned. You can define the display length (M) and the number of decimals (D). This is not required and will default to 10,2, where 2 is the number of decimals and 10 is the total number of digits (including decimals). Decimal precision can go to 24 places for a FLOAT.

**• DOUBLE(M,D)** − A double precision floating-point number that cannot be unsigned. You can define the display length (M) and the number of decimals (D). This is not required and will default to 16,4, where 4 is the number of decimals. Decimal precision can go to 53 places for a DOUBLE. REAL is a synonym for DOUBLE.

• **DECIMAL(M,D)** − An unpacked floating-point number that cannot be unsigned. In the unpacked decimals, each decimal corresponds to one byte. Defining the display length (M) and the number of decimals (D) is required. NUMERIC is a synonym for DECIMAL.

**Date and Time Types**

**The MySQL date and time datatypes are as follows −**

**• DATE** − A date in YYYY-MM-DD format, between 1000-01-01 and 9999-12-31. For example, December 30th, 1973 would be stored as 1973-12-30.

**• DATETIME** − A date and time combination in YYYY-MM-DD HH:MM:SS format, between 1000- 01-01 00:00:00 and 9999-12-31 23:59:59. For example, 3:30 in the afternoon on December 30th , 1973 would be stored as 1973-12-30 15:30:00.

**• TIME − Stores the time in a HH:MM: SS format.**

**• YEAR(M)** − Stores a year in a 2-digit or a 4-digit format. If the length is specified as 2 (for example YEAR (2)), YEAR can be between 1970 to 2069 (70 to 69). If the length is specified as 4, then YEAR can be 1901 to 2155. The default length is 4.

**String Types**

Although the numeric and date types are fun, most data you'll store will be in a string format. This list describes the common string datatypes in MySQL.

**• CHAR(M) −** A fixed-length string between 1 and 255 characters in length (for example CHAR(5)), right-padded with spaces to the specified length when stored. Defining a length is not required, but the default is 1.

**• VARCHAR(M)** − A variable-length string between 1 and 255 characters in length. For example, VARCHAR(25). You must define a length when creating a VARCHAR field.

## EXPERIMENT-1

### OBJECTIVE

Introduction: Objective, scope and outcome of the course.

**THEORY**

Data base system is a system to achieve an organized, store a large number of dynamical associated data, facilitate for multi-user accessing to computer hardware, software and data, that it is a computer system with database technology.

* Data constitute the building blocks of information
* Information is produced by processing data
* Information is used to reveal the meaning of data.
* Accurate, relevant and timely information is the key to good decision making
* Good decision making is the key to organizational survival in a global environment

**SCOPE**

The scope of this course is to explore to Design database schema for a given application and apply normalization. Acquire skills in using SQL commands for data definition and data manipulation. Develop solutions for database applications using procedures, cursors and triggers

**OUTCOMES**

1. In drawing the ER, EER, and UML Diagrams.
2. In analyzing the business requirements and producing a viable model for the
3. implementation of the database.
4. In converting the entity-relationship diagrams into relational tables.
5. To develop appropriate Databases to a given problem that integrates ethical,
6. social, legal, and economic concerns.

**EXPERIMENT-2**

**OBJECTIVE**

Design Database and create required tables Bank or College using SQL statements

**INTRODUCTION**

Introduction about Database and SQL

**What is Database?**

A database management system (or DBMS) is essentially nothing more than a computerized data-keeping system. Users of the system are given facilities to perform several kinds of operations on such a system for either manipulation of the data in the database or the management of the database structure itself.

**DDL - Data Definition Language**

|  |  |
| --- | --- |
| **Sr.No.** | **Command & Description** |
| 1 | **CREATE**  Creates a new table, a view of a table, or other object in the database. |
| 2 | **ALTER**  Modifies an existing database object, such as a table. |
| 3 | **DROP**  Deletes an entire table, a view of a table or other objects in the database. |

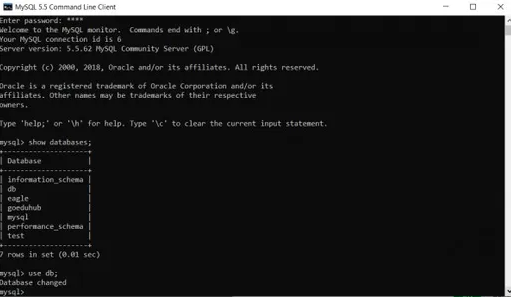
**DML - Data Manipulation Language**

|  |  |
| --- | --- |
| **Sr.No.** | **Command & Description** |
| 1 | **SELECT**  Retrieves certain records from one or more tables. |
| 2 | **INSERT**  Creates a record. |
| 3 | **UPDATE**  Modifies records. |
| 4 | **DELETE**  Deletes records. |

**DCL - Data Control Language**

|  |  |
| --- | --- |
| **Sr.No.** | **Command & Description** |
| 1 | **GRANT**  Gives a privilege to user. |
| 2 | **REVOKE**  Takes back privileges granted from user. |

**Step-1 :**if you already have created data bases then use them else create new database.



**Step-2:** creation of database

**Syntax:**   create database database name;

Capture1.PNG

**Step-3:**usethe data base you have created

Cap.PNG

**Step-4:** creation of table in existing database

**Syntax:**

create table table\_name ( column1 datatype , column2 datatype , column3 datatype, ....);create table college(stu\_id int(10),stu\_name varchar(30), stu\_dept varchar(20),phonenumber int(10),dob date,address varchar(50),fathers\_name varchar(25),mothers\_name varchar(25));

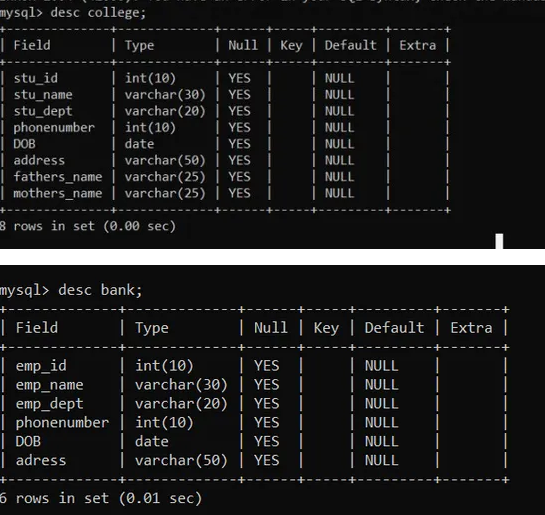
create table college(stu\_id int(10),stu\_name varchar(30), stu\_dept varchar(20),phonenumber int(10),dob date,address varchar(50)));

**Step-5:**

desc or describe command shows the structure of table which include name of the column, data-type of column and the nullability which means, that column can contain null values or not.

**Syntax:**

describe tablename;   or      desc tablename;

****

**EXPERIMENT-3**

**OBJECTIVE**

Apply the constraints like Primary key, foreign key, NOT NULL to the tables.

**INTRODUCTION**

**Primary key:** The PRIMARY KEY constraint uniquely identifies each record in a table. Primary keys must contain UNIQUE values, and cannot contain NULL values. A table can have only one primary key and, in the table, this primary key can consist of single or multiple columns (fields).

**NOT NULL**: By default, a column can hold NULL values. The constraint enforces a column to NOT accept NULL values. This enforces a field to always contain a value, which means that NOT NULL you cannot insert a new record, or update a record without adding a value to this field.

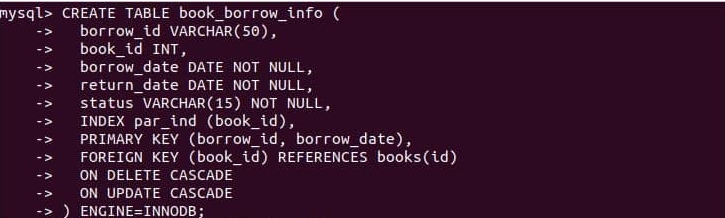
**Foreign key**: the foreign key constraint is used to prevent actions that would destroy links between tables. A foreign key is a field (or collection of fields) in one table, that refers to the [primary key](https://www.w3schools.com/sql/sql_primarykey.asp) in another table. The table with the foreign key is called the child table, and the table with the primary key is called the referenced or parent table.

**Step-1** :

[create](https://search.oracle.com/search/search?group=MySQL&q=CREATE) [database](https://search.oracle.com/search/search?group=MySQL&q=DATABASE) library;  
[use](https://search.oracle.com/search/search?group=MySQL&q=USE) library;  
  
**Step-2** :

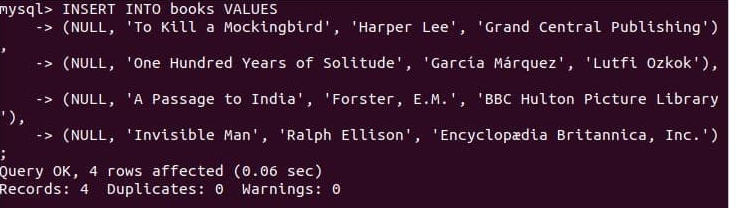
[create](https://search.oracle.com/search/search?group=MySQL&q=CREATE) [table](https://search.oracle.com/search/search?group=MySQL&q=TABLE) books (  
id [int](https://search.oracle.com/search/search?group=MySQL&q=INT) [not](https://dev.mysql.com/doc/refman/5.1/en/non-typed-operators.html) [null](https://search.oracle.com/search/search?group=MySQL&q=NULL) [auto\_increment](https://search.oracle.com/search/search?group=MySQL&q=AUTO_INCREMENT),  
title [varchar](https://search.oracle.com/search/search?group=MySQL&q=VARCHAR)(50) [not](https://dev.mysql.com/doc/refman/5.1/en/non-typed-operators.html) [null](https://search.oracle.com/search/search?group=MySQL&q=NULL),  
author [varchar](https://search.oracle.com/search/search?group=MySQL&q=VARCHAR)(50) [not](https://dev.mysql.com/doc/refman/5.1/en/non-typed-operators.html) [null](https://search.oracle.com/search/search?group=MySQL&q=NULL),  
publisher [varchar](https://search.oracle.com/search/search?group=MySQL&q=VARCHAR)(50) [not](https://dev.mysql.com/doc/refman/5.1/en/non-typed-operators.html) [null](https://search.oracle.com/search/search?group=MySQL&q=NULL),  
[primary key](https://search.oracle.com/search/search?group=MySQL&q=PRIMARY%20KEY) (id)  
) [engine](https://search.oracle.com/search/search?group=MySQL&q=ENGINE)=[innodb](https://search.oracle.com/search/search?group=MySQL&q=INNODB);  
  
**Step-3**

[create](https://search.oracle.com/search/search?group=MySQL&q=CREATE) [table](https://search.oracle.com/search/search?group=MySQL&q=TABLE) borrowers (  
id [varchar](https://search.oracle.com/search/search?group=MySQL&q=VARCHAR)(50) [not](https://dev.mysql.com/doc/refman/5.1/en/non-typed-operators.html) [null](https://search.oracle.com/search/search?group=MySQL&q=NULL),  
name [varchar](https://search.oracle.com/search/search?group=MySQL&q=VARCHAR)(50) [not](https://dev.mysql.com/doc/refman/5.1/en/non-typed-operators.html) [null](https://search.oracle.com/search/search?group=MySQL&q=NULL),  
address [varchar](https://search.oracle.com/search/search?group=MySQL&q=VARCHAR)(50) [not](https://dev.mysql.com/doc/refman/5.1/en/non-typed-operators.html) [null](https://search.oracle.com/search/search?group=MySQL&q=NULL),  
email [varchar](https://search.oracle.com/search/search?group=MySQL&q=VARCHAR)(50) [not](https://dev.mysql.com/doc/refman/5.1/en/non-typed-operators.html) [null](https://search.oracle.com/search/search?group=MySQL&q=NULL),  
[primary key](https://search.oracle.com/search/search?group=MySQL&q=PRIMARY%20KEY) (id)  
) [engine](https://search.oracle.com/search/search?group=MySQL&q=ENGINE)=[innodb](https://search.oracle.com/search/search?group=MySQL&q=INNODB);

****

**Step-4:**

[insert](https://search.oracle.com/search/search?group=MySQL&q=INSERT) [into](https://search.oracle.com/search/search?group=MySQL&q=INTO) books [values](https://search.oracle.com/search/search?group=MySQL&q=VALUES)  
([null](https://search.oracle.com/search/search?group=MySQL&q=NULL), 'to kill a mockingbird', 'harper lee', 'grand central publishing'),  
([null](https://search.oracle.com/search/search?group=MySQL&q=NULL), 'one hundred years of solitude', 'garcia marquez', 'lutfi ozkok'),  
([null](https://search.oracle.com/search/search?group=MySQL&q=NULL), 'a passage to india', 'forster, e.m.', 'bbc hulton picture library'),  
([null](https://search.oracle.com/search/search?group=MySQL&q=NULL), 'invisible man', 'ralph ellison', 'encyclopædia britannica, inc.');  
  
[insert](https://search.oracle.com/search/search?group=MySQL&q=INSERT) [into](https://search.oracle.com/search/search?group=MySQL&q=INTO) book borrow\_info  [values](https://search.oracle.com/search/search?group=MySQL&q=VALUES)  
('123490', 1, '2020-02-15', '2020-02-25', 'returned'),  
('157643', 2, '2020-03-31', '2020-03-10', 'pending'),  
('174562', 4, '2020-04-04', '2020-04-24', 'borrowed'),  
('146788', 3, '2020-04-10', '2020-01-20', 'borrowed');

****

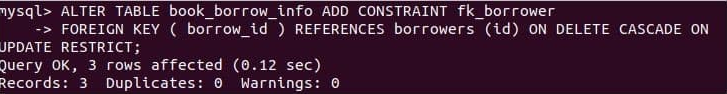
**Step-5:**

[insert](https://search.oracle.com/search/search?group=MySQL&q=INSERT) [into](https://search.oracle.com/search/search?group=MySQL&q=INTO) book\_borrow\_info [values](https://search.oracle.com/search/search?group=MySQL&q=VALUES)  
('195684', 10, '2020-04-15', '2020-04-30', 'returned');

****

**Step-6:**

alter table book\_borrow\_info add constraint fk\_borrower  
foreign key ( borrow\_id ) references borrowers (id) on delete cascade on update restrict;

****

**EXPERIMENT-4**

**OBJECTIVE**

Write SQL statement for implementing ALTER, UPDATE and DELETE.

**INTRODUCTION**

mysql alter statement is used when you want to change the name of your table or any table field. It is also used to add or delete an existing column in a table.

Drop a mysql database quickly with the mysql drop database command. It will delete the database along with all the tables, indexes, and constraints permanently

mysql update query is a DML statement used to modify the data of the mysql table within the database The mysql delete statement is used to remove the data from the mysql table.

S**tep-1: Add column in table**

**Syntax**

Alter table\_name

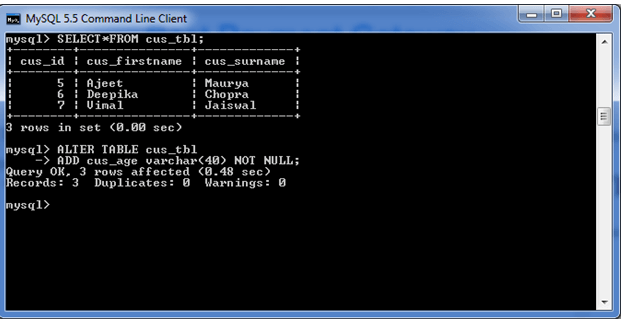
Add new\_column\_name column\_definition

[ first | after column\_name ];

alter table cus\_tbl

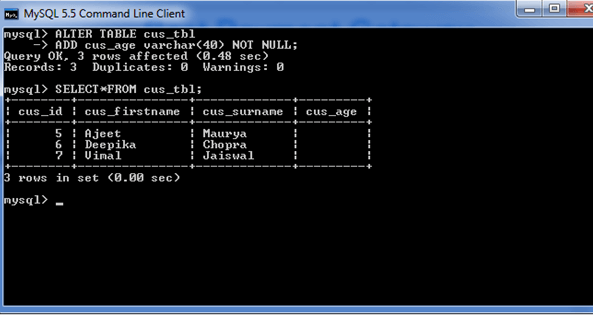
add cus\_age varchar (40) NOT NULL;

Select\* from cus\_tbl;



**Step-2:**

Select \* fromcus\_tbl;

****

**Step-3:**

Modify command is used to change the column definition of the table.

**Syntax**

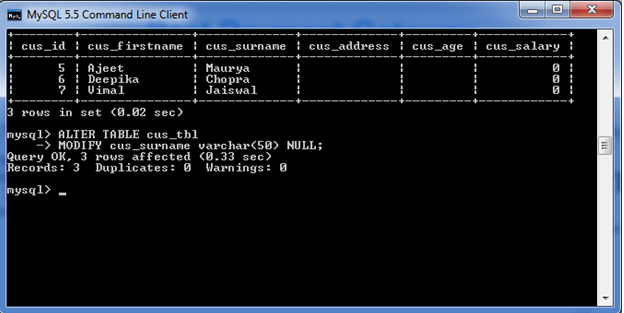
alter table table\_name

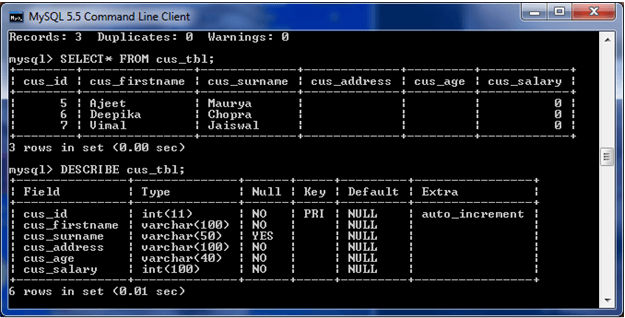
modify column\_name column\_definition

[ first | after column\_name ];

alter table cus\_tbl

modify cus\_surname varchar(50) null;





**Step-4:**

**Syntax**

alter table table\_name

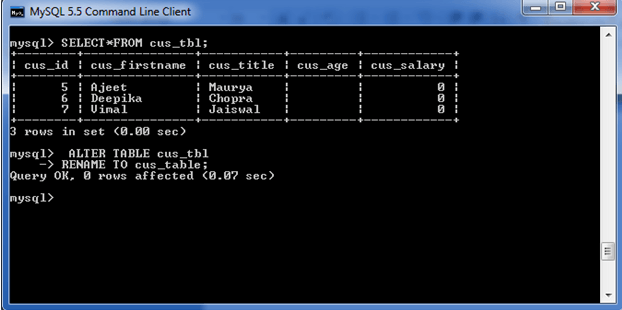
change column old\_name new\_name

column\_definition

[ first | after column\_name ]

alter table cus\_tbl

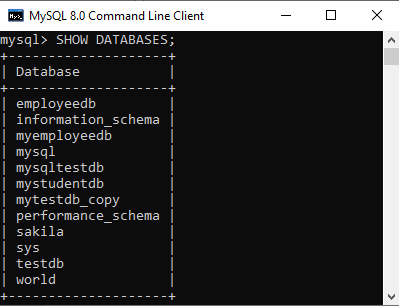
rename to cus\_table;



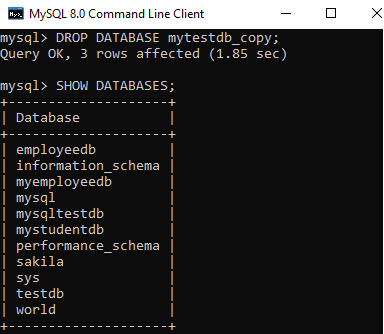
**Step-5:**

**Syntax:**

drop database [if exists] database\_name;

****

drop database mytestdb\_copy;

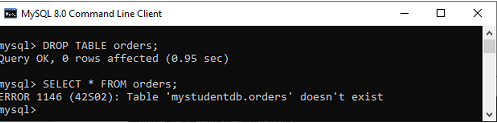
****

**Step-6:**

**Syntax:**

drop table  table\_name;

drop table  orders;



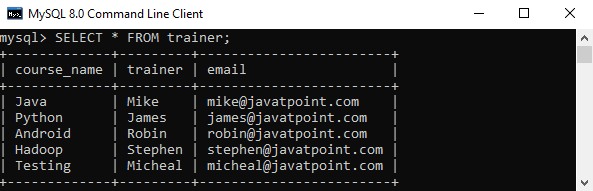
**Step-7:**

**Syntax:**

update table\_name

set column\_name1 = new-value1,

column\_name2=new-value2, ...    [where Clause]

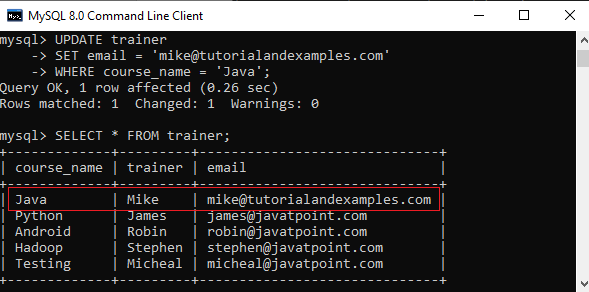


update trainer

set email = 'mike@tutorialandexamples.com'

where course\_name = 'java';

select \* from trainer;



**Step-8:**

**Syntax:**

delete from tablename <where condition>

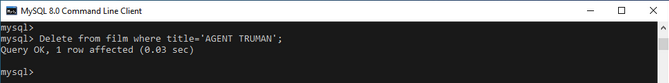
<order by ...>

<limit count>

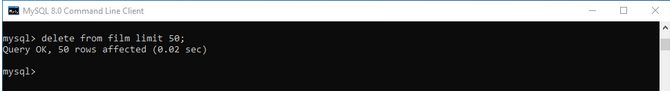
delete from language;



delete from film where title=’AGENT TRUMAN’;



delete from film limit 50;

****

**EXPERIMENT-5**

**OBJECTIVE**

Write SQL statement for implementing common SQL commands like SELECT, WHERE, ORDER BY, DESC BY.

**INTRODUCTION**

The MYSQL ORDER BY Clause is used to sort the records in ascending or descending order.

MySQL DISTINCT clause is used to remove duplicate records from the table and fetch only the unique records. The DISTINCT clause is only used with the SELECT statement.

**ASC:** It is optional. It sorts the result set in ascending order by expression (default, if no modifier is provider).

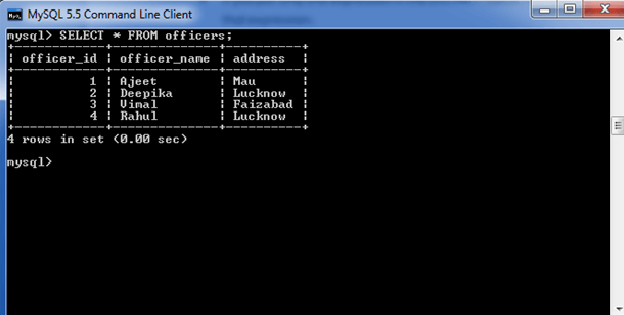
**DESC:** It is also optional. It sorts the result set in descending order by expression.

The MYSQL GROUP BY Clause is used to collect data from multiple records and group the result by one or more column. It is generally used in a SELECT statement. The MYSQL GROUP BY Clause is used to collect data from multiple records and group the result by one or more column. It is generally used in a SELECT statement.

MySQL DISTINCT clause is used to remove duplicate records from the table and fetch only the unique records. The DISTINCT clause is only used with the SELECT statement.

select distinct expressions  from tables

[where conditions];

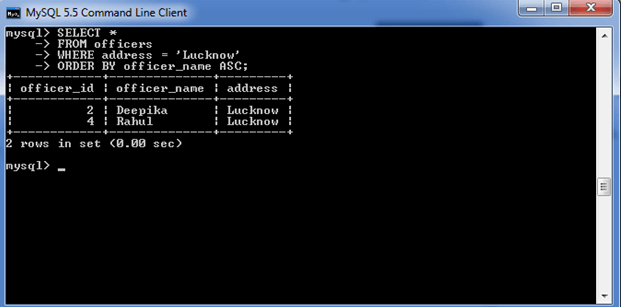


**Step-1:**

select \*  from officers

where address = 'lucknow'

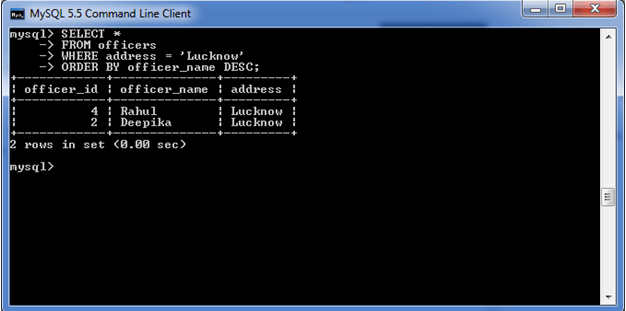
order by officer\_name asc;

****

select \*  from officers

where address = 'lucknow'

order by officer\_name desc;



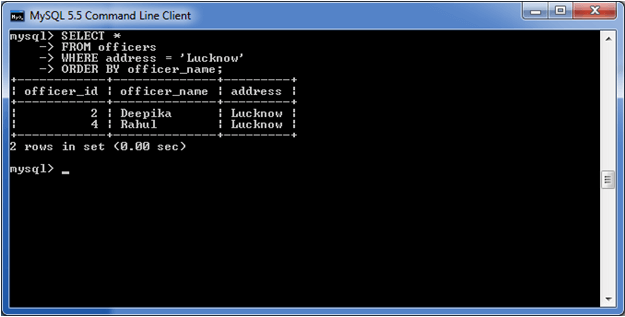
**Step-2:**

ORDER BY clause without specifying the ASC and DESC modifier

select \*  from officers

where address = 'lucknow'

order by officer\_name;



**EXPERIMENT-6**

**OBJECTIVE**

Write the query for implementing the following functions: GROUP BY(), MAX(), MIN(), AVG(),SUM() and COUNT().

**INTRODUCTION**

The MYSQL GROUP BY Clause is used to collect data from multiple records and group the result by one or more column. It is generally used in a SELECT statement.

The MYSQL GROUP BY Clause is used to collect data from multiple records and group the result by one or more column. It is generally used in a SELECT statement. Some aggregate functions like COUNT, SUM, MIN, MAX, AVG etc. can be apply on the grouped column.

Aggregate\_function: It specifies a function such as SUM, COUNT, MIN, MAX, or AVG etc. tables: It specifies the tables, from where you want to retrieve the records. There must be at least one table listed in the FROM clause.

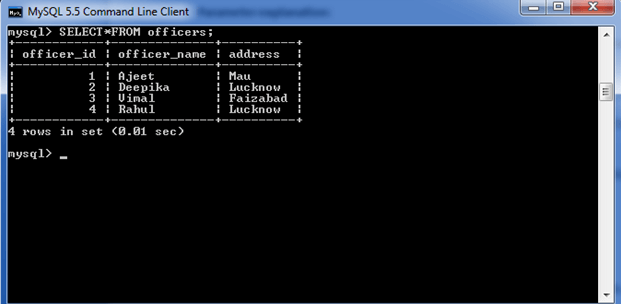
**Syntax:**

Select expression1, expression2, expression\_n, aggregate\_function (expression)

from tables

[where conditions]

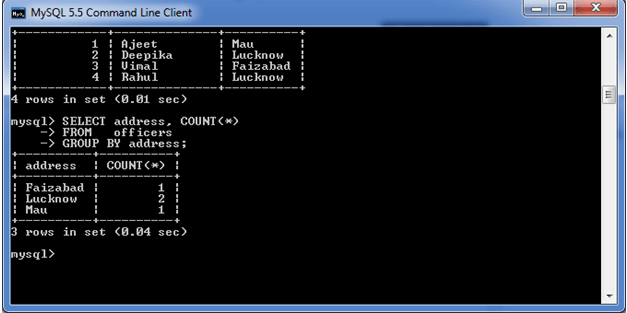
group by expression1, expression2, ... expression\_n;

**Clause with COUNT function**

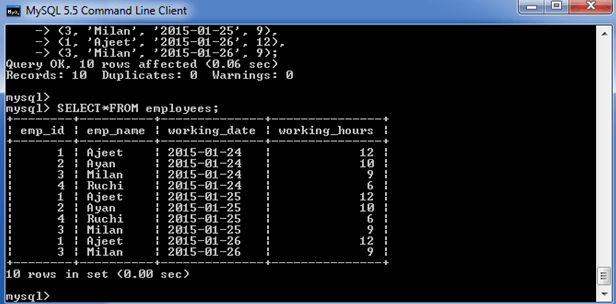
**Step1:**

select address, count(\*)

from   officers   group by address;



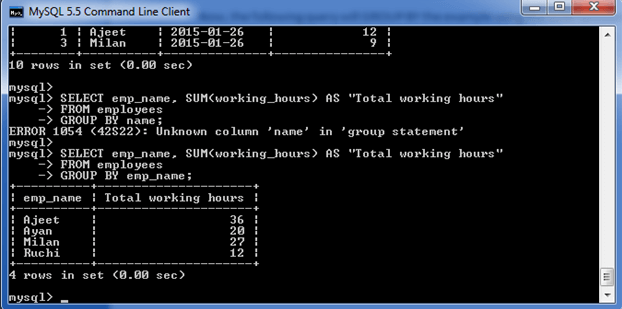
**Clause with SUM function**

****

**Step 2:**

Select emp\_name, sum(working\_hours) as "total working hours"

from employees  group by emp\_name;

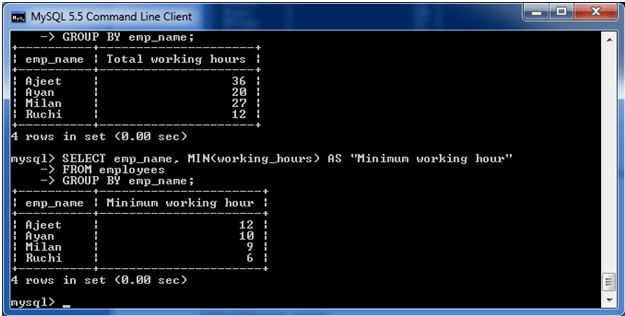


**GROUP BY Clause with MIN function**

**Step 3:**

Select emp\_name, min(working\_hours) as "minimum working hour"

from employees  group by emp\_name;

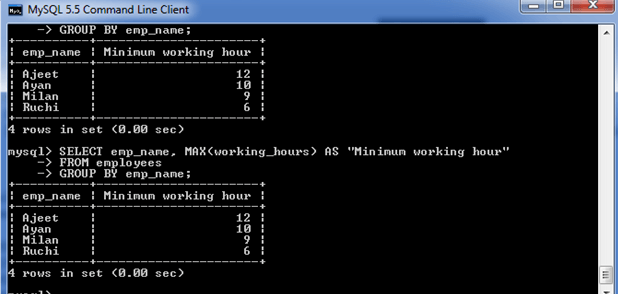


**GROUP BY Clause with MAX function**

**Step 4:**

Select emp\_name, max (working\_hours) as "minimum working hour"

from employees  group by emp\_name;



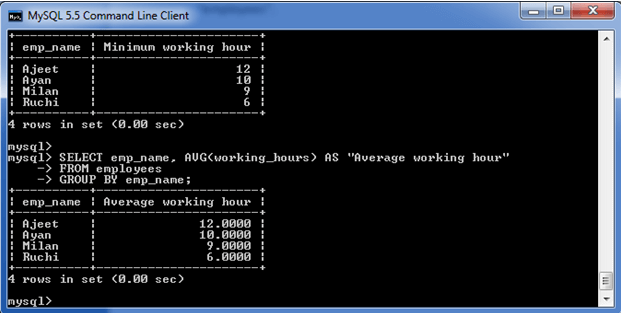
**GROUP BY Clause with AVG function**

**Step 5:**

Select emp\_name, avg(working\_hours) as "average working hour"

from employees

group by emp\_name;



**EXPERIMENT-7**

**OBJECTIVE**

Write the query to implement Joins

**INTRODUCTION**

MySQL JOINS are used with SELECT statement. It is used to retrieve data from multiple tables. It is performed whenever you need to fetch records from two or more tables.

There are three types of [MySQL](https://www.javatpoint.com/mysql-tutorial) joins:

* INNER JOIN
* LEFT OUTER JOIN
* RIGHT OUTER JOIN

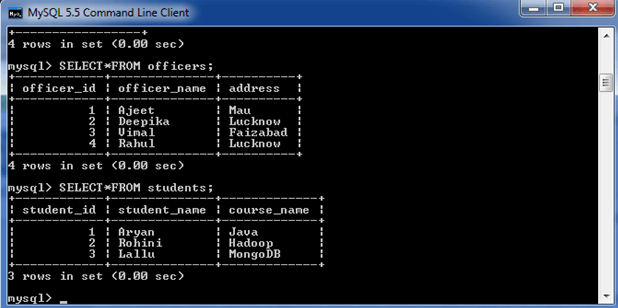
1. INNER JOIN

The  [Inner join](https://www.javatpoint.com/mysql-inner-join) is used to return all rows from multiple tables where the join condition is satisfied

**Syntax:**

Select columns from table1

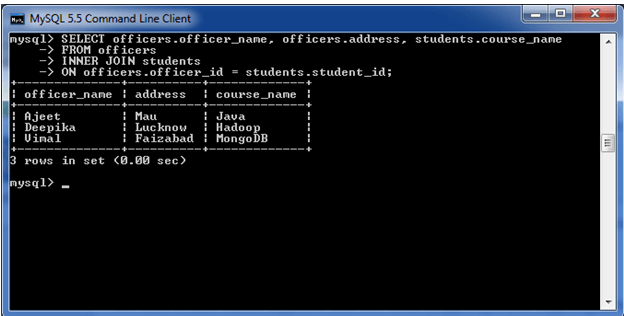
inner join table2  on table1.column = table2.column;



**Step 1:**

Select officers.officer\_name, officers.address, students.course\_name

from officers   inner join students  on officers.officer\_id = students.student\_id;



1. **LEFT OUTER JOIN**

Returns all rows from the left hand table specified in the ON condition and only those rows from the other table where the join condition is fulfilled.

**Syntax:**

Select columns  from table1

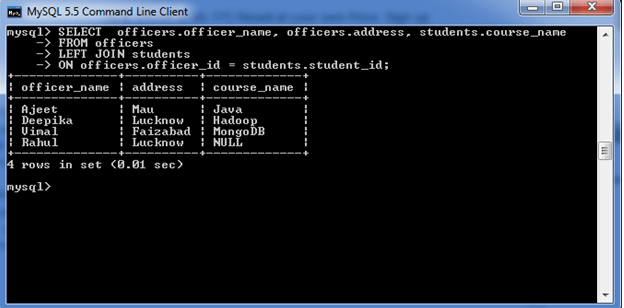
left [outer] join table2

on table1.column = table2.column;

**Step 2:**

Select  officers.officer\_name, officers.address, students.course\_name

from officers  left join students  on officers.officer\_id = students.student\_id;



1. **RIGHT OUTER JOIN**

Right Outer Join returns all rows from the RIGHT- hand table specified in the ON condition and only those rows from the other table where he join condition is fulfilled

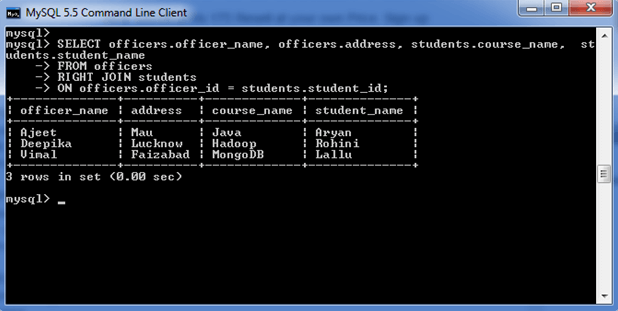
**Syntax:**

Select columns from table1

right [outer] join table2  on table1.column = table2.column;

**Step 3:**

Select officers.officer\_name, officers.address, students.course\_name, students.student\_name  from officers  right join students  on officers.officer\_id = students.student\_id;



**EXPERIMENT-8**

**OBJECTIVE**

Perform the queries for triggers.

**INTRODUCTION**

It is a special type of stored procedure that is invoked automatically in response to an event. Each trigger is associated with a table, which is activated on any DML statement such as INSERT, UPDATE, or DELETE**.**

Types of Triggers in MySQL?

We can define the maximum six types of actions or events in the form of triggers:

[**Before Insert**](https://www.javatpoint.com/mysql-before-insert-trigger)**:** It is activated before the insertion of data into the table.

[**After Insert**](https://www.javatpoint.com/mysql-after-insert-trigger)**:** It is activated after the insertion of data into the table.

[**Before Update**](https://www.javatpoint.com/mysql-before-update-trigger)**:** It is activated before the update of data in the table.

[**After Update**](https://www.javatpoint.com/mysql-after-update-trigger)**:** It is activated after the update of the data in the table.

[**Before Delete**](https://www.javatpoint.com/mysql-before-delete-trigger)**:** It is activated before the data is removed from the table.

[**After Delete**](https://www.javatpoint.com/mysql-after-delete-trigger)**:** It is activated after the deletion of data from the table.

**Syntax:**

Create trigger trigger\_name

(after | before) (insert | update | delete)

 on table\_name for each row

begin

--variable declarations

--trigger code

 end;

**Step 1:**

Create table employee(

name varchar(45) NOT NULL,

occupation varchar(35) NOT NULL,

working\_date date,

working\_hours varchar(10));

Insert into employee Values

('Robin', 'Scientist', '2020-10-04', 12),

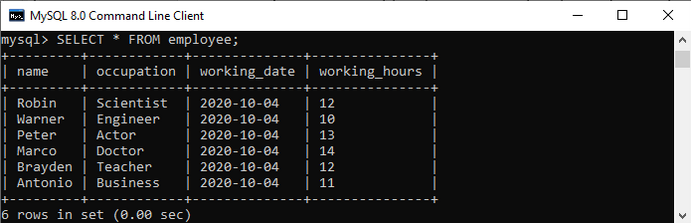
('Warner', 'Engineer', '2020-10-04', 10),

('Peter', 'Actor', '2020-10-04', 13),

('Marco', 'Doctor', '2020-10-04', 14),

('Brayden', 'Teacher', '2020-10-04', 12),

('Antonio', 'Business', '2020-10-04', 11);

****

**create a** [**before insert trigger**](https://www.javatpoint.com/mysql-before-insert-trigger)**.**

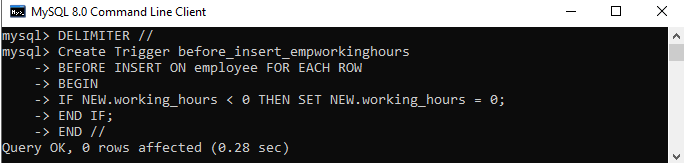
**Step 2:**

Create trigger before\_insert\_empworkinghours

before insert on employee for each row

begin if new.working\_hours < 0 then set new.working\_hours = 0;

end if;



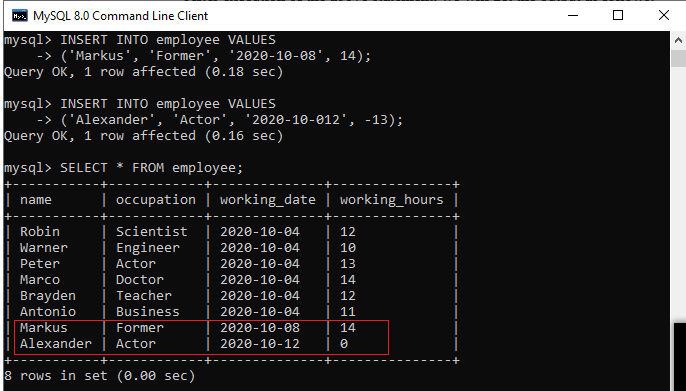
**Step 3:**

insert into employee values

('markus', 'former', '2020-10-08', 14);

 insert into employee values

('alexander', 'actor', '2020-10-012', -13);



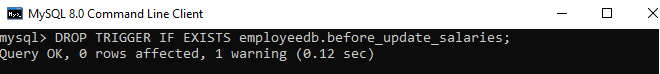
drop/delete/remove a trigger in MySQL using the drop trigger statement

**Syntax:**

drop trigger [if exists] [schema\_name.]trigger\_name;

**Step 4:**

Drop Trigger employeedb.before\_update\_salaries;



**EXPERIMENT-9**

**OBJECTIVE**

Perform the following operation for demonstration the insertion, updation and deletion.

**INTRODUCTION**

The **UPDATE** is used to update the values of one or multiple columns of a table.

The DELETEstatement is used to remove the data from the MySQL table.

To demonstrate **UPDATE** and **DELETE** statements, sakila database on MySQL Server using thefilm and languagetable of the sakila database.

**Syntax:**

update <low\_priority>|<ignore> --modifiers

tablename

set

col\_name1 = NewValue\_1,

 col\_name2 = NewValue\_2,

 col\_name3 = NewValue\_3,

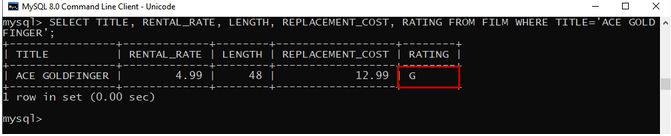
    ...

WHERE

 condition;

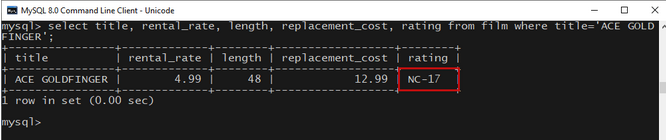
**Step 1:**

select title, rental\_rate, length, replacement\_cost, rating from film where title='ace gold finger';

****

**Step 2:**

update film set rating ='NC-17' where title='ACE GOLDFINGER'

****

**Step 3:**

delete from language;

****

**Step 4:**

Delete from film where title=’AGENT TRUMAN’;



**EXPERIMENT-10**

**OBJECTIVE**

Using the referential integrity constraints

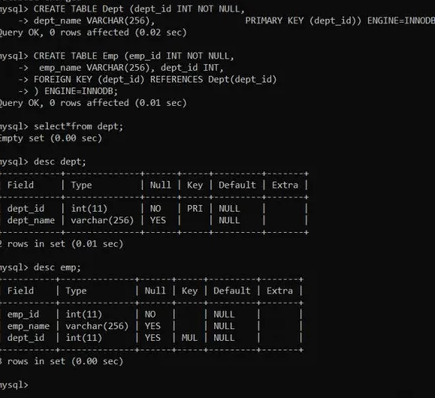
**INTRODUCTION**

**Referential Integrity** is set of constraints applied to foreign key which prevents entering a row in child table (where you have foreign key) for which you don't have any corresponding row in parent table. Referential Integrity prevents your table from having incorrect or incomplete relationship.

**Step 1:**

create table dept (dept\_id int not null,  
dept\_name varchar(256), primary key (dept\_id)) engine=innodb;

create table emp (emp\_id int not null, emp\_name varchar(256), dept\_id int,foreign key (dept\_id) references dept(dept\_id)) engine=innodb;

****

**Step 2:**

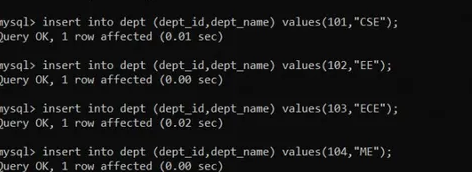
insert into dept (dept\_id,dept\_name) values(101,"CSE");

insert into dept (dept\_id,dept\_name) values(102,"EE");

insert into dept (dept\_id,dept\_name) values(103,"ECE");

insert into dept (dept\_id,dept\_name) values(104,"ME");

insert into dept (dept\_id,dept\_name) values(105,"IT");



**Step 3:**

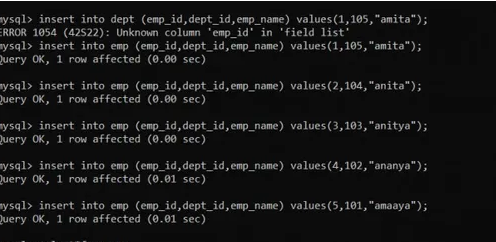
insert into emp (emp\_id,dept\_id,emp\_name) values(1,105,"amita");

insert into emp (emp\_id,dept\_id,emp\_name) values(2,104,"anita");

insert into emp (emp\_id,dept\_id,emp\_name) values(3,103,"anitya");

insert into emp (emp\_id,dept\_id,emp\_name) values(4,102,"ananya");

insert into emp (emp\_id,dept\_id,emp\_name) values(5,101,"amaaya");

****

**Step 4:**

delete from dept;

select \* from emp;

**EXPERIMENT-11**

**OBJECTIVE**

Write the query for creating the users and their role

**INTRODUCTION**

MySQL is a popular and widely used database management system that stores and organizes data and allows users to retrieve it. It comes with a vast array of options that grant users certain permissions to tables and databases.

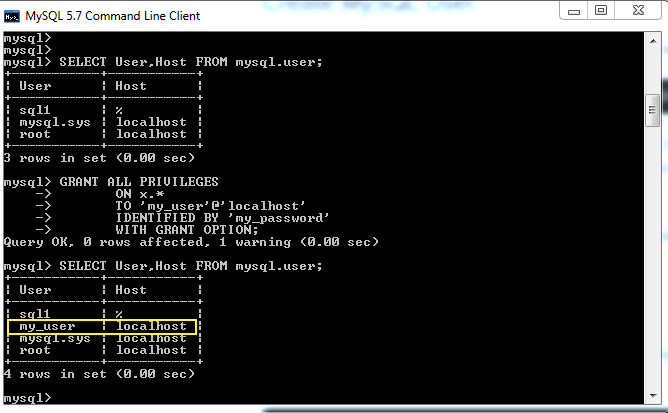
create MySQL accounts (users) two ways:

by using account-management statements intended for creating accounts and establishing their privileges, such as CREATE USER and GRANT.

First, use the mysql program to connect to the server as the MySQL root user:   
shell> mysql --user=root mysql.

**Step 1:**

Grant all privileges   
on x.\*   
to 'my\_user'@'localhost'  
identified by 'my\_password'   
with grant option;

****

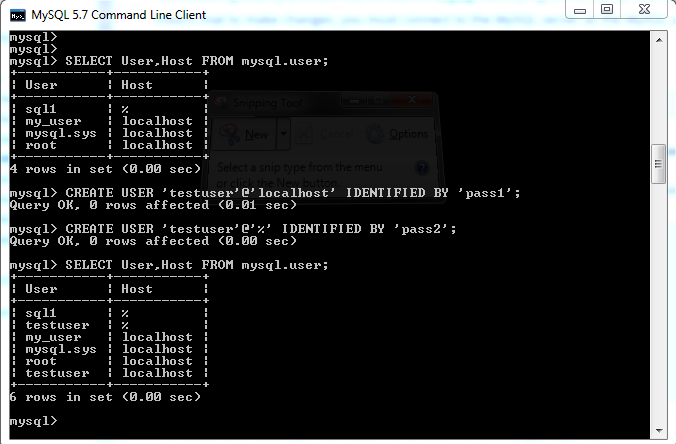
**Step 2:**

Select user,host from mysql.user;

**Step 3:**

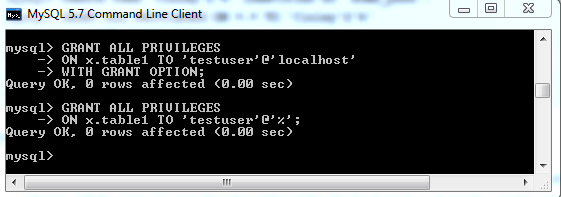
Using create user command

create user 'testuser'@'localhost' identified by 'pass1';  
create user 'testuser'@'%' identified by 'pass2';



**Step 4:**

Grant all privileges   
on x.table1 to 'testuser'@'localhost'   
with grant option;  
  
grant all privileges   
on x.table1 to 'testuser'@'%';

****

list of other common permissions that users can be granted with:

CREATE: allows them to create new tables or databases

- SELECT: allows them to use the Select command to read through databases

- DROP: allows them to them to delete tables or databases

- INSERT: allows them to insert rows into tables

- DELETE: allows them to delete rows from tables

- UPDATE: allow them to update table rows

- ALL PRIVILEGES: as we saw previously, this would allow a MySQL user all access to a designated database (or if no database is selected across the system)

- GRANT OPTION: allows them to grant or remove other users' privileges

**EXPERIMENT-12**

**OBJECTIVE**

Write the query to create the views.

**Introduction**

Views in SQL are kind of virtual tables. A view also has rows and columns as they are in a real table in the database.

We can create a view by selecting fields from one or more tables present in the database. A View can either have all the rows of a table or specific rows based on certain condition.

We can create View using CREATE VIEW statement..

**Syntax:**

CREATE VIEW view\_name AS SELECT column1, column2.....

FROM table\_name WHERE condition;

view\_name: Name for the View

table\_name: Name of the table

condition: Condition to select rows

Creating View from a table:

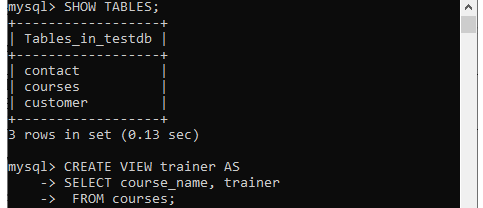
Suppose our database has a table course, and we are going to create a view based on this table. Thus, the below example will create a VIEW name "trainer" that creates a virtual table made by taking data from the table courses.

CREATE VIEW trainer AS

SELECT course\_name, trainer

FROM courses;

Once the execution of the CREATE VIEW statement becomes successful, MySQL will create a view and stores it in the database.



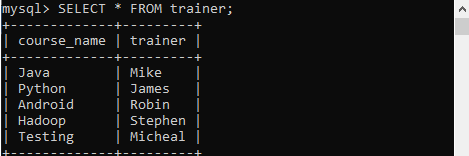
To see the created VIEW

We can see the created view by using the following syntax:

SELECT \* FROM view\_name;

Let's see how it looks the created VIEW:

SELECT \* FROM trainer;



**EXPERIMENT-13**

**OBJECTIVE**

Declare a Cursor.

**Introduction**

To handle a result set inside a stored procedure, you use a cursor. A cursor allows you to iterate a set of rows returned by a query and process each row individually. A cursor is a SELECT statement that is defined within the declaration section of your stored program in MySQL.

**Syntax**

The syntax to declare a cursor in MySQL is:

DECLARE cursor\_name CURSOR FOR select\_statement;

**Example**

CREATE TABLE books(

ID INT PRIMARY KEY,

TITLE VARCHAR(100),

AUTHOR VARCHAR(40),

DATE VARCHAR(40)

);

insert 5 records in tutorials table using books statements –

insert into books values(1, 'Java', 'Krishna', '2019-09-01');

insert into books values(2, 'JFreeCharts', 'Satish', '2019-05-01');

insert into books values(3, 'JavaSprings', 'Amit', '2019-05-01');

insert into books values(4, 'Android', 'Ram', '2019-03-01');

insert into books values(5, 'Cassandra', 'Pruthvi', '2019-04-06');

Let us create another table to back up the data −

CREATE TABLE backup (

ID INT,

TITLE VARCHAR(100),

AUTHOR VARCHAR(40),

DATE VARCHAR(40)

);

DELIMITER //

CREATE PROCEDURE ExampleProc()

BEGIN

DECLARE done INT DEFAULT 0;

DECLARE tutorialID INTEGER;

DECLARE tutorialTitle, tutorialAuthor, tutorialDate VARCHAR(20);

DECLARE cur CURSOR FOR SELECT \* FROM tutorials;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

OPEN cur;

label: LOOP

FETCH cur INTO tutorialID, tutorialTitle, tutorialAuthor, tutorialDate;

INSERT INTO backup VALUES(tutorialID, tutorialTitle, tutorialAuthor,

tutorialDate);

IF done = 1 THEN LEAVE label;

END IF;

END LOOP;

CLOSE cur;

END//

DELIMITER ;

mysql> CALL Example Proc;

mysql> select \* from backup;

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

| ID | TITLE | AUTHOR | DATE |

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

| 1 | Java | Krishna | 2019-09-01 |

| 2 | JFreeCharts | Satish | 2019-05-01 |

| 3 | JavaSprings | Amit | 2019-05-01 |

| 4 | Android | Ram | 2019-03-01 |

| 5 | Cassandra | Pruthvi | 2019-04-06 |

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

**EXPERIMENT-14**

**Objective**

Creating a function

**Introduction**

A function always returns a value using the return statement. The function can be used in SQL queries.

Syntax

CREATE FUNCTION function\_name [ (parameter datatype [, parameter datatype]) ]

RETURNS return\_datatype

BEGIN

Declaration\_section

Executable\_section

END;

create a function that returns the customer occupation based on the age using the below statement.

DELIMITER $$

CREATE FUNCTION Customer\_Occupation (

age int

)

RETURNS VARCHAR (20)

DETERMINISTIC

BEGIN

DECLARE customer\_occupation VARCHAR (20);

IF age > 35 THEN

SET customer\_occupation = 'Scientist';

ELSEIF (age <= 35 AND

age >= 30) THEN

SET customer\_occupation = 'Engineer';

ELSEIF age < 30 THEN

SET customer\_occupation = 'Actor';

END IF;

-- return the customer occupation

RETURN (customer\_occupation);

END$$

DELIMITER;

SHOW FUNCTION STATUS WHERE db = 'mysqltestdb';

MySQL Stored Function