|  |
| --- |
| *Agnel Charities*  **Fr. C. Rodrigues Institute of Technology, Vashi**  **Department of Electronics and Telecommunication Engg.**  **SUB:-Linux & Networking & Server Configuration ( LNSC) ECL-604** |
| **Implement MYSQL on Ubuntu** |

|  |  |
| --- | --- |
| **Experiment**  **No** | 12 |
| **AIM** | **Implement MYSQL on Ubuntu** |
| **THEO**  **RY** | MySQL tutorial provides basic and advanced concepts of MySQL. Our MySQL tutorial is designed for beginners and professionals. 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**.  **Installation of MYSQL:**  **Step 1: Update/Upgrade Package Repository**   1. Open the terminal and run the following command:   **sudo apt update**  2. Enter your password and wait for the update to finish.  3. Next, run:  **sudo apt upgrade**  4. Enter Y when prompted to continue with the upgrade and hit ENTER. Wait for the upgrade to finish.  **Step 2: Install MySQL**  1. After successfully updating the package repository, install MySQL Server by running the following command:  **sudo apt install mysql-server**  2. When asked if you want to continue with the installation, answer Y and hit ENTER.  3. Check if MySQL was successfully installed by running:  **mysql –version**  **Step 3: Securing MySQL**  The MySQL instance on your machine is insecure immediately after installation.  1. Secure your MySQL user account with password authentication by running the included security script:  sudo mysql\_secure\_installation  Enter your password and answer Y when asked if you want to continue setting up the VALIDATE PASSWORD component. The component checks to see if the new password is strong enough.\_installation  3. Choose one of the three levels of password validation:  0 - Low. A password containing at least 8 characters.  1 - Medium. A password containing at least 8 characters, including numeric, mixed case characters, and special characters.  2 - Strong. A password containing at least 8 characters, including numeric, mixed case characters, and special characters, and compares the password to a dictionary file.  Enter 0, 1, or 2 depending on the password strength you want to set. The script then instructs you to enter your password and re-enter it afterward to confirm.  Any subsequent MySQL user passwords need to match your selected password strength.  **Step 4: Check if MySQL Service Is Running**  Upon successfully installing MySQL, the MySQL service starts automatically.  Verify that the MySQL server is running by running:  **sudo systemctl status mysql**  **Step 5: Log in to MySQL Server**  Finally, to log in to the MySQL interface, run the following command:  **sudo mysql -u root** |
| **TASK 1** | **CREATE DATABASE**  Let’s create a movies database.  1. Create a database using the CREATE statement:  CREATE DATABASE movies;  2. Next, verify that the database was created by [showing a list of all databases](https://phoenixnap.com/kb/how-to-list-all-databases-mysql). Use the SHOW statement:  SHOW DATABASES;  The terminal prints out a list of databases and information about the time it took to perform the query:  Creating a database in MySQL.  3. Select the database to make changes to it by using the USE statement:  USE movies;  **Create a Table**  We’ll create a table containing information about two movies:  In the process of creating a table, you need to specify the following information:  Column names – We are creating the title, genre, director, and release year columns for our table.  Varchar of the columns containing characters – Specifies the maximum number of characters stored in the column.  The integer of the columns containing numbers – Defines numeric variables holding whole numbers.  Not null rule – Indicates that each new record must contain information for the column.  Primary key – Sets a column that defines a record.  1. Create a table using the CREATE command. Using the information from our movies example, the command is:  CREATE TABLE movies(title VARCHAR(50) NOT NULL,genre VARCHAR(30) NOT NULL,director VARCHAR(60) NOT NULL,release\_year INT NOT NULL,PRIMARY KEY(title));  Creating a table.  2. Verify that the table is created using the DESCRIBE command:  DESCRIBE movies;  The terminal prints out information about the table:  Field – Indicates column name.  Type – Specifies data type for the column (varchar for characters, int for numbers).  Null – Indicates whether the column can remain with null values.  Key – Displays the primary column.  Default – Displays the column’s default value.  Extra – Indicates additional information about the columns.  Verifying the table creation.  3. Insert movie information in column order – title, genre, director, and release year. Use the INSERT command:  INSERT INTO movies VALUE ("Joker", "psychological thriller", "Todd Phillips", 2019);  Loading data into table.  4. Repeat the previous step with the second movie. Use the SELECT command to display the table:  SELECT \* FROM movies;  The terminal prints out the movie table:  Displaying created table. |
| **TASK 2s** | **INSERT/UPDATE/ALTER TABLES**  Once a table has been created in the MySQL database, we can **insert records** to this table by executing the following command:  **mysql> INSERT INTO Student VALUES (1, ‘Ayesha’, 24);**  You can change these values according to your own choice.    When our new record will be successfully added to our table, we will receive the message shown in the image below on the MySQL shell. In the very same manner, we can insert another record.  The **UPDATE command in MySQL** is used to modify the values of any specific record. For example, if you want to change the values of any record of your table, then you have to execute the following command:  mysql> UPDATE Student SET StudentName='Saad', StudentAge=24 WHERE StudentID=2;    We can verify if the said changes have taken place or not by using the same SELECT statement once again to view the records of our table. You can see from the following image that our specified record has been updated successfully.    The **ALTER command in MySQL** is used to Add or Drop a specified column from a table. For example, if we want to delete any specific column from our table, then we will execute the following command:  Here, you can replace StudentAge with any column of your choice that you want to drop.    After the successful execution of this operation, you will receive the message shown in the image below on the MySQL shell.    To check if the specified column has been dropped or not, we will run the SELECT command once again.  we can **drop a table** by executing the following command:    The successful execution of this command will drop the selected table. The message shown in the image below confirms this deletion. |
| **TASK 3** | **JOINS** MySQL INNER JOIN – join three tables example The following products, orders and orderdetails tables:    This query uses two INNER JOIN clauses to join three tables: orders, orderdetails, and products:    This picture shows the partial output:  MySQL Inner Join - join three tables example |
| **CONCLUSI-ON** | Some of the most widely-used SQL statements and clauses include the following:   * **ALTER**: Modifies the structure of a database object. * **CREATE**: Creates a database object, such as a table or database. * **DELETE**: Removes one or more existing rows from the database. * **DROP**: Permanently deletes an object from the database. * **FROM**: Indicates which table to use for the query. * **INSERT**: Adds rows to the database. * **JOIN**: A clause specifying how to combine and assemble data from multiple tables. * **SELECT**: Retrieves data from one or more tables. This command does not alter the database or change any data. * **UPDATE**: Modifies one or more existing rows. * **WHERE**: A clause to identify the rows a query should operate on. It is typically used with a comparison operator. * The wildcard **\* operator** is often used in conjunction with the SELECT command. This command instructs SQL to display all columns in the output. |