

Database Programming with MySQL

Are you registered with
Onlinevarsity.com?

Yes



No



Did you download this book
from **Onlinevarsity.com?**

Yes



No



Scores

For each **YES** you score **50**

For each **NO** you score **0**

If you score less than 100 this book is illegal.

Register on **www.onlinevarsity.com**

Database Programming with MySQL Learner's Guide

© 2014 Aptech Limited

All rights reserved.

No part of this book may be reproduced or copied in any form or by any means – graphic, electronic or mechanical, including photocopying, recording, taping, or storing in information retrieval system or sent or transferred without the prior written permission of copyright owner Aptech Limited.

All trademarks acknowledged.

APTECH LIMITED

Contact E-mail: ov-support@onlinevarsity.com

First Edition - 2014



Dear Learner,

We congratulate you on your decision to pursue an Aptech Worldwide course.

Aptech Ltd. designs its courses using a sound instructional design model – from conceptualization to execution, incorporating the following key aspects:

- Scanning the user system and needs assessment

Needs assessment is carried out to find the educational and training needs of the learner

Technology trends are regularly scanned and tracked by core teams at Aptech Ltd. TAG* analyzes these on a monthly basis to understand the emerging technology training needs for the Industry.

An annual Industry Recruitment Profile Survey# is conducted during August - October to understand the technologies that Industries would be adapting in the next 2 to 3 years. An analysis of these trends & recruitment needs is then carried out to understand the skill requirements for different roles & career opportunities.

The skill requirements are then mapped with the learner profile (user system) to derive the Learning objectives for the different roles.

- Needs analysis and design of curriculum

The Learning objectives are then analyzed and translated into learning tasks. Each learning task or activity is analyzed in terms of knowledge, skills and attitudes that are required to perform that task. Teachers and domain experts do this jointly. These are then grouped in clusters to form the subjects to be covered by the curriculum.

In addition, the society, the teachers, and the industry expect certain knowledge and skills that are related to abilities such as *learning-to-learn, thinking, adaptability, problem solving, positive attitude etc.* These competencies would cover both cognitive and affective domains.

A precedence diagram for the subjects is drawn where the prerequisites for each subject are graphically illustrated. The number of levels in this diagram is determined by the duration of the course in terms of number of semesters etc. Using the precedence diagram and the time duration for each subject, the curriculum is organized.

- Design & development of instructional materials

The content outlines are developed by including additional topics that are required for the completion of the domain and for the logical development of the competencies identified. Evaluation strategy and scheme is developed for the subject. The topics are arranged/organized in a meaningful sequence.

The detailed instructional material – Training aids, Learner material, reference material, project guidelines, etc.- are then developed. Rigorous quality checks are conducted at every stage.

➤ Strategies for delivery of instruction

Careful consideration is given for the integral development of abilities like thinking, problem solving, learning-to-learn etc. by selecting appropriate instructional strategies (training methodology), instructional activities and instructional materials.

The area of IT is fast changing and nebulous. Hence considerable flexibility is provided in the instructional process by specially including creative activities with group interaction between the students and the trainer. The positive aspects of web based learning –acquiring information, organizing information and acting on the basis of insufficient information are some of the aspects, which are incorporated, in the instructional process.

➤ Assessment of learning

The learning is assessed through different modes – tests, assignments & projects. The assessment system is designed to evaluate the level of knowledge & skills as defined by the learning objectives.

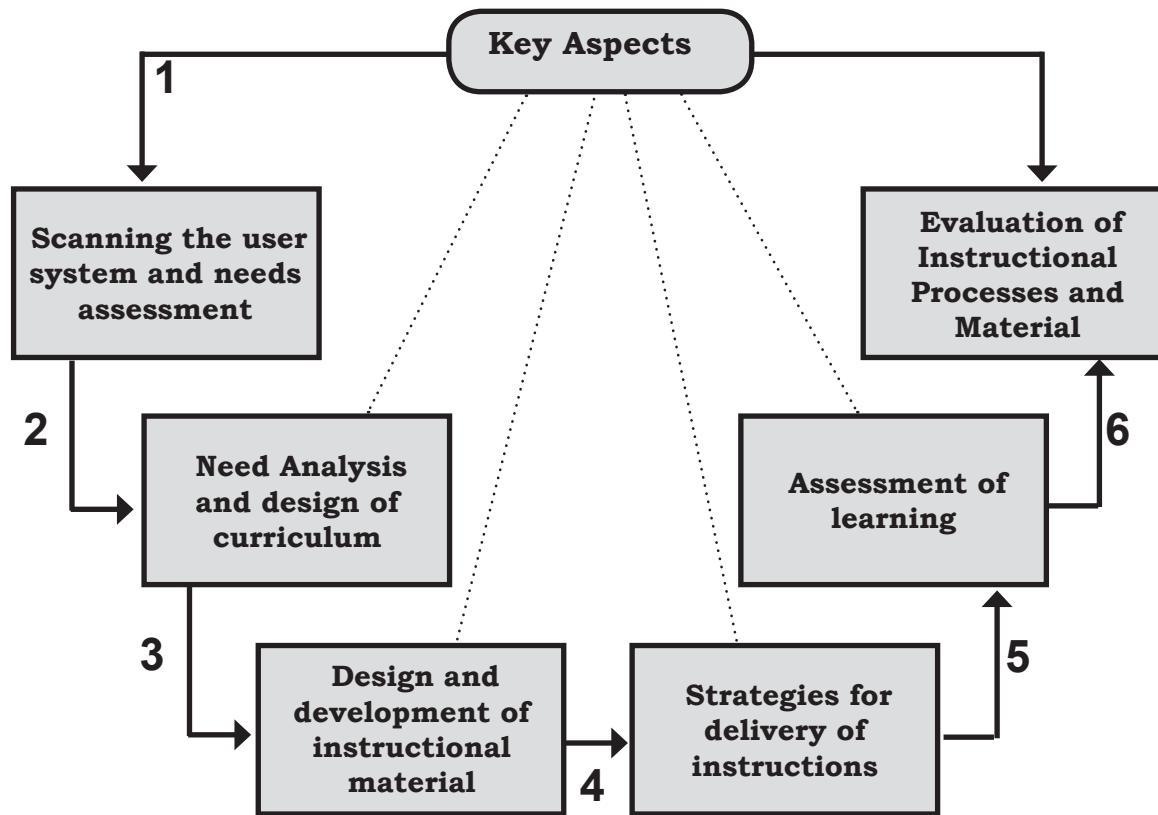
➤ Evaluation of instructional process and instructional materials

The instructional process is backed by an elaborate monitoring system to evaluate - on-time delivery, understanding of a subject module, ability of the instructor to impart learning. As an integral part of this process, we request you to kindly send us your feedback in the reply pre-paid form appended at the end of each module.

*TAG – Technology & Academics Group comprises members from Aptech Ltd., professors from reputed Academic Institutions, Senior Managers from Industry, Technical gurus from Software Majors & representatives from regulatory organizations/forums.

Technology heads of Aptech Ltd. meet on a monthly basis to share and evaluate the technology trends. The group interfaces with the representatives of the TAG thrice a year to review and validate the technology and academic directions and endeavors of Aptech Ltd.

Aptech New Products Design Model



ASK to LEARN

Questions
in your
mind?



are here to HELP

Post your queries in **ASK to LEARN** @

www.onlinevarsity.com

Preface

MySQL 5.1 is a Database Management System that enables us to enter, organize and select data from a database. MySQL is open source software. It is available for free download on the Internet. The advantage of MySQL is that it provides flexibility of modifying its source code. It has almost all the features of the commercial databases in the market. It also provides with different tools that facilitates the user to work on databases.

In this book, we will deal with databases and tables. Databases are used to store information, so that the user is able to retrieve the information at a later stage. These databases consist of one or more tables. Queries are used to retrieve data from the databases. Different accounts can be created for different users. These accounts are managed by the administrator.

This book is the result of a concentrated effort of the Design Team, which is continuously striving to bring you the best and the latest in Information Technology. The process of design has been a part of the ISO 9001 certification for Aptech-IT Division, Education Support Services. As part of Aptech's quality drive, this team does intensive research and curriculum enrichment to keep it in line with industry trends.

We will be glad to receive your suggestions.

Design Team



Visit
Frequently Asked Questions
@

Table of Contents

Sessions

1. Introduction to MySQL as an Open Source Database
2. Installing and Configuring MySQL
3. Installing and Configuring MySQL (Lab)
4. Using MySQL
5. Using MySQL (Lab)
6. Implementing SQL Queries Using MySQL - I
7. Implementing SQL Queries Using MySQL - I (Lab)
8. Implementing SQL Queries Using MySQL - II
9. Implementing SQL Queries Using MySQL - II (Lab)
10. Using Joins
11. Using Joins (Lab)
12. Using Basic Functions in MySQL - I
13. Using Basic Functions in MySQL - I (Lab)
14. Using Basic Functions in MySQL - II
15. Using Basic Functions in MySQL - II (Lab)
16. Controlling and Managing MySQL database
17. Controlling and Managing MySQL database (Lab)

Get
WORD WISE



Visit
Glossary@

www.onlinevarsity.com

Objectives

At the end of this session, the student will be able to:

- *List the features of MySQL.*
- *State the advantages of MySQL over other RDBMS.*
- *Compare MySQL with other RDBMS.*
- *State the advantages of PHP in MySQL.*
- *Explain open source software licenses.*

1.1 Introduction

A database is a systematic collection of data. A Database Management System (DBMS) is required to store, access, delete, or otherwise organize data in a database.

MySQL is an open source DBMS. You can freely download, modify, and use open source software without having to pay any fees or royalty to the original author.

In this session, you will learn about MySQL, its features, and its advantages over other Relational Database Management Systems (RDBMS). A RDBMS is based on relational model as specified by E.F. Codd. In addition, you will learn about the advantages of MySQL as an open source database and use of HyperText PreProcessor (PHP) with MySQL.

1.2 Overview of MySQL

Consider a library which lends books to its members. Traditionally, the details of books, members, and lending are maintained manually using ledgers. As the number of books increases, managing and searching for books, members, and lending details becomes difficult. This information can be stored in rows and columns in a table. A database can be created to store these tables. In addition, a DBMS can be used to manage the databases.

A DBMS can be defined as a software program that stores and manages databases. A database is a system used to store the data in a structured format. In other words, database can be defined as an organized collection of data. DBMS is responsible for managing the various database operations such as adding, accessing, and processing of data. A DBMS helps you to manage data in two ways:

- It provides an interface to manage data

Session 1

Introduction to MySQL as an Open Source Database

- It supports connectivity to other applications that can be used to manage data

Both, DBMS and RDBMS, perform the same task of storing and managing data. One of the key differences between DBMS and RDBMS is that RDBMS splits large amount of data into smaller tables and establishes relationship between the tables. DBMS stores large amount of data in a single table. Also, the RDBMS is based on a relational model whereas DBMS is not.

MySQL is an open source RDBMS. MySQL uses the standardized Structured Query Language (SQL) to manage the database. MySQL is developed and distributed by MySQL AB, a company founded by the MySQL developers. In 2008, Sun Microsystems acquired MySQL AB. In 2010, Oracle acquired Sun Microsystems and hence, MySQL is now owned by Oracle Corp.

1.2.1 Features of MySQL

MySQL was designed to achieve speed, robustness, and ease of use. The features of MySQL are as follows:

- Technical Features:
 - Is written in C and C++
 - Is tested with different compilers
 - Is compatible with multiple operating systems
 - Has support for multiple storage engines; both transactional and non-transactional
 - Has Application Programming Interfaces (APIs) for accessing MySQL databases available in many languages, including C, C++, Eiffel, Java, Perl, PHP, Python, Ruby, and Tcl
 - Is using multiple kernel threads or processing units, if available, for data processing
 - Is using multiple processors where available, boosting performance
 - Has ability to divert memory resources from inactive threads to active threads for faster processing
 - Has commands and features to retrieve, update, and delete data from several tables
 - Has support for compatibility to be used as a separate application or as an embedded library

Session 1

Introduction to MySQL as an Open Source Database

- Column Types:
 - Includes multiple column or data types including numeric, date and time, and string
 - Includes mapping of data types from other databases to MySQL data types
 - Includes fixed-length and variable-length strings
- Commands and Functions:
 - Has support for all MySQL operators and functions in the `SELECT` statement and the `WHERE` clause
 - Has support for tables from different databases in one statement
 - Has support for table and column aliases
 - Has support for displaying information about databases, tables, and indexes using the `SHOW` command
 - Has support for displaying query resolution information using the `EXPLAIN` command
 - Has full support for `SQL GROUP BY` and `ORDER BY` clauses, group functions and left and right outer joins
 - Has support for the use of function names as table or column name
- Security:
 - Has support for in-built data encryption and decryption
 - Has support for user account privileges
 - Has support for password encryption
- Scalability and Limits:
 - Handles large databases that have upto 5 billion rows
 - Allows upto 64 indexes per table
 - Allows upto 16 keys per table

Concepts

Session 1

Introduction to MySQL as an Open Source Database

➤ Connectivity:

- Supports connectivity on any platform to MySQL server using Transmission Control Protocol/Internet Protocol (TCP/IP) sockets
- Supports connectivity on Windows NT, 2000, XP, 2003, and Vista using named pipes or shared-memory connections
- Supports connectivity on UNIX systems using UNIX domain socket files

➤ Localization:

- Displays error messages in languages, such as Czech, French, German, Japanese, Korean, Norwegian, Polish, and Russian
- Supports Unicode and various character sets
- Allows data to be stored, sorted, and compared using the chosen character set

➤ Clients and tools:

- Provides built-in support to check, optimize, and repair tables
- Provides “mysql” tool to execute individual SQL commands or SQL commands stored in a file
- Provides “mysqlaccess” tool to check host, user, and database privileges
- Provides “mysqladmin” tool to manage the database server
- Provides “mysqldump” tool to backup the contents of one or more MySQL databases to a file
- Provides “mysqlhotcopy” tool to backup a single database or table on to the same computer
- Provides “mysqlimport” tool to import data into a MySQL table from a file
- Provides “mysqlshow” tool to display information about the databases, tables, and columns
- Provides “mysqld _ safe” tool that enables safe start up of the MySQL server

Session 1

Introduction to MySQL as an Open Source Database

Concepts

1.3 Advantages of MySQL Over Other RDBMS

There are many commercial Database Management Systems such as Oracle, Microsoft SQL Server, and Sybase available in the market. These Database Management Systems are robust, reliable, and support most of the features that a user wants. However, it is impossible for these databases to compete with MySQL with regards to price, as MySQL is available for free. In addition, for commercial Database Management Systems, the initial setup cost is more expensive, resource intensive, and time consuming, whereas with MySQL, this is not the case. This is one of the key advantages of MySQL.

Also, as the source for MySQL is fully available, you can customize MySQL as required. There are many troubleshooting techniques, command help, and syntax help that are available. This information is available in blogs, forums, and lists that do not require paid subscriptions. However, troubleshooting techniques, command help, and syntax help for commercial databases may require a paid subscription.

Typically, open source software tends to be updated more frequently than commercial software because many users contribute to its development. As a result, new features are available more often than for commercial databases.

MySQL provides different versions that work on different versions of Linux, UNIX, Microsoft, Windows, and other operating systems. MySQL also supports various built-in and third-party GUI tools for faster and easier design, implementation, and administration.

Following are the other advantages that MySQL offers over other RDBMS:

- **Reliable:** Supports tables that can store and handle large number of records.
- **Ease of Use:** Provides a modular and flexible architecture that makes it easy to manage and customize.
- **Cross Platform Support:** Supports different operating systems, such as Linux, UNIX, and Microsoft Windows.
- **Views:** Supports views where data is copied into temporary or virtual tables during processing. This feature ensures data security.
- **Stored Procedures:** Supports stored procedures and functions. This allows you to implement business logic at the database level.
- **Triggers:** Supports triggers. This feature also enables you to implement business logic during data processing.

Session 1

Introduction to MySQL as an Open Source Database

1.4 Comparing MySQL As An Open Source Database With Other RDBMSes

There are many open source and commercial databases available today. Some of the popular open source databases include mSQL, PostgreSQL, and InstantDB. As mentioned earlier, Oracle, MS SQL Server, and Sybase are some popular commercial databases.

The early editions of MySQL did not support all the SQL features. For example, transaction support and stored procedures were not available in the older versions of MySQL. The latest versions of MySQL, however, provide full SQL and transaction support.

The commercial databases support almost all the features that are present in MySQL, but the performance of MySQL is better.

One drawback in MySQL is that it does not support advanced SQL3 features such as object oriented data types. PostgreSQL supports advanced SQL3 features and is a better choice as an open-source DBMS. However, PostgreSQL has a major disadvantage in its hidden limit of 8K of data per row.

PostgreSQL is more powerful but MySQL is faster. MySQL does not need a vacuum procedure as PostgreSQL. Vacuum procedure refers to the process of optimizing the data stored in the database. PostgreSQL withstands higher loads. The latest versions of PostgreSQL and MySQL support features, such as sub-selects, stored procedures, triggers, unions, and views. However, older versions of MySQL did not support all these features. In addition, because these features are new to MySQL, there are some performance issues.

However, MySQL provides more user-friendly command interface so it is popular among Web developers. Also, MySQL supports more data types and functions as compared to mSQL.

InstantDB competes well with MySQL when you consider the different features. The only feature of MySQL that InstantDB is unable to compete with is performance. MySQL is faster as compared to InstantDB.

1.5 Advantages of PHP In MySQL Environment

A scripting tool enables you to control one or more applications when executed. PHP is a scripting tool designed for Web development. PHP supports embedding scripts into HTML code. Developers can use PHP scripts to create HTML Web pages that can read and write data from a database.

PHP is a scripting language that is executed at run-time. It enables interaction of the application with the database. You can use PHP and MySQL together to manage data on the Web. PHP is compatible with MySQL. You can also store and manage information from the database. Figure 1.1 displays the interaction between the client, server, and database.

Session 1

Introduction to MySQL as an Open Source Database

Concepts

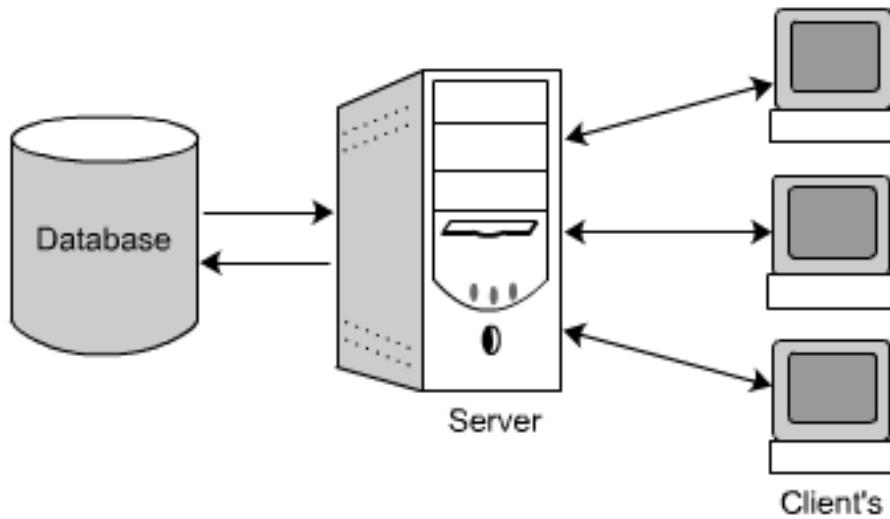


Figure 1.1: Interaction between Client, Server, and the Database

Consider a database connected to a server as shown in figure 1.1. The server is connected to several clients. The database, server, and client maintain a two-way communication. This system appears to be simple; however, it has certain limitations. For example, when a client requests for data, the browser sends a request to the server. The server locates the data from the database, and returns it to the browser. If several clients make a request to the server for the same data, then the server will return the data to all the clients. This results in slow performance of the system.

The advantage of using PHP is that the database can be accessed directly through a Web page. In this scenario, the client will request for a PHP file. The PHP preprocessor will connect to the database, retrieve the data, convert the data into HTML format, and send it to the browser.

Following are some examples of real-world Websites where databases are used:

- **Online Ticket Reservation:** In an online reservation system you can reserve a seat using the Internet. Your action updates the backend or the database of this booking system. You can access several parts of the database by changing the Uniform Resource Locator (URL).
- **Message Boards:** On the Internet, message boards are widely present that run on MySQL and PHP. It is an online discussion site where messages can be posted. Message boards running on PHP and MySQL are more efficient because you need to update only one page and the changes are automatically reflected in others.
- **Marketing Websites:** Consider that a large Website is required to be updated. The Website can be updated using few PHP scripts as the information related to these pages is stored in MySQL database. The PHP scripts accesses the MySQL database to obtain the information about the pages.

Session 1

Introduction to MySQL as an Open Source Database

- **Advertising Banners:** Consider a Website where several advertisement banners are present on the site. These banners are stored in a database on the server. You can call a PHP script to display each banner. In order to insert, modify, or delete a banner you have to access the database. A PHP script can be written to select the database from which the banner will be displayed. The PHP script would select and display the correct banners for the pages on the site.

To run a PHP script, you will need to install the following software:

- Web server
- PHP
- MySQL

PHP supports different operating systems such as Mac OS X, Linux, UNIX, and Windows.

Both PHP and MySQL are open source. This feature makes them cost effective as compared to other software products.

MySQL also supports the command line interface. This interface enables the PHP page to access the database and display the query results.

You can use PHP scripts to control the administration activities of the database. However, it is better to install a PHPMyAdmin on the server. PHPMyAdmin is an administrative interface for MySQL databases. It consists of a set of free scripts for administration of the database.

1.6 Open Source Software Licensing

There are many types of software licenses available. An open-source software license permits users to read, access, change, and reuse the source code of a software product. Open-source software does not necessarily mean free software.

The advantage of using open-source software is that the developer can customize as per requirements. There is no limit on the customization. Also, there are various troubleshooting and performance tips freely available on the Internet for open-source software. However, open-source license does not allow you to sell customized software. In addition, you will have to provide the modified source code when you distribute the software.

However, open-source software must comply with certain conditions and some of them are as follows:

- The source code must be distributed along with the binary.
- The software can be modified as required and redistributed under the same terms as the original software.

Session 1

Introduction to MySQL as an Open Source Database

- When the software is used as a part of development of other applications, the software must be redistributed with the application without any royalty or fee.

Note: For a detailed description of the open-source license terms and conditions, visit: <http://www.opensource.org/docs/osd>

Concepts

1.6.1 Licensing of MySQL

MySQL is available under two licensing schemes. They are:

- **General Public License (GPL)** – Applies to developers who use, and/or distribute open source software under the GPL.
- **Commercial License** – Applies to developers who only use MySQL to develop their own executables and not the source code.

Session 1

Introduction to MySQL as an Open Source Database



Summary

- A database stores data in a structured format. A Database Management System is responsible to store, access, and delete data from a database.
- MySQL is an open source RDBMS. It was developed and distributed by MySQL AB, which is now owned by Oracle.
- There are many commercial Relational Database Management Systems available, such as Oracle, Microsoft SQL, and Sybase that support most of the data management features.
- MySQL works on different operating systems, such as Mac OS X, Linux, UNIX, and Windows.
- Unlike in traditional Database Management Systems that are proprietary, in open source software, you can modify the source code to customize the features.
- PHP is a scripting language that enables interaction with a database. You can use PHP and MySQL to store and manage data on the Web.
- In order to run a PHP script, you will need to install a Web server, PHP and MySQL.
- Open-source software licenses allow you to read, access, change, and reuse the source code of a software product.
- MySQL is available under General Public License and Commercial License.

Session 1

Introduction to MySQL as an Open Source Database



Check Your Progress

Concepts

1. The `mysqlshow` command displays information about the _____.
 - a. Databases and tables on the server
 - b. Tables and columns in a database
 - c. Columns and rows in a table
 - d. Version of MySQL Server
2. _____ is an administrative interface for MySQL databases.
 - a. mysqladmin
 - b. PHPMyAdmin
 - c. admin
 - d. SQL
3. To execute a PHP script, you will need to install _____.
 - a. Web Server, PHP, MySQL
 - b. PHP, MySQL
 - c. MySQL, Web Server
 - d. PHP, Web Server
4. MySQL allows upto _____ indexes per table.
 - a. 16
 - b. 32
 - c. 46
 - d. 64



Check Your Progress

5. _____ tool manages users of MySQL.
- a. mysqladmin
 - b. mysqlaccess
 - c. mysql
 - d. mysqlshow

Objectives

At the end of this session, the student will be able to:

- *Explain the various distribution options of MySQL.*
- *Explain the installation process of MySQL on Microsoft Windows.*
- *Explain the installation process of MySQL on Red Hat Enterprise Linux.*
- *Explain the configuration process of MySQL using Scripts.*
- *Explain initialization of MySQL at startup.*

2.1 Introduction

MySQL is an Open Source software that enables you to use and modify the source code. It is available in binary and source distribution formats. You can use MySQL on different operating systems such as Microsoft Windows, Linux, or UNIX. It is easy to install, maintain, and configure.

In this session, you will learn to install MySQL on different platforms and initialize MySQL at the startup. You will also learn to configure MySQL installation using scripts and my.cnf configuration file.

2.2 Evaluating the Various Distribution Options of MySQL

You can install MySQL using either the binary distribution or the source distribution. The binary distribution contains a setup program that performs the installation. If you use the source distribution that contains the source code of MySQL, you will have to compile the code before installation.

Normally, you use the binary distribution for installation. This is because the binary distribution is a pre-compiled, ready-to-install distribution. Binary distribution is available in different formats that are compatible with a specific operating system. For example, RPM files or compressed ZIP or tar files are available for installing MySQL on Linux. Thus, binary installations are fast and easy to install.

The installation, using the binary distribution under Linux or UNIX, creates the following directories, as shown in figure 2.1.

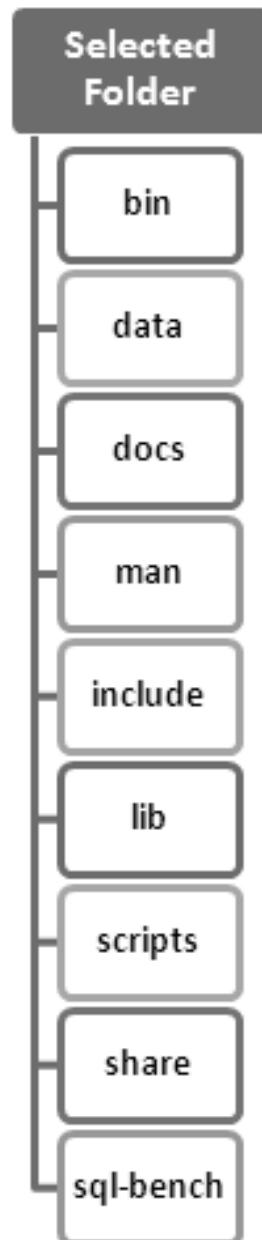


Figure 2.1: Default Directories under Source Directory for a Linux/Unix Binary Installation

Table 2.1 lists the default folders under source directory and its contents.

Directory	Content
bin	Contains client utilities and mysqld daemon
data	Contains databases and log files
docs	Contains documentation in the Info format

Session 2

Installing and Configuring MySQL

Concepts

Directory	Content
man	Contains UNIX manual pages
include	Contains header files
lib	Contains library files
scripts	Contains mysql_install_db file
share	Contains support files such as error messages
sql-bench	Contains benchmarks of MySQL database

Table 2.1: Default Directories Under Source Directories

The only disadvantage of a binary distribution is that customization in terms of directory locations, and so forth is not possible. If you need a custom MySQL installation, then you may prefer to use the source distribution of MySQL.

You can use source distribution to create a distribution:

- Having customized directory structure
- Having a custom set of features by enabling or disabling features
- From the latest development source (binaries are available only for release candidates)

If you do not customize the installation folder structure, the distribution generated by compiling the source uses the same folder structure as the binary distribution.

2.3 Installing and Configuring MySQL on Different Platforms

You must ensure that the platform or the Operating System (OS) will support MySQL before starting the installation process of MySQL. The OS or platforms that support MySQL are as follows:

- Solaris
- Linux
- Microsoft Windows XP/Vista/7
- Microsoft Windows Server 2003/2008
- Mac OS X
- FreeBSD
- AIX

Session 2

Installing and Configuring MySQL

Note: Visit <http://www.mysql.com/support/supportedplatforms/database.html> for a detailed list of supported platforms.

2.3.1 Installing and Configuring MySQL on Microsoft Windows

The Microsoft Windows installer files are required to install MySQL on Microsoft Windows platform. You will have to download the mysql-5.1.56-win32.msi file from <http://dev.mysql.com/downloads/mysql/5.1.html>.

The steps followed for installing MySQL on Microsoft Windows XP are as follows:

1. Browse and locate the downloaded installation package, and double click the file. The **Open File Security Warning** dialog box is displayed.
2. Click **Run**. The Windows Installer prepares the installation process and **MySQL Server 5.1- Setup Wizard** dialog box is displayed, as shown in figure 2.2.

Session 2

Installing and Configuring MySQL

Concepts



Figure 2.2: MySQL Server 5.1 - Setup Wizard

3. Click **Next** to display the **License Agreement** pane in the **MySQL Server 5.1 – Setup Wizard** dialog box, as shown in figure 2.3.

Session 2

Installing and Configuring MySQL

Concepts

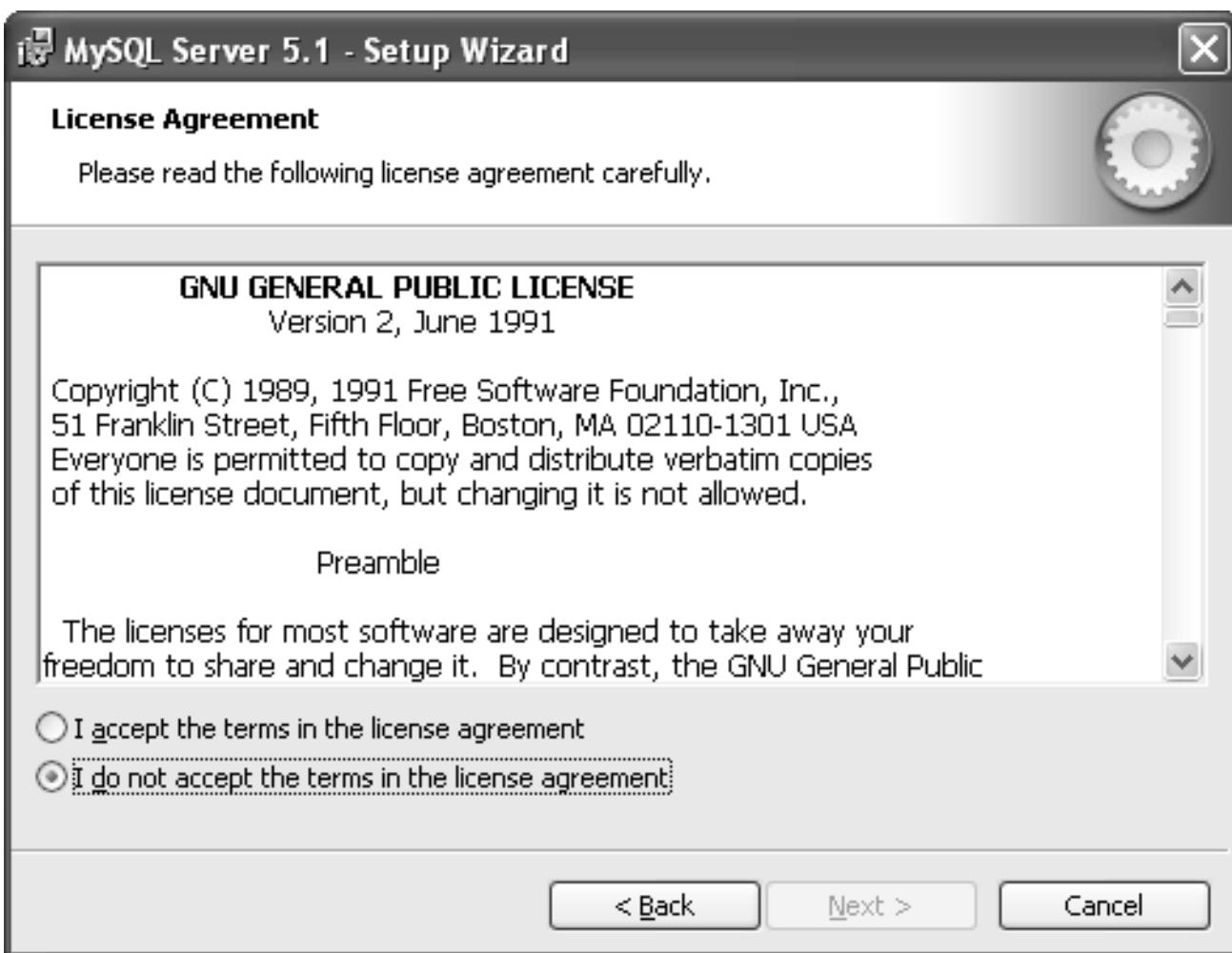


Figure 2.3: End User License Agreement Dialog Box

4. Select 'I accept the terms in the license agreement'.
5. Click **Next** to display the **Setup Type** pane in the **MySQL Server 5.1 – Setup Wizard** dialog box, as shown in figure 2.4.

Session 2

Installing and Configuring MySQL

Concepts



Figure 2.4: Choose Setup Type Dialog Box

6. Click **Typical**.
7. Click **Next** to display the **Ready to Install the Program** pane in the **MySQL Server 5.1 – Setup Wizard** dialog box, as shown in figure 2.5.

Session 2

Installing and Configuring MySQL

Concepts



Figure 2.5: Ready to Install the Program Dialog Box

8. Click **Install**. The Setup Wizard completes the installation process and the **MySQL Enterprise** dialog box is displayed, as shown in figure 2.6.

Session 2

Installing and Configuring MySQL

Concepts

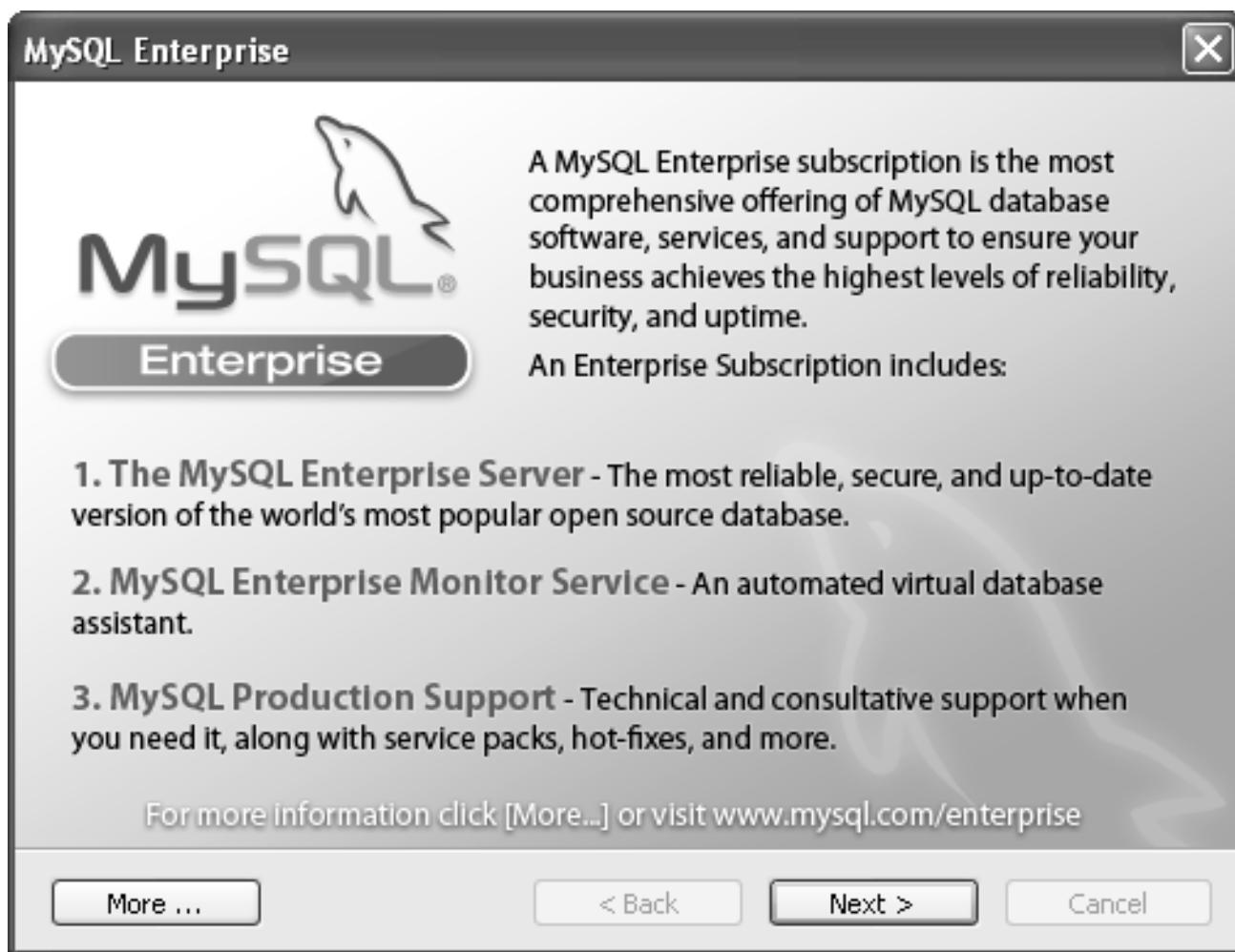


Figure 2.6: MySQL Enterprise Dialog Box

9. Click **Next**. The **MySQL Enterprise Monitor Service** pane of the **MySQL Enterprise** dialog box is displayed, as shown in figure 2.7.

Session 2

Installing and Configuring MySQL

Concepts

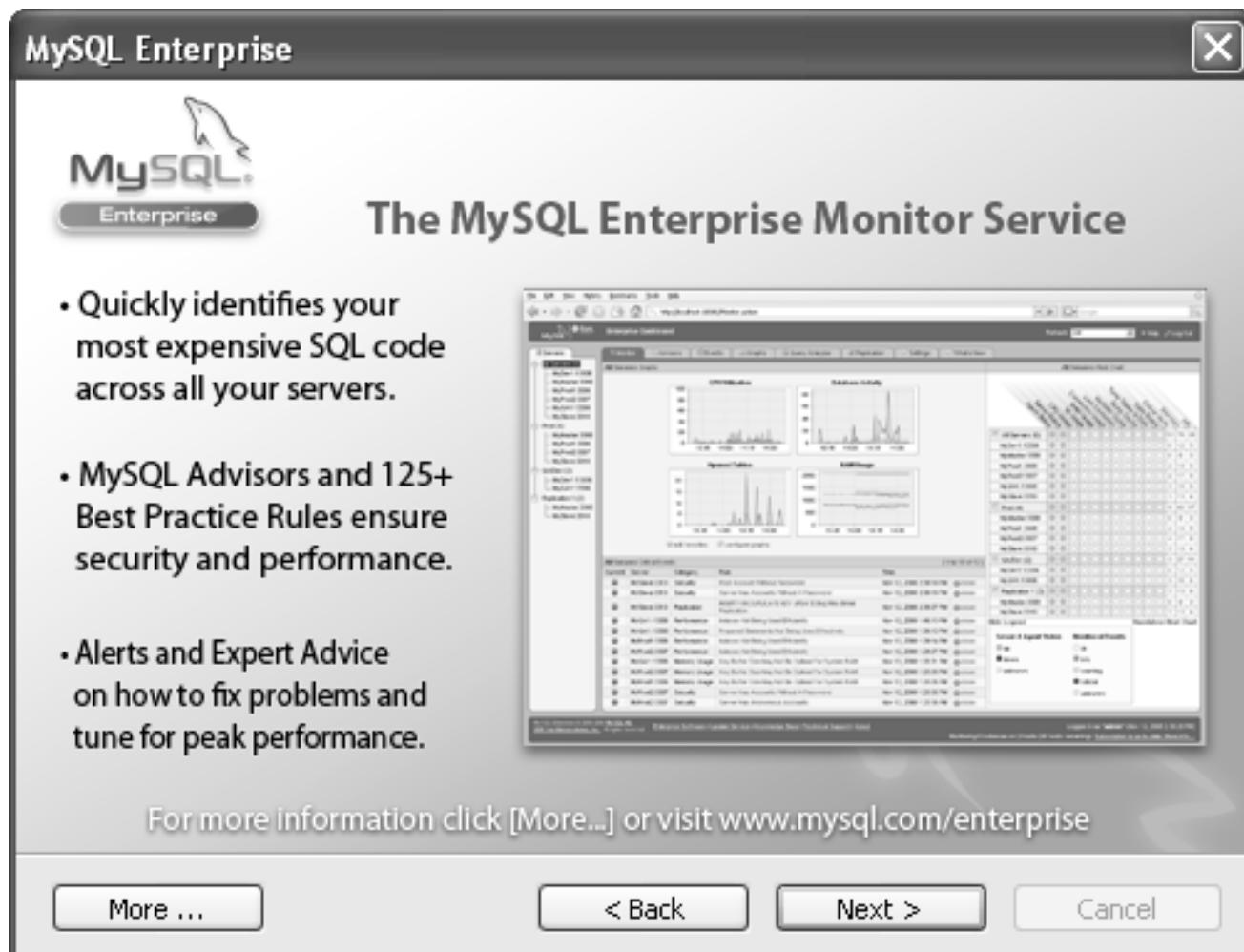


Figure 2.7: MySQL Enterprise Monitor Service Dialog Box

10. Click **Next**. The **MySQL Server 5.1 - Setup Wizard** dialog box displays the completion of the installation process by displaying the **Wizard Completed** pane, as shown in figure 2.8.

Session 2

Installing and Configuring MySQL

Concepts



Figure 2.8: Wizard Completed Dialog Box

11. Click **Finish**. The **MySQL Server Instance Configuration Wizard** pane is displayed, as shown in figure 2.9.

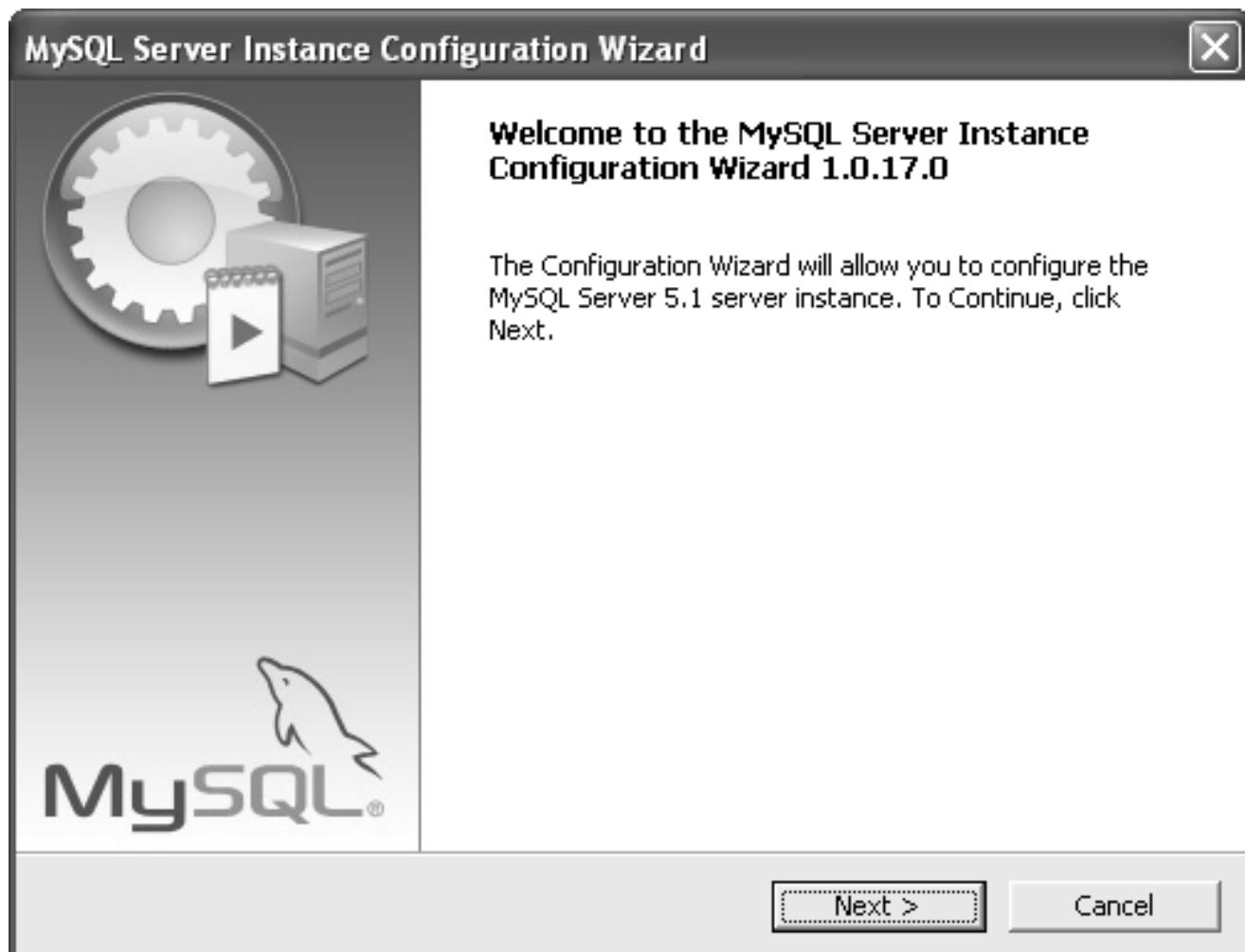


Figure 2.9: MySQL Server Instance Configuration Wizard

12. Click **Next** to display the **MySQL Server Instance Configuration** pane of the **MySQL Server Instance Configuration Wizard** dialog box, as shown in figure 2.10.

Session 2

Installing and Configuring MySQL

Concepts



Figure 2.10: MySQL Server Instance Configuration Wizard - Configuration Type

13. Click **Standard Configuration**.
14. Click **Next** to display the **Windows Options** pane of the **MySQL Server Instance Configuration Wizard**, as shown in figure 2.11.

Session 2

Installing and Configuring MySQL

Concepts



Figure 2.11: Windows Options Dialog Box

Options available include:

- **Install As Windows Service** – starts MySQL server at system startup
 - **Launch the MySQL Server automatically** – starts MySQL with Windows
 - **Include Bin Directory in Windows PATH** – copies the server and client executables in the Windows PATH variable. These variables can be invoked from the command line
15. Select the **Install As Windows Service** check box.
16. Select the **Include Bin Directory in Windows PATH** check box.
17. Click **Next**. A dialog box appears displaying the different security options, as shown in figure 2.12.

Session 2

Installing and Configuring MySQL

Concepts



Figure 2.12: Security Options Dialog Box

The different options available include:

- **Modify Security Settings** – changes the default security settings
- **New root password** – accepts password for the root user
- **Confirm** – accepts the password for the root user
- **Enable root access from remote machines** – allows root users to connect to MySQL from other computers
- **Create An Anonymous Account** – specifies to generate a hidden login account

18. Clear the **Modify Security Settings** check box.

Session 2

Installing and Configuring MySQL

Note: You can assign a password for the root account by entering a password in the text boxes.

19. Click **Next** to display configuration tasks in the **MySQL Server Instance Configuration Wizard** as shown in figure 2.13.

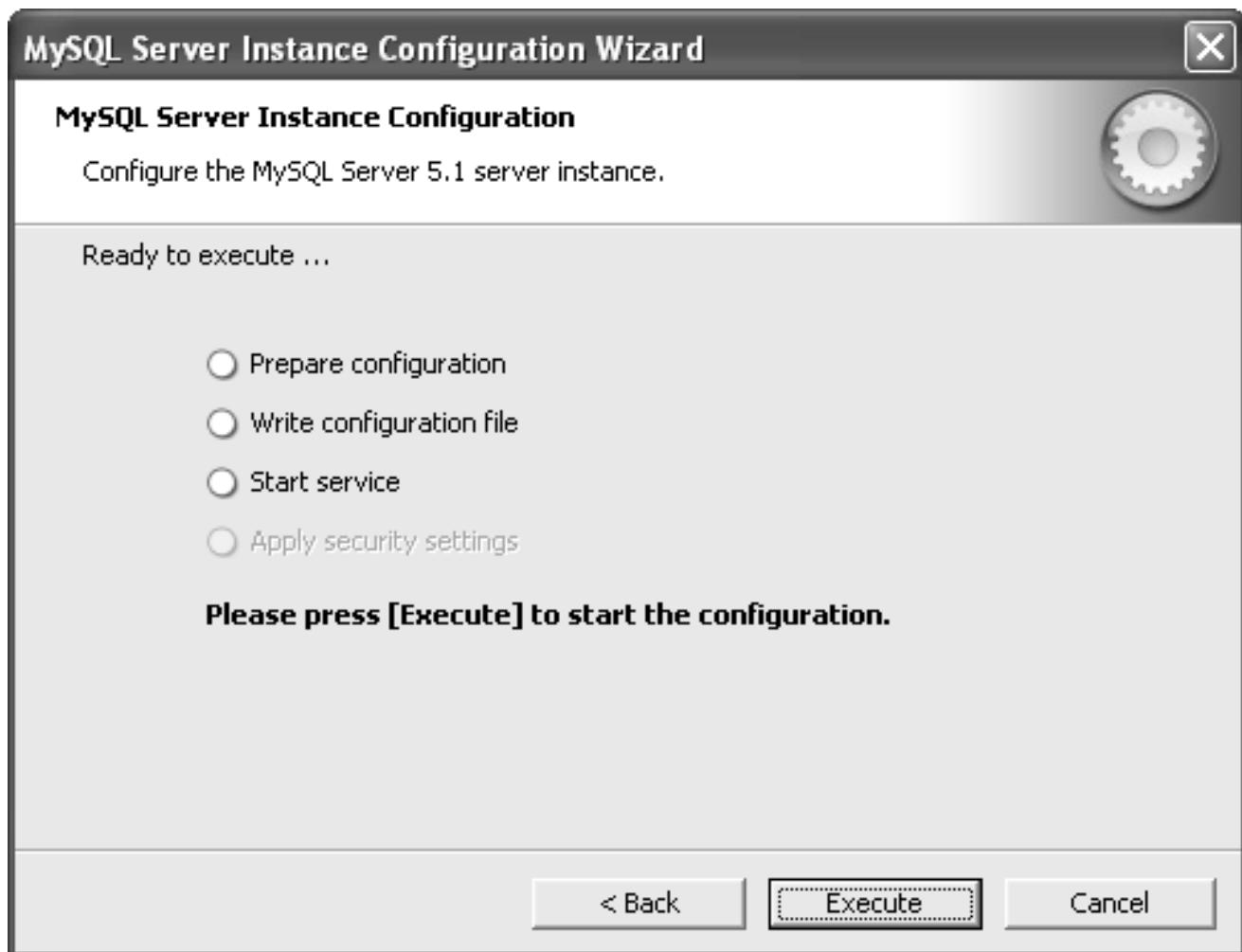


Figure 2.13: MySQL Server Instance Configuration Wizard Displaying Configuration

20. Click **Execute**. The wizard configures the settings and a confirmation message box is displayed as shown in figure 2.14.

Session 2

Installing and Configuring MySQL

Concepts



Figure 2.14: Confirmation Message Box

21. Click **Finish**.

The following steps helps to start MySQL:

1. Start the MS-DOS command prompt.
2. Change the directory to C:\Program Files\MySQL\MySQL Server 5.1 folder.
3. Enter the following command at the command prompt:

```
mysql
```

MySQL starts and displays the 'mysql' prompt, as shown in figure 2.15.

Session 2

Installing and Configuring MySQL

Concepts



The screenshot shows a Windows command prompt window titled 'C:\WINDOWS\system32\cmd.exe - mysql'. The window displays the MySQL monitor interface. The text output includes:

```
C:\Program Files\MySQL\MySQL Server 5.1>mysql
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.1.56-community MySQL Community Server (GPL)

Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights
This software comes with ABSOLUTELY NO WARRANTY. This is free soft
and you are welcome to modify and redistribute it under the GPL v2

Type 'help;' or '\h' for help. Type '\c' to clear the current input
mysql> _
```

Figure 2.15: MySQL Prompt

4. To view the default databases after installation, enter the following command at the prompt:

```
SHOW DATABASES;
```

Figure 2.16 displays all the databases present on the server.

Session 2

Installing and Configuring MySQL

Concepts

```
C:\WINDOWS\system32\cmd.exe - mysql
C:\Program Files\MySQL\MySQL Server 5.1>mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.1.56-community MySQL Community Server (GPL)

Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.
This software comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to modify and redistribute it under the GPL v2 license

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

mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| test |
+-----+
2 rows in set (0.08 sec)

mysql>
```

Figure 2.16: Default Databases After MySQL Installation

2.3.2 Installing MySQL on Red Hat Enterprise Linux

You will use the RPM packages to install MySQL on Red Hat Enterprise Linux. The different RPM packages required are as follows:

```
MySQL-client-community-5.1.56-1.rhel5.i386.rpm
MySQL-server-community-5.1.56-1.rhel5.i386.rpm
```

These packages can be downloaded from the Downloads section of the MySQL Website. You must select **Red Hat and Oracle Enterprise Linux** to view the list of packages that can be used to install MySQL.

You must check the existence of previous versions of MySQL before installation. If a previous version of MySQL is already present in the system, you will need to first uninstall the previous version and then install the new version.

Note: You must have root privileges to install MySQL. You can also login as root to execute the installation of MySQL.

The client and server RPM packages downloaded from the MySQL server Website will be executed to install MySQL server.

Session 2

Installing and Configuring MySQL

1. To execute the server RPMs, double click the **MySQL-server-community-5.1.56-1.rhel5.i386.rpm** package. The installation prompts for confirmation, as shown in figure 2.17.

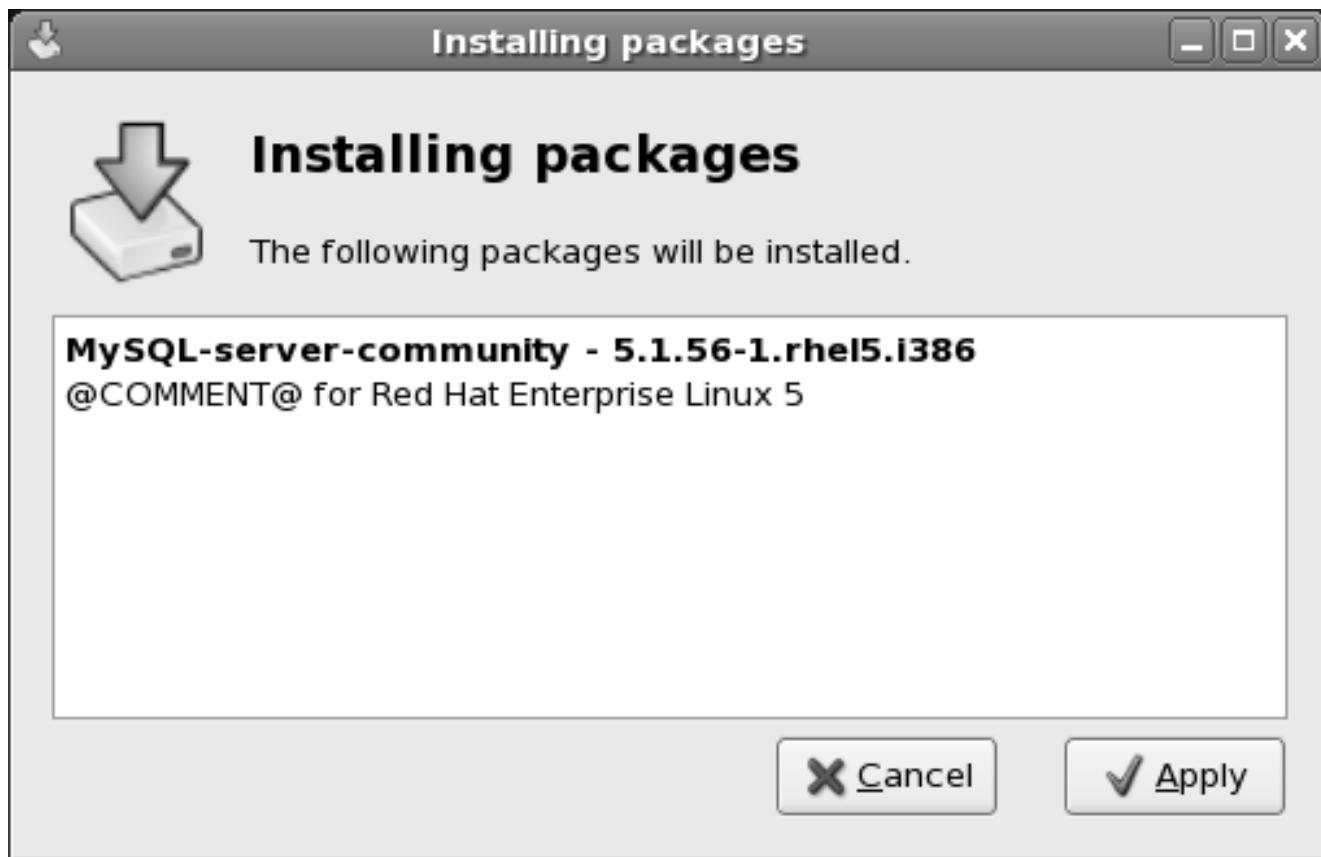


Figure 2.17: MySQL-Server-RPM Installation

2. Click **Apply**. The installation process completes and a confirmation message is displayed.
3. To execute the client RPMs, double click the **MySQL-client-community - 5.1.56-1.rhel5.i386.rpm** package. The installation prompts for confirmation, as shown in figure 2.18.

Session 2

Installing and Configuring MySQL

Concepts

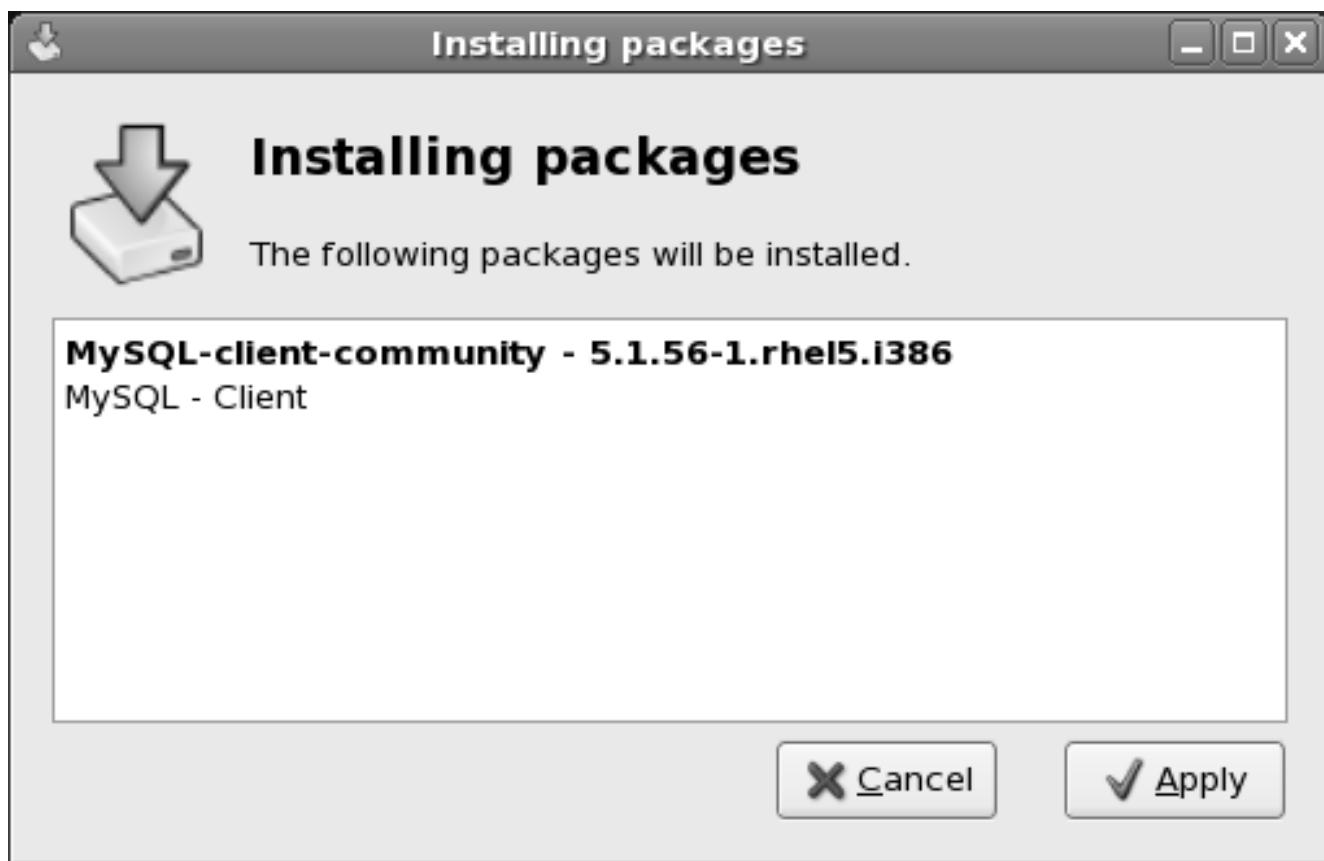


Figure 2.18: MySQL Client RPM Installation

4. Click **Apply**. The installation process completes and a confirmation message is displayed.
5. Start the terminal in Linux.
6. To start the MySQL service, enter the following command at the command prompt:

```
/etc/init.d/mysql start
```

Note: This command also adds MySQL service to the startup. It means that MySQL service automatically starts during system boot.

Figure 2.19 displays the output of the command.

Session 2

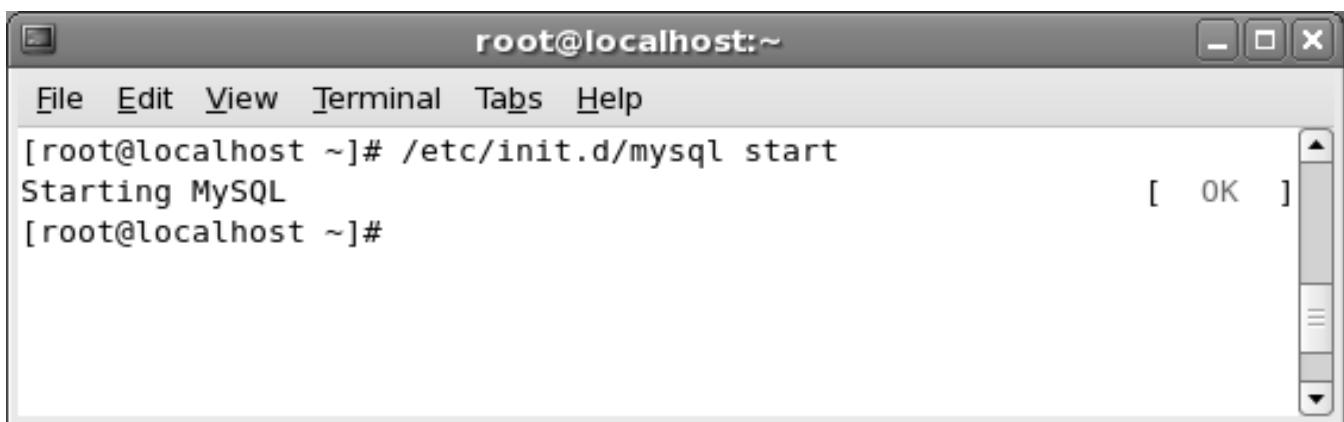


Figure 2.19: Initializing and Starting MySQL Service

7. To start MySQL, enter the following command at the command prompt:

```
mysql -u root
```

Figure 2.20 displays the output of the command.

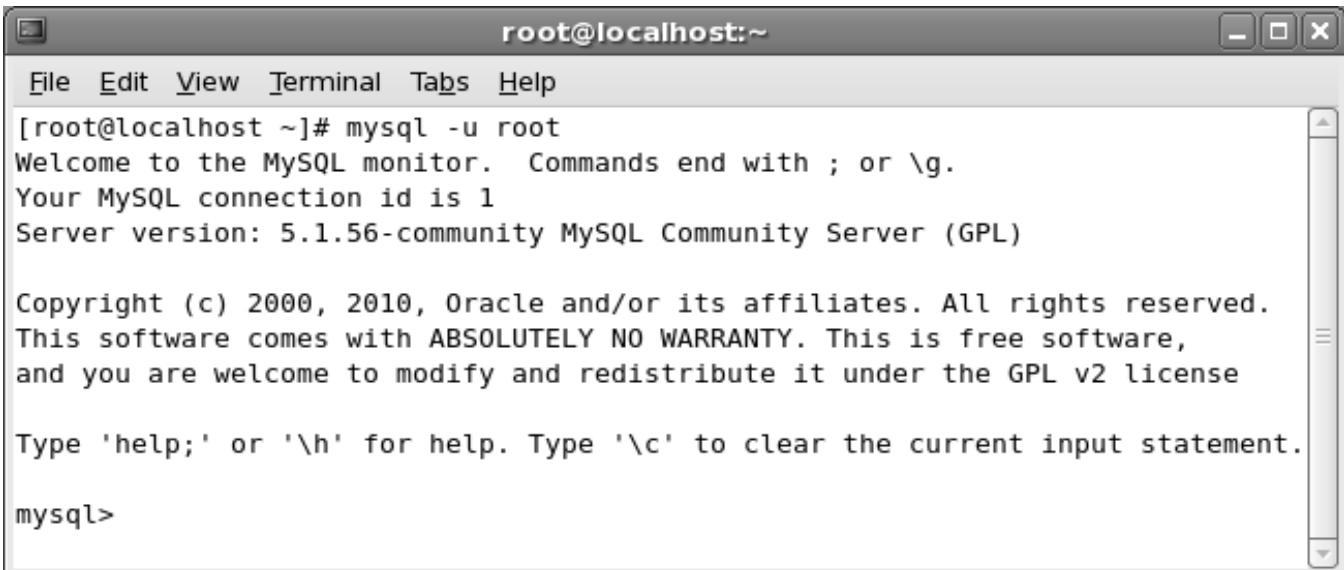


Figure 2.20: MySQL Prompt - Startup Screen

Note: If you login to Linux as the root user, you can connect to the MySQL prompt by entering the following command at the command prompt:

```
mysql
```

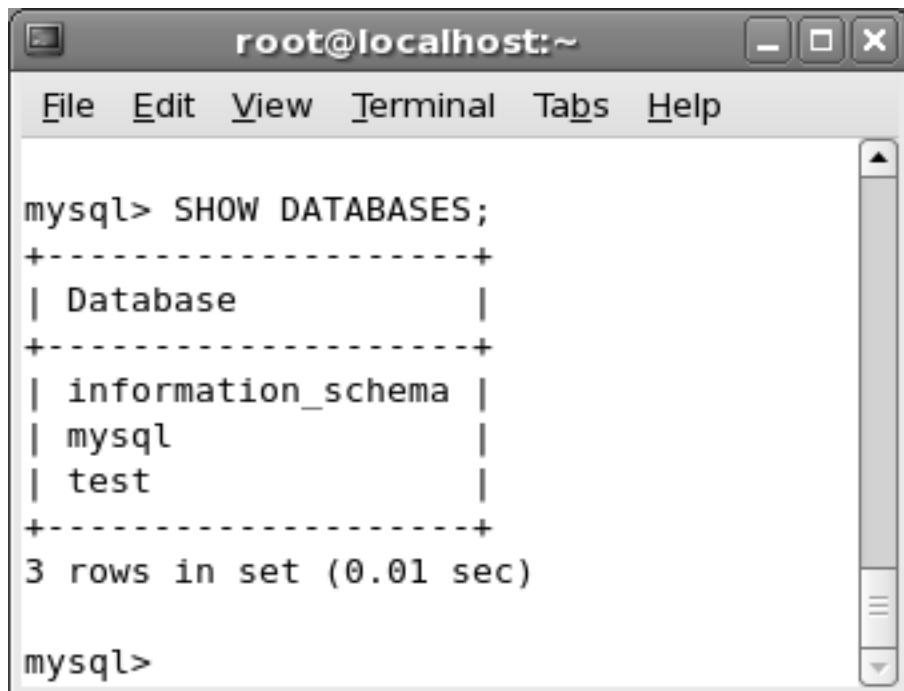
Session 2

Installing and Configuring MySQL

To view the list of default databases present on the server after installation, enter the following command at the command prompt:

```
SHOW DATABASES;
```

Figure 2.21 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area displays the following MySQL command output:

```
mysql> SHOW DATABASES;
+-----+
| Database      |
+-----+
| information_schema |
| mysql          |
| test           |
+-----+
3 rows in set (0.01 sec)

mysql>
```

Figure 2.21: SHOW DATABASES Command

2.4 Using Scripts to Customize the Configuration of MySQL

The configure script enables you to control the MySQL distribution. You can build MySQL by using the various options of configure on the command line. You can configure MySQL using the combination of command-line options, configuration files, and environment variables.

The environment variables used by MySQL are listed in table 2.2.

Variable	Description
CXX	specifies the name of C++ compiler
CC	specifies the name of C compiler
CFLAGS	specifies instructions for the C compiler
CXXFLAGS	specifies instructions for the C++ compiler
DBI_USER	specifies the default user name for Perl DBI

Session 2

Installing and Configuring MySQL

Variable	Description
DBI_TRACE	specifies trace options for Perl DBI
HOME	specifies the default path for the MySQL history file. The default path is \$HOME/.mysql_history
LD_RUN_PATH	specifies the directories to be searched by the dynamic linker
MYSQL_DEBUG	enables debugging options
MYSQL_GROUP_SUFFIX	specifies to read groups with a suffix
MYSQL_HISTFILE	specifies the default path for the MySQL history file. The default path is \$HOME/.mysql_history
MYSQL_HOME	specifies the path of the directory where my.cnf file is located
MYSQL_HOST	specifies the default host name for the command line client
MYSQL_PS1	specifies the command prompt for the command line client
MYSQL_PWD	specifies the default password while connecting to MySQL
MYSQL_TCP_PORT	specifies the default TCP/IP port number
MYSQL_UNIX_PORT	specifies the default UNIX socket file name
PATH	enables the shell to search for MySQL programs
TMPDIR	specifies the directory to create temporary files
TZ	specifies the local time zone
UMASK	specifies the mode to create files
UMASK_DIR	specifies the mode to create directories
USER	specifies the user name on Windows when connecting to MySQL

Table: 2.2: Environment Variables in MySQL

The configuration options are stored in configuration files. These configuration files are available in the binary source code. You will be required to download the source code and extract the files to a folder before making changes to the configuration files. You must download the **MySQL-community-5.1.56-1.rhel5.src.rpm** package to extract the source code.

Note: The source code for MySQL is also available in RPM packages under the Source Code section of the Downloads page on the MySQL Website.

To extract the RPM package containing the source code:

1. Create a folder named **mysql** under the root directory.
2. Select **Applications → Accessories → Archive Manager**. The Archive Manager screen is displayed as shown in figure 2.22.

Session 2

Installing and Configuring MySQL

Concepts

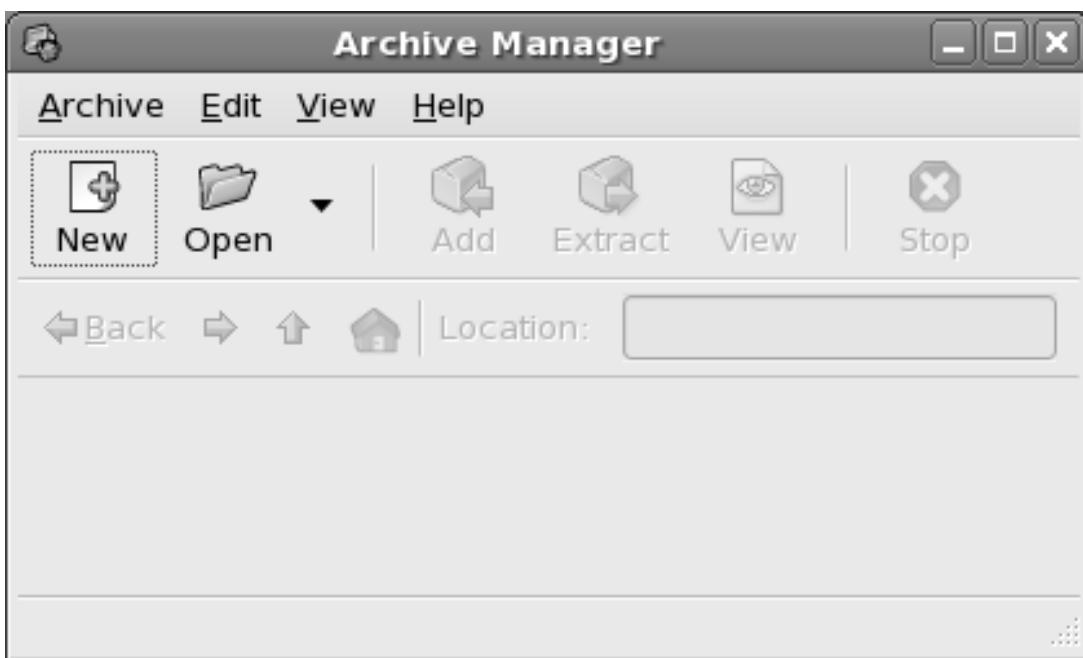


Figure 2.22: Archive Manager

3. Select **Archive → Open**. The Open dialog box is displayed as shown in figure 2.23.

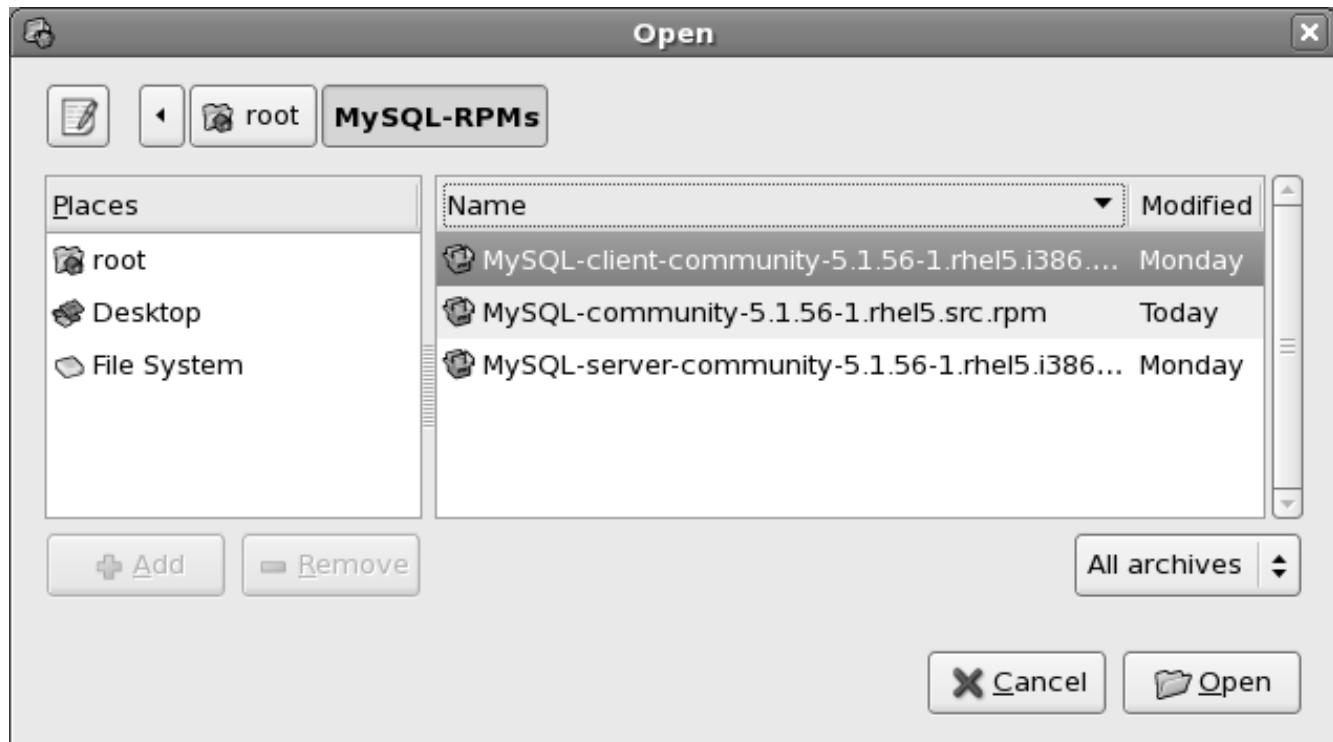


Figure 2.23: Open Dialog Box

Session 2

Installing and Configuring MySQL

4. Browse to the folder containing the **MySQL-community-5.1.56-1.rhel5.src.rpm** file.
5. Click **Open**. The contents of the RPM package are displayed as shown in figure 2.24.

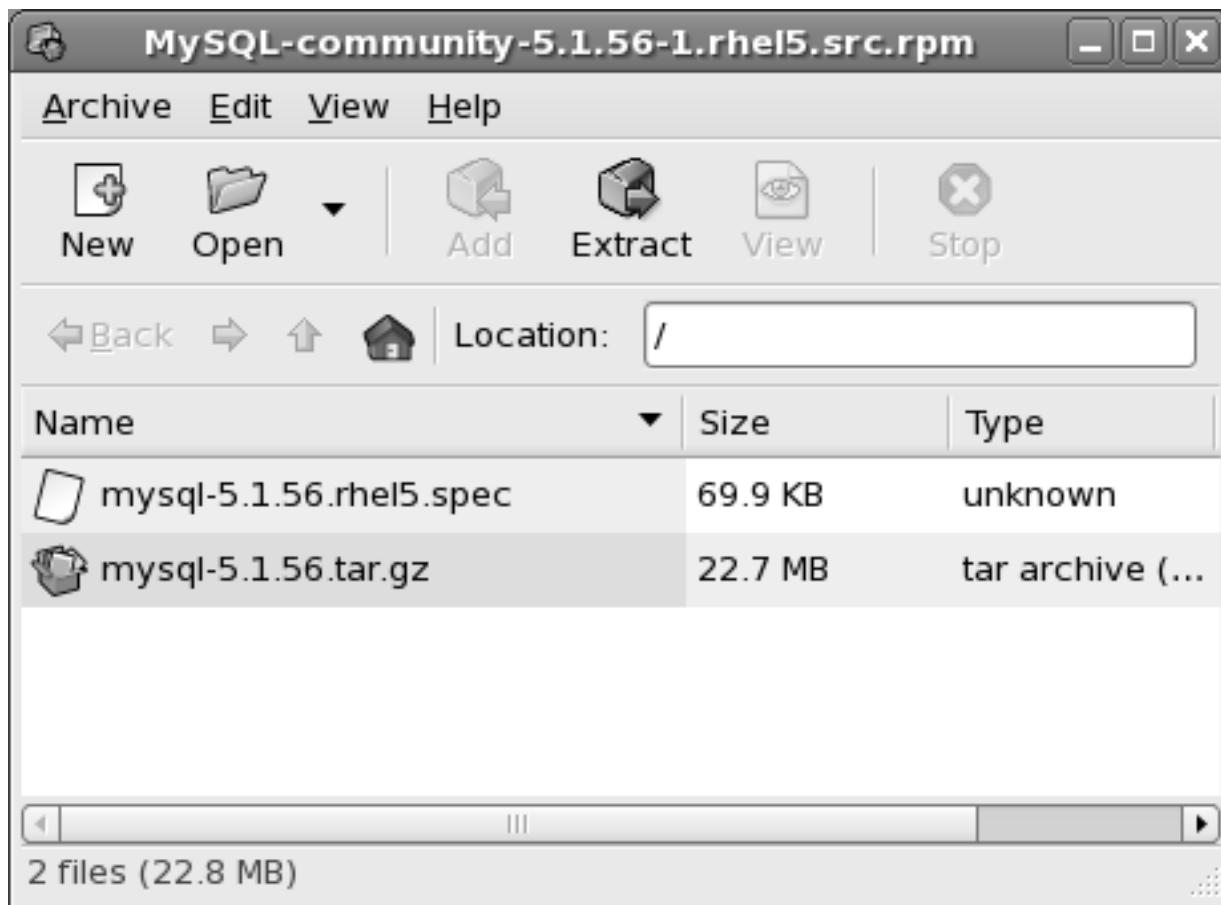


Figure 2.24: RPM Package Contents

6. Select **Archive → Extract**. The Extract dialog box is displayed, as shown in figure 2.25.

Session 2

Installing and Configuring MySQL

Concepts



Figure 2.25: Extract Dialog Box

7. Browse to the **mysql** folder in the **Extract in folder** drop-down box.
8. Click **Extract**. The application extracts the package contents.
9. Right-click the **mysql-5.1.56.tar.gz** file.
10. Select **Extract Here** from the popup menu. The file contents are extracted as shown in figure 2.26.

Session 2

Installing and Configuring MySQL

Concepts

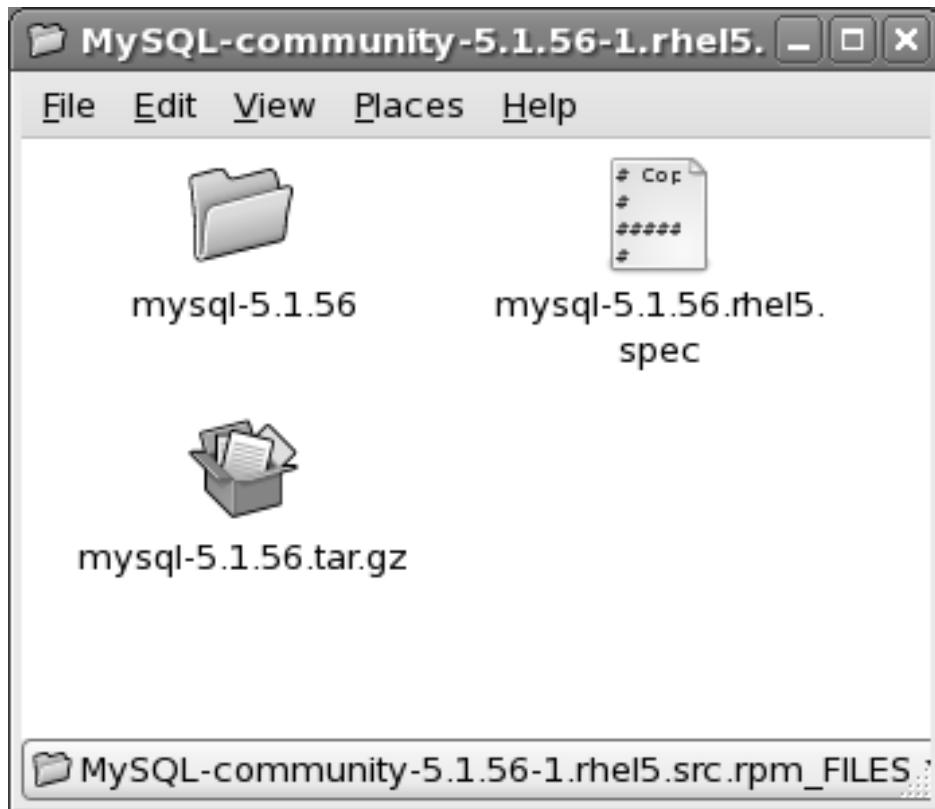


Figure 2.26: Extracted Files

After extracting the package contents, you will create a symbolic link. A symbolic link enables you to move a database directory on a specific disk. You will name the link as root and use it to start the MySQL client.

11. Right click the **mysql-5.1.56** folder and select **Open In Terminal**. The Terminal window is displayed as shown in figure 2.27.

Session 2

Installing and Configuring MySQL

Concepts

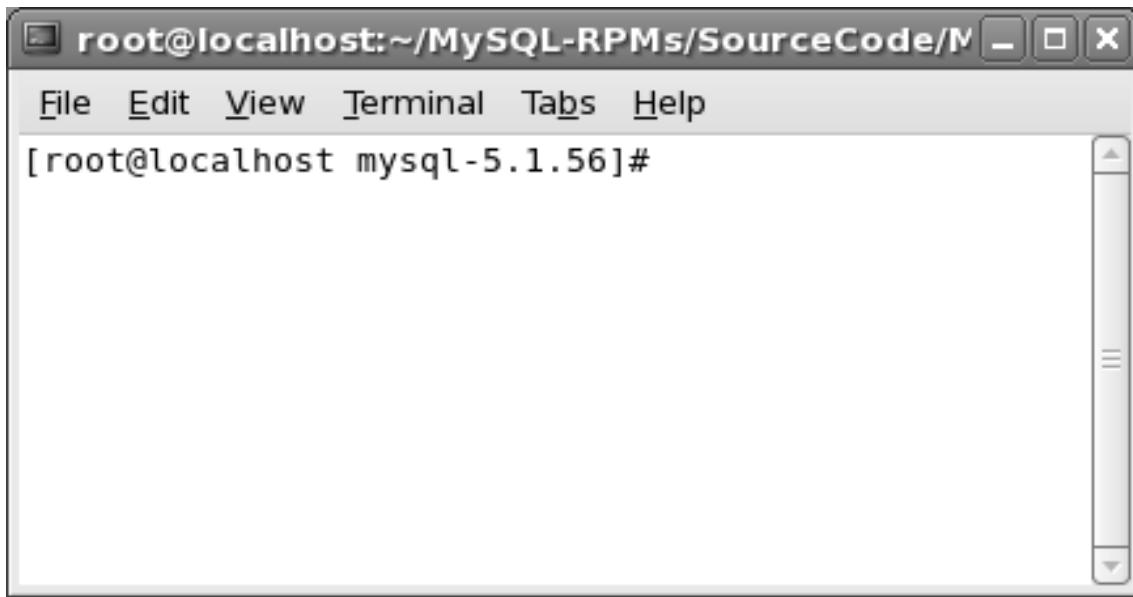


Figure 2.27: Terminal Window

12. Enter the following command at the command prompt:

```
ln -s MySQL-community-5.1.56.rhel5.src root
```

Figure 2.28 displays the output of the command.

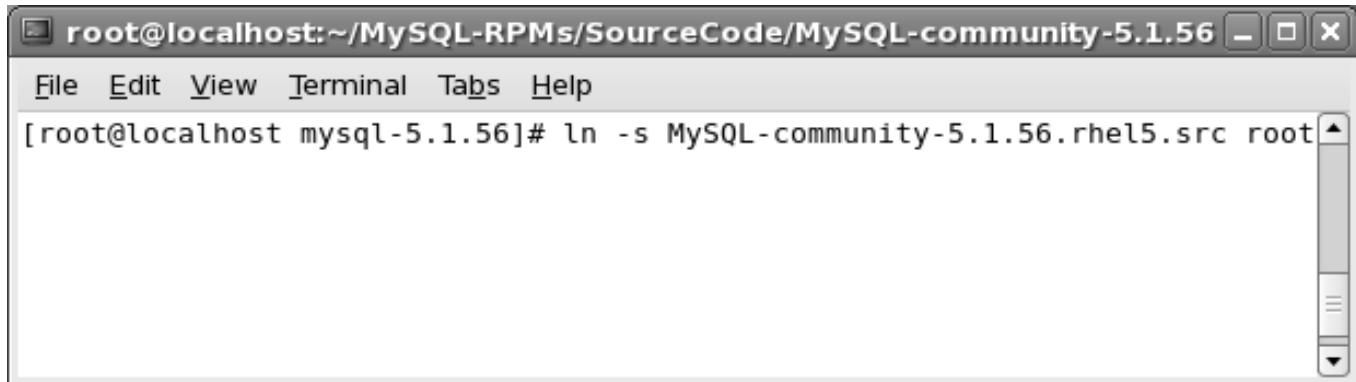


Figure 2.28: Creating a Symbolic Link

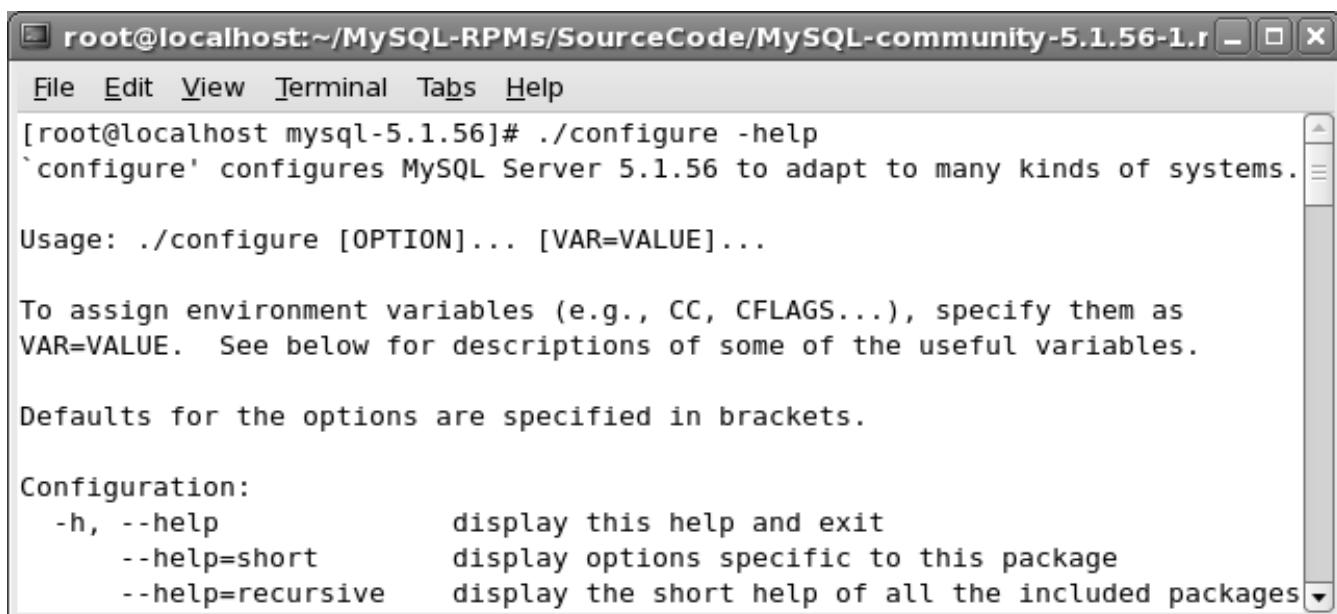
13. To view the options for configuration, enter the following command at the command prompt:

```
./configure --help
```

Figure 2.29 displays the output of the command.

Session 2

Installing and Configuring MySQL



The screenshot shows a terminal window titled "root@localhost:~/MySQL-RPMs/SourceCode/MySQL-community-5.1.56-1.r". The window contains the following text:

```
[root@localhost mysql-5.1.56]# ./configure --help
'configure' configures MySQL Server 5.1.56 to adapt to many kinds of systems.

Usage: ./configure [OPTION]... [VAR=VALUE]...

To assign environment variables (e.g., CC, CFLAGS...), specify them as
VAR=VALUE. See below for descriptions of some of the useful variables.

Defaults for the options are specified in brackets.

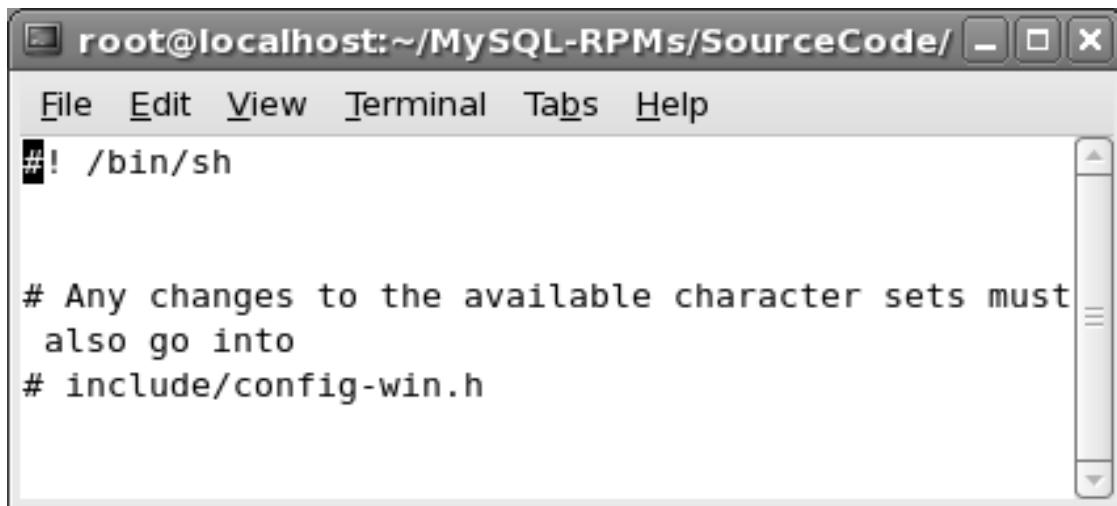
Configuration:
  -h, --help           display this help and exit
  --help=short         display options specific to this package
  --help=recursive    display the short help of all the included packages
```

Figure 2.29: Help Options of Configure

14. To open the configuration file, enter the following command at the command prompt:

```
vi configure
```

Figure 2.30 displays the configuration file in vi editor.



The screenshot shows a terminal window titled "root@localhost:~/MySQL-RPMs/SourceCode/". The window contains the following text:

```
#!/bin/sh

# Any changes to the available character sets must
# also go into
# include/config-win.h
```

Figure 2.30: Configure file in vi Editor

The following are commonly used options with the configure command to customize the distribution of MySQL:

Session 2

Installing and Configuring MySQL

- To compile only MySQL client and library programs without the server, use configure with the --without-server option as shown:

```
./configure --without-server
```

- To relocate log files and database directories from /usr/local/var location, use configure with the --prefix option.

```
./configure --prefix=/usr/local/mysql  
./configure --prefix=/usr/local\--localstatedir= /usr/local/mysql/data
```

The first command changes the installation default location from /usr/local to /usr/local/mysql. The second command maintains the chosen default installation location, and places the database directories from /usr/local/var to /usr/local/mysql/data.

- To build binary distributions, to increase the speed, or to work with the bugs, use configure with the -with-client-Idflags option.

```
./configure -with-client-Idflags=-all-static\--with-mysqld-Idflags=-all-  
static
```

- To configure MySQL for not using DEFAULT value in the columns that cannot contain NULL values. Use configure with the option DDONT_USE_DEFAULT_FIELDS as shown:

```
CXXFLAGS=-DDONT_USE_DEFAULT_FIELDS ./configure
```

CXXFLAGS is a flag used for C++ compiler. This is one of the environment variables. With the help of this command, it is specified that no columns will be accepting NULL value. This will help the INSERT command to avoid errors while inserting values in the tables.

- To change the default character set, use configure with the --with-charset option as shown:

```
./configure --with-charset=CHARSET
```

You can use the character sets, such as big5, dec8, cp850, hp8, koi8r, latin1, latin2, swe7, ascii, ujis, sjis, hebrew, tis620, euckr, koi8u, gb2312, greek, cp1250, gbk, latin5, armSCII8, utf8, ucs2, cp866, keybcs2, macce, macroman, cp852, latin7, utf8mb4, cp1251, utf16, cp1256, cp1257, utf32, binary, geostd8, cp932, and eucjpm as the default character set in MySQL.

- To configure MySQL with codes for debugging, use configure with the --with-debug option as shown:

Session 2

Installing and Configuring MySQL

```
./configure --with-debug
```

You can debug errors and get the output with the help of this command.

2.5 Controlling MySQL Options Using my.cnf file

MySQL reads the default startup options for the server and the client. These default options are stored in the my.cnf file.

On Microsoft Windows OS, MySQL reads the default options from my.ini file. The MySQL Server Instance Configuration wizard generates this file and stores it in the MySQL Server 5.1 folder under the **C:\Program Files** folder.

On the Red Hat Enterprise Linux platform, the files from which MySQL reads default options are listed in table 2.3.

File	Purpose
/etc/my.cnf	Defines global options
/etc/mysql/my.cnf	Defines global options
\$MYSQL_HOME/my.cnf	Defines server specific options
~/.my.cnf	Defines user specific options

Table 2.3: Default Option Files For MySQL in Red Hat Enterprise Linux

The programs that support option files are mysql, mysqladmin, mysqld, mysqld_safe, mysql, server, mysqldump, mysqlimport, mysqlshow, mysqlcheck, myisamchk, and myisampack.

An option file consists of the following details:

- **Comments** - starts lines with symbols, such as '#' or ';'. A comment is non-executable statement in a file that is added to provide extra information to the user
- **Group** - specifies the name for which the user wants to set the options
- **Option** - specifies the --option on the command line
- **option = value** - specifies the value for the option. It is the same as specifying --option=value on the command line
- **set-variable = variable value** - assigns a variable. It is the same as --set-variable variable=value on the command line

Session 2

Installing and Configuring MySQL

To open the **my.cnf** file:

1. Browse to the **mysql-test** folder under the mysql source code folder.
2. Right click the include folder and select **Open In Terminal**. The Terminal window appears.
3. Enter the following command at the command prompt:

```
vi default_my.cnf
```

Figure 2.31 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~/MySQL-RPMs/SourceCode/MySQL-community-5.1.56-1.rhel". The window contains the following text:

```
# Use default setting for mysqld processes
!include default_mysqld.cnf

[mysqld.1]

# Run the master.sh script before starting this process
#!run-master-sh

log-bin=          master-bin

[mysqlbinlog]
disable-force-if-open

# mysql_fix_privilege_tables.sh does not read from [client] so it
# need its own section
[mysql_fix_privilege_tables]
socket=           @client.socket
port=             @client.port
user=             @client.user
password=         @client.password

[ENV]
"default_my.cnf" 25L, 597C
```

Figure 2.31: default_my.cnf in vi editor

In figure 2.31, the lines beginning with the # symbol are comments. These lines are not executable and provide information about this file. The file contains:

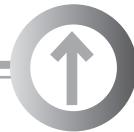
- Socket

Session 2

Installing and Configuring MySQL

Concepts

- Port Number
- User Name
- Password



Summary

- You can use binary distribution format instead of source distribution format while installing MySQL. After the installation of binary distribution, default directories are created in the source directory.
- Binary distribution format contains setup program that installs every component for the server from the start. Source distribution format contains all the codes and support files for making the files executable.
- Solaris, Linux, Windows XP/2003/Vista/2007, Mac OS X, FreeBSD, and AIX are the different operating systems that support MySQL.
- You can customize the configuration of MySQL using `./configure` command. You can also use various options of the `configure` command to configure MySQL on the command line.
- The environment variables enable you to edit or alter the `configure` file.
- MySQL can be configured using the combination of command-line options, configuration files, and environment variables.
- The programs like `mysql`, `mysqladmin`, `mysqld`, `mysqld_safe`, `mysql.server`, `mysqldump`, `mysqlimport`, `mysqlshow`, `mysqlcheck`, `myisamchk`, and `myisampack` supports option files of `my.cnf` file.
- The OS can be configured to start MySQL at startup.



Check Your Progress

1. To start MySQL client session in MS-DOS prompt, type _____.
 - a. mysql -u root
 - b. mysql
 - c. mysqld-shareware –standalone
 - d. mysql -u
2. Which of the following environment variable locates the MySQL history file?
 - a. HOME
 - b. MYSQL_PSL
 - c. MYSQL_HOST
 - d. MYSQL_HISTFILE
3. Which of the following options for MySQL can be specified in the **my.cnf** file?
 - a. Global
 - b. User
 - c. Server
 - d. Comment
4. In binary distribution format, the installation is configured using _____.
 - a. System Variables
 - b. Environment variables
 - c. Server Variables
 - d. Scripts



Check Your Progress

Concepts

5. Which command is used to configure the installation of MySQL in Red Hat Enterprise Linux?
 - a. ./configure
 - b. ./configure -help
 - c. configure
 - d. vi configure

GROWTH
RESEARCH
OBSERVATION
UPDATES
PARTICIPATION



Objectives

At the end of this session, the student will be able to:

- *Install MySQL on Microsoft Windows.*
- *Configure MySQL on Microsoft Windows.*
- *Install MySQL on Red Hat Linux.*
- *Configure MySQL on Red Hat Enterprise Linux.*

The steps given in the session are detailed, comprehensive and carefully thought through. This has been done so that the learning objectives are met and the understanding of the software is complete. Please follow the steps carefully.

Part I - For the first 1.5 hours:

Install MySQL

You can install MySQL on the following platforms:

- Microsoft Windows
- Red Hat Enterprise Linux

Note: You can download the mysql-5.1.56-win32.msi file to install MySQL from their official Website
<http://dev.mysql.com/downloads/mysql/5.1.html>

Installing and Configuring MySQL on Microsoft Windows

Download the MySQL executable file in a temporary folder named mysqltmp. To install MySQL on Microsoft Windows XP platform, perform the following steps:

Note: You will require administrative rights on Microsoft Windows XP to install MySQL. Contact the system administrator for administrative permissions to install MySQL.

1. **To install MySQL on Microsoft Windows XP, run `mysql-5.1.56-win32.msi` from `mysqltmp` folder. The Open File – Security Warning dialog box is displayed as shown in figure 3.1.**

Session 3

Installing and Configuring MySQL (Lab)



Figure 3.1: Open File – Security Warning Dialog Box

2. Click Run. The Microsoft Windows installer prepares the installation process and the MySQL Server 5.1 - Setup Wizard dialog box is displayed as shown in figure 3.2.

Session 3

Installing and Configuring MySQL (Lab)



Lab Guide

Figure 3.2: MySQL Server 5.1 Setup Wizard

3. Click Next. The License Agreement pane of the MySQL Server 5.1 - Setup Wizard dialog box is displayed as shown in figure 3.3.

Session 3

Installing and Configuring MySQL (Lab)

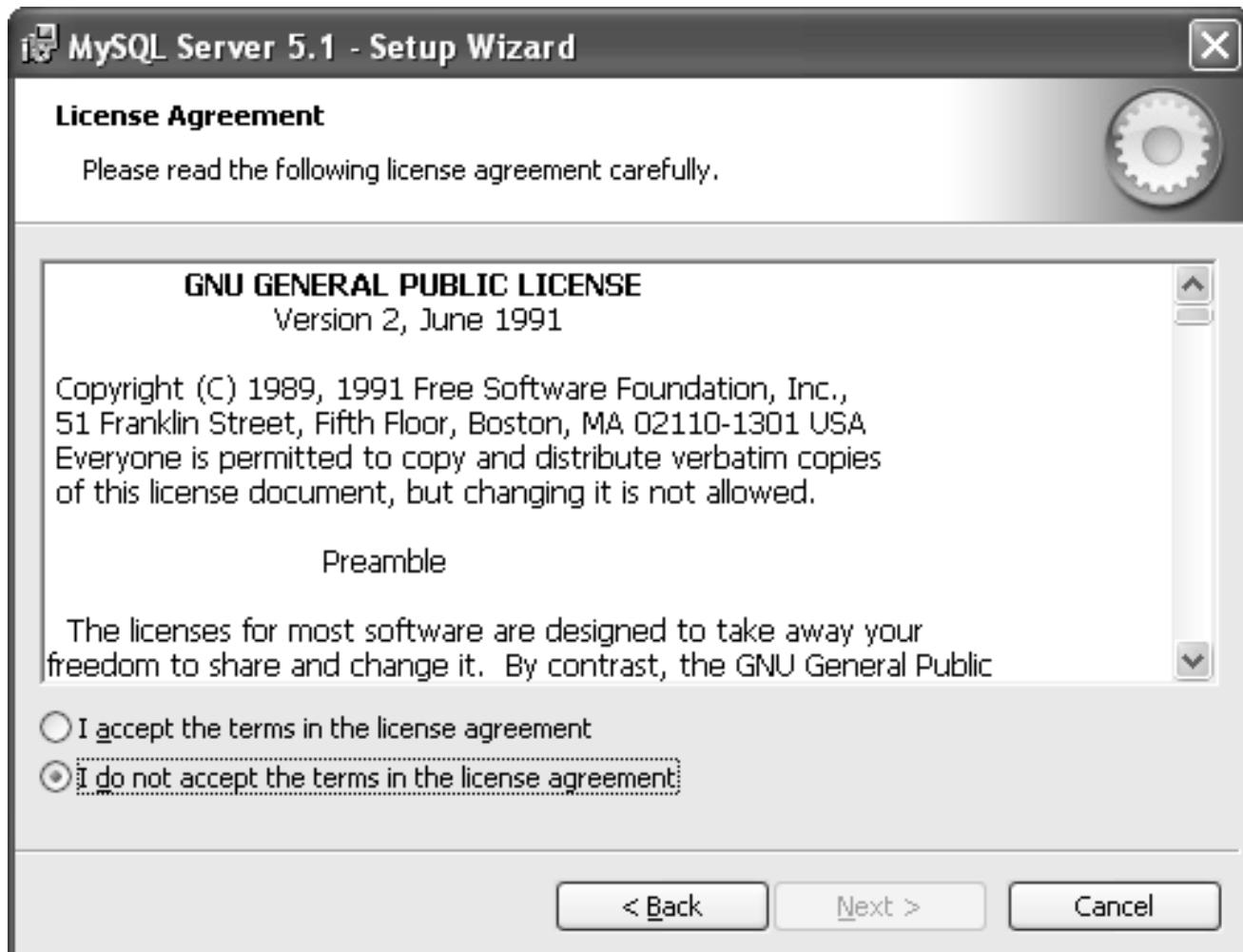
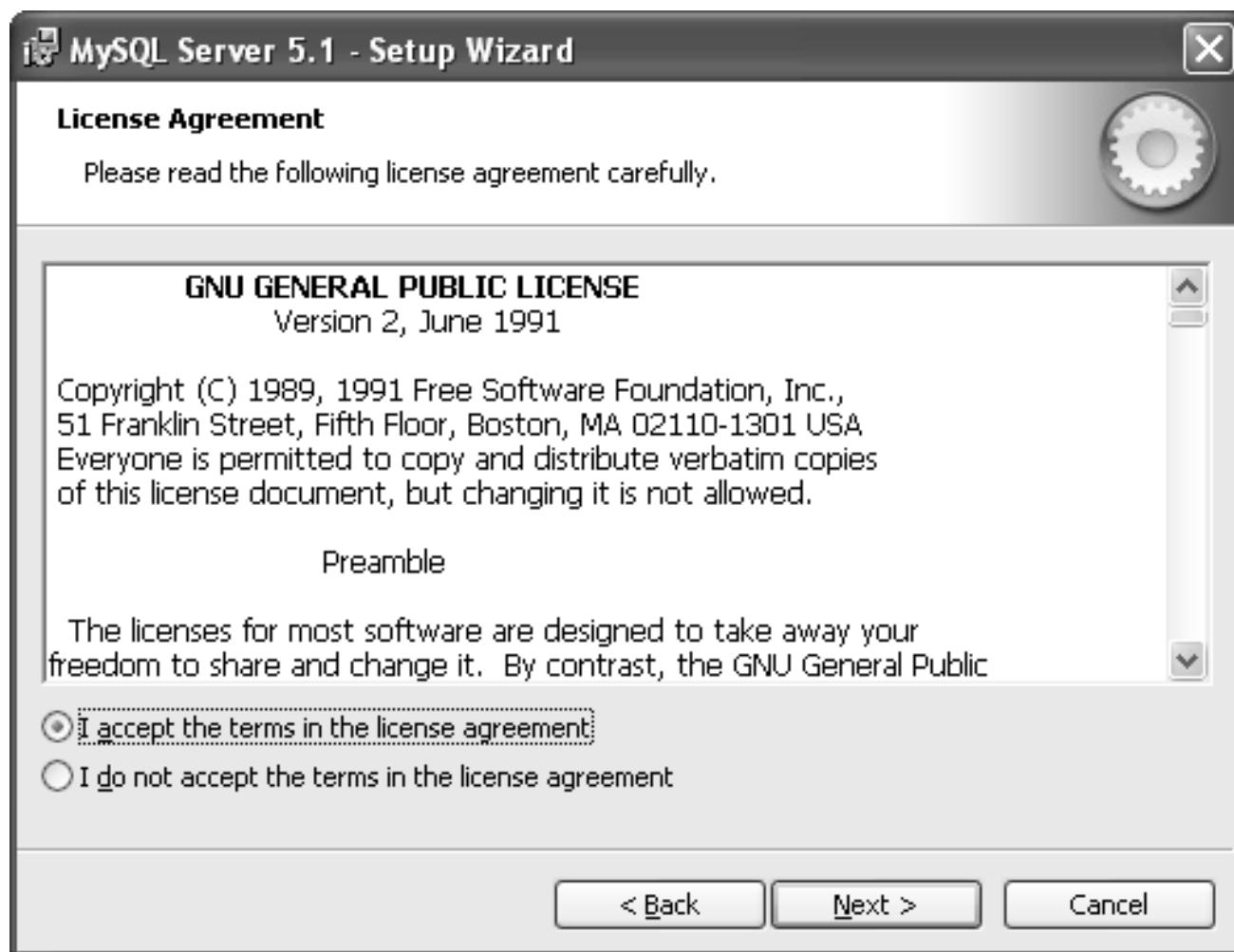


Figure 3.3: License Agreement Dialog Box

4. Click 'I accept the terms in the license agreement' option and the MySQL Server 5.1 - Setup Wizard activates the Next button as shown in figure 3.4.

Session 3

Installing and Configuring MySQL (Lab)



Lab Guide

Figure 3.4: License Agreement Dialog Box after Selection of Terms

5. Click Next. The 'Setup Type' pane of the MySQL Server 5.1 - Setup Wizard dialog box is displayed as shown in figure 3.5.

Session 3

Installing and Configuring MySQL (Lab)



Figure 3.5: Setup Type Dialog Box

6. Click Typical.
7. Click Next The 'Ready to Install the Program' pane of the MySQL Server 5.1 - Setup Wizard dialog box is displayed as shown in figure 3.6.

Session 3

Installing and Configuring MySQL (Lab)



Figure 3.6: Reviewing Installation Settings

8. Click **Install**. The Setup Wizard completes the installation process and the MySQL Enterprise dialog box is displayed as shown in figure 3.7.

Session 3

Installing and Configuring MySQL (Lab)

Lab Guide

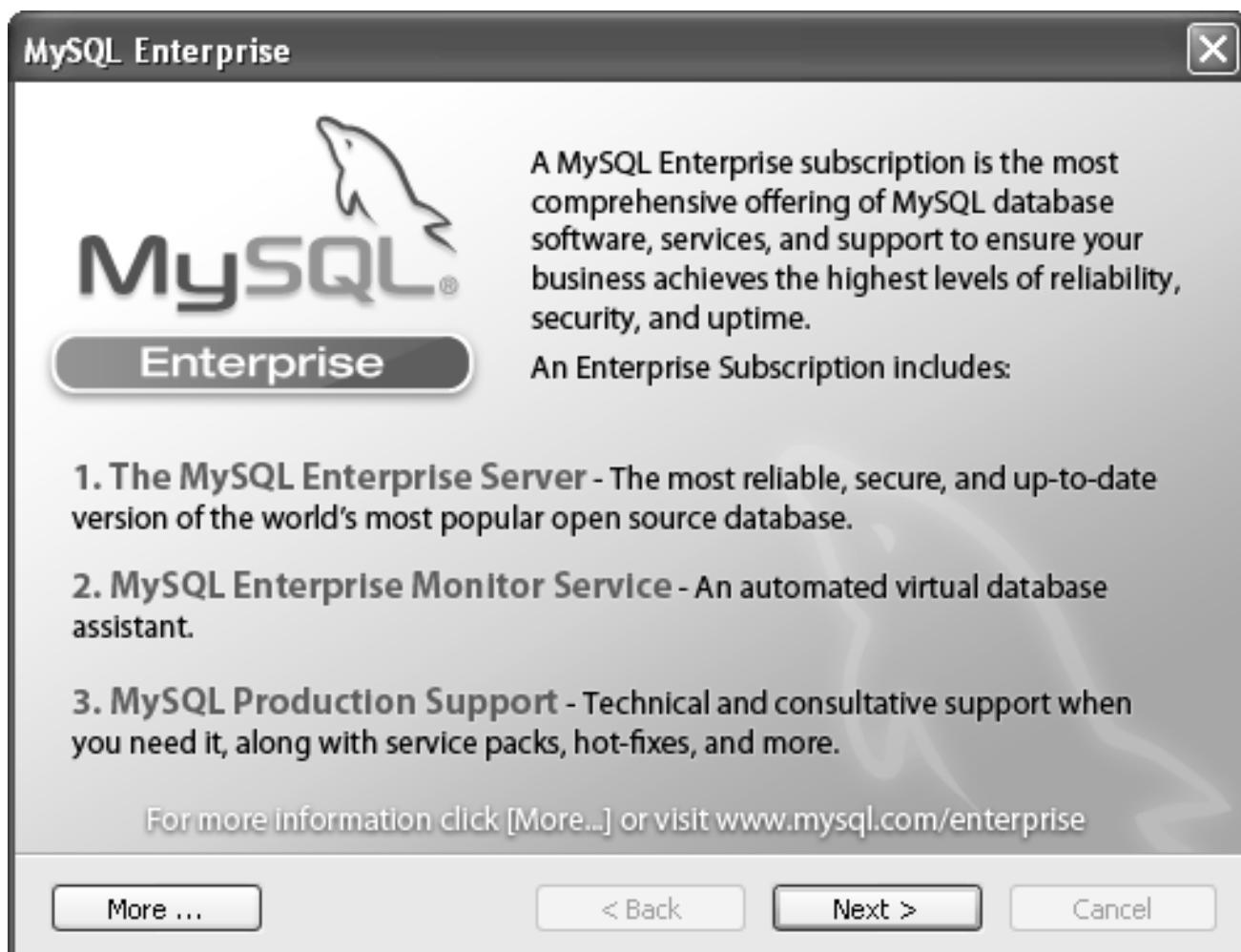
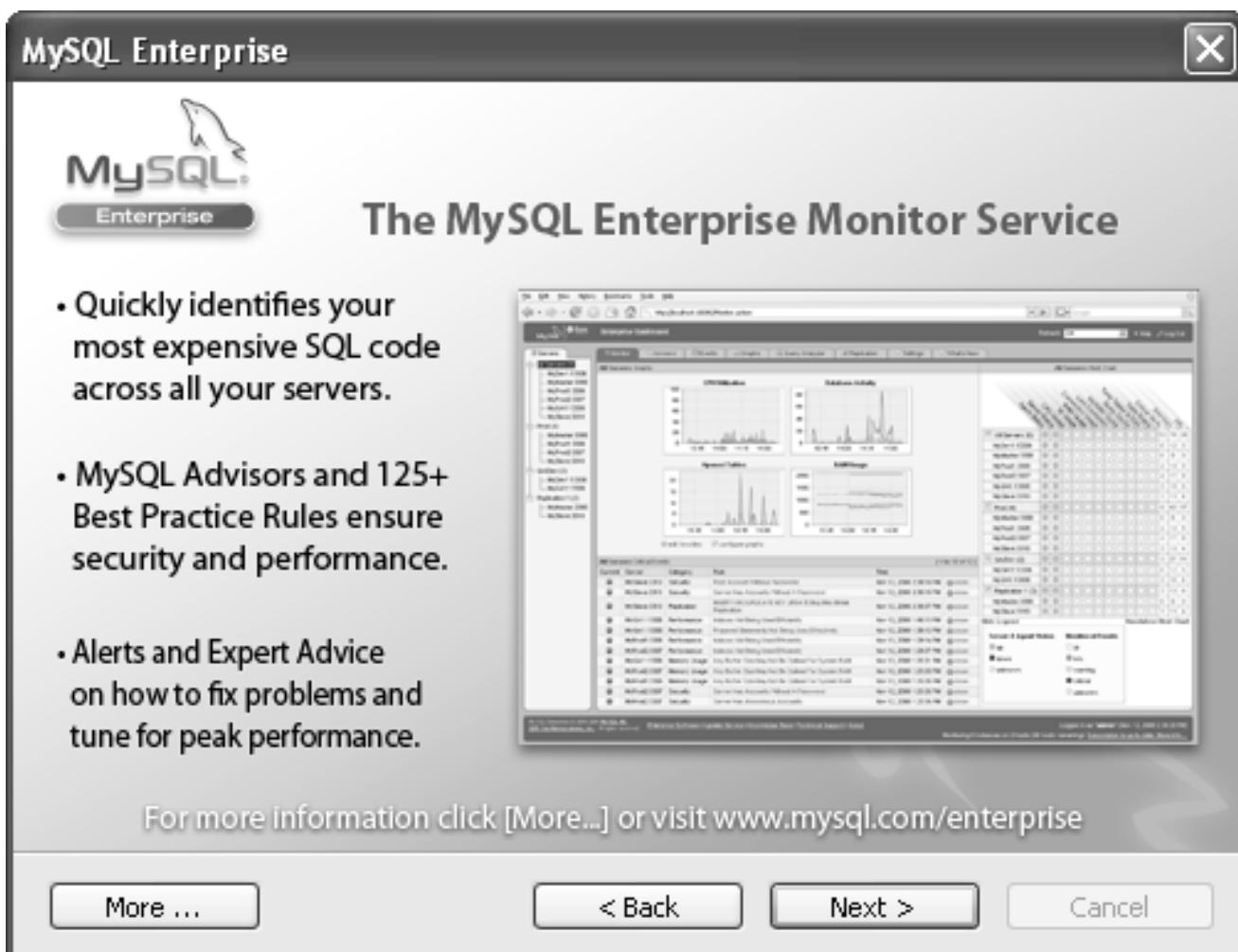


Figure 3.7: MySQL Enterprise Dialog Box

9. Click Next. The MySQL Enterprise Monitor Service pane is displayed as shown in figure 3.8.

Session 3

Installing and Configuring MySQL (Lab)



Lab Guide

Figure 3.8: MySQL Enterprise Monitor Service Dialog Box

10. Click Next. The Setup Wizard completes the installation process and displays the Wizard Completed pane as shown in figure 3.9.

Session 3

Installing and Configuring MySQL (Lab)

Lab Guide



Figure 3.9: MySQL Installation Complete

11. Click Finish. The MySQL Server Instance Configuration Wizard dialog box is displayed as shown in figure 3.10.

Session 3

Installing and Configuring MySQL (Lab)

Lab Guide

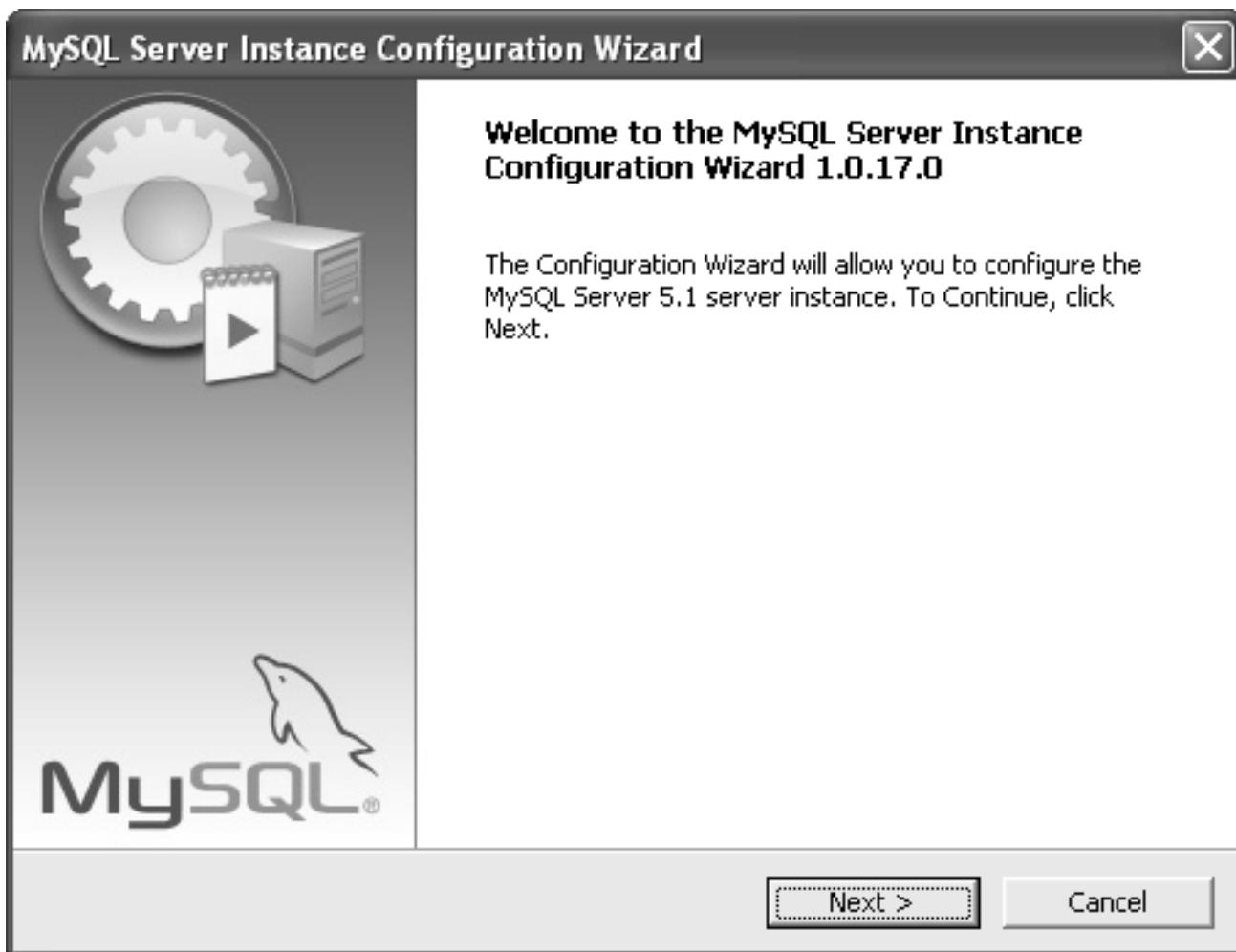


Figure 3.10: MySQL Server Instance Configuration Wizard

12. Click Next. The MySQL Server Instance Configuration pane is displayed as shown in figure 3.11.

Session 3

Installing and Configuring MySQL (Lab)

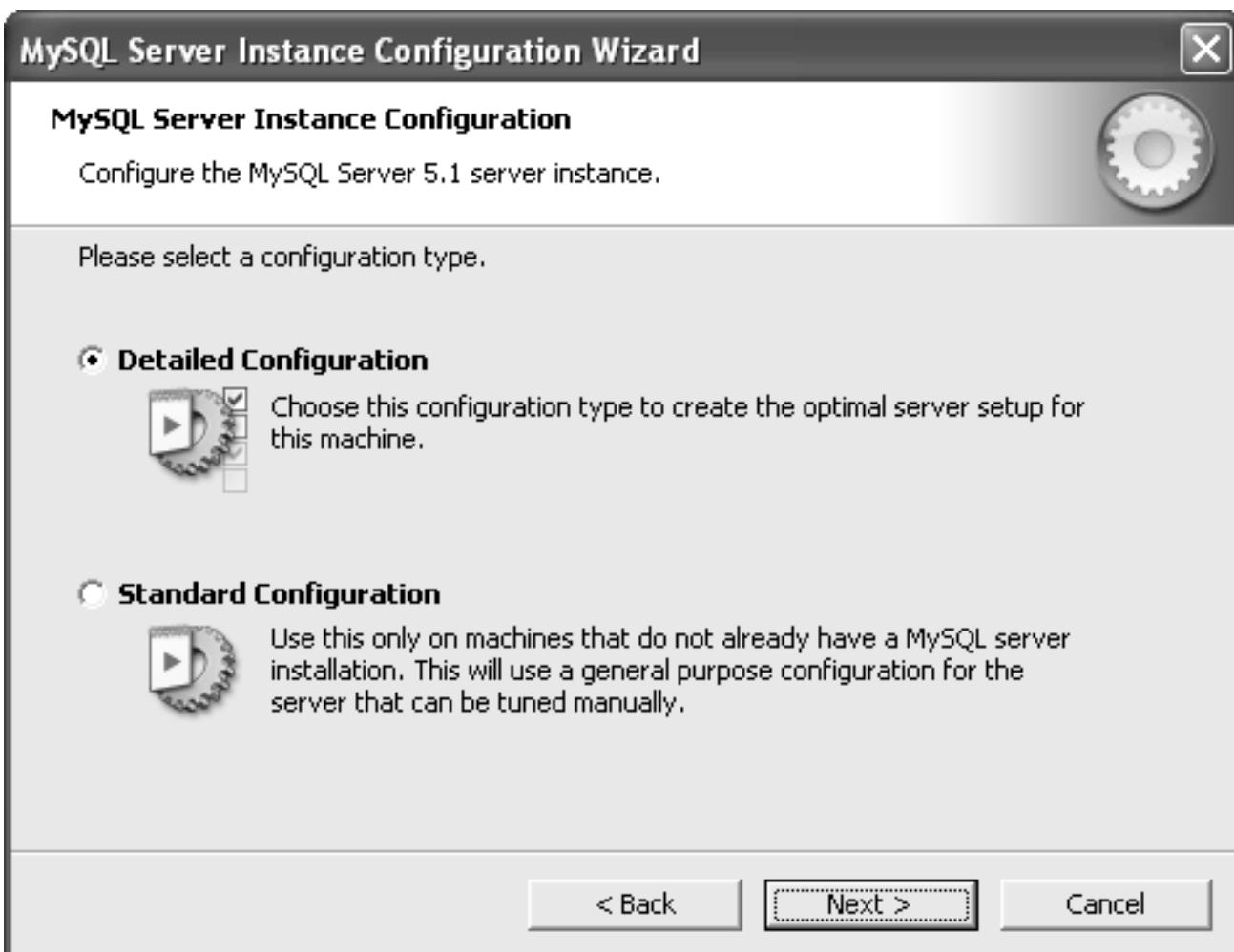


Figure 3.11: MySQL Configuration Type Dialog Box

13. Click Standard Configuration.
14. Click Next. The Windows Options pane is displayed as shown in figure 3.12.

Session 3

Installing and Configuring MySQL (Lab)

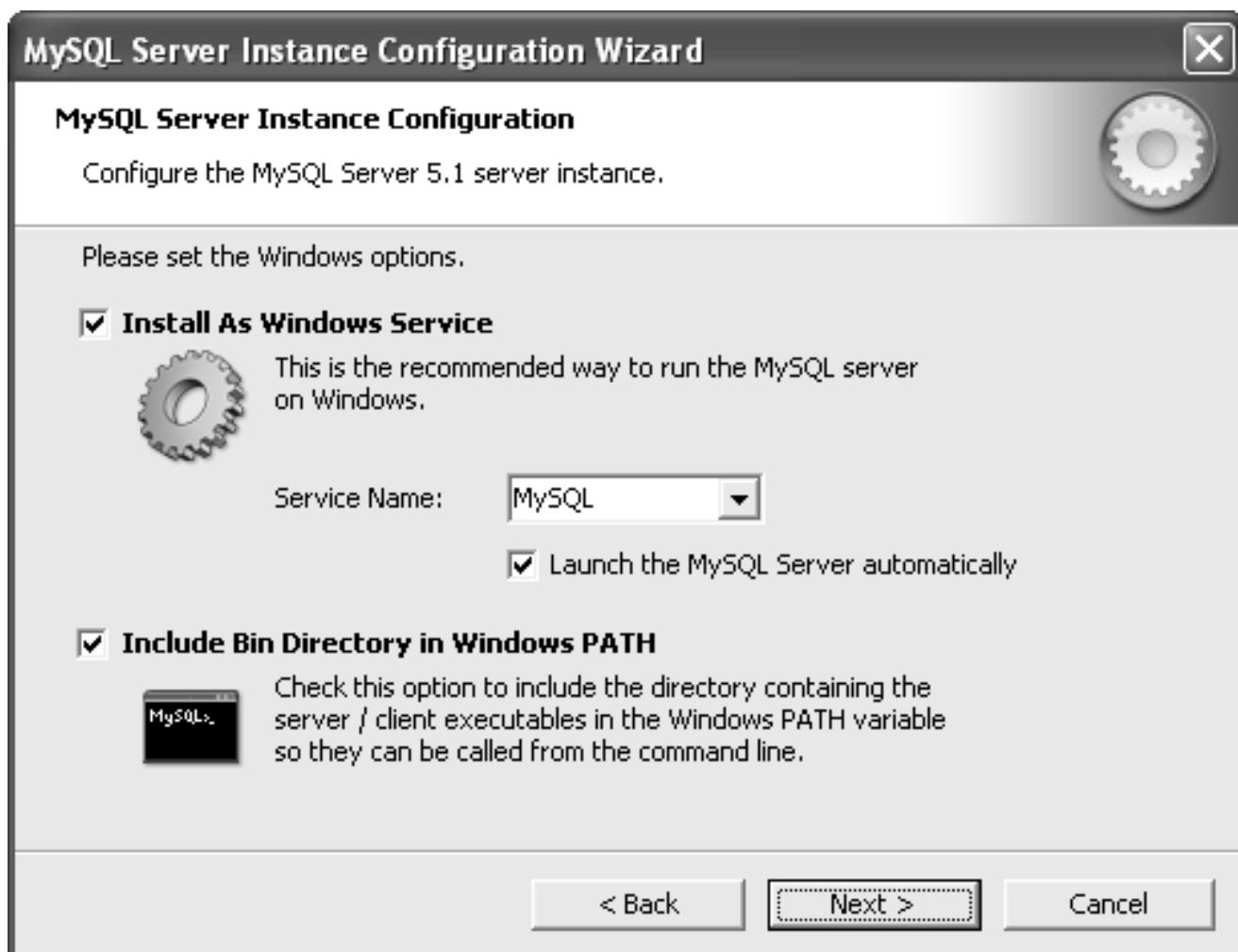


Figure 3.12: Windows Options Dialog Box

15. Select 'Install As Windows Service' check box.
16. Select 'Include Bin Directory in Windows PATH' check box.
17. Click Next. The Security Options pane is displayed as shown in figure 3.13.

Session 3

Installing and Configuring MySQL (Lab)



Figure 3.13: Security Options Dialog Box

18. Clear the 'Modify Security Settings' check box.
19. Click Next. The MySQL Server Instance Configuration pane is displayed for configuration of the tasks. Figure 3.14 displays the MySQL Server Configuration Tasks.

Note: You can specify a password for the root user in MySQL by entering the password in the text boxes.

Session 3

Installing and Configuring MySQL (Lab)

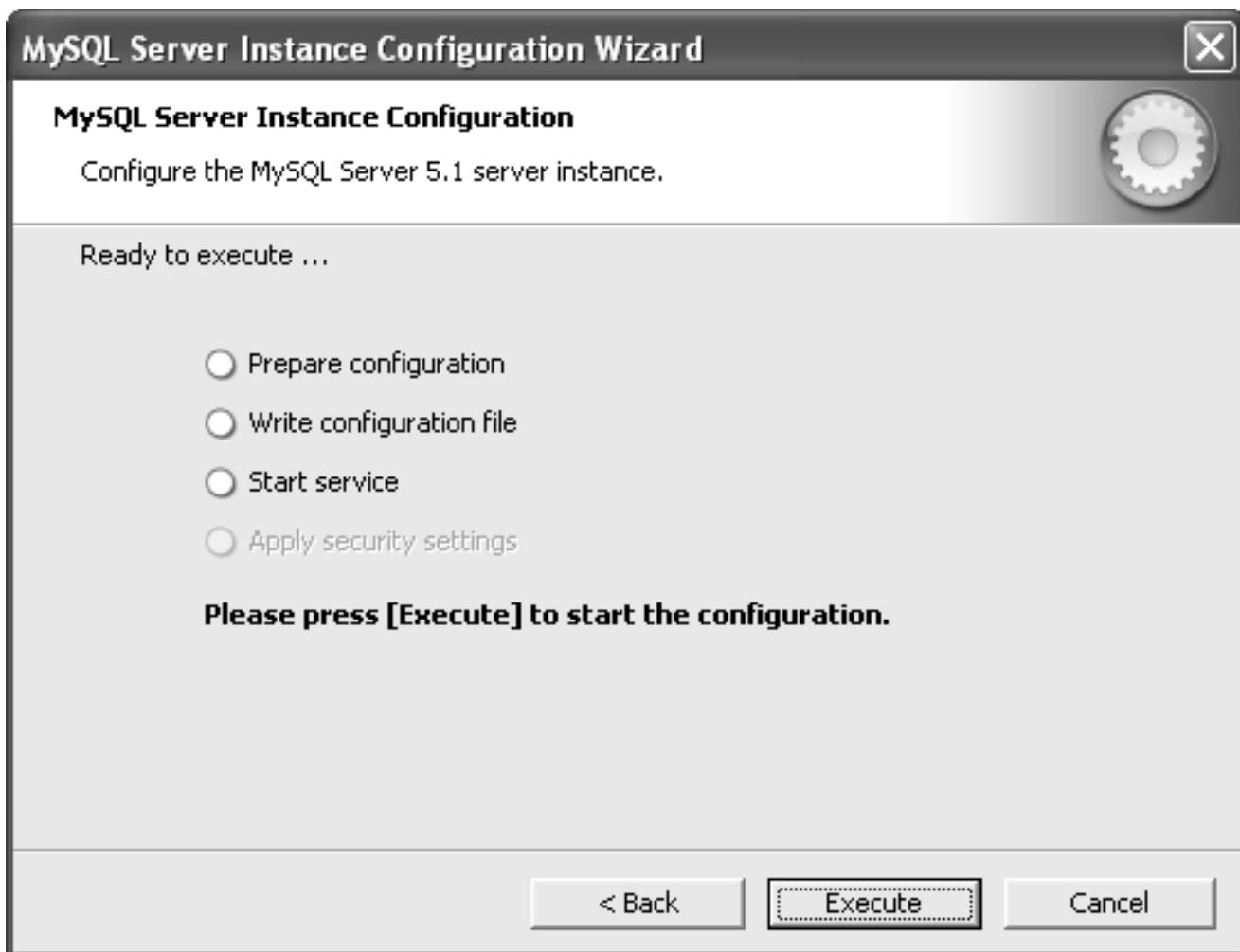


Figure 3.14: MySQL Server Configuration Tasks

20. Click Execute. The wizard configures the settings and displays a confirmation message box, as shown in figure 3.15.

Session 3

Installing and Configuring MySQL (Lab)

Lab Guide

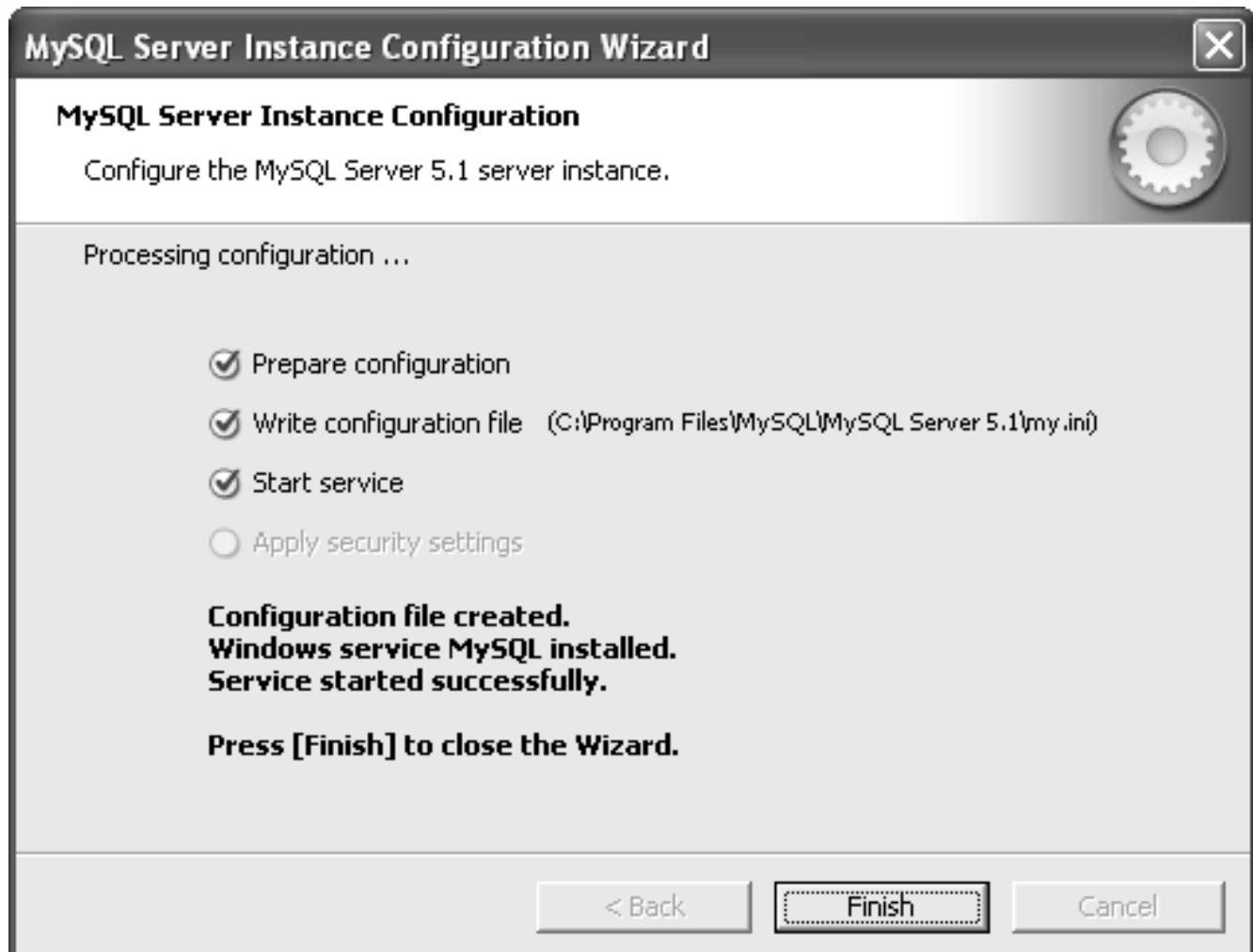


Figure 3.15: Confirmation Message Box

21. Click Finish.
22. Start the command prompt.
23. Change the directory to C:\Program Files\MySQL Server 5.1.
24. Type mysql.
25. Press Enter. The MySQL prompt appears as shown in figure 3.16.

Session 3

Installing and Configuring MySQL (Lab)

Lab Guide

```
C:\Program Files\MySQL\MySQL Server 5.1>mysql
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.1.56-community MySQL Community Server <GPL>

Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights
This software comes with ABSOLUTELY NO WARRANTY. This is free soft
and you are welcome to modify and redistribute it under the GPL v2

Type 'help;' or '\h' for help. Type '\c' to clear the current input
mysql> =
```

Figure 3.16: MySQL Prompt

Installing MySQL on Linux

Before installing MySQL on Red Hat Linux, create a group and a user with the GUI environment.

Note: You must login as root or have root privileges to create a group and user account.

1. Click System → Administration → Users and Groups to create a group and a user for accessing MySQL. The User Manager dialog box is displayed as shown in figure 3.17.

Note: If you are not a root user, Red Hat Enterprise Linux will prompt you to enter the root password in step 1. You cannot create a group or user account if you do not specify the root password.

Session 3

Installing and Configuring MySQL (Lab)



Figure 3.17: User Manager Dialog Box

2. Click File → Add Group to create a new group. The Create New Group dialog box is displayed.
3. Type mysql in the Group Name box as shown in figure 3.18.

Session 3

Installing and Configuring MySQL (Lab)



Lab Guide

Figure 3.18: Create New Group Dialog Box

4. Click OK.
5. Click the Groups tab in the User manager dialog box as shown in figure 3.19.

Session 3

Installing and Configuring MySQL (Lab)

Lab Guide



Figure 3.19: Groups Tab

6. Select mysql.
7. Click File → Properties. The Group Properties dialog box is displayed as shown in figure 3.20.

Session 3

Installing and Configuring MySQL (Lab)



Lab Guide

Figure 3.20: Group Properties Dialog Box

8. Click Group Users tab as shown in figure 3.21.

Session 3

Installing and Configuring MySQL (Lab)

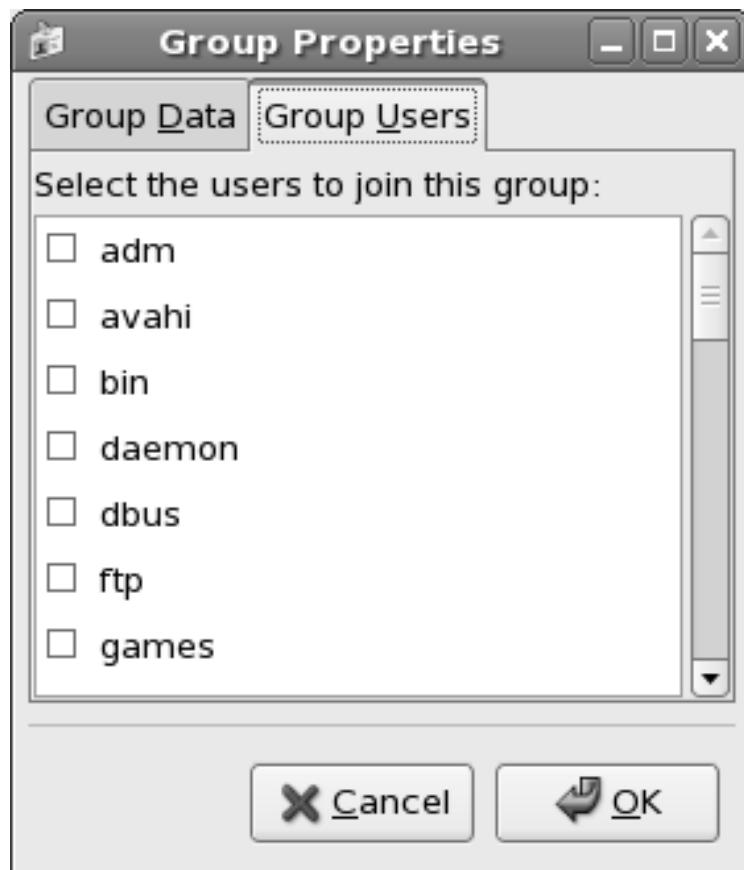


Figure 3.21: Group Users Tab

9. Select bin.
10. Click OK.
11. Click File → Add User to add a new user to the mysql group. The Create New User dialog box is displayed.
12. Enter the User Name and the Password as shown in figure 3.22.

Session 3

Installing and Configuring MySQL (Lab)

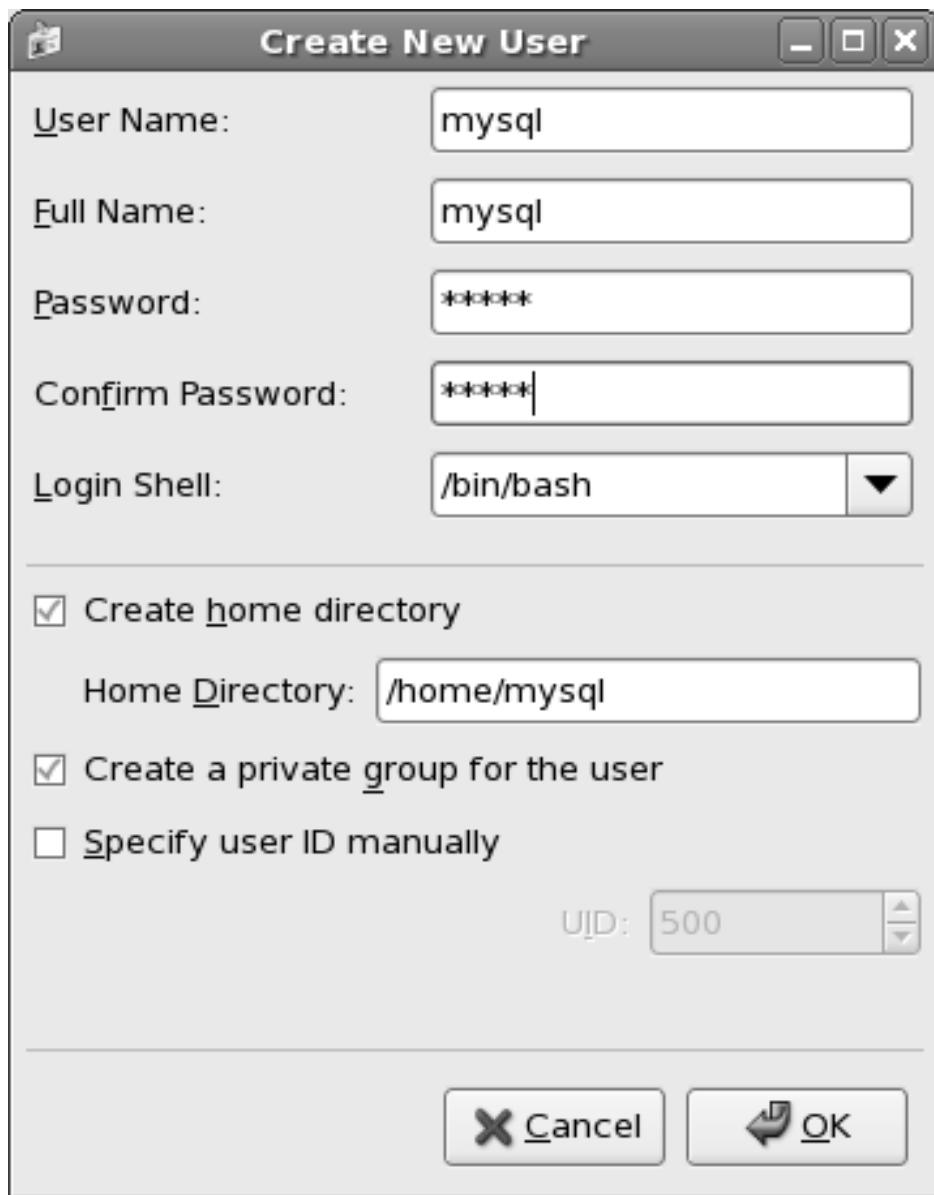


Figure 3.22: Create New User Dialog Box

13. Click OK.

You will install MySQL on Linux using RPM packages. RPM packages are the binary distribution format of MySQL. The following RPM packages are required to install MySQL:

```
MySQL-client-community-5.1.56-1.rhel5.i386.rpm;
```

```
MySQL-server-community-5.1.56-1.rhel5.i386.rpm;
```

Session 3

Installing and Configuring MySQL (Lab)

Note: You can download these RPM packages from the MySQL Website: <http://www.mysql.com/downloads/mysql/5.1.html#downloads>. You must select Red Hat & Oracle Enterprise Linux from the list of operating systems to view the RPM packages to download.

To install MySQL server 5.1.56 on Red Hat Enterprise Linux 5:

1. Double-click `MySQL-server-community-5.1.56-1.rhel5.i386.rpm` to install MySQL Server package. The Installing packages dialog box is displayed as shown in figure 3.23.

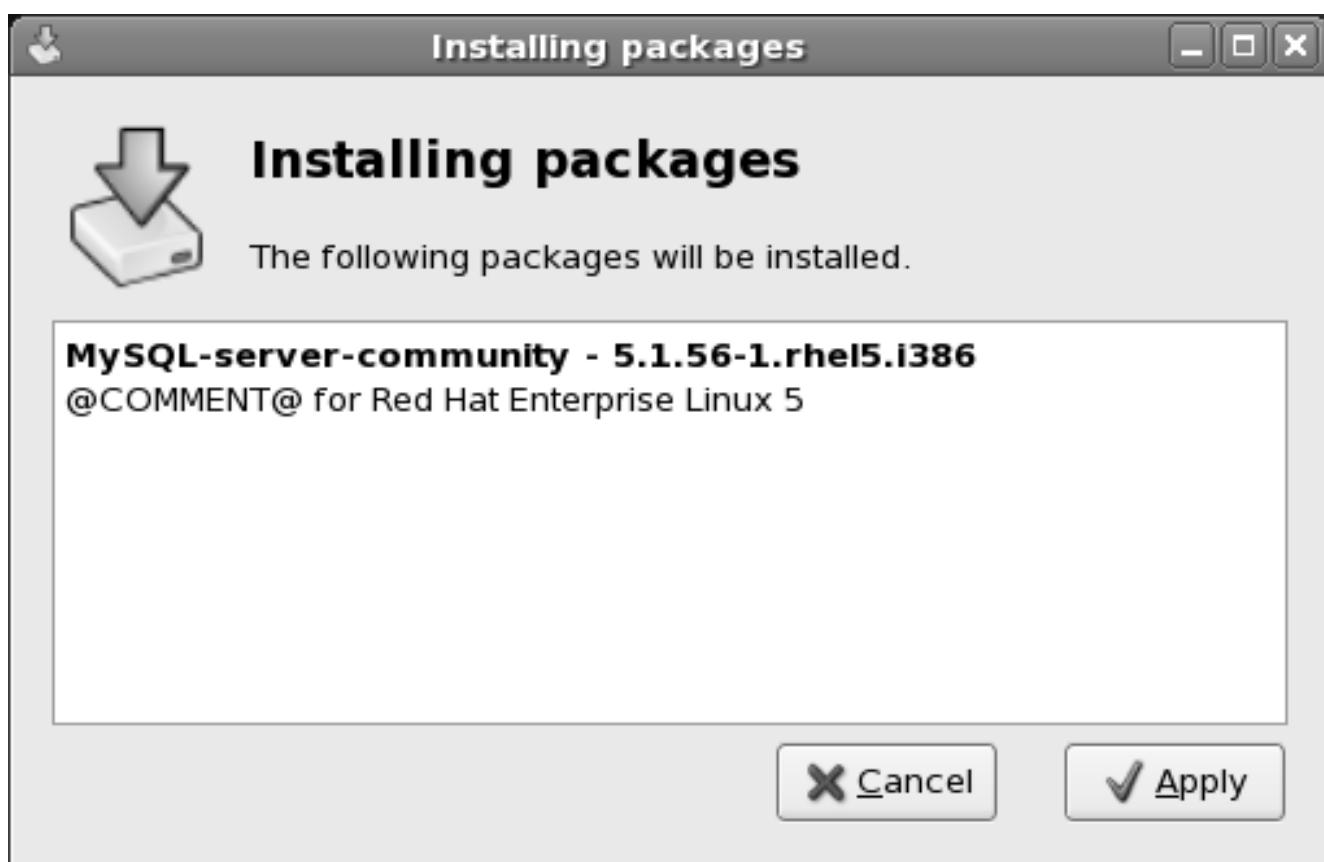
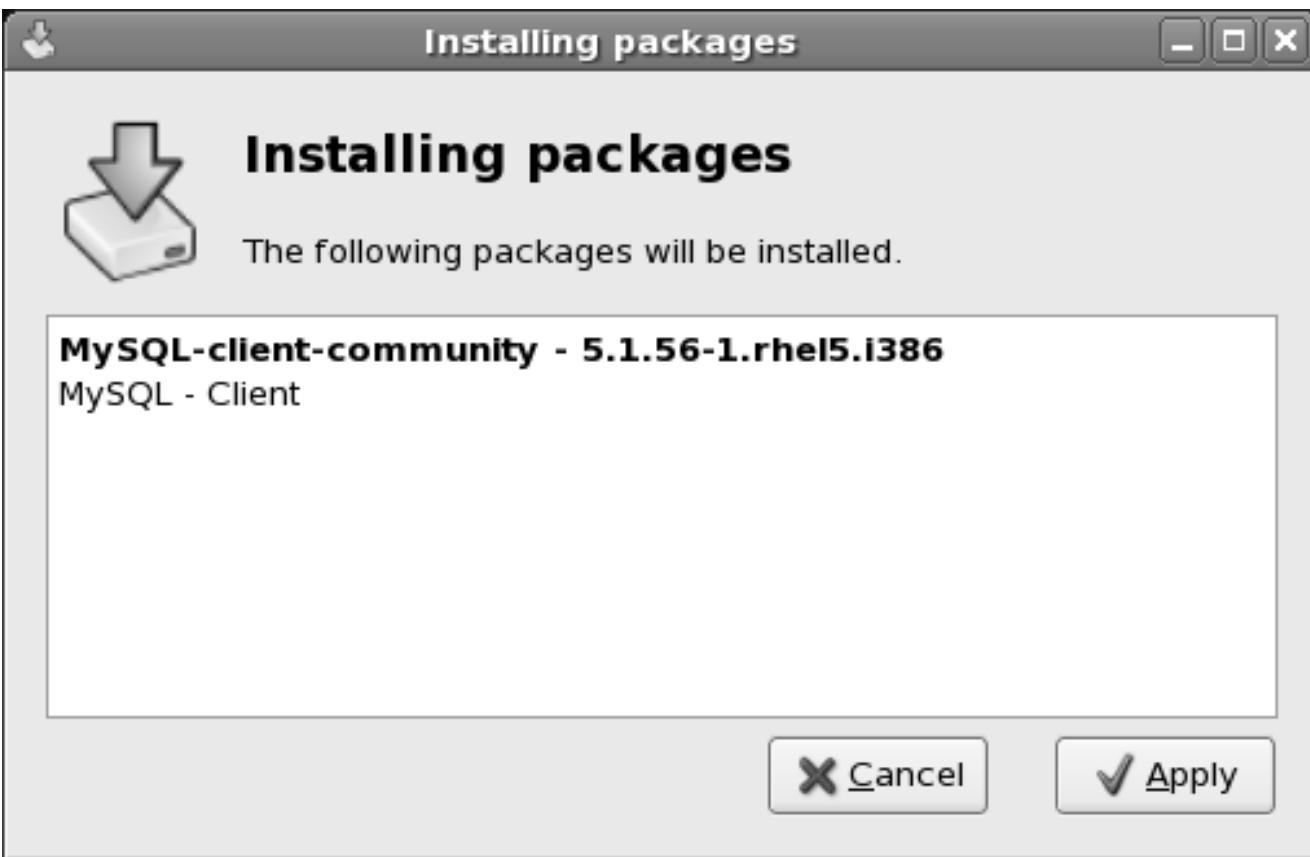


Figure 3.23: Installing Server Packages Dialog Box

2. Click Apply. The installation completes.
3. Double-click `MySQL-client-community-5.1.56-1.rhel5.i386.rpm` to install MySQL client package,. The Installing packages dialog box is displayed as shown in figure 3.24.

Session 3

Installing and Configuring MySQL (Lab)



Lab Guide

Figure 3.24: Installing Client Packages Dialog Box

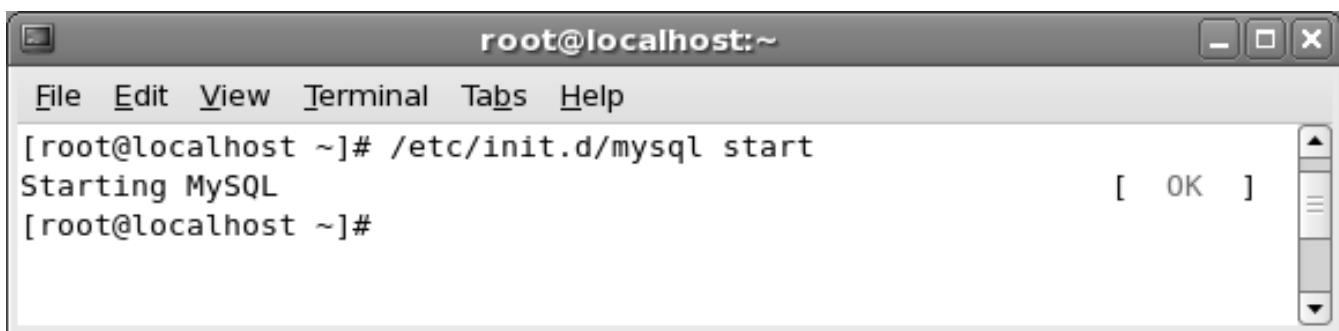
4. Click Apply. The installation completes.
5. To configure the MySQL service to start automatically with the operating system, open the Linux terminal and enter the following command at the command prompt:

```
/etc/init.d/mysql start;
```

Figure 3.25 displays the output of the command.

Session 3

Installing and Configuring MySQL (Lab)



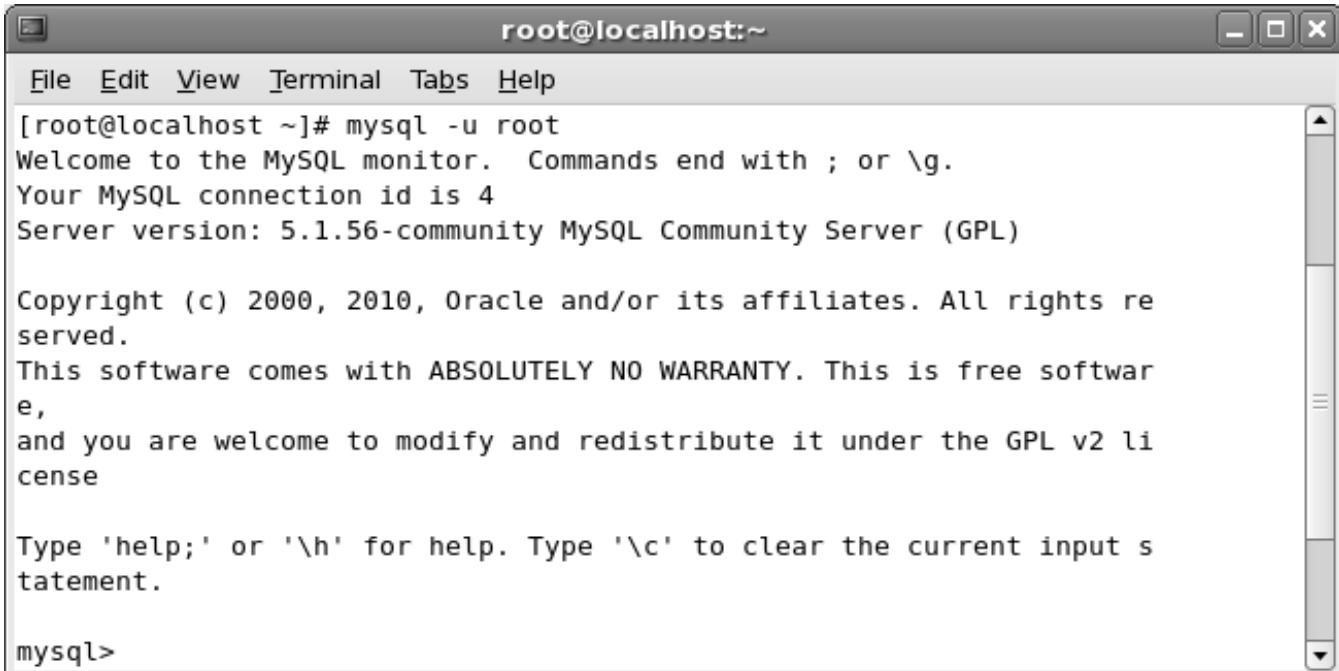
A terminal window titled "root@localhost:~". The window contains the following text:
File Edit View Terminal Tabs Help
[root@localhost ~]# /etc/init.d/mysql start
Starting MySQL
[root@localhost ~]# [OK]
The window has standard Linux-style window controls at the top right.

Figure 3.25: Initializing MySQL

6. To start the MySQL client, enter the following command at the command prompt:

```
mysql -u root;
```

Figure 3.26 displays the output of the command.



A terminal window titled "root@localhost:~". The window contains the following text:
File Edit View Terminal Tabs Help
[root@localhost ~]# mysql -u root
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.1.56-community MySQL Community Server (GPL)

Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.
This software comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to modify and redistribute it under the GPL v2 license

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

mysql>

Figure 3.26: Starting MySQL Client

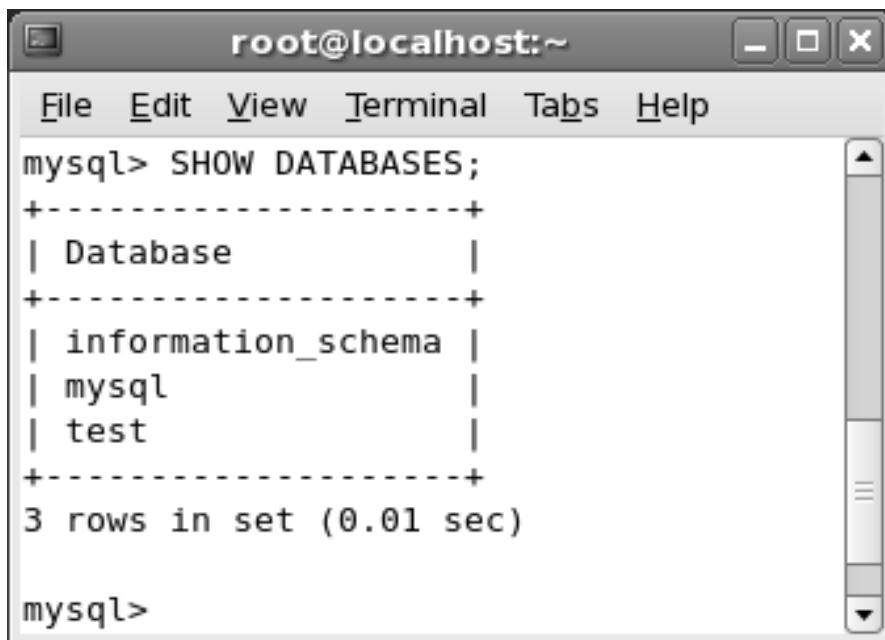
Session 3

Installing and Configuring MySQL (Lab)

7. To view the list of default databases, enter the following command at the command prompt:

```
SHOW DATABASES;
```

Figure 3.27 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area displays the following text:

```
mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| test |
+-----+
3 rows in set (0.01 sec)

mysql>
```

Figure 3.27: SHOW DATABASES Command

Configure MySQL Installation Using Configuration Scripts

The source code must be downloaded and extracted to a folder before making changes to the configuration files. You will be required to download the source code for MySQL to configure the installation. The source code is available in the Downloads section of MySQL Website under Source Code category. The file to be downloaded is:

- MySQL-community-5.1.56-1.rhel5.src.rpm

1. Select Applications → Accessories → Archive Manager. The Archive Manager dialog box is displayed as shown in figure 3.28.

Session 3

Installing and Configuring MySQL (Lab)

Lab Guide

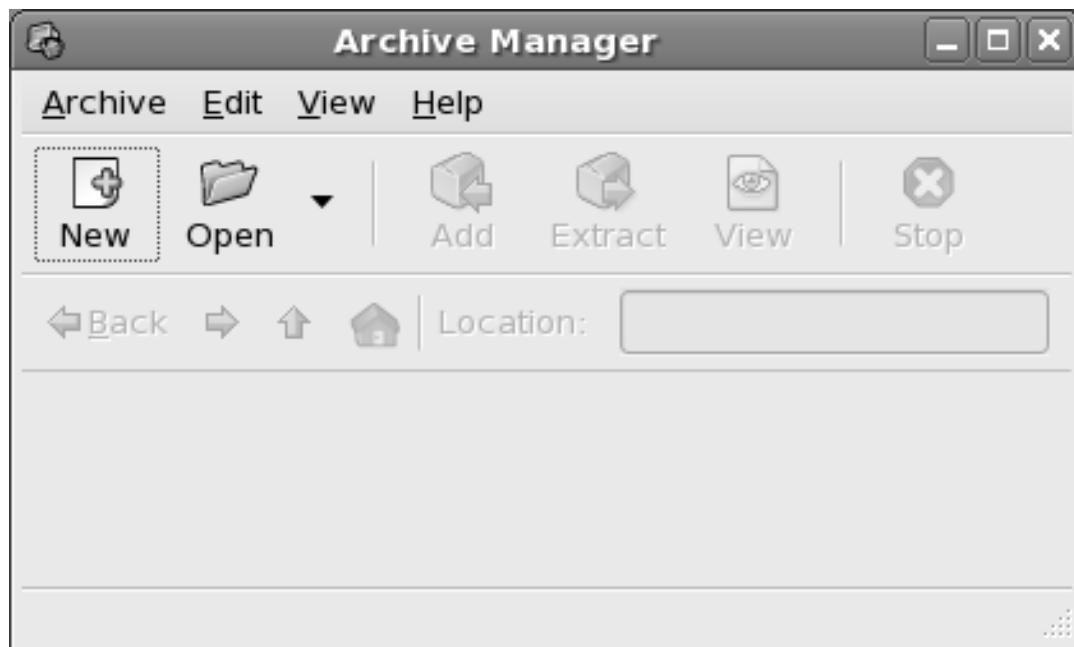


Figure 3.28: Archive Manager Dialog Box

2. Select Archive → Open. The Open dialog box is displayed as shown in figure 3.29.

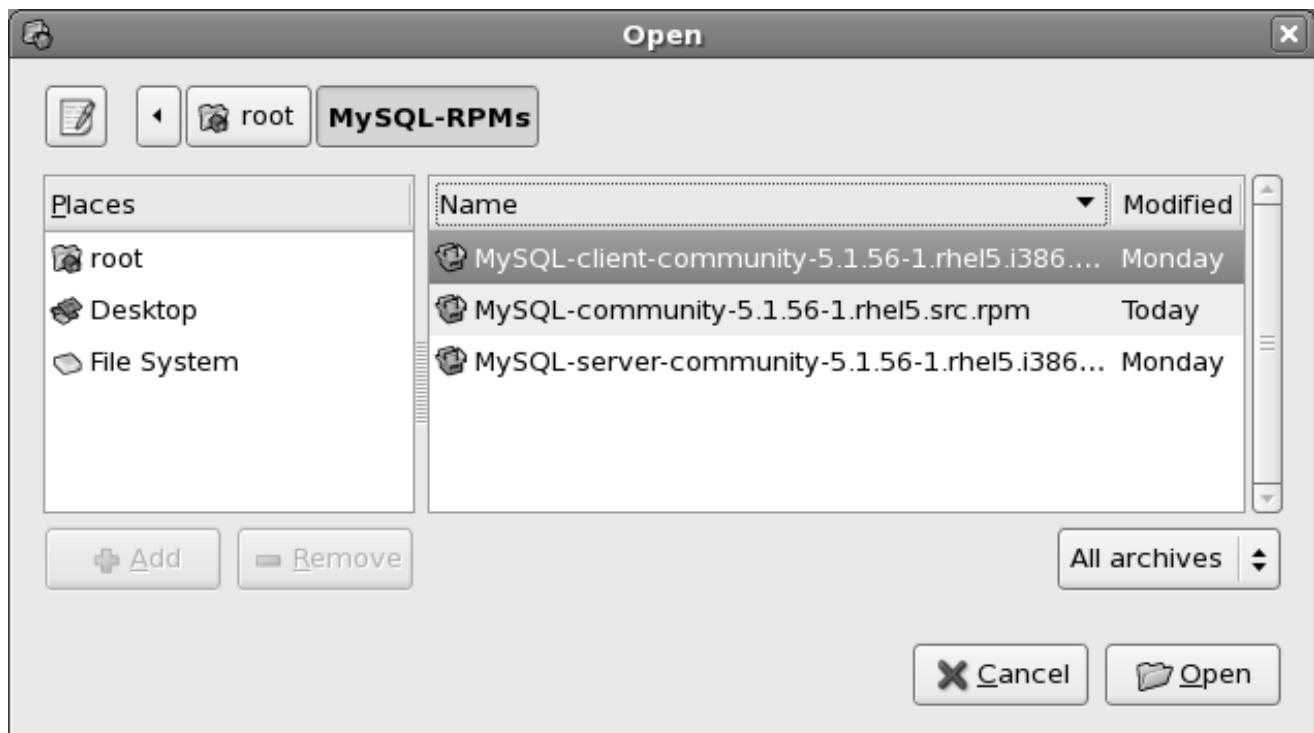
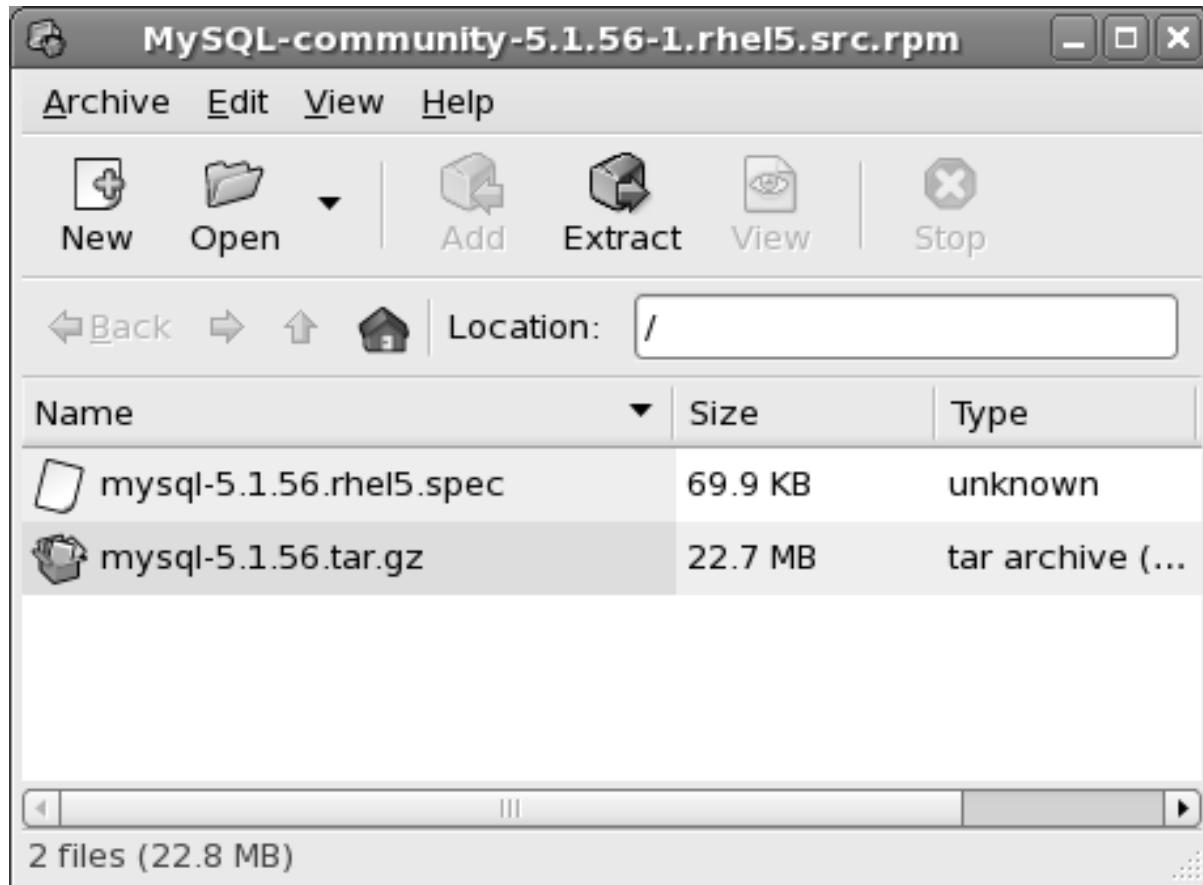


Figure 3.29: Open Dialog Box

Session 3

Installing and Configuring MySQL (Lab)

3. Browse to the folder that contains MySQL-community-5.1.56-1.rhel5.src.rpm.
4. Select MySQL-community-5.1.56-1.rhel5.src.rpm.
5. Click Open as shown in figure 3.30 to display the contents of the RPM package.



Lab Guide

Figure 3.30: RPM Package Contents

6. Minimize the Archive Manager window and create a folder named mysql under the root directory.
7. Select Archive → Extract. The Extract dialog box is displayed as shown in figure 3.31.

Session 3

Installing and Configuring MySQL (Lab)

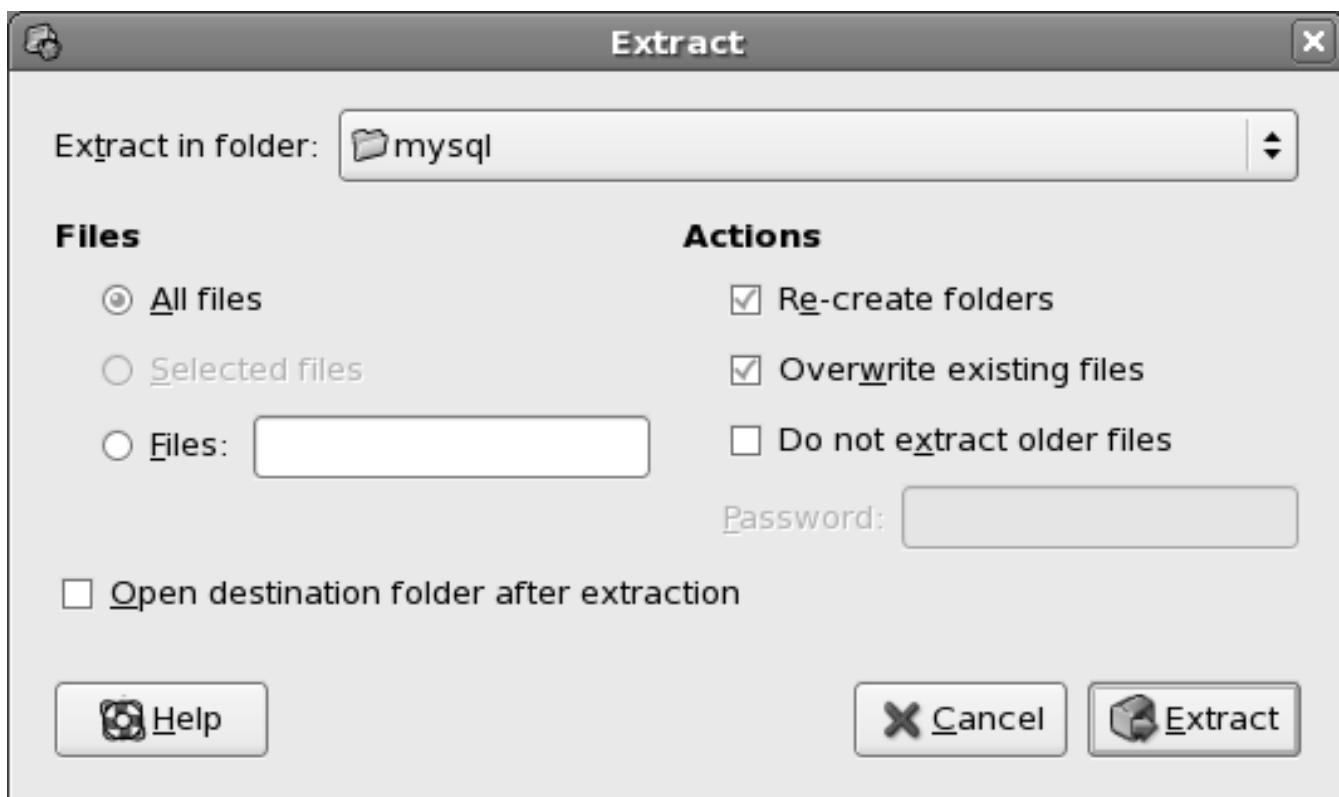


Figure 3.31: Extract Dialog Box

8. Browse to the mysql folder in the 'Extract in folder' drop-down box.
9. Click Extract. The application extracts the package contents to the specified location.
10. Right click the `mysql-5.1.56.tar.gz` file and select Extract Here. The file contents are extracted as shown in figure 3.32.

Session 3

Installing and Configuring MySQL (Lab)

Lab Guide

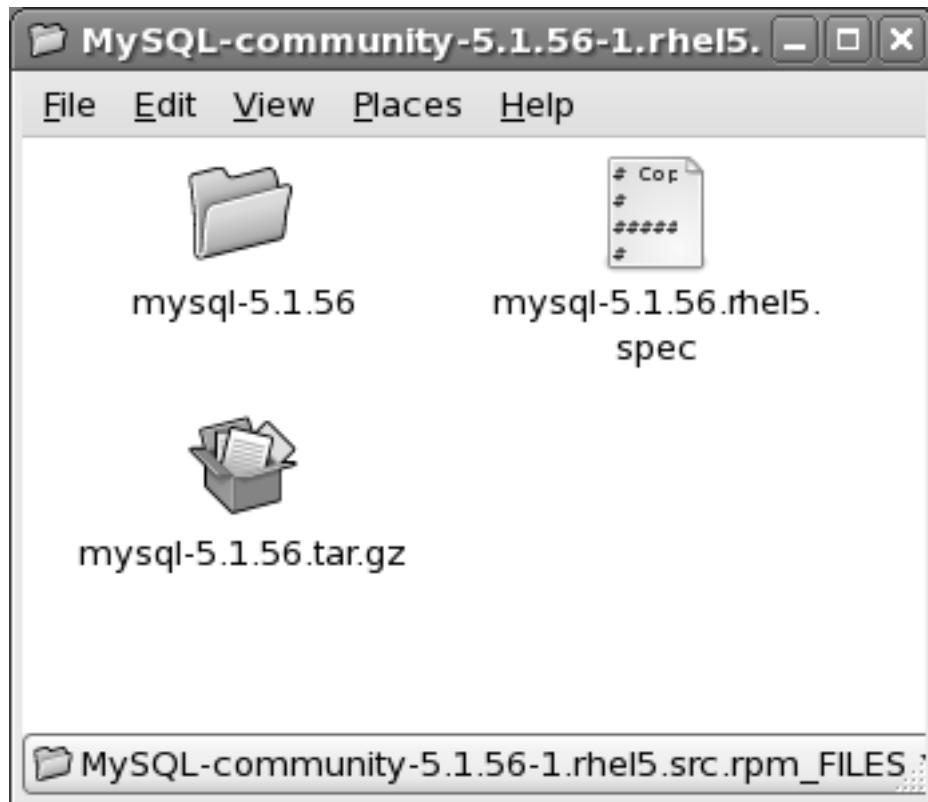


Figure 3.32: Extracted Package Contents

11. Right click the `mysql-5.1.56` folder to display the context menu and select Open In Terminal. The Terminal window is displayed as shown in figure 3.33.

Session 3

Installing and Configuring MySQL (Lab)

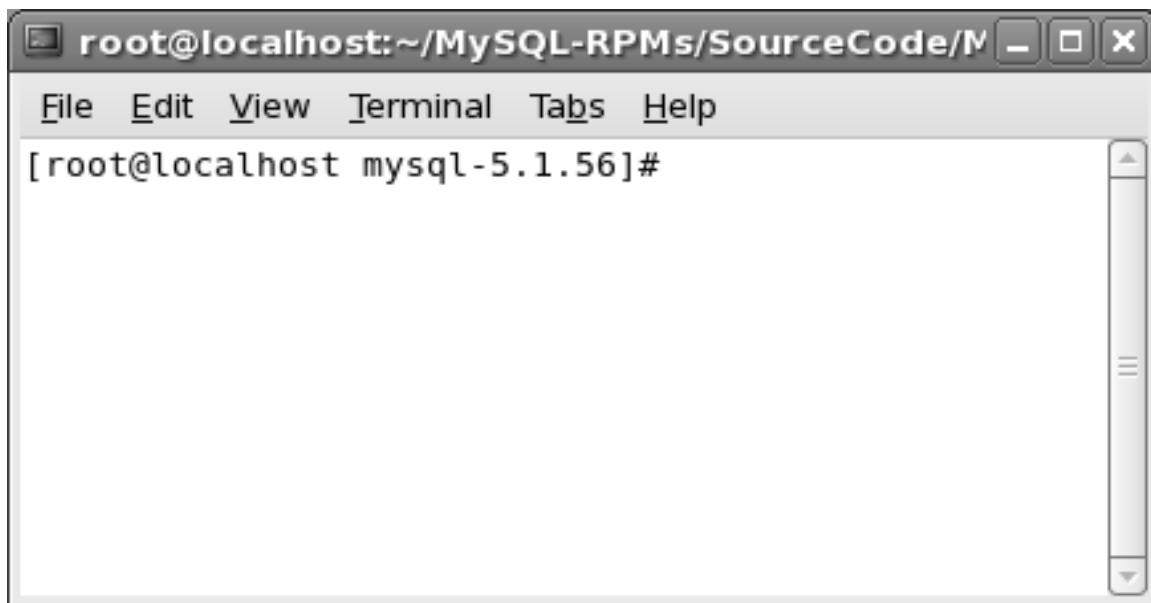


Figure 3.33: Terminal Window

12. To create a symbolic link, enter the following command at the command prompt:

```
ln -s MySQL-community-5.1.56.rhel5.src root;
```

Figure 3.34 displays the output of the command.

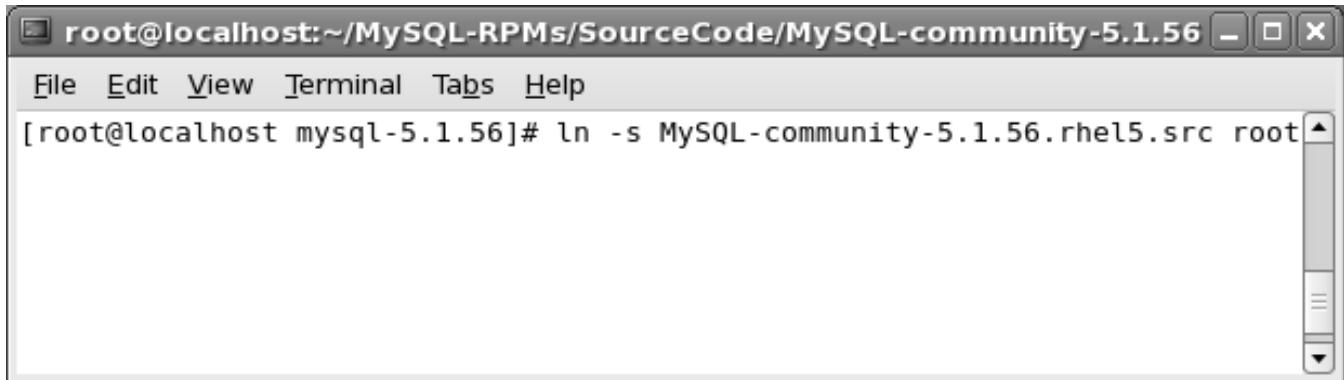


Figure 3.34: Creating a Symbolic Link

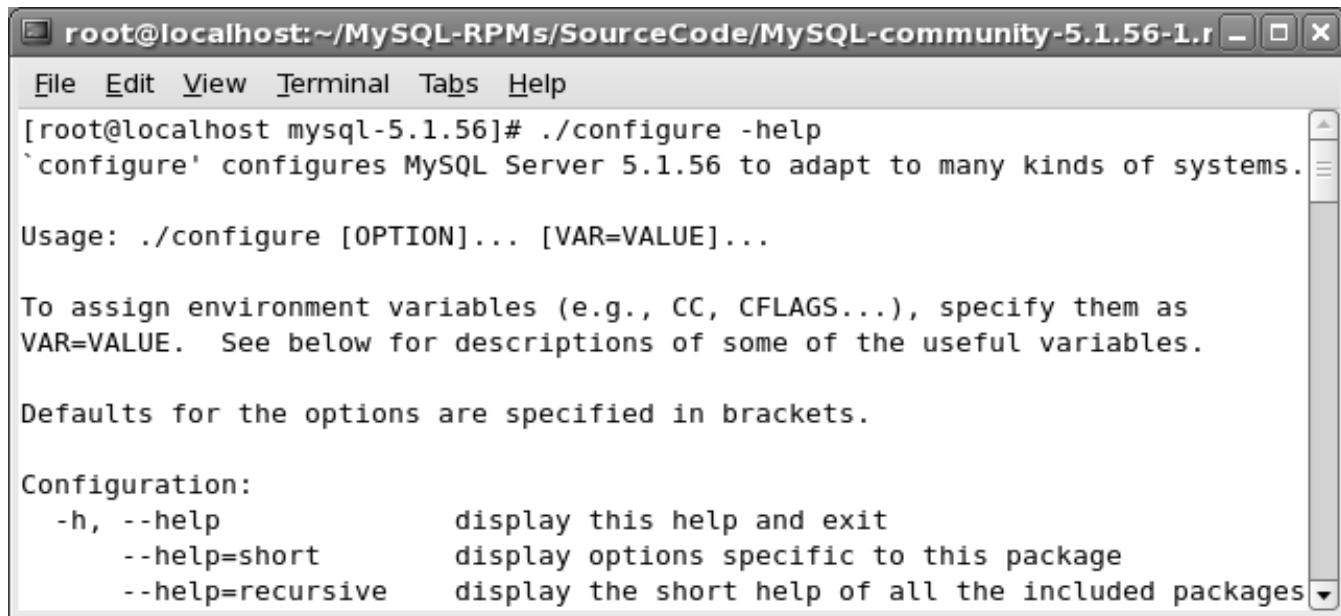
Session 3

Installing and Configuring MySQL (Lab)

13. To view the options for configuration, enter the following command at the command prompt:

```
./configure --help;
```

Figure 3.35 displays the output of the command.



The screenshot shows a terminal window titled 'root@localhost:~/MySQL-RPMs/SourceCode/MySQL-community-5.1.56-1.r'. The window contains the following text:

```
[root@localhost mysql-5.1.56]# ./configure --help
`configure' configures MySQL Server 5.1.56 to adapt to many kinds of systems.

Usage: ./configure [OPTION]... [VAR=VALUE]...

To assign environment variables (e.g., CC, CFLAGS...), specify them as
VAR=VALUE. See below for descriptions of some of the useful variables.

Defaults for the options are specified in brackets.

Configuration:
  -h, --help                  display this help and exit
  --help=short                 display options specific to this package
  --help=recursive              display the short help of all the included packages
```

Figure 3.35: Help Options of Configure

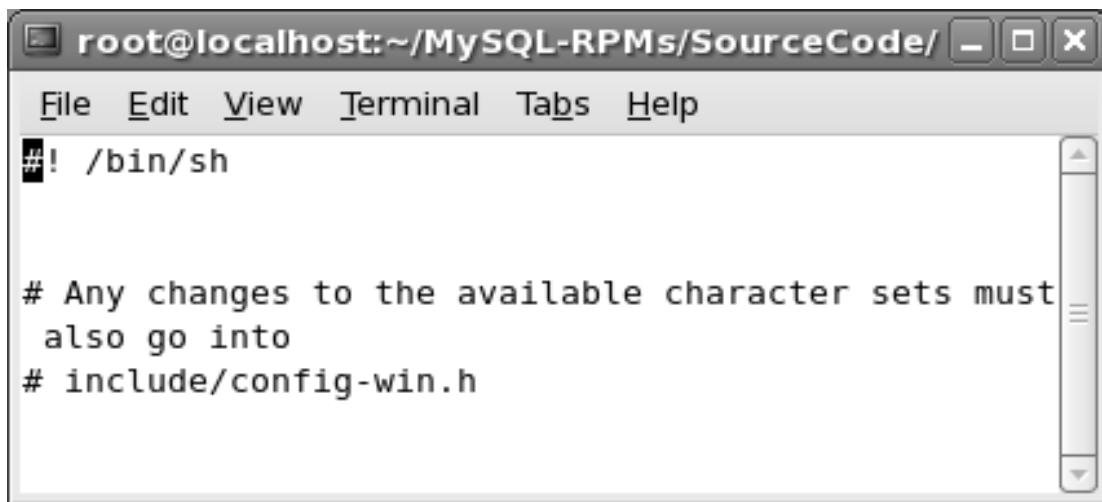
14. To open the configuration file, enter the following command at the command prompt:

```
vi configure;
```

Figure 3.36 displays the output of the command.

Session 3

Installing and Configuring MySQL (Lab)



The screenshot shows a terminal window titled "root@localhost:~/MySQL-RPMs/SourceCode/". The window contains a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". Below the menu is a code editor window showing the following text:

```
#! /bin/sh

# Any changes to the available character sets must
# also go into
# include/config-win.h
```

Figure 3.36: Configure File in vi Editor

15. To open the my.cnf file, browse to the /mysql/mysql-5.1.56/include folder.
16. Right click the include folder and select Open In Terminal. The Terminal window is displayed.
17. Enter the following command at the command prompt:

```
vi default_my.cnf;
```

Figure 3.37 displays the output of the command.

Session 3

Installing and Configuring MySQL (Lab)

Lab Guide



The screenshot shows a terminal window titled "root@localhost:~/MySQL-RPMs/SourceCode/MySQL-community-5.1.56-1.rhel". The window displays the contents of the "default_my.cnf" file. The file includes sections for mysqld, mysqlbinlog, mysql_fix_privilege_tables, and ENV, with various configuration parameters like log-bin, disable-force-if-open, socket, port, user, password, and command-line options.

```
# Use default setting for mysqld processes
!include default_mysqld.cnf

[mysqld.1]

# Run the master.sh script before starting this process
#!run-master-sh

log-bin=          master-bin

[mysqlbinlog]
disable-force-if-open

# mysql_fix_privilege_tables.sh does not read from [client] so it
# need its own section
[mysql_fix_privilege_tables]
socket=          @client.socket
port=            @client.port
user=            @client.user
password=        @client.password

[ENV]
"default_my.cnf" 25L, 597C
```

Figure 3.37: default_my.cnf File

Session 3

Installing and Configuring MySQL (Lab)

Lab Guide



Do It Yourself

1. Configure the installation of MySQL using the `--without-server` option.
2. Configure the installation of MySQL using the `--prefix` option.
3. Configure the installation of MySQL using the `--with-client-Idflags` option.
4. Configure the installation of MySQL using the `DDONT_USE_DEFAULT_FIELDS` option.
5. Configure the installation of MySQL using the `--with-charset` option.
6. Configure the installation of MySQL using the `--with-debug` option.

Objectives

At the end of this session, the student will be able to:

- *Explain database.*
- *Explain the data types.*
- *Identify the different types of data.*
- *Explain the creation of a table.*
- *Explain Normalization.*
- *Identify the different forms of normalization.*
- *Explain Indexes and Referential Integrity.*

4.1 Introduction

A database is a collection of organized and related data. Databases store information in an organized manner in the form of a table. The table structure consists of rows and columns which holds data. A database management system can be used to handle the data in a database.

A database can contain duplicate records in tables if the database is not normalized and tables are not indexed. This can increase redundancy of the data in the database. You can normalize the database and index the tables to eliminate redundancy and enhance the speed of data search and retrieval operations.

In this session, you will learn how to create a database. You will identify the different types of data and how to create a table using different data types. In addition, you will learn how to normalize a database, and implement indexes and referential integrity in a database.

4.2 Basic Concepts in MySQL

Before you can use MySQL, there are certain rules that must be adhered to by the developer. This will help you to understand the naming conventions in MySQL. It will also help you to understand, how commands are used in MySQL.

4.2.1 Naming Conventions

While naming a database and its components, such as tables, columns, and so forth you will have to follow certain rules. For example, you will have to ensure that the name does not exceed the specified length or use illegal characters.

Note: MySQL allows reserved words to be used as names. However, these words have to be enclosed in quotes when used as object names.

The following naming conventions are recommended for MySQL:

- **Legal Characters** - You can use any alphabet (a-z or A-Z) or digits (0-9) in the name. You can also use characters, such as '_' or '\$'. A name can start with a digit but cannot contain only digits because you cannot distinguish it from a number. You cannot use '.' within the name because it acts as the separator. For example, db_name.table_name where db_name is the database name and table_name is the name of the table in the database. You also cannot use the separator characters such as '/' or '\'.
- **Length of names** - MySQL restricts the length of the name of the databases, indexes, or columns. It can have a maximum length of 64 characters. However, MySQL allows alias names to be 256 characters in length and compound statement labels are restricted to 16 characters. Examples of compound statement labels include join and conditional operators, such as INNER JOIN, OUTER JOIN, LEFT INNER JOIN, RIGHT OUTER JOIN, FROM, and WHERE.

Note: When you create an alias for a column name in a CREATE VIEW statement, the length must not exceed 64 characters.

4.2.2 MySQL Commands

You can enter SQL commands on a single line or split them across multiple lines. All MySQL commands should be terminated by a semicolon. MySQL waits for a semicolon (;) to execute a command. Alternatively, you can use the 'g' command.

4.2.3 Case Sensitive Conventions

In MySQL, case sensitivity varies with the object in question.

Note: Read more about using reserved words as identifiers at <http://dev.mysql.com/doc/refman/5.1/en/reserved-words.html>

➤ SQL Statements and Keywords

The SQL statements and keywords are not case sensitive. A user can write SQL statements in up-

per or lower case. Consider the statements as shown:

```
SELECT VERSION ( );
select version( );
```

Both the statements are equivalent. Both the commands will return the version of MySQL being used. The following combinations are also valid:

```
Select VERSION ( );
SeLecT version ( );
```

➤ Database and Table names

Case sensitivity of databases, tables, and trigger names depend on the operating system. If the underlying operating system is case sensitive, then the names of databases, tables, and triggers. For example, Microsoft Windows is not case sensitive and so databases, tables, and triggers names in MySQL running on Microsoft Windows are not case-sensitive. However, Linux is case sensitive and so databases, tables, and triggers names under a Linux installation of MySQL are case-sensitive. This is because these objects are stored as files on the file system.

Note: You cannot use different cases while referring to the same object in the same SQL statement. For example, if you refer to a database as 'MyDB', then you cannot refer to it as 'mydb' or 'MYDB' elsewhere in the same statement.

➤ Columns, Column Aliases, Index, Stored Routines, and Events Names

Names of columns, column aliases, index, stored routines, and events are not case sensitive. You can use one of the following SQL statements to display the column 'FNAME' from a SAMPLE table:

```
SELECT FNAME from SAMPLE ( );
SELECT fname from SAMPLE ( );
```

➤ Log Files

Log file names are case sensitive.

4.3 Creating a MySQL database

MySQL is a Relational Database Management System. You can use MySQL to manage relational databases. A relational database consists of different tables in which data is stored such that there is minimal redundancy. A relational database also contains relationships or links to related tables.

Session 4

Using MySQL

You can create a database if you have proper privileges.

The syntax for creating a database is:

```
CREATE DATABASE [ IF NOT EXISTS ] <dbname>;
```

where,

CREATE DATABASE – adds a new database to the MySQL instance

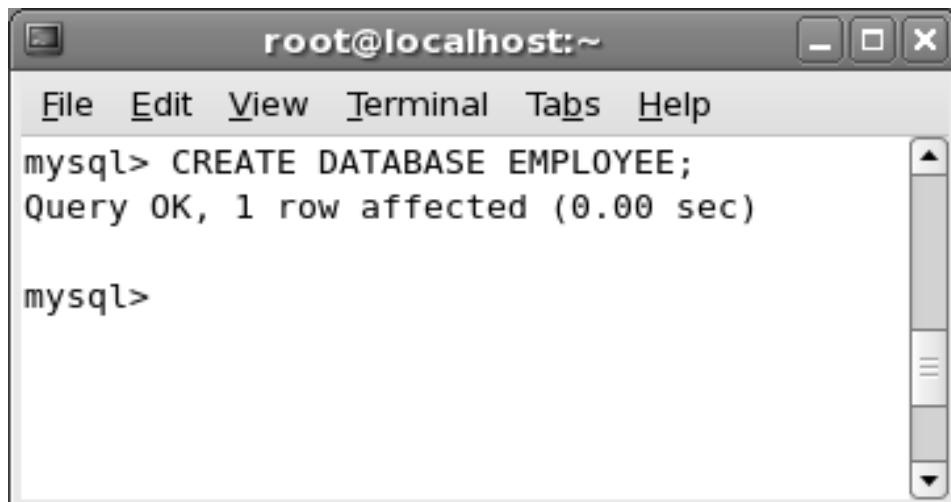
IF NOT EXISTS – checks for the existence of databases with the same name before adding the new database. MySQL does not allow you to create databases with same names

dbname – specifies a name for the new database

For example, to create a database named EMPLOYEE for storing employee information, such as personal details, qualifications, salary details, and so on, enter the following command at the command prompt:

```
CREATE DATABASE EMPLOYEE;
```

Figure 4.1 displays the output of the command.



A screenshot of a terminal window titled "root@localhost:~". The window has a standard Linux-style interface with a title bar, menu bar, and scroll bars. The terminal session shows the command "mysql> CREATE DATABASE EMPLOYEE;" being entered, followed by the output "Query OK, 1 row affected (0.00 sec)". The prompt "mysql>" appears again at the bottom of the window.

Figure 4.1: Creating a Database

Figure 4.1 displays the number of rows affected, and the time taken to execute the command.

4.4 Working with MySQL Tables

As compared to a DBMS, RDBMS stores data in different tables to ensure consistency and faster search and retrieval operations. A table stores data in rows and columns. A field in a table is called a column and row is called a record. A record is also defined as collection of fields. The columns in a table contain different types of data.

4.4.1 Describing Data Types

Data types define the type of the data that will be stored in a column. Data types also define the size and the type of data that can be stored in the column.

MySQL supports different data types to store values. Following are the different categories under which the data can be categorized:

- Numeric data types
- Date and time data types
- String data types
- Complex data types

Numeric Data Type

A numeric data type stores data in number form. For example, a column called Product_Id containing information on the product number can be defined as numeric. There are several numeric data types available in MySQL.

Note: Numbers can be classified as signed and unsigned. A signed number is preceded with negative or positive sign. An unsigned number does not have any sign and is assumed to be positive.

Table 4.1 lists the numeric data types in MySQL.

Numeric Data Type	Storage In Bytes	Description	Synonyms
BIGINT (length)	8	Stores unsigned numbers in the range of 0 to 18,446,744,073,709,551,615, and signed numbers in the range of -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	DEC NUMERIC

Session 4

Using MySQL

Concepts

Numeric Data Type	Storage In Bytes	Description	Synonyms
DECIMAL(length, decimal)	Precision +2	Stores floating point numbers and cannot be unsigned	
DOUBLE	8	Stores double precision floating point numbers. Range for a negative number is between -1.7976931348623157E+308 and -2.2250738585072014E-308,0. Range for a positive number is between 2.2250738585072014E-308 and 1.7976931348623157E+308	DOUBLE PRECISION REAL
FLOAT(length, decimal)	4	Stores single precision floating point numbers. Range for a negative number is between -3.402823466E+38 and -1.175494351E-38,0. Range for a positive number is between 1.175494351E-38 and 3.402823466E+38	
INT(length)	4	Stores integer numbers. Range for a signed number is from -2,147,483,648 to 2,147,483,647. Range for an unsigned number is from 0 to 4,294,967,295	INTEGER
MEDIUMINT(length)	3	Stores integer numbers. Range for a signed number is from -8,388,608 to 8,388,607. Range for an unsigned number is from 0 to 16,777,215	
SMALLINT(length)	2	Stores integer numbers. Range for a signed number is from -32,768 to 32,767. Range for an unsigned number is from 0 to 65,535	
TINYINT	1	Stores integer numbers. Range for a signed number is from -128 to 127. Range for an unsigned number is from 0 to 255	BOOLEAN

Table 4.1: Numeric Data Types in MySQL

String Data Type

A string represents a sequence of characters. A string data type is enclosed in single quotes or double quotes. It enables you to enter alphabets from a-z, A-Z, and numbers from 0-9. For example to store employee names in a column called Ename, you can use the string data type. The number of characters specified defines the length of a character string.

Session 4

Using MySQL

Table 4.2 lists the string data types supported by MySQL.

Data Type	Storage In Bytes	Description
CHAR (Length)	Length	Stores character data set and can have a maximum length of 255 characters.
BINARY (Length)	Length	Stores binary byte string upto 255 characters.
VARCHAR ()	Length+1	Stores variable length string values and can have a length from 0 to 65,535. Uses one byte to store 255 characters. Uses additional two bytes to if values require more than 255 characters.
VARBINARY	Length+2	Stores binary strings and has a maximum length same as that of VARCHAR.
BLOB	Length+2	Stores large amount of variable binary data such as images and can have a maximum length of 65,535. Uses an additional byte of storage if values exceed 65,536.
TINYBLOB	Length+1	Stores upto 255 bytes.
MEDIUMBLOB	Length+3	Stores text in size upto 16 MB. Uses additional three bytes of storage if values exceed the specified length.
LONGBLOB	Length+4	Stores text values whose size exceeds 4 GB. Uses additional four bytes of storage if values exceed the specified length.
TEXT	Length+2	Stores characters and can be upto 65,535 characters in length. Uses additional two bytes of storage if values exceed the specified length.
TINYTEXT	Length+1	Stores short text values and can have a length upto 255 characters.
MEDIUMTEXT	Length+3	Stores medium-sized text whose size is upto 16 MB. Uses additional three bytes of storage if values exceed the specified length.
LONGTEXT	Length+4	Stores large text values whose size exceeds 4 GB. Uses additional four bytes of storage if values exceed the specified length.

Table 4.2: String Data Types in MySQL

Date Data Type

A date data type stores date and time information. For instance, you can store the date of birth of employees in date type column. The Date data type column stores date as string value in the default format, 'YYYY-MM-DD'.

Table 4.3 lists the date data types in MySQL.

Session 4

Using MySQL

Data Type	Format	Storage In Bytes	Description
DATE	YYYY-MM-DD	3	Stores a date type of data in the range of January 1,1000 to December 31,9999
DATETIME	YYYY-MM-DD hh:mm:ss	8	Stores date and time. The range for it is '1000-01-01 00:00:00' to '9999-12-31 23:59:59'
TIME	hh:mm:ss	3	Stores time
TIMESTAMP	YYYY-MM-DD hh:mm:ss	4	Stores timestamp
YEAR	YYYY	1	Stores a year in the range of 1901 to 2155

Table 4.3: Date Data Types in MySQL

Complex Data Type

Complex data type is a special data type. It defines the list of possible values that can be inserted in a column. For example, you can assign the `SET` or `ENUM` data type to a column where a distinct string value will be chosen from a list of predefined values.

Table 4.4 lists the complex data types in MySQL.

Complex Data Type	Storage In Bytes	Description
ENUM	1,2	Stores one value out of a possible 65535 number of options in a predefined list of possible values. This data type can contain only one distinct string value from the predefined list of values. The values are represented internally as integers. Eg: ENUM('abc','def','ghi')
SET	1,2, 3, 4 or 8	Stores a list of values from a predefined set of values. Maximum of 64 values can be stored in a SET data type column. The SET data type can contain zero or more values that can be specified in the list.

Table 4.4: Complex Data Types in MySQL

The `SET` data type is similar to `ENUM`. While an `ENUM` data type restricts you to selecting one of many predefined values, `SET` data type allows you to choose more than one of the predefined values.

Consider the following example:

```
CREATE TABLE MY_CITY
(CITY_ID INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
CITY_NAME ENUM('CHICAGO','DALLAS','COLUMBUS'));
```

Session 4

Using MySQL

The code will create a table `MY_CITY` with two columns, `CITY_ID` and `CITY_NAME`. The data type for `CITY_NAME` has been defined as `ENUM`. When you add records to the table using the `INSERT` command, this data type will allow you to use one of the city names specified in the `ENUM` list as the column value when you add rows to the table.

For example, you can insert `CHICAGO`, `DALLAS`, or `COLUMBUS` as values for the column `CITY_NAME`, but cannot insert `NEW YORK` as a value because it is not a part of the `ENUM` list. When you use an invalid value for the column `CITY_NAME` in the `INSERT` query, the query is ignored.

Consider the same code with `SET` as the data type for the `CITY_NAME` field.

```
CREATE TABLE MY_CITY
(CITY_ID INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
CITY_NAME SET('CHICAGO','DALLAS','COLUMBUS'));
```

When you assign `SET` data type to the `CITY_NAME` field, you can use one or more city names from the `SET` list as a column value when you add rows to the table.

For example, you can insert `('CHICAGO','DALLAS')`, `('CHICAGO','COLUMBUS')` or `('CHICAGO','COLUMBUS','DALLAS')` as values for the column `CITY_NAME`. If you specify an invalid value, such as `NEW YORK`, in the `INSERT` query for the `CITY_NAME` column, MySQL ignores the value and displays a warning. The command to insert values would be:

```
INSERT INTO MY_CITY (CITY_NAME) VALUES ('CHICAGO','DALLAS');
```

The `SET` data type uses storage space in the form of bytes to store the number of elements defined in the data type. Table 4.5 lists the storage structure of `SET` data type depending upon the number of elements.

Number Of Elements	Storage In Bytes
1-8	1
9-16	2
17-24	3
25-32	4
33-64	8

Table 4.5: Storage Structure of `SET` Data Type

4.4.2 Creating Tables

A relational database contains several related tables. The table name can be upto 64 characters in length. The maximum length of a field name is 64 characters. Also, you can specify the table type or the

Session 4

Using MySQL

storage engine while creating the table.

MySQL supports two types of tables:

- **Transaction-safe tables:** enables you to execute transactions or data manipulation commands without losing data. You can undo the changes made to transaction-safe tables.
- **Non-transaction safe tables:** also allows you to execute transaction or data manipulation commands. However, you cannot undo the changes made to non-transaction safe tables because the changes are permanent.

Following are the features of transaction-safe tables:

- Provides safety from data loss and allows MySQL to recover data, from a backup and the transaction log, in case of a crash or hardware failure
- Enables to combine and execute statements using the `COMMIT` command
- Enables to undo the changes made to the data using the `ROLLBACK` command
- Enables concurrency while simultaneously reading data from tables
- Examples of transaction-safe tables are InnoDB

Following are the features of non-transaction safe tables:

- Faster than the transaction-safe tables as no transaction overhead is required
- Require less disk space
- Require less memory for updates

Storage Engines

MySQL supports storage engines for different table types. Storage engines handle both transaction-safe and non-transaction safe tables. Table 4.6 lists the storage engines supported in MySQL:

Storage Engine	Feature
MyISAM	Is the default storage engine to handle tables
InnoDB	Provides commit, rollback, and crash recovery characteristics to secure data
MERGE	Allows developers to create tables with identical column and index information

Session 4

Using MySQL

Concepts

Storage Engine	Feature
MEMORY (HEAP)	Allows developers to create tables that store contents in the memory. MEMORY tables can have upto 64 indexes per table, 16 columns per index, and a maximum key length of 3072 bytes
EXAMPLE	Allows developers to create dummy tables where no data will be stored
FEDERATED	Provides access to data from tables in remote databases without copying data to the local server. This engine does not create a replica or a copy of the database or tables to the local server while executing a query on the remote server
ARCHIVE	Allows developers to store large amount of data without indexing
CSV	Allows developers to store data in text files using comma-separated values
BLACKHOLE	Allows developers to accept data without storing. Tables that have this storage engine will always return an empty result. This storage engine can be used in distributed database design where data is automatically replicated, but not stored locally

Table 4.6: Storage Engines in MySQL

You will use the CREATE command to add a new table to a database.

The syntax for CREATE command is:

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name (create_clause,...) [table_
options] [IGNORE | REPLACE] [AS] Select ...;
```

where,

CREATE TABLE – adds a new table to the database

tbl_name – specifies a name for the table

Table 4.7 lists the description for the options in the CREATE TABLE syntax:

Option	Description
TEMPORARY	Table exists as long as the current user is connected
IF NOT EXISTS	Creates table only if it does not exists in the database
create_clause	Defines column
table_options	Creates different types of tables such as ISAM, INNODB, and so on
IGNORE	If table already exists system ignores and does not replace the previous table
REPLACE	Replaces the table if a table with the same name exists in the database
Select	Enables to copy records from an existing table

Table 4.7: CREATE TABLE Options

Session 4

Using MySQL

The syntax for `create_clause` in the `CREATE TABLE` command is:

```
column type [NOT NULL | NULL] [DEFAULT value] [AUTO_INCREMENT] [PRIMARY KEY] [REFERENCE];
```

where,

`column type` – defines the data type for the column

Table 4.8 lists the options in the `create_clause`.

Option	Description
<code>AUTO_INCREMENT</code>	Specifies that the column value must be auto incremented. This option works only if the columns contain positive values.
<code>DEFAULT value</code>	Inserts the specified value when no value for that column is entered. The default value has to be a constant. The values cannot be a function or expression.
<code>NOT NULL</code>	Specifies that a column cannot contain a null value. When a null value is inserted, the system prompts with an error message.
<code>NULL</code>	Specifies that the column can contain null values.
<code>PRIMARY KEY</code>	Defines a column as the primary key while creating a table.
<code>REFERENCES</code>	Assigns the InnoDB storage engine.

Table 4.8: Options in the `create_clause`

The syntax for `REFERENCE` options is:

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name  
REFERENCE tbl_name[(index_col_name,...)] [MATCH FULL | MATCH PARTIAL | MATCH SIMPLE] [ON DELETE reference_option] [ON UPDATE reference_option];
```

where,

`CREATE TABLE` – adds a new table to the database

`tbl_name` – specifies a name for the table

`REFERENCE` – defines a relationship between the index and another table

`tbl_name` – specifies the name for the referenced table

`index_col_name` – specifies the column to be indexed

Session 4

Using MySQL

Table 4.9 lists the options in the REFERENCE clause.

Concepts

Option	Description
index_col_name	Defines an index on the column
ON DELETE	Used for cascading tables
ON UPDATE	Used for cascading tables

Table 4.9: Options in the REFERENCE clause

While creating a table, you can also specify the storage engine.

Consider an example of a customer table with columns, such as `customer_id` and `name` and the table is assigned to a storage engine, INNODB.

```
CREATE TABLE customer (customer_id INT NOT NULL, name CHAR (15)) ENGINE = INNODB;
```

Similarly, you can create a table of the required type.

You will create three tables namely `EMP_DETAILS`, `EMP_DEPARTMENT`, and `SALARY_DETAILS` in the `EMPLOYEE` database.

The data stored in the three respective tables are as follows:

- **EMP_DETAILS:** Stores the personal details of the employees, such as identification number, name, address, and phone number.
- **EMP_DEPARTMENT:** Stores the department information, such as identification number, department, date of joining, and designation of the employee.
- **SALARY_DETAILS:** Stores the salary information of the employee, such as identification number, basic salary, house rent allowance, traveling allowance, and gross salary.

You will first have to select the database to create the tables. You will select the `EMPLOYEE` database with the help of `USE` command.

The syntax for USE command is:

```
USE DATABASE;
```

where,

`USE` – switches to the specified database

Session 4

Using MySQL

DATABASE - specifies the name of the database

The column E_ID contains the employee code, so you will use the INT data type. Similarly, you will define suitable data types for each column, depending on the data.

Accordingly, create an **EMP_DETAILS** table with the structure listed in table 4.10.

Column Name	Data Type	Length
E_ID	INT	3
E_FNAME	CHAR	10
E_LNAME	CHAR	15
E_ADDRESS	CHAR	15
E_PHONE_NO	INT	6

Table 4.10: EMP_DETAILS Table Structure

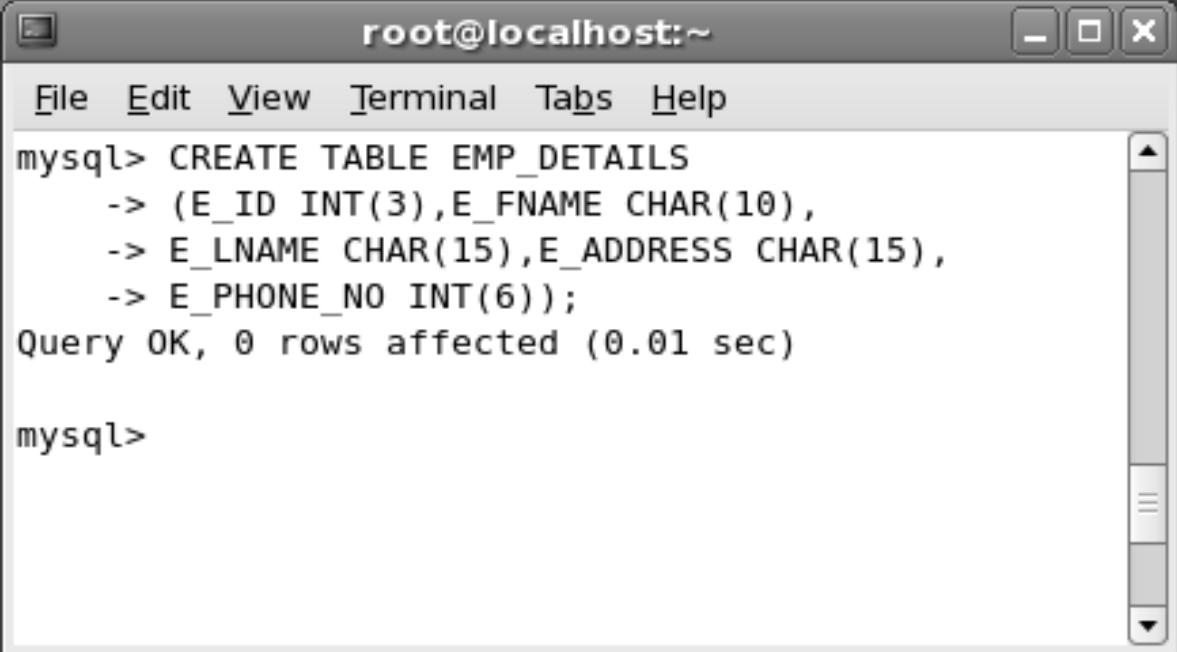
To create the **EMP_DETAILS** table, enter the following command at the command prompt:

```
CREATE TABLE EMP_DETAILS(E_ID INT(3), E_FNAME CHAR(10), E_LNAME CHAR(15),
E_ADDRESS CHAR(15), E_PHONE_NO INT(6));
```

Figure 4.2 displays the output of the command.

Session 4

Using MySQL



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". Below the menu, the MySQL prompt "mysql>" appears twice. The first instance shows the execution of a "CREATE TABLE" command:

```
mysql> CREATE TABLE EMP_DETAILS
    -> (E_ID INT(3), E_FNAME CHAR(10),
    -> E_LNAME CHAR(15), E_ADDRESS CHAR(15),
    -> E_PHONE_NO INT(6));
Query OK, 0 rows affected (0.01 sec)
```

The second instance of "mysql>" is at the bottom of the window.

Concepts

Figure 4.2: Creating a Table

MySQL provides the `DESCRIBE` command to view the structure of a table.

The syntax for the `DESCRIBE` command is:

```
DESCRIBE table [column];
```

Or

```
DESC table [column];
```

where,

`DESCRIBE` – displays the field structure of the table

`table` – specifies the name of the table to display the structure

`column` – displays information for a specific column of the table

To view the structure of the `EMP_DETAILS` table, enter the following command at the prompt:

```
DESC EMP_DETAILS;
```

Session 4

Using MySQL

Figure 4.3 displays the output of the DESCRIBE command.

The screenshot shows a terminal window titled "root@localhost:~". The window contains the following text:

```
mysql> DESCRIBE EMP_DETAILS;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| E_ID | int(3) | YES | | NULL | |
| E_FNAME | char(10) | YES | | NULL | |
| E_LNAME | char(15) | YES | | NULL | |
| E_ADDRESS | char(15) | YES | | NULL | |
| E_PHONE_NO | int(6) | YES | | NULL | |
+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql>
```

Figure 4.3: DESCRIBE Command

Figure 4.3 displays the following information for a table:

- Columns in the table
- Data type used for a column
- Acceptance of null values for a column. A column having a `NULL` value indicates that the value is not available and is not known
- Type of key such as Primary key
- Default value for a column
- Extra information or unique characteristics of a column
- Number of rows in the table

Now, you will create an `EMP_DEPARTMENT` table with the structure listed in table 4.11.

Session 4

Using MySQL

Concepts

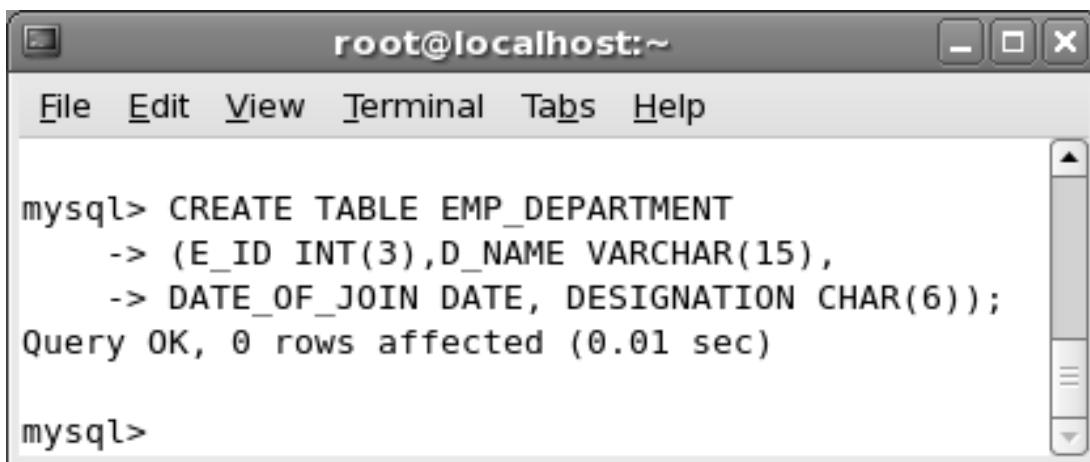
Column Name	Data Type	Length
E_ID	INT	3
D_NAME	VARCHAR	15
DATE_OF_JOIN	DATE	
DESIGNATION	CHAR	6

Table 4.11: Structure of EMP_DEPARTMENT Table

To create the `EMP_DEPARTMENT` table, enter the following command at the command prompt:

```
CREATE TABLE EMP_DEPARTMENT(E_ID INT(3),D_NAME VARCHAR(15), DATE_OF_JOIN  
DATE, DESIGNATION CHAR(6));
```

Figure 4.4 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a standard Linux-style title bar with icons for minimize, maximize, and close. Below the title bar is a menu bar with options: File, Edit, View, Terminal, Tabs, Help. The main area of the terminal shows the MySQL command-line interface. The user has entered the following SQL command:

```
mysql> CREATE TABLE EMP_DEPARTMENT  
-> (E_ID INT(3),D_NAME VARCHAR(15),  
-> DATE_OF_JOIN DATE, DESIGNATION CHAR(6));  
Query OK, 0 rows affected (0.01 sec)
```

After the command, the MySQL prompt "mysql>" is visible again.

Figure 4.4: Creating an EMP_DEPARTMENT Table

Similarly, create a `SALARY_DETAILS` table with the structure listed in table 4.12.

Column Name	Data Type	Length
E_ID	INT	3
BASIC_SAL	DECIMAL	(12,2)
HRA	DECIMAL	(7,2)
TA	DECIMAL	(7,2)
GROSS_SAL	REAL	(15,2)

Table 4.12: Structure of SALARY_DETAILS Table

In table 4.12, the data type of `BASIC_SAL` field is `DECIMAL`. The length of the field has been specified as

Session 4

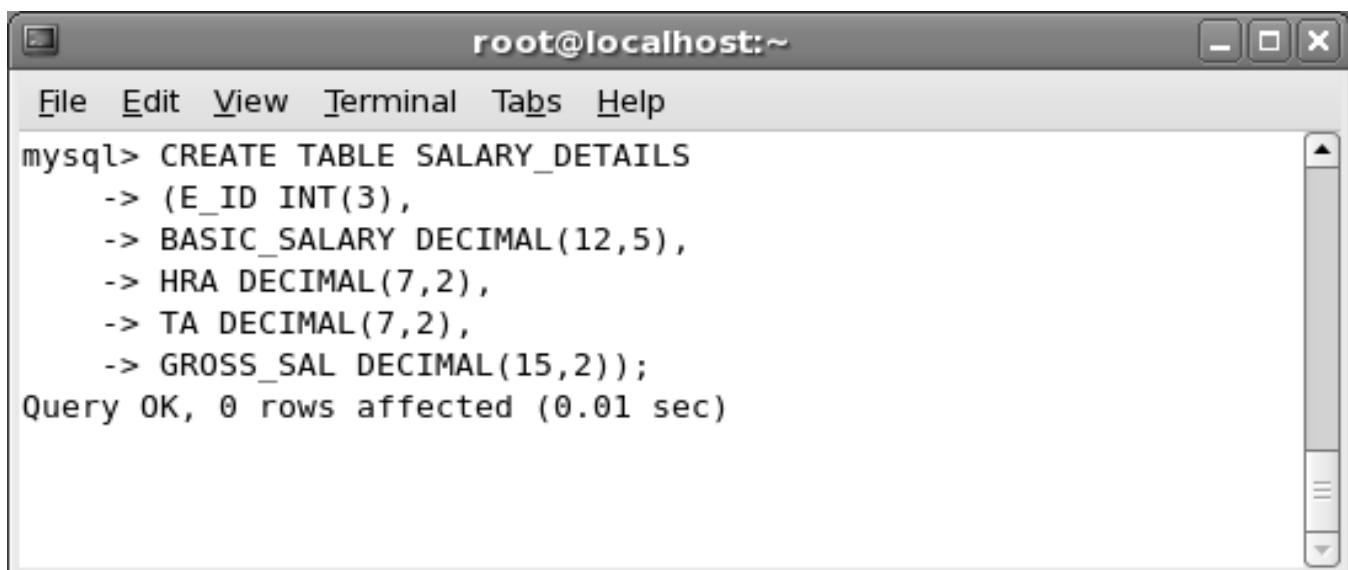
Using MySQL

specified as (12, 2). In this instance, the value before the comma in the bracket defines the total length of the column including the decimal point. The value after the comma specifies the number of digits after the decimal point.

To create a `SALARY_DETAILS` table, enter the following command at the command prompt:

```
CREATE TABLE SALARY_DETAILS(E_ID INT(3), BASIC_SALARY DECIMAL(12,5), HRA DECIMAL(7,2), TA DECIMAL(7,2), GROSS_SAL DECIMAL(15,2));
```

Figure 4.5 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following text:

```
File Edit View Terminal Tabs Help
mysql> CREATE TABLE SALARY_DETAILS
      -> (E_ID INT(3),
      -> BASIC_SALARY DECIMAL(12,5),
      -> HRA DECIMAL(7,2),
      -> TA DECIMAL(7,2),
      -> GROSS_SAL DECIMAL(15,2));
Query OK, 0 rows affected (0.01 sec)
```

Figure 4.5: `SALARY_DETAILS` Table

4.5 Normalization

Normalization is the process of structuring the data in a table to minimize duplication and inconsistency. Normalization requires breaking down of a larger table into two or more smaller tables. The tables are linked by defining a relationship between them. Normalization improves the clarity in a database. You can normalize a database by using a set of rules while creating the database.

You must understand the following terms before normalizing a database:

- **Entity** - represents an object about which you want to store the information
- **Attributes** - represents the component of entity type or identifying qualities
- **Domain** - defines the range for the attributes

Session 4

Using MySQL

- **Relationship** - represents association between entities

Concepts

4.5.1 Defining Keys

Keys help to establish a relationship between columns. You can define a field as a key. These keys uniquely identify a record. The different types of keys include:

- **Primary Key** - Is used as the unique identifier and has only one attribute
- **Composite Key** - Is a primary key having more than one attribute
- **Foreign Key** - Is a column or a set of column in a table which refers to a column or a set of column in another table

To normalize a database, you will use a sample table where you store department number, department name, employee identification number, employee name, employee category, and wages of the employee per hour.

Table 4.13 lists the data stored in a sample table for normalization.

D_No	D_Name	E_Id	E_Name	Emp_Category	Wages
10	Research	41	Charles	I	100
		46	Jack	II	60
		50	Peter	III	40
20	Marketing	51	Mike	I	100
		70	George	III	40

Table 4.13: Sample Table Data for Normalization

4.5.2 Different Forms of Normalization

You can apply different forms of normalization on a database to remove redundancy and improve clarity. You must follow certain set of rules while normalizing the database so that the database contains a minimum number of duplicate records.

- **First Normal Form or 1NF**

An entity is said to be in the first normal form, if all the attributes are single valued. In other words, there should not be any repeating attributes.

Session 4

Using MySQL

The following are the rules for a table to be in the first normal form:

- **All attributes are atomic:** Field holds a value for one object.
- **All columns have only one instance:** Columns must be defined to contain a specific type of record. For example, you can store the first and the last name of employees in different columns, such as `F_Name` and `L_Name`. This makes the data retrieval process easier.

Table 4.14 displays the sample table structure after applying the First Normal Form.

D_No	D_Name	E_Id	E_Name	Emp_Category	Wages
10	Research	41	Charles	I	100
10	Research	50	Peter	II	60
10	Research	46	Jack	III	40
20	Marketing	51	Mike	I	100
20	Marketing	70	George	III	40

Table 4.14: First Normal Form

In Table 4.14, the `D_NO` does not uniquely identify a record, so the primary key must be a combination of `D_NO` and `E_ID`. These two fields are used to uniquely identify the records.

➤ Second Normal Form or 2NF

An entity is said to be in the second normal form, if it is in the first normal form and all non key attributes entirely depend on the unique identifier of the entity.

After normalizing the sample table using the first normal form, the value for `D_NO` repeats several times. Also, the field `D_NAME` depends only on the `D_NO` field which is part of the primary key and not the entire primary key. Therefore, you will apply the second normal form and split the tables, as shown in tables 4.15, 4.16 and 4.17.

EMP_DEPARTMENT TABLE

D_No	E_Id
10	41
10	50
10	46
20	51
20	70

Table 4.15: EMP_DEPARTMENT Table in Second Normal Form

Session 4

Using MySQL

EMPLOYEE TABLE:

E_Id	E_Name	Emp_Category	Wages
41	Charles	I	100
50	Peter	II	60
46	Jack	III	40
51	Mike	I	100
70	George	III	40

Concepts

Table 4.16: EMPLOYEE Table in Second Normal Form

DEPARTMENT TABLE:

D_No	D_Name
10	Marketing
20	Research

Table 4.17: DEPARTMENT Table in Second Normal Form

➤ Third Normal Form or 3NF

An entity is said to be in the third normal form, if it is already in the second normal form and no non-identifying attributes are dependent on any other non-identifying attributes.

Consider the following data in the EMPLOYEE table, as shown in table 4.18.

E_Id	E_Name	Emp_Category	Wages
41	Charles	I	100
50	Peter	II	60
46	Jack	III	60
51	Mike	I	60
70	George	III	40

Table 4.18: EMPLOYEE Table

In table 4.18, the employee Mike is of EMP_CATEGORY I and the wages should be 100 instead of 60. You are redundantly storing the EMP_CATEGORY and WAGES details. All the attributes should depend only on the primary key according to the third normal form. Therefore, you will split the table, as shown in tables 4.19 and 4.20.

Session 4

Using MySQL

EMPLOYEE table:

E_Id	E_Name	Emp_Category
41	Charles	I
50	Peter	II
46	Jack	III
51	Mike	I
70	George	III

Table 4.19: EMPLOYEE Table in Third Normal Form

RATE table:

Emp_Category	Wages
I	100
II	60
III	40

Table 4.20: EMPLOYEE Table Split in Third Normal Form

➤ **Boyce-Codd Normal Form or BCNF**

An entity is said to be in the Boyce-Codd normal form, if it is already in the third normal form and contains only one unique identifier.

The tables in the earlier example are already in the Boyce-Codd normal form.

4.6 Database Concepts

Referential integrity is a feature that prevents you from storing inconsistent data. You implement referential integrity when you define a relationship between two tables in a database. Consider an example, where Table Y has a foreign key pointing to a field in Table X, referential integrity prevents addition of a new record in Table Y that cannot be linked to Table X. In addition, whenever you delete a record from Table X, any record linked to that record in Table Y is also deleted. Also, when you alter or update a linked field in Table X, all the records linked to it in Table Y are updated accordingly.

You can use indexing on databases to make search operations faster. Normally, you index the fields in a table, on which the search is to be performed. Also, you can index several fields in a table at a time.

4.6.1 Using Indexes

Indexes are lists that specify the location of data in a table within a database. Indexes support faster data

Session 4

Using MySQL

storage and retrieval capabilities. Indexes in a database are similar to the indexes in books. Consider the example where you use the index to find some information on a topic in a book instead of reading the entire book. Indexes in a database use the same concept.

You can define indexes on a field in a table within a database. The only disadvantage of indexing is that it uses more storage space. In addition, data manipulation commands, such as insert, update, and delete require longer duration to execute.

You can create an index in a table by indexing the fields. You must define the index on a field that will sort the records in the table. The primary key serves as the index key when the index key is not specified explicitly.

You must remember the following points while indexing fields in a table:

- Fields should have unique values
- Fields on which the table is to be sorted
- Fields should have data types such as text, number, date or time

4.6.2 Referential Integrity

You can use referential integrity to maintain consistency of records in different tables. It means that when a record in a table refers to a corresponding record in another table, that corresponding record should exist. Referential integrity can be defined by linking a foreign key of a table to the primary key of another. This means that a foreign key of one table must have a corresponding row and primary key in the related table.

Consider an example, where you have two tables, `EMP_DETAILS` and `EMP_SALARY`. Employees can be uniquely identified with the `E_ID`. Hence this is the primary key in the `EMP_DETAILS` table. The salary details stored in the `EMP_SALARY` table will have `E_ID` as the foreign key.

In this example, referential integrity of the database will not allow you to make inconsistent changes in either of these two tables.

Consider the following records stored in the `EMP_DETAILS` and `EMP_DEPARTMENT` table as shown in tables 4.21 and 4.22 respectively.

`EMP_DETAILS`

<code>E_ID</code>	<code>E_FNAME</code>	<code>E_LNAME</code>	<code>E_ADDRESS</code>	<code>E_PHONE_NO</code>
101	Jack	Williams		765432
102	Peter	Adams		712832

Table 4.21: `EMP_DETAILS` Table

Session 4

Using MySQL

EMP_DEPARTMENT

Concepts

E_ID	D_NAME	DATE_OF_JOIN	DESIGNATION
101	Sales	2001-01-01	Supervisor
102	Sales	2000-02-05	Worker
103	Research	2003-04-06	Worker

Table 4.22: EMP_DEPARTMENT Table

You can identify inconsistency in the data from the tables 4.21 and 4.22. The employee with the E_ID as 103 does not have a corresponding entry in the EMP_DETAILS. You will use referential integrity to remove inconsistency from the tables.

To implement referential integrity, you will have to use a foreign key.

The syntax for defining a foreign key while creating a table is:

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] table (create_clause,...) [table_options] [IGNORE [REPLACE] select],FOREIGN KEY(col_name) REFERENCES <table_name>(referenced_column);
```

where,

FOREIGN KEY – defines a foreign key

col_name – specifies the name of the column to be used for the foreign key

REFERENCES – defines a relationship with other tables

table_name – specifies the name of the table to define a relationship

referenced_column – specifies the column of the table to define a relationship

Consider the following example where the E_ID is defined as a primary key in the EMP_DETAILS table.

```
CREATE TABLE EMP_DETAILS (E_ID INT,E_FNAME CHAR (10),E_LNAME CHAR(15),ADDRESS CHAR(15),E_PHONE_NO INT(6),PRIMARY KEY (E_ID));
```

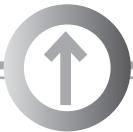
Consider the following example, where the E_ID is defined as a Foreign key referencing E_ID column of EMP_DETAILS table.

Session 4

Using MySQL

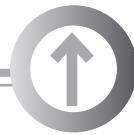
```
CREATE TABLE EMP_DEPARTMENT (E_ID INT, D_NAME CHAR(), DATE_OF_JOIN DATE, DESIGNATION CHAR(6), FOREIGN KEY (E_ID) REFERENCES EMP_DETAILS (E_ID));
```

In the example, `EMP_DETAILS` is the master table and `EMP_DEPARTMENT` is the child table. This relationship enables to store consistent records in the tables. Referential integrity monitors addition and deletion of records from these tables. MySQL will generate an error message when you attempt to add a new record in the child table that does not have a corresponding record in the parent table. Similarly, MySQL will generate an error message when you delete a record from the child table that has a corresponding entry in the master table.



Summary

- Databases are used to store information and can contain more than one table.
- A table consists of rows and columns. A column in a table is known as a field and row is known as a record.
- MySQL supports data types, such as numeric, string, date, and complex.
- Numeric data type column consists of different types of numbers. A number can be classified as signed or unsigned. A signed number stores a negative or positive value.
- String data type columns consist of data in the form of characters.
- Date data type column have date values entered as string values in the form of 'YYYY-MM-DD' format.
- MySQL provides the `CREATE` command to generate new databases and tables.
- Normalization removes redundancies from the database.
- A Primary key is a field in the table which is used as the unique identifier. A primary key having more than one attribute is called Composite key.
- A Foreign key in a table is a Primary Key in another table.
- An entity is said to be in the first normal form if all the attributes are single valued.
- An entity is said to be in the second normal form, if it is in the first normal and all non key attributes entirely depend on the unique identifier of the entity.
- An entity is said to be in the third normal form, if it is in the second normal form and no non-identifying attributes are dependent on any other non-identifying attributes.
- An entity is said to be in the Boyce-Codd normal form if it is already in the third normal form and only one unique identifier exists.
- Referential integrity means that when you have a record in a table that refers to corresponding record in another table then that particular record must exist.
- Indexing enables faster search and retrieval operations in a database.



Check Your Progress

1. The Boolean data type is similar to _____.
 - a. TINYTEXT
 - b. MEDIUMTEXT
 - c. TINYINT
 - d. MEDIUMINT

2. A foreign key in a table can also be a _____.
 - a. Primary key in another table
 - b. Composite key in another table
 - c. Primary key and composite key in another table
 - d. Foreign key in another table

3. An entity is said to be in the first normal form if _____.
 - a. all the attributes are single valued
 - b. all the attributes are multiple valued
 - c. all the attributes are not null valued
 - d. there is no top to bottom ordering of rows

4. Which type of key is used by the child table for referential integrity?
 - a. Primary key
 - b. Foreign key
 - c. Composite key
 - d. Indexes



Check Your Progress

5. Which of the following options helps to remove redundancies from a database?
 - a. Referential integrity
 - b. Indexing
 - c. Normalization
 - d. Boyce-Codd Rule

Objectives

At the end of this session, the student will be able to:

- *Create a database.*
- *Create a table using different data types.*

The steps given in this session are detailed, comprehensive, and carefully thought through in order to meet the learning objectives and understand the tool completely. Please follow the steps carefully.

Part I - For the first 1.5 hours:

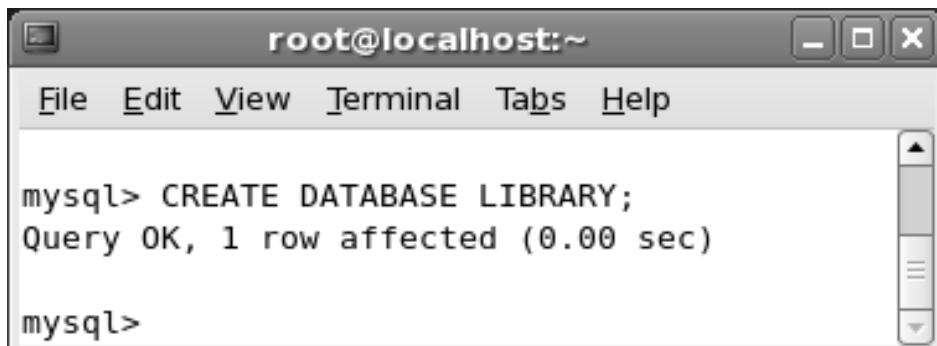
Creating a Database

A relational database consists of several tables that store a particular set of data. Data in all these tables are related to each other. You will create a **LIBRARY** database to store details about the books, user's category, and date of issue.

1. Open the Linux terminal and start MySQL.
2. To create the **LIBRARY** database to store details about the books, user's category, and date of issue, enter the following command at the command prompt:

```
CREATE DATABASE LIBRARY;
```

Figure 5.1 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a standard Linux desktop interface with a title bar, menu bar, and scroll bars. The terminal content shows the following MySQL command and its execution:

```
mysql> CREATE DATABASE LIBRARY;
Query OK, 1 row affected (0.00 sec)

mysql>
```

Figure 5.1: Creation of a Database

Session 5

Using MySQL (Lab)

Creating a Table Using Different Data Types

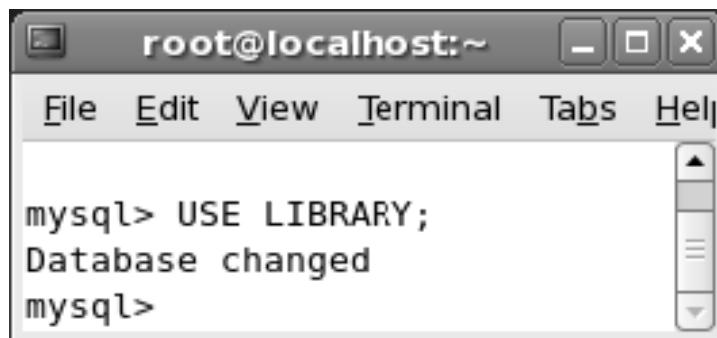
Tables store information in a row and column format. A row is known as a record and a column is called a field. You can use different data types for fields depending on the data stored in the field. Create the following tables in the LIBRARY database:

- BOOK_DETAILS
- USER_DETAILS
- CATEGORY
- ISSUE_DETAILS

3. In order to create tables in the LIBRARY database, you must activate the database. To activate the LIBRARY database, enter the following command at the command prompt:

```
USE LIBRARY;
```

Figure 5.2 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area of the terminal shows the MySQL command-line interface. The user has entered the command "USE LIBRARY;" and received the response "Database changed". The prompt "mysql>" is visible again.

Figure 5.2: Activation of the LIBRARY Database

First, you will create a table to store the book details, such as book identification number, book name, author, publication, edition, and description about the book. The BOOK_DETAILS table has the structure as listed in table 5.1.

Field Name	Data Type
BOOK_ID	INT
BOOKNAME	VARCHAR (35)
AUTHOR	CHAR (25)
EDITION	CHAR (15)

Session 5

Using MySQL (Lab)

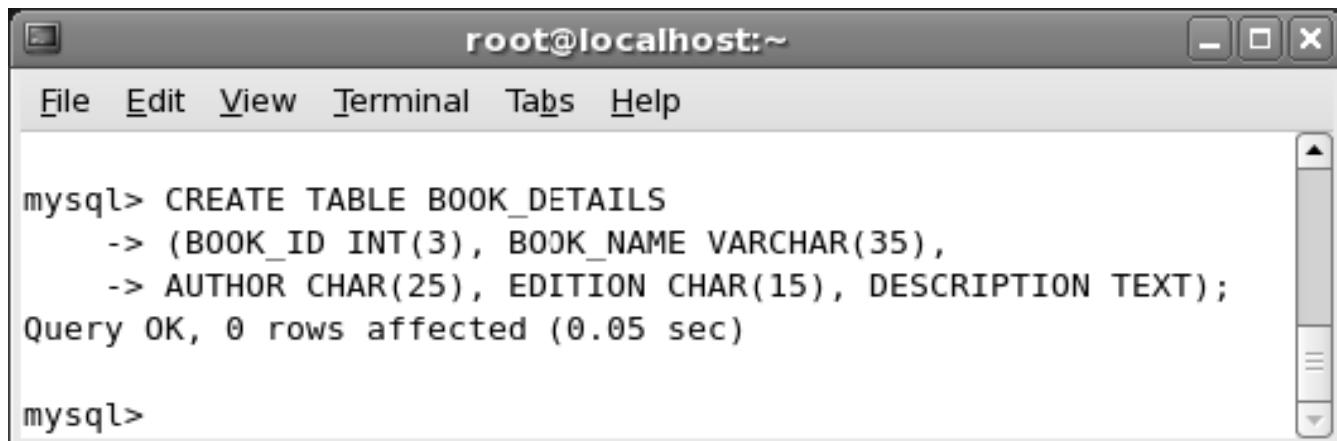
Field Name	Data Type
DESCRIPTION	TEXT

Table 5.1: BOOK_DETAILS Table Structure

4. To create the BOOK_DETAILS table to store the book details, such as book identification number, book name, author, edition, and description, enter the following command at the command prompt:

```
CREATE TABLE BOOK_DETAILS(BOOK_ID INT(3),BOOK_NAME VARCHAR(35)  
, AUTHOR CHAR(25), EDITION CHAR(15), DESCRIPTION TEXT);
```

Figure 5.3 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a standard Linux-style title bar with icons for minimize, maximize, and close. Below the title bar is a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The main area of the terminal shows the MySQL command-line interface. The user has entered the following SQL command to create the "BOOK_DETAILS" table:

```
mysql> CREATE TABLE BOOK_DETAILS  
-> (BOOK_ID INT(3), BOOK_NAME VARCHAR(35),  
-> AUTHOR CHAR(25), EDITION CHAR(15), DESCRIPTION TEXT);  
Query OK, 0 rows affected (0.05 sec)
```

After the command, the user types "mysql>" again, indicating they are still in the MySQL prompt.

Figure 5.3: Creation of BOOKDETAILS Table

5. To view the structure of the BOOK_DETAILS table, enter the following command at the command prompt:

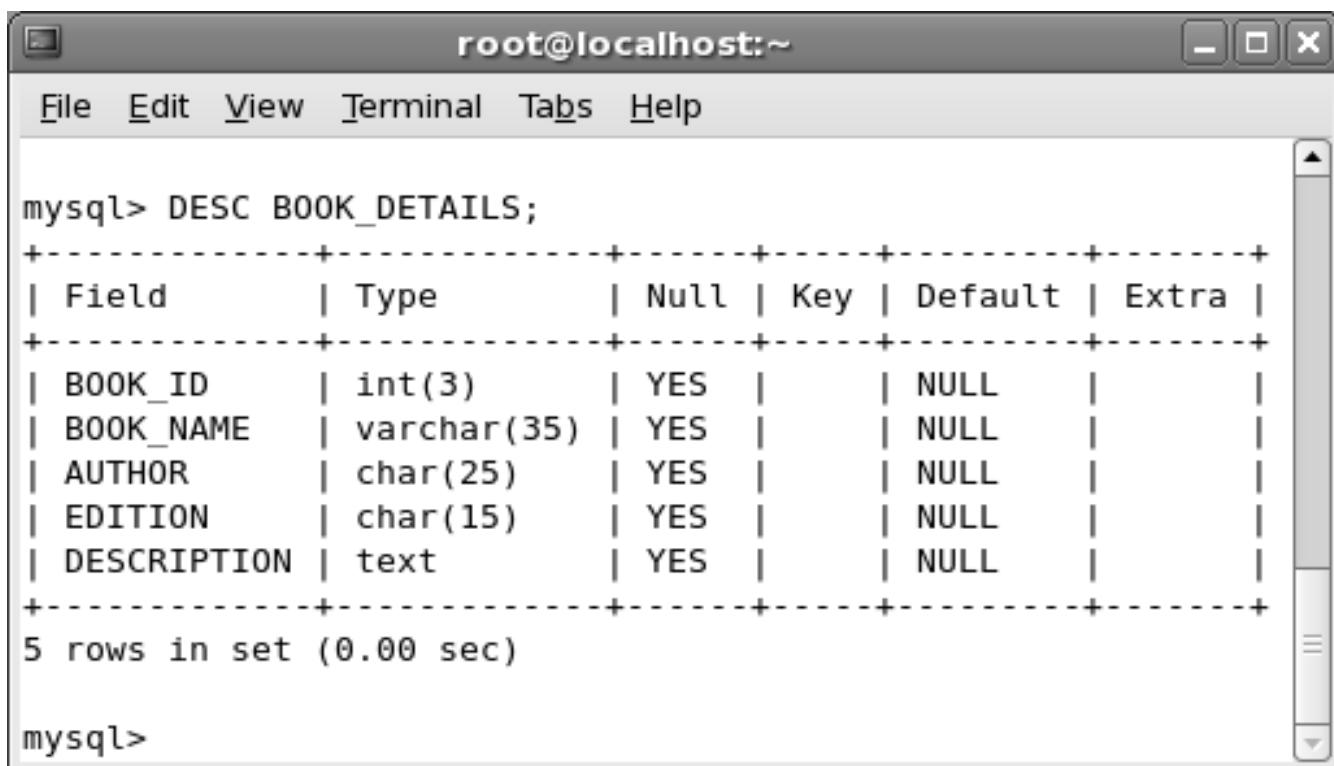
```
DESC BOOK_DETAILS;
```

Figure 5.4 displays the output of the command.

Session 5

Using MySQL (Lab)

Lab Guide



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area displays the following MySQL command and its results:

```
mysql> DESC BOOK_DETAILS;
+-----+-----+-----+-----+-----+
| Field      | Type       | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| BOOK_ID    | int(3)     | YES  |     | NULL    |       |
| BOOK_NAME   | varchar(35) | YES  |     | NULL    |       |
| AUTHOR      | char(25)    | YES  |     | NULL    |       |
| EDITION     | char(15)    | YES  |     | NULL    |       |
| DESCRIPTION | text        | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql>
```

Figure 5.4: BOOK_DETAILS Table Structure

Create a table to store the user details, such as the user identification number, first name, last name, address, and phone number. The `USER_DETAILS` table has the structure as listed in table 5.2.

Field Name	Data Type
USER_ID	INT (3)
NAME	VARCHAR (25)
LOCATION	CHAR (15)
PHONE_NO	INT (6)

Table 5.2: USER_DETAILS Table Structure

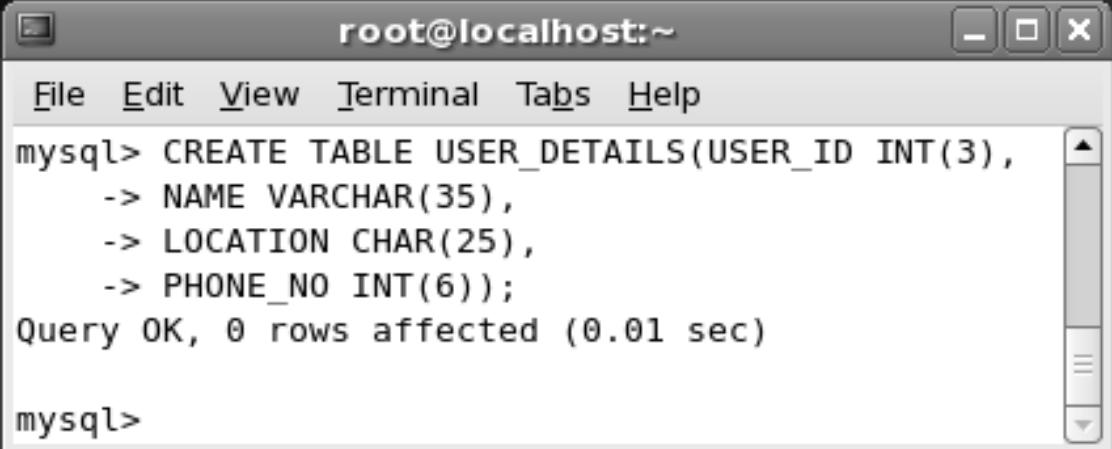
6. To create a `USER_DETAILS` table to store the user details, such as the user identification number, first name, last name, address, and phone number, enter the following command at the command prompt:

```
CREATE TABLE USER_DETAILS(USER_ID INT (3), NAME VARCHAR (35),
                           LOCATION CHAR (25), PHONE_NO INT (6));
```

Figure 5.5 displays the output of the command.

Session 5

Using MySQL (Lab)



```
root@localhost:~ File Edit View Terminal Tabs Help mysql> CREATE TABLE USER_DETAILS(USER_ID INT(3), -> NAME VARCHAR(35), -> LOCATION CHAR(25), -> PHONE_NO INT(6)); Query OK, 0 rows affected (0.01 sec)

mysql>
```

Lab Guide

Figure 5.5: Creation of USER_DETAILS Table

Now, create a category table to store information, such as user identification number, membership type of the user, facility provided to the user, and the issue status (whether any book is issued to the user or not). The CATEGORY table has the structure as listed in table 5.3.

Field Name	Data Type
USER_ID	INT (3)
TYPE	VARCHAR (35)
FACILITY	CHAR (25)
ISSUE_STATUS	BOOLEAN

Table 5.3: CATEGORY Table Structure

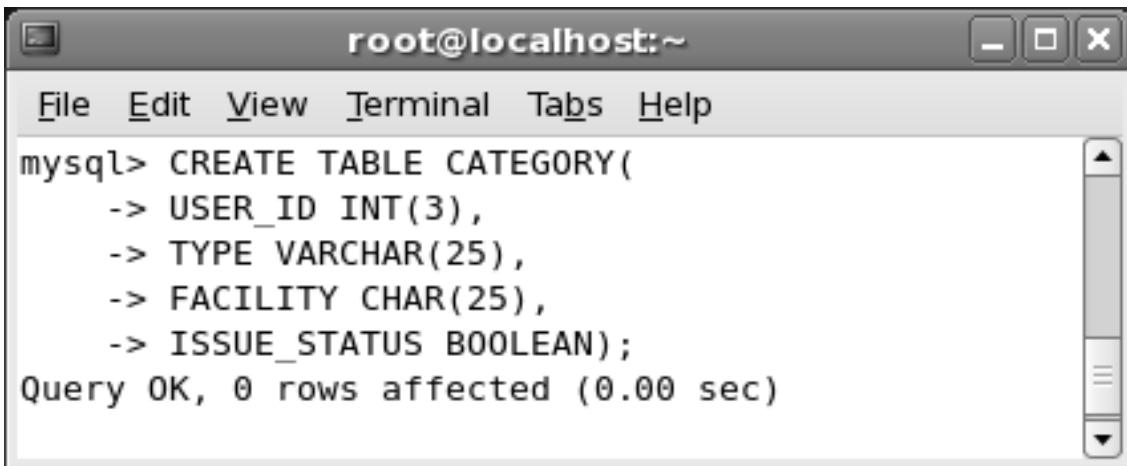
7. To create a CATEGORY table to store information, such as user identification number, membership type of the user, facility provided to the user, and the issue status, enter the following command at the command prompt:

```
CREATE TABLE CATEGORY (USER_ID INT(3), TYPE VARCHAR(25),  
FACILITY CHAR(25), ISSUE_STATUS BOOLEAN);
```

Figure 5.6 displays the output of the command.

Session 5

Using MySQL (Lab)



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its execution results:

```
mysql> CREATE TABLE CATEGORY(
    -> USER_ID INT(3),
    -> TYPE VARCHAR(25),
    -> FACILITY CHAR(25),
    -> ISSUE_STATUS BOOLEAN);
Query OK, 0 rows affected (0.00 sec)
```

Figure 5.6: Creation of CATEGORY Table

Finally, create an `ISSUE_DETAILS` table to store the user ID, book ID, book name, date of issue, date of return, and fine. The `ISSUE_DETAILS` table has the structure as listed in table 5.4.

Field Name	Data Type
USER_ID	INT (3)
BOOK_ID	INT (3)
BOOK_NAME	CHAR (35)
DATE_OF_ISSUE	DATE
DATE_OF_RETURN	DATE
FINE	DECIMAL (5, 2)

Table 5.4: ISSUE_DETAILS Table Structure

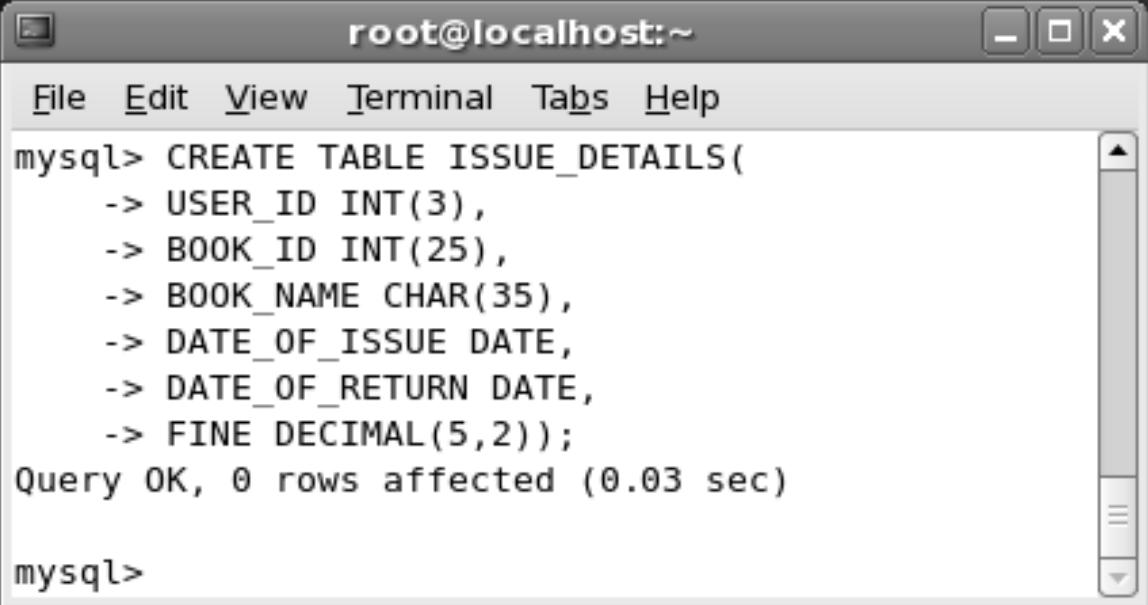
8. To create a `ISSUE_DETAILS` table to store the user ID, book ID, book name, date of issue, date of return, and fine, enter the following command at the command prompt:

```
CREATE TABLE ISSUE_DETAILS (USER_ID INT(3), BOOK_ID INT(25)
, BOOK_NAME CHAR (35), DATE_OF_ISSUE DATE, DATE_OF_RETURN DATE, FINE
DECIMAL (5, 2));
```

Figure 5.7 displays the output of the command.

Session 5

Using MySQL (Lab)



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its execution results:

```
mysql> CREATE TABLE ISSUE_DETAILS(
    -> USER_ID INT(3),
    -> BOOK_ID INT(25),
    -> BOOK_NAME CHAR(35),
    -> DATE_OF_ISSUE DATE,
    -> DATE_OF_RETURN DATE,
    -> FINE DECIMAL(5,2));
Query OK, 0 rows affected (0.03 sec)

mysql>
```

Lab Guide

Figure 5.7: Creation of ISSUE_DETAILS Table



Do It Yourself

Create a database for storing the product sales detail information. The `Product_Sales` database contains two tables, `Stock_Details` and `Sales_Details`. The `Stock_Details` table contains fields, such as code, name, description, cost price, and date of manufacture of the product.

1. Create a `PRODUCT_SALES` database.
2. Create a `STOCK_DETAILS` table having the structure as listed in table 5.5.

Field Name	Data Type
Product_Code	INT(4)
Product_Name	CHAR(15)
Description	TEXT
Cost_Price	DECIMAL(8, 2)
Date_of_Manufacture	DATE

Table 5.5: STOCK_DETAILS Table Structure

3. Create a `SALES_DETAILS` table having the structure as listed in table 5.6.

Field Name	Data Type
Product_Code	INT(4)
Selling_Price	DECIMAL(8, 2)
Qty	INT(4)
Units_Sold	INT(4)
Month	SET('JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL', 'AUG', 'SEP', 'OCT', 'NOV', 'DEC')
Location	TEXT

Table 5.6: SALES_DETAILS Table Structure

Objectives

At the end of this session, the student will be able to:

- *Describe the commands to view and alter a database.*
- *Explain the commands to retrieve data from a table.*
- *Describe the commands to modify the table definitions.*
- *Describe the commands to delete the table definitions.*

6.1 Introduction

MySQL supports Structured Query Language (SQL) queries to handle data. You can use queries in MySQL to create databases and tables, insert data into tables, and retrieve data from tables. The SQL queries are short and single line statements. There may be multiple statements in a single query.

In this session, you will learn to write queries to view tables in a database and modify the database. You will also learn to use queries to retrieve data from a table and modify the table structure. In addition, you will learn to delete a table from the database.

6.2 Viewing and Altering Databases on the File System

While working with databases, you may need information about the number of databases present in the server, the names and other details of these databases, or you may need details about the structure of any database. You may also have to modify the structure of the database. MySQL enables you to view and alter an existing database on the file system through queries.

6.2.1 Viewing Information about Databases on the File System

Before executing any query in MySQL, you can list the databases that are managed by the server. You will use the `SHOW` command to display the list of databases and index keys along with the privileges. The `SHOW` command provides information about databases, tables, columns, and server status.

To view a list of databases present on the server, use the following syntax:

```
SHOW DATABASES;
```

Session 6

Implementing SQL Queries Using MySQL - I

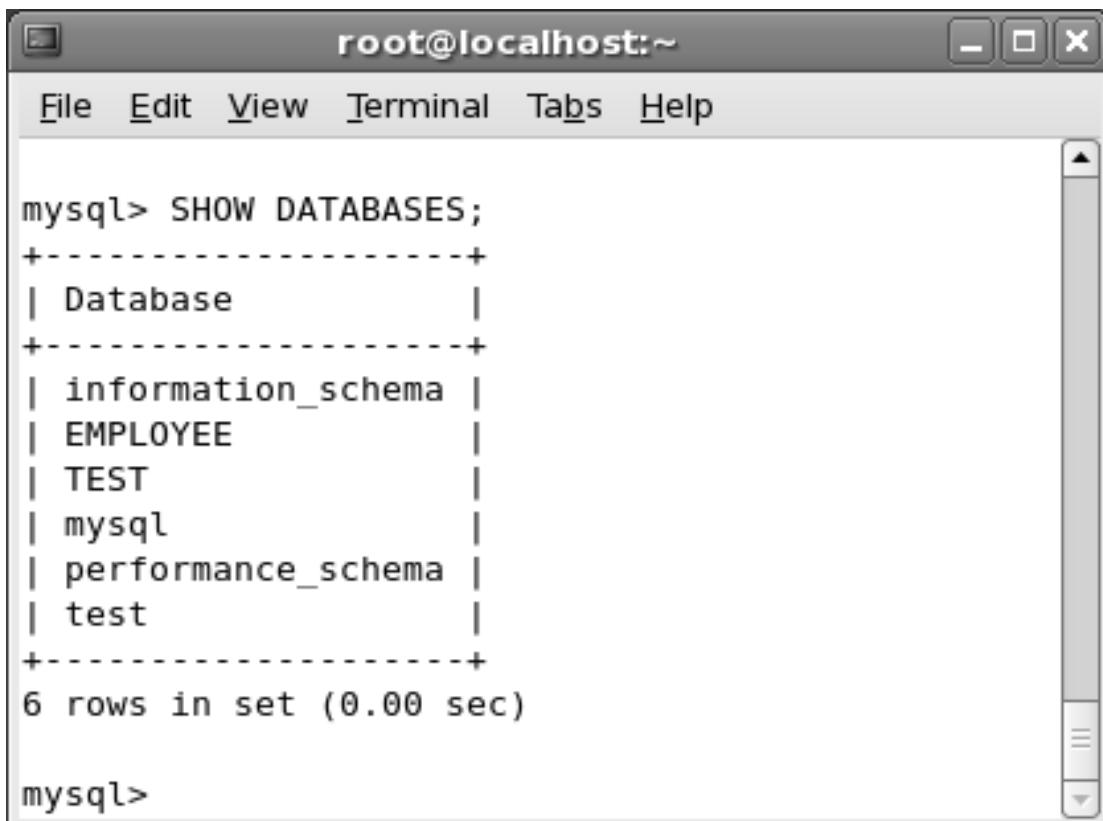
where,

Concepts

SHOW – displays the object specified in the clause

DATABASES – displays the databases present in the instance of MySQL

Figure 6.1 displays the output of the SHOW DATABASES command.



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following text:

```
mysql> SHOW DATABASES;
+-----+
| Database      |
+-----+
| information_schema |
| EMPLOYEE       |
| TEST           |
| mysql          |
| performance_schema |
| test           |
+-----+
6 rows in set (0.00 sec)

mysql>
```

Figure 6.1: SHOW DATABASES Command

Figure 6.1 displays the output of the SHOW DATABASES command. The output lists all the databases present on the server. It displays the total number of databases in the form of number of rows. In addition, it also displays the time taken to execute the command in seconds.

To view a list of databases that begin or contain specified characters in the database name, use the following syntax:

```
SHOW DATABASES [LIKE <condition>];
```

Session 6

Implementing SQL Queries Using MySQL - I

where,

SHOW DATABASES – displays the list of databases present in the instance of MySQL

LIKE <condition> – contains conditions that must be satisfied before displaying the list of databases

The LIKE clause can use conditions with wildcard characters such as ‘%’ and ‘_’. This clause is optional. The ‘%’ represents any string of zero or more characters. The ‘_’ represents any single character.

For example, to view all databases that begin with the letter ‘E’, enter the following command at the command prompt:

```
SHOW DATABASES LIKE 'E%';
```

Figure 6.2 displays the output of the command. The output lists all the databases starting with the letter ‘E’, present on the server.

```
root@localhost:~  
File Edit View Terminal Tabs Help  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
mysql> SHOW DATABASES;  
+-----+  
| Database |  
+-----+  
| information_schema |  
| EMPLOYEE |  
| mysql |  
| performance_schema |  
| test |  
+-----+  
5 rows in set (0.04 sec)  
  
mysql> SHOW DATABASES LIKE 'E%';  
+-----+  
| Database (E%) |  
+-----+  
| EMPLOYEE |  
+-----+  
1 row in set (0.00 sec)  
  
mysql>
```

Figure 6.2: Using LIKE clause with SHOW DATABASES

Session 6

Implementing SQL Queries Using MySQL - I

To view a list of tables in a database, use the following syntax:

```
SHOW TABLES [FROM database_name] [LIKE <condition>];
```

where,

SHOW TABLES – displays the tables from the selected database

FROM database_name – displays the tables from the database specified in the clause

The FROM clause enables you to specify the database name for which tables are to be listed. You can use this keyword when you have not activated a database from the server with the USE command.

LIKE <condition> – displays the tables from the database based on the condition specified in the clause.

For example, to view the tables of the EMPLOYEE database that has been set as current database through the USE command, enter the following command at the command prompt:

```
SHOW TABLES;
```

Figure 6.3 displays the output of the command.

Session 6

Implementing SQL Queries Using MySQL - I

Concepts

```
root@localhost:~  
File Edit View Terminal Tabs Help  
Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.  
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
mysql> USE EMPLOYEE  
Reading table information for completion of table and column names  
You can turn off this feature to get a quicker startup with -A  
Database changed  
mysql> SHOW TABLES;  
+-----+  
| Tables_in_EMPLOYEE |  
+-----+  
| EMP_DEPARTMENT |  
| EMP_DETAILS |  
| SALARY_DETAILS |  
+-----+  
3 rows in set (0.00 sec)  
mysql>
```

Figure 6.3: Tables in the EMPLOYEE Database

Figure 6.3 displays the output of the command. The output lists all the tables present in the EMPLOYEE database. You must select the database with `USE` command before displaying the tables using the command shown in figure 6.3.

Note: You can also use '`mysqlshow database_name`' command for displaying all the tables in a database.

To view a list of tables in a database when the database has not been selected earlier through the `USE` command, enter the following command at the command prompt:

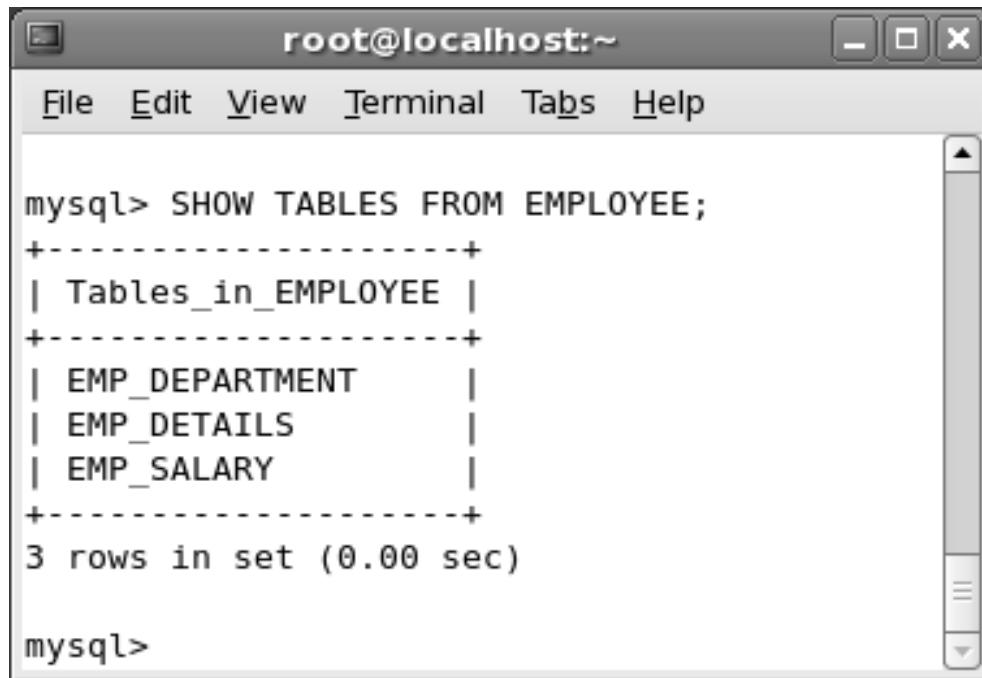
```
SHOW TABLES FROM EMPLOYEE;
```

Figure 6.4 displays the output of the command.

Session 6

Implementing SQL Queries Using MySQL - I

Concepts



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The main area displays the following MySQL command and its output:

```
mysql> SHOW TABLES FROM EMPLOYEE;
+-----+
| Tables_in_EMPLOYEE |
+-----+
| EMP_DEPARTMENT      |
| EMP_DETAILS          |
| EMP_SALARY           |
+-----+
3 rows in set (0.00 sec)

mysql>
```

Figure 6.4: Listing the Tables in EMPLOYEE Database

You will use the `FROM` keyword here because the `EMPLOYEE` database has not been activated with the `USE` command before displaying the tables.

To view tables that begin with the letter 'E' from the `EMPLOYEE` database, enter the following command at the command prompt:

```
SHOW TABLES FROM EMPLOYEE LIKE 'E%';
```

Figure 6.5 displays the output of the command.

Session 6

Implementing SQL Queries Using MySQL - I

Concepts

The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL session:

```
owners.

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

mysql> SHOW TABLES FROM EMPLOYEE;
+-----+
| Tables_in_EMPLOYEE |
+-----+
| EMP_DEPARTMENT      |
| EMP_DETAILS          |
| SALARY_DETAILS       |
+-----+
3 rows in set (0.00 sec)

mysql> SHOW TABLES FROM EMPLOYEE LIKE 'E%';
+-----+
| Tables_in_EMPLOYEE (E%) |
+-----+
| EMP_DEPARTMENT          |
| EMP_DETAILS              |
+-----+
2 rows in set (0.00 sec)

mysql>
```

Figure 6.5: Listing the Tables Beginning with 'E'

Figure 6.5 displays all the tables present in the `EMPLOYEE` database starting with letter 'E' and any number of characters after the letter E. The `LIKE` keyword is used to display only those tables, which matches with the specified letter.

To view the column structure of a table from the database, use the following syntax:

```
SHOW COLUMNS FROM table_name [FROM database_name] [LIKE clauses];
```

where,

`table_name` – specifies the name of the table

`COLUMNS` – displays the columns from the specified table

`table_name` – specifies the name of the table that contains the columns

Session 6

Implementing SQL Queries Using MySQL - I

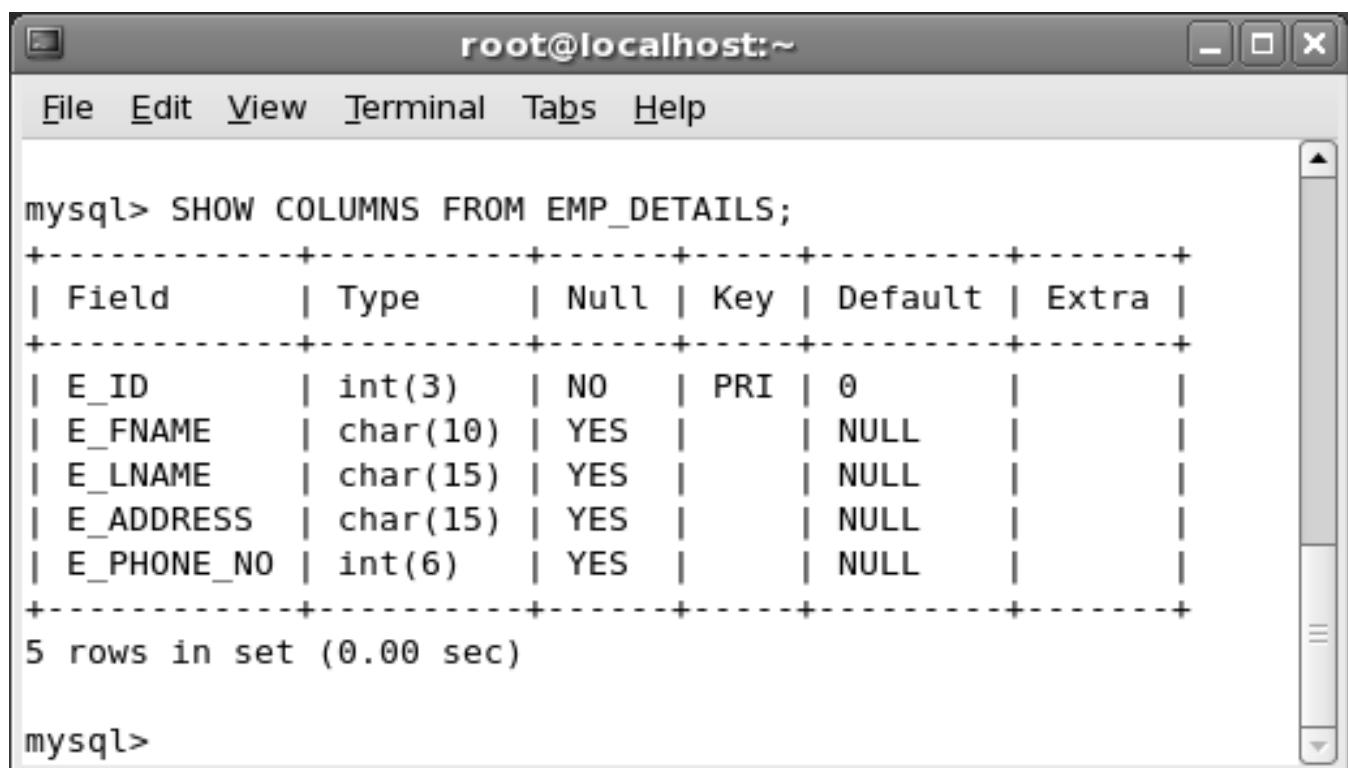
database_name – specifies the name of the database

LIKE – specifies conditions, if any

For example, to view the column structure of the EMP_DETAILS table, enter the following command at the command prompt:

```
SHOW COLUMNS FROM EMP_DETAILS;
```

Figure 6.6 displays the output of the command. Here, the database name is omitted because the EMPLOYEE database is the current database.



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following text:

```
mysql> SHOW COLUMNS FROM EMP_DETAILS;
+-----+-----+-----+-----+
| Field      | Type       | Null | Key | Default | Extra |
+-----+-----+-----+-----+
| E_ID        | int(3)     | NO   | PRI  | 0        |          |
| E_FNAME     | char(10)   | YES  |       | NULL     |          |
| E_LNAME     | char(15)   | YES  |       | NULL     |          |
| E_ADDRESS   | char(15)   | YES  |       | NULL     |          |
| E_PHONE_NO  | int(6)     | YES  |       | NULL     |          |
+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql>
```

Figure 6.6: Columns of EMP_DETAILS

Figure 6.6 displays the columns present in the EMP_DETAILS table.

Table 6.1 lists the description of each of columns displayed in figure 6.6.

Column	Description
FIELD	Indicates column name
TYPE	Indicates the data type for a column

Session 6

Implementing SQL Queries Using MySQL - I

Concepts

Column	Description
NULL	Specifies that the column can contain empty values
KEY	Specifies if the column is indexed
DEFAULT	Specifies the default value of the column
EXTRA	Specifies additional characteristics for the columns

Table 6.1: Detailed Information of Columns from EMP_DETAILS

To display the same output with the EMPLOYEE database specified explicitly, enter the following command at the command prompt:

```
SHOW COLUMNS FROM SALARY_DETAILS FROM EMPLOYEE;
```

Figure 6.7 displays the output of the command.

The screenshot shows a terminal window titled "root@localhost:~". The window contains two sets of command-line output. The first set is a direct copy of the provided text. The second set is the result of running the command in the terminal, showing the actual database context.

```
root@localhost:~  
File Edit View Terminal Tabs Help  
+-----+-----+-----+-----+-----+  
| Field | Type | Null | Key | Default | Extra |  
+-----+-----+-----+-----+-----+  
| E_ID | int(3) | YES | | NULL | |  
| E_FNAME | char(10) | YES | | NULL | |  
| E_LNAME | char(15) | YES | | NULL | |  
| E_ADDRESS | char(15) | YES | | NULL | |  
| E_PHONE_NO | int(6) | YES | | NULL | |  
+-----+-----+-----+-----+-----+  
5 rows in set (0.00 sec)  
  
mysql> SHOW COLUMNS FROM SALARY_DETAILS FROM EMPLOYEE;  
+-----+-----+-----+-----+-----+  
| Field | Type | Null | Key | Default | Extra |  
+-----+-----+-----+-----+-----+  
| E_ID | int(3) | YES | | NULL | |  
| BASIC_SALARY | decimal(12,2) | YES | | NULL | |  
| HRA | decimal(7,2) | YES | | NULL | |  
| TA | decimal(7,2) | YES | | NULL | |  
| GROSS_SAL | decimal(15,2) | YES | | NULL | |  
+-----+-----+-----+-----+-----+  
5 rows in set (0.01 sec)  
  
mysql>
```

Figure 6.7: Columns of SALARY_DETAILS with database specified

Session 6

Implementing SQL Queries Using MySQL - I

Figure 6.7 displays the columns present in the `SALARY_DETAILS` table along with the information about each column. The first `FROM` keyword specifies the table to use for retrieving data. The second `FROM` keyword is used because you have not selected the database earlier with the `USE` command. Here, it specifies `EMPLOYEE` as the database in which the table `SALARY_DETAILS` is present.

Similarly, to view columns that start with the letter ‘H’ of the `SALARY_DETAILS` table from the `EMPLOYEE` database, enter the following command at the command prompt:

```
SHOW COLUMNS FROM SALARY_DETAILS LIKE 'H%';
```

Figure 6.8 displays the output of the command.

```
root@localhost:~  
File Edit View Terminal Tabs Help  
+-----+-----+-----+-----+  
5 rows in set (0.00 sec)  
  
mysql> SHOW COLUMNS FROM SALARY_DETAILS FROM EMPLOYEE;  
+-----+-----+-----+-----+  
| Field | Type | Null | Key | Default | Extra |  
+-----+-----+-----+-----+  
| E_ID | int(3) | YES | | NULL | |  
| BASIC_SALARY | decimal(12,2) | YES | | NULL | |  
| HRA | decimal(7,2) | YES | | NULL | |  
| TA | decimal(7,2) | YES | | NULL | |  
| GROSS_SAL | decimal(15,2) | YES | | NULL | |  
+-----+-----+-----+-----+  
5 rows in set (0.01 sec)  
  
mysql> SHOW COLUMNS FROM SALARY_DETAILS LIKE 'H%';  
+-----+-----+-----+-----+  
| Field | Type | Null | Key | Default | Extra |  
+-----+-----+-----+-----+  
| HRA | decimal(7,2) | YES | | NULL | |  
+-----+-----+-----+-----+  
1 row in set (0.00 sec)  
  
mysql>
```

Figure 6.8: Columns of `SALARY_DETAILS` starting with letter ‘H’

Figure 6.8 displays the columns present in the `SALARY_DETAILS` table starting with letter ‘H’. You can use the `LIKE` keyword to display only those columns that start with letter ‘H’.

A database index is a data structure that speeds up retrieval operations on a table. Indexes can be created

Session 6

Implementing SQL Queries Using MySQL - I

using one or more columns of a database table. They enable faster searches and efficient organization of data.

To view the index in a table from a database, use the following syntax:

```
SHOW INDEX FROM table_name [FROM database_name];
```

where,

SHOW INDEX - displays the index information of the table specified in the `table_name` clause of the command

`FROM table_name` – specifies the name of the table to retrieve the index

`FROM database_name` - specifies the name of the database where the table exists

The `SHOW INDEX` command displays index information as shown in table 6.2.

Column	Description
Table	Displays name of the table
Non_unique	Displays 0 if the index cannot contain duplicate value
Key_name	Displays index name
Seq_in_index	Displays sequence number of columns in index, starting with 1
Column_name	Displays column name
Collation	Displays the sorting order of columns in the index
Cardinality	Displays number of unique values in the index
Sub_part	Displays number of indexed characters when the column is partly indexed

Table 6.2: Indexes on a Table

For example, to view the index in the `EMP_DETAILS` table of the `EMPLOYEE` database, enter the following command at the command prompt:

```
SHOW INDEX FROM EMP_DETAILS FROM EMPLOYEE;
```

Figure 6.9 displays the output of the command.

Session 6

Implementing SQL Queries Using MySQL - I

Concepts

```
mysql> SHOW INDEX FROM EMP_DETAILS FROM EMPLOYEE;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Table      | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null |
| Index_type | Comment    | Index_comment |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| EMP_DETAILS |          0 | PRIMARY   |           1 | E_ID       |          |          |          |        |        |
|             | A           |           6 |           NULL | NULL      |          |          |          |        |        |
| BTREE       |          |           |           |           |          |          |          |        |        |
| EMP_DETAILS |          1 | E_ID      |           1 | E_ID       |          |          |          |        |        |
|             | A           |           6 |           NULL | NULL      |          |          |          |        |        |
| BTREE       |          |           |           |           |          |          |          |        |        |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql>
```

Figure 6.9: Index Columns of EMP_DETAILS

Alternatively, you can use the following syntax:

```
SHOW INDEX FROM database_name.table_name;
```

where,

SHOW INDEX – displays the table index

database_name – specifies the name of the database that contains the table

Session 6

Implementing SQL Queries Using MySQL - I

`table_name` – specifies the name of the table for which you want to display the index

For example, to display index keys of `EMP_DETAILS` from `EMPLOYEE` database, enter the following command at the command prompt:

```
SHOW INDEX FROM EMPLOYEE.EMP_DETAILS;
```

This command is more compact and concise.

To view the server status, use the following syntax:

```
SHOW STATUS;
```

where,

`SHOW` – displays the object specified in the clause

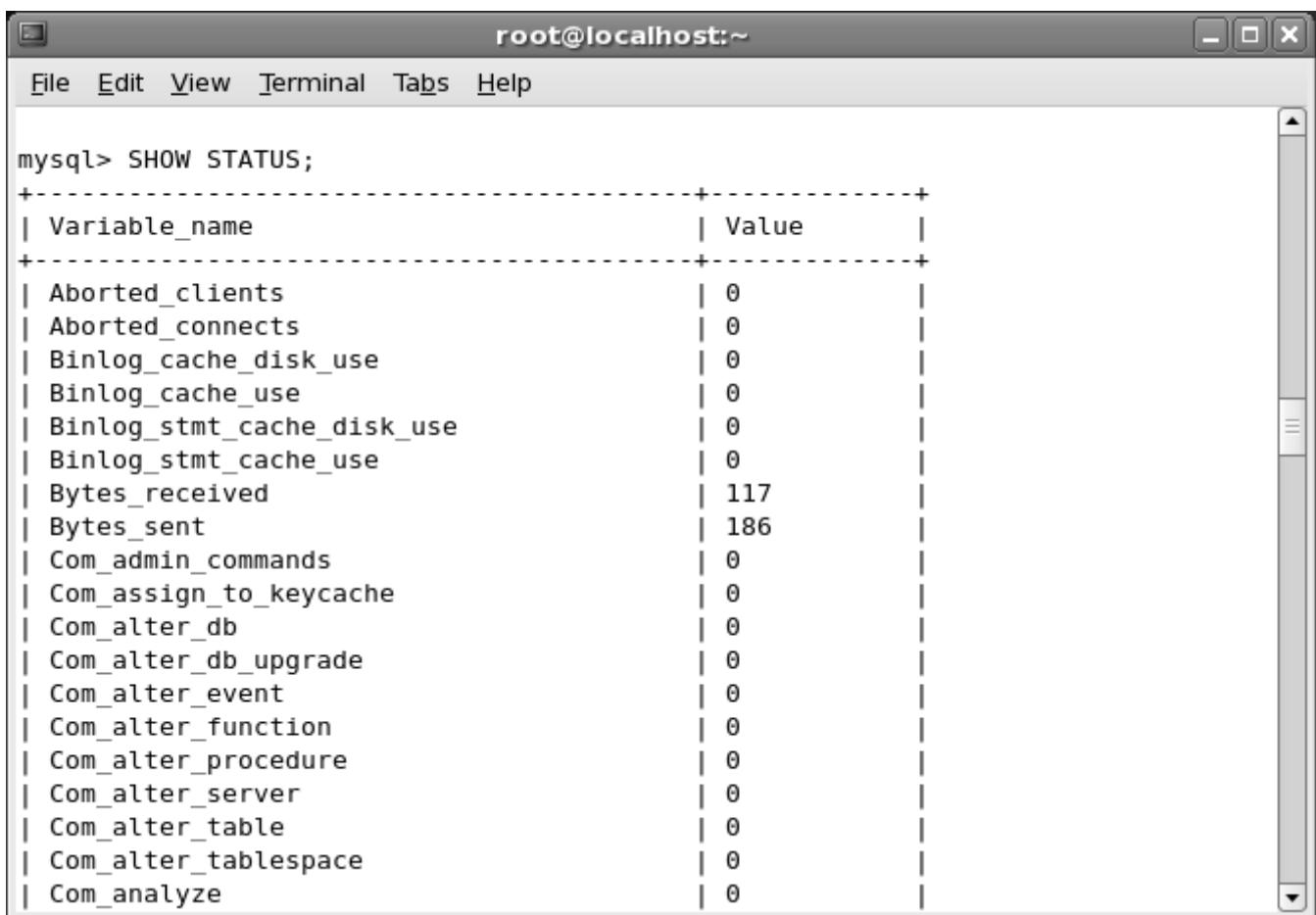
`STATUS` – displays information about the server

Figure 6.10 displays the output of the command.

Session 6

Implementing SQL Queries Using MySQL - I

Concepts



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The main area displays the output of the MySQL command "SHOW STATUS;". The output is a table with two columns: "Variable_name" and "Value". The table contains numerous system variables, many of which have a value of 0. Some non-zero values include "Bytes_received" at 117 and "Bytes_sent" at 186.

Variable_name	Value
Aborted_clients	0
Aborted_connects	0
Binlog_cache_disk_use	0
Binlog_cache_use	0
Binlog_stmt_cache_disk_use	0
Binlog_stmt_cache_use	0
Bytes_received	117
Bytes_sent	186
Com_admin_commands	0
Com_assign_to_keycache	0
Com.Alter_db	0
Com.Alter_db_upgrade	0
Com.Alter_event	0
Com.Alter_function	0
Com.Alter_procedure	0
Com.Alter_server	0
Com.Alter_table	0
Com.Alter_tablespace	0
Com.Analyze	0

Figure 6.10: Server Status

Note: The numbers and format may be different in various servers.

To view the values of the system variables, use the following syntax:

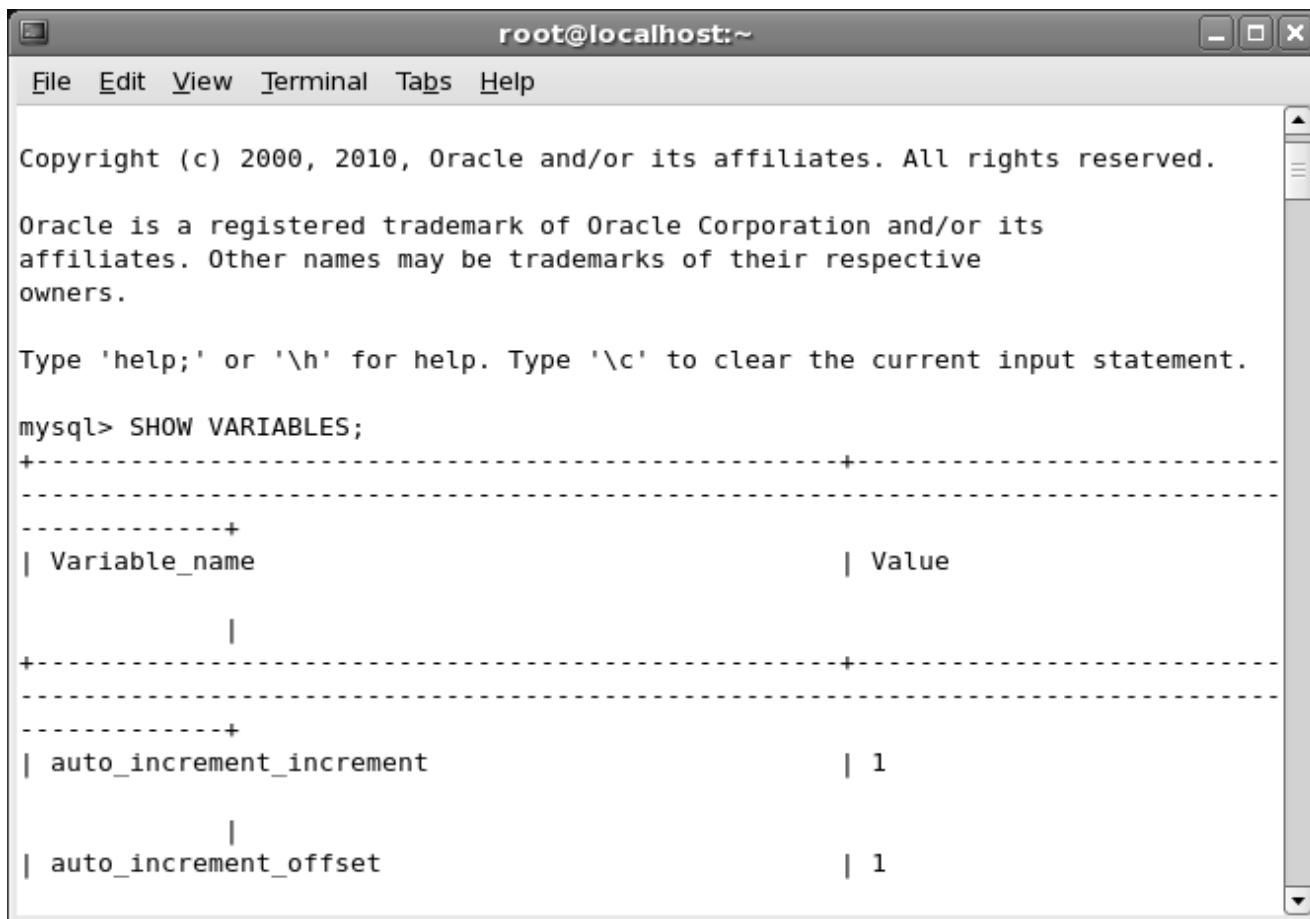
```
SHOW VARIABLES;
```

Figure 6.11 displays the output of the command.

Session 6

Implementing SQL Queries Using MySQL - I

Concepts



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following text:

```
Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.  
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
mysql> SHOW VARIABLES;  
+-----+-----+  
| Variable_name | Value |  
+-----+-----+  
| auto_increment_increment | 1 |  
+-----+-----+  
| auto_increment_offset | 1 |
```

Figure 6.11: System Variables

You can change the default values of the variables using the command-line options when MySQL starts. The `SET` statement can also be used to edit the default values at runtime.

Note: The list of names and values may be different for various servers.

You can also view the system variable information using the `mysqladmin` command.

Similarly, to view only the variables that match a specified condition use the following syntax:

```
SHOW VARIABLES [LIKE <condition>];
```

For example, to display the variables that start with 'HAVE', enter the following command at the command prompt:

```
SHOW VARIABLES LIKE 'HAVE%';
```

Session 6

Implementing SQL Queries Using MySQL - I

Figure 6.12 displays the output of the command.

```
root@localhost:~
```

```
File Edit View Terminal Tabs Help
```

```
318 rows in set (0.00 sec)
```

```
mysql> SHOW VARIABLES LIKE 'HAVE%';
+-----+-----+
| Variable_name      | Value   |
+-----+-----+
| have_compress      | YES     |
| have_crypt         | YES     |
| have_csv           | YES     |
| have_dynamic_loading | YES    |
| have_geometry      | YES     |
| have_innodb         | YES     |
| have_ndbcluster    | NO      |
| have_openssl        | DISABLED |
| have_partitioning   | YES     |
| have_profiling      | YES     |
| have_query_cache    | YES     |
| have_rtree_keys     | YES     |
| have_ssl            | DISABLED |
| have_symlink        | YES     |
+-----+-----+
14 rows in set (0.00 sec)

mysql>
```

Figure 6.12: Server Variables starting with 'HAVE'

To view the running threads or processes, use the following syntax:

```
SHOW [FULL] PROCESSLIST;
```

You must have SUPER privileges to view all the threads. The user accounts that do not have SUPER privileges can view only their threads.

Note: If the [FULL] option is not specified, MySQL will only display the first 100 characters of each process.

The SHOW TABLE STATUS command is similar to the SHOW TABLE command except that it displays more information about each table. The syntax for SHOW TABLE STATUS command is as follows:

Session 6

Implementing SQL Queries Using MySQL - I

```
SHOW TABLE STATUS [FROM database_name] [LIKE <condition>];
```

where,

SHOW TABLE STATUS – displays information about tables in the database

FROM database_name – enables you to specify the database to select tables. You can use the FROM clause if you have not activated the database with the USE command.

LIKE <condition> – specifies one or more conditions to be satisfied before displaying the table information

Table 6.3 lists the information returned by the SHOW TABLE STATUS command.

Column	Description
Name	Displays the name of the table
Type	Displays type of table
Row_format	Displays storage format of the row (such as fixed, dynamic, or compressed)
Rows	Displays number of rows
Avg_row_length	Displays average row length
Data_length	Displays length of the data file
Max_data_length	Displays max length of the data file
Index_length	Displays length of the index file
Data_free	Displays number of unused allocated bytes
Auto_increment	Displays next auto increment value
Create_time	Displays when the table was created
Update_time	Displays the date of last updation of data file
Check_time	Displays when the table was last checked
Comment	Displays the comments for a column. The comment can be 255 characters long.

Table 6.3: Information Returned by SHOW TABLE command

For example, to view the status of all the tables of the EMPLOYEE database, enter the following command at the command prompt:

```
SHOW TABLE STATUS FROM EMPLOYEE;
```

Figure 6.13 displays the output of the command.

Session 6

Implementing SQL Queries Using MySQL - I

```
root@localhost:~
```

File Edit View Terminal Tabs Help

```
mysql> SHOW TABLE STATUS FROM EMPLOYEE;
```

Name	Engine	Version	Row_format	Rows	Avg_row_length	Data_length	Max_data_length	Index_length	Data_free	Auto_increment	Create_time	Update_time	Check_time	Collation	Checksum	Create_options	Comment
EMP_DEPARTMENT	InnoDB	10	Compact	0	0	16384	0	0	10485760	NULL	2011-03-03 16:51:43	NULL	NULL	latin1_swedish_ci	NULL		
EMP_DETAILS	InnoDB	10	Compact	0	0	16384	0	0	10485760	NULL	2011-03-03 16:48:12	NULL	NULL	latin1_swedish_ci	NULL		
SALARY_DETAILS	InnoDB	10	Compact	0	0	16384	0	0	10485760	NULL	2011-03-03 16:48:12	NULL	NULL	latin1_swedish_ci	NULL		

Figure 6.13: Status of Tables in EMPLOYEE Database

To view a list of grants that are assigned for a user, use the following syntax:

```
SHOW GRANTS FOR user;
```

where,

SHOW – displays information specified in the clause

GRANTS – displays privileges or account rights

FOR – specifies the object to display the privileges

user – specifies the type of object for which privileges are to be displayed

Session 6

Implementing SQL Queries Using MySQL - I

For example, to display the rights granted to the root user, enter the following command at the command prompt:

```
SHOW GRANTS FOR root@localhost;
```

Concepts

Figure 6.14 displays the output of the command. The output shows the list of grants for the root user.

The screenshot shows a terminal window with the title bar 'root@localhost:~'. The menu bar includes 'File', 'Edit', 'View', 'Terminal', 'Tabs', and 'Help'. The main pane displays the following MySQL command and its output:

```
mysql> SHOW GRANTS FOR root@localhost;
+-----+
| Grants for root@localhost |
+-----+
| GRANT ALL PRIVILEGES ON *.* TO 'root'@'localhost' WITH GRANT OPTION |
| GRANT PROXY ON ''@'' TO 'root'@'localhost' WITH GRANT OPTION |
+-----+
2 rows in set (0.00 sec)

mysql>
```

Figure 6.14: Rights granted to root user

6.2.2 Altering Database on the File System

Alteration of a database involves making changes to the database. You can make changes to the data in the database, as well as the database structure. MySQL provides the `ALTER DATABASE` command to modify the global characteristics or attributes of a database stored in the `db.opt` file of the database directory. You will use the `CHARACTER SET` clause to modify the default database character set. In addition, you will use the `COLLATE` clause to modify the default database collation. A collation in MySQL database is a set of rules used in data comparisons.

The syntax for modifying the character set of a database is:

```
ALTER DATABASE database_name DEFAULT CHARACTER SET charset_name;
```

where,

`ALTER DATABASE` – edits the database

`database_name` – specifies the name of the database on which you need to make changes

Session 6

Implementing SQL Queries Using MySQL - I

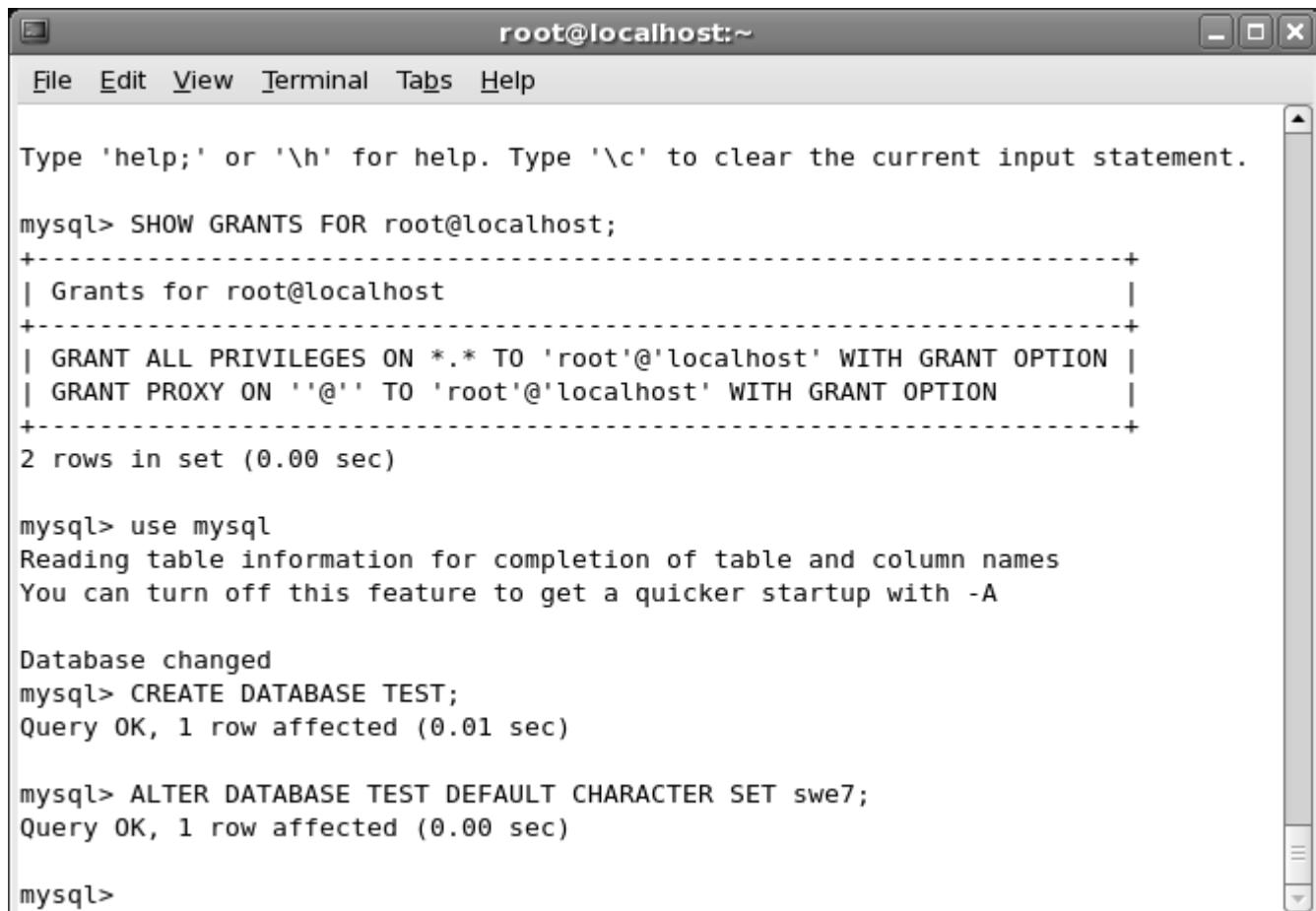
DEFAULT CHARACTER SET – specifies the default character set for the database

charset_name – specifies the type of character set for the database

For example, to modify the character set of the database, create a temporary database named TEST and then alter the database by entering the following command at the command prompt:

```
ALTER DATABASE TEST DEFAULT CHARACTER SET swe7;
```

Figure 6.15 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL session:

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

mysql> SHOW GRANTS FOR root@localhost;
+-----+-----+
| Grants for root@localhost          |
+-----+-----+
| GRANT ALL PRIVILEGES ON *.* TO 'root'@'localhost' WITH GRANT OPTION |
| GRANT PROXY ON ''@'' TO 'root'@'localhost' WITH GRANT OPTION          |
+-----+
2 rows in set (0.00 sec)

mysql> use mysql
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> CREATE DATABASE TEST;
Query OK, 1 row affected (0.01 sec)

mysql> ALTER DATABASE TEST DEFAULT CHARACTER SET swe7;
Query OK, 1 row affected (0.00 sec)

mysql>
```

Figure 6.15: ALTER Database

The syntax to modify the collation of a database is:

```
ALTER DATABASE database_name DEFAULT COLLATE collation_name;
```

Session 6

Implementing SQL Queries Using MySQL - I

Concepts

Note: You must have ALTER privileges to use the ALTER DATABASE command.

6.3 Retrieving Data Using SELECT, FROM, DISTINCT, and WHERE Clauses

A table contains structured data in the form of rows and columns. The table must contain data to display. You cannot view the contents of a table if it does not contain data. Therefore, you must insert data before retrieving data from tables. MySQL provides the INSERT command to add data to the table. The syntax for INSERT command is:

```
INSERT INTO table_name {VALUES | VALUE} (value1, value2...);
```

where,

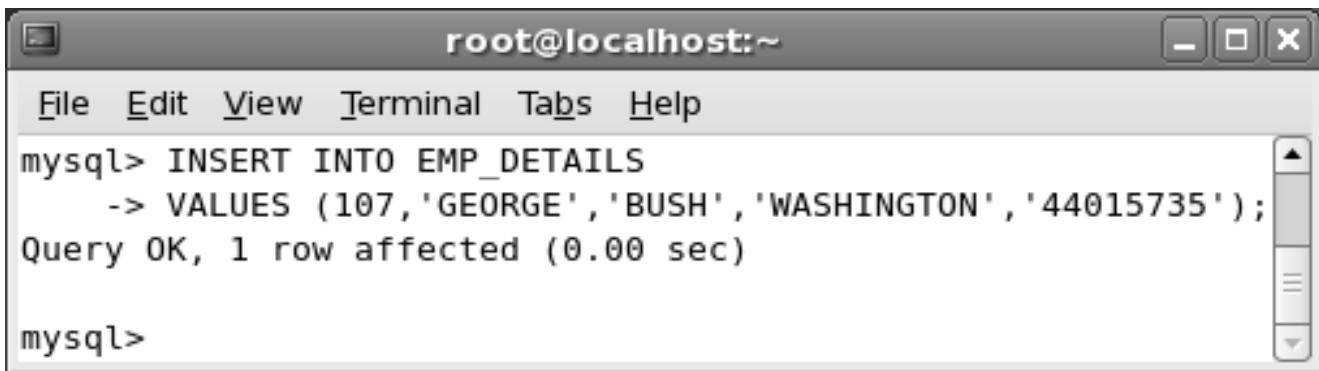
INSERT INTO – adds a new record to the table

table_name – specifies the name of the table to add the record

{VALUES | VALUE} – any one of these two may be used to specify the values and either may be used for a single values list or multiple lists

value1 – specifies the data that will be added to the column

Figure 6.16 displays the insertion of records in the table.



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following text:

```
File Edit View Terminal Tabs Help
mysql> INSERT INTO EMP_DETAILS
-> VALUES (107, 'GEORGE', 'BUSH', 'WASHINGTON', '44015735');
Query OK, 1 row affected (0.00 sec)

mysql>
```

Figure 6.16: INSERT Values

Figure 6.16 displays the method of inserting values in the table. The character value is specified with single quotation mark before and after the string.

You can use the SELECT command to retrieve data from one or more tables. The SELECT command has two clauses:

Session 6

Implementing SQL Queries Using MySQL - I

- `FROM` - specifies the table name whose records are to be retrieved.
- `WHERE` - specifies the condition, based on which the records are retrieved. This clause is optional.

The syntax for retrieving all the records of a table is:

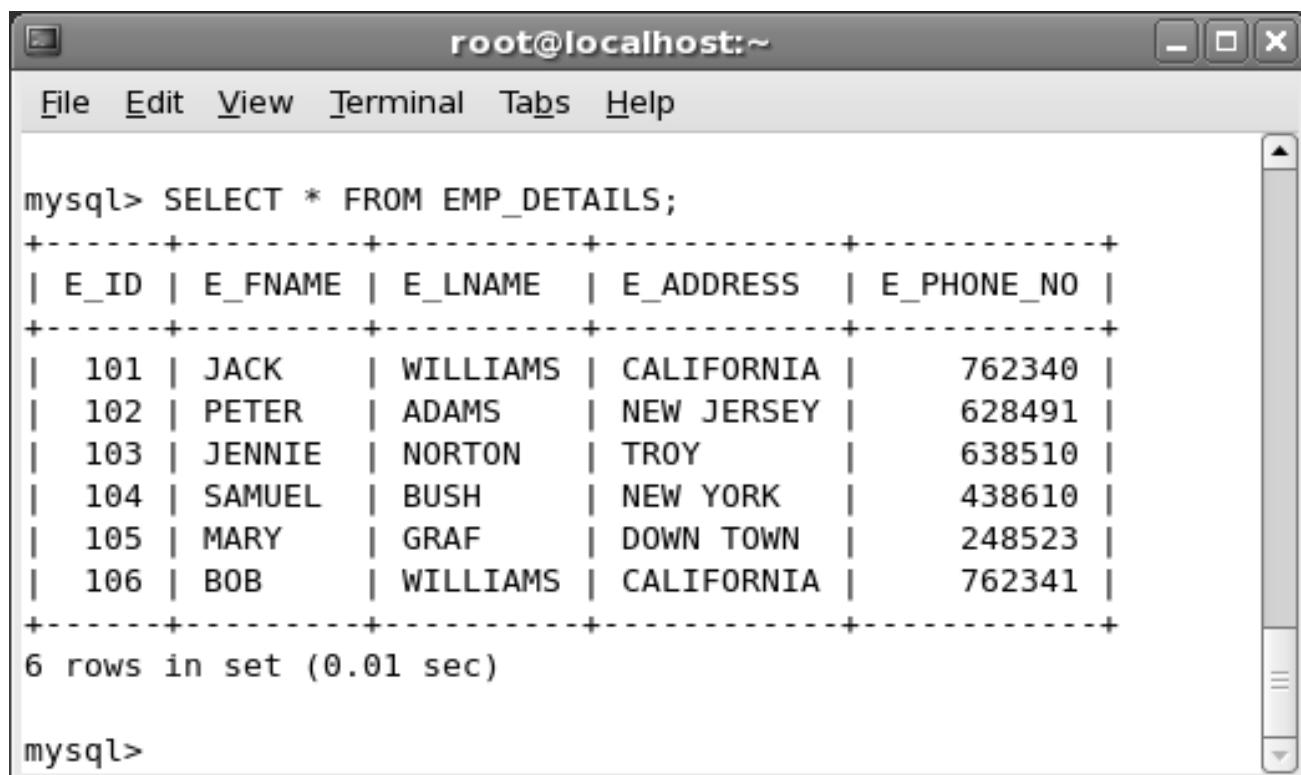
```
SELECT [*] FROM table_name;
```

where, '*' displays all the columns of the specified table.

For example, to view all records of the `EMP_DETAILS` table, enter the following command at the command prompt:

```
SELECT * FROM EMP_DETAILS;
```

Figure 6.17 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The main area displays the following MySQL command and its output:

```
mysql> SELECT * FROM EMP_DETAILS;
+-----+-----+-----+-----+-----+
| E_ID | E_FNAME | E_LNAME | E_ADDRESS | E_PHONE_NO |
+-----+-----+-----+-----+-----+
| 101  | JACK    | WILLIAMS | CALIFORNIA | 762340   |
| 102  | PETER   | ADAMS    | NEW JERSEY  | 628491   |
| 103  | JENNIE  | NORTON   | TROY        | 638510   |
| 104  | SAMUEL  | BUSH     | NEW YORK   | 438610   |
| 105  | MARY    | GRAF     | DOWN TOWN  | 248523   |
| 106  | BOB     | WILLIAMS | CALIFORNIA | 762341   |
+-----+-----+-----+-----+-----+
6 rows in set (0.01 sec)

mysql>
```

The output shows six rows of data from the `EMP_DETAILS` table, with columns labeled `E_ID`, `E_FNAME`, `E_LNAME`, `E_ADDRESS`, and `E_PHONE_NO`. The data includes names like JACK, PETER, JENNIE, SAMUEL, MARY, and BOB, along with their addresses and phone numbers.

Figure 6.17: Displaying all Records of the EMP_DETAILS

You can use the `DISTINCT` keyword in the `SELECT` command to display columns that do not contain duplicate data. The `DISTINCT` keyword displays only those rows containing unique values in the specified column.

Session 6

Implementing SQL Queries Using MySQL - I

To display unique information from columns using `DISTINCT` keyword with `SELECT` command, use the syntax:

```
SELECT DISTINCT column_name FROM table_name;
```

where,

`SELECT` – displays data from the table

`DISTINCT` – is the keyword that displays unique information from columns. MySQL does not display duplicate records when you use this clause

`column_name` – specifies the name of the column

`table_name` – specifies the name of the table where the column exists

This command will display the unique records from the columns specified in the `column_name` keyword.

For example, to display only the unique department names from the `EMP_DEPARTMENT` table, enter the following command at the command prompt:

```
SELECT DISTINCT D_NAME FROM EMP_DEPARTMENT;
```

Figure 6.18 displays the output of the command.

Session 6

Implementing SQL Queries Using MySQL - I

```
root@localhost:~
```

```
File Edit View Terminal Tabs Help
```

```
mysql> SELECT DISTINCT D_NAME FROM EMP_DEPARTMENT;
```

D_NAME
RESEARCH
MARKETING
DEVELOPMENT
SALES

```
4 rows in set (0.01 sec)
```

```
mysql>
```

Figure 6.18: Displaying Unique Department Names

The `SELECT` command also enables you to view specific columns of a table.

The syntax for viewing only specific columns of a table is:

```
SELECT column_name1, column_name2 FROM table_name;
```

where,

`SELECT` – displays the information

`column_name1` – specifies the name of the column to be retrieved

`column_name2` – specifies the name of the column to be retrieved

`table_name` – specifies the name of the table that contains the columns

This command will display only those columns that are specified in the `column_name` clause of the command.

For example, to view the `E_FNAME`, `E_LNAME`, and `E_ADDRESS` columns of the `EMP_DETAILS` table, enter the following command at the command prompt:

Session 6

Implementing SQL Queries Using MySQL - I

```
SELECT E_FNAME, E_LNAME, E_ADDRESS FROM EMP_DETAILS;
```

Figure 6.19 displays the output of the command.

The screenshot shows a terminal window titled "root@localhost:~". The window contains the following text:

```
mysql> SELECT E_FNAME,E_LNAME,E_ADDRESS
-> FROM EMP_DETAILS;
+-----+-----+-----+
| E_FNAME | E_LNAME | E_ADDRESS |
+-----+-----+-----+
| JACK    | WILLIAMS | CALIFORNIA |
| PETER   | ADAMS    | NEW JERSEY  |
| JENNIE  | NORTON   | TROY        |
| SAMUEL  | BUSH     | NEW YORK   |
| MARY    | GRAF     | DOWN TOWN  |
| BOB     | WILLIAMS | CALIFORNIA |
+-----+-----+-----+
6 rows in set (0.00 sec)

mysql>
```

The terminal window has a standard Linux-style interface with a menu bar (File, Edit, View, Terminal, Tabs, Help) and a scroll bar on the right side.

Concepts

Figure 6.19: Displaying Specific Columns

Figure 6.19 displays the first name, last name, and address of all employees from the `EMP_DETAILS` table.

You can also use the `SELECT` command to make calculations. These calculations do not require a reference to any table.

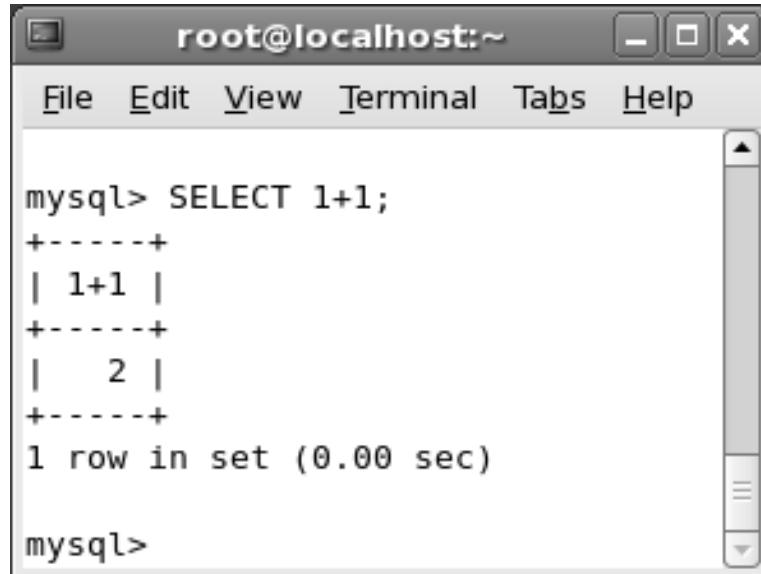
```
SELECT 1+1;
```

Figure 6.20 displays the output of the command.

Session 6

Implementing SQL Queries Using MySQL - I

Concepts



A screenshot of a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area displays the following MySQL command and its result:

```
mysql> SELECT 1+1;
+----+
| 1+1 |
+----+
|    2 |
+----+
1 row in set (0.00 sec)

mysql>
```

Figure 6.20: Computed Result

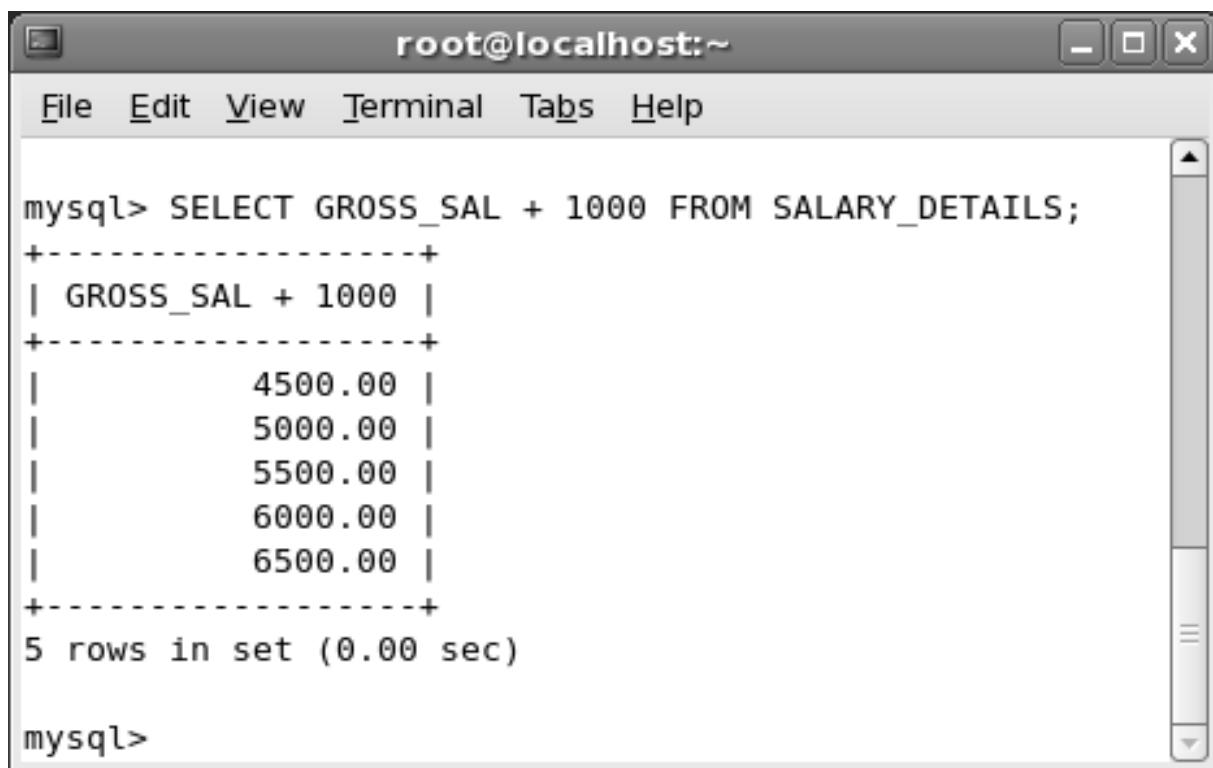
To perform arithmetic operations using the `SELECT` command, consider the following example:

```
SELECT GROSS_SAL + 1000 FROM SALARY_DETAILS;
```

Figure 6.21 displays the output of the command.

Session 6

Implementing SQL Queries Using MySQL - I



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its output:

```
mysql> SELECT GROSS_SAL + 1000 FROM SALARY_DETAILS;
+-----+
| GROSS_SAL + 1000 |
+-----+
|      4500.00 |
|      5000.00 |
|      5500.00 |
|      6000.00 |
|      6500.00 |
+-----+
5 rows in set (0.00 sec)

mysql>
```

A vertical scrollbar is visible on the right side of the terminal window.

Figure 6.21: Computed Gross Salary of Employees

Figure 6.21 displays the calculated sum of gross salary and 1000 in the `GROSS_SAL` column for each of the rows of the table.

6.3.1 SELECT command Using WHERE Clause

The `WHERE` clause contains one or more conditions that must be satisfied before the query retrieves a row. It searches for data according to the condition specified in the query and narrows the selection of data. The `WHERE` clause uses logical and conditional operators in the query.

The syntax to specify a condition using the `WHERE` clause is:

```
SELECT * FROM table_name WHERE <condition to satisfy>;
```

where,

`SELECT` – retrieves the data

`*` – specifies to retrieve all the columns of the table

Session 6

Implementing SQL Queries Using MySQL - I

table_name – specifies the name of the table

WHERE – specifies the clause to filter data

<condition to satisfy> – contains conditions to satisfy before retrieving and displaying data

For example, to view the records of EMP_SALARY table, where the basic salary is greater than 2500, enter the following command at the command prompt:

```
SELECT * FROM EMP_SALARY WHERE BASIC_SALARY > 2500;
```

Figure 6.22 displays the output of the command.

```
root@localhost:~  
File Edit View Terminal Tabs Help  
mysql> SELECT * FROM EMP_SALARY  
-> WHERE BASIC_SALARY > 2500;  
+-----+-----+-----+-----+-----+-----+  
| E_ID | BASIC_SALARY | HRA | TA | GROSS_SAL | D_NAME |  
+-----+-----+-----+-----+-----+-----+  
| 103 | 3000.00 | 1000.00 | 500.00 | 4500.00 | DEVELOPMEN |  
| 104 | 3500.00 | 1000.00 | 500.00 | 5000.00 | SALES |  
| 105 | 4000.00 | 1000.00 | 500.00 | 5500.00 | SALES |  
+-----+-----+-----+-----+-----+-----+  
3 rows in set (0.01 sec)  
  
mysql>
```

Figure 6.22: Displaying Records Using WHERE Clause

Similarly, to view the records of the EMP_DETAILS table where the last name is NORTON, enter the following command at the command prompt:

```
SELECT * FROM EMP_DETAILS WHERE E_LNAME = 'NORTON';
```

Figure 6.23 displays the output of the command.

Session 6

Implementing SQL Queries Using MySQL - I

Concepts

```
root@localhost:~  
File Edit View Terminal Tabs Help  
  
mysql> SELECT * FROM EMP_DETAILS WHERE E_LNAME = 'NORTON';  
+-----+-----+-----+-----+  
| E_ID | E_FNAME | E_LNAME | E_ADDRESS | E_PHONE_NO |  
+-----+-----+-----+-----+  
| 103 | JENNIE | NORTON | TROY | 638510 |  
+-----+-----+-----+-----+  
1 row in set (0.00 sec)  
  
mysql>
```

Figure 6.23: Displaying Records Using WHERE Clause With Text

Figure 6.23 displays all records from `EMP_DETAILS` table that have the last name as 'NORTON'.

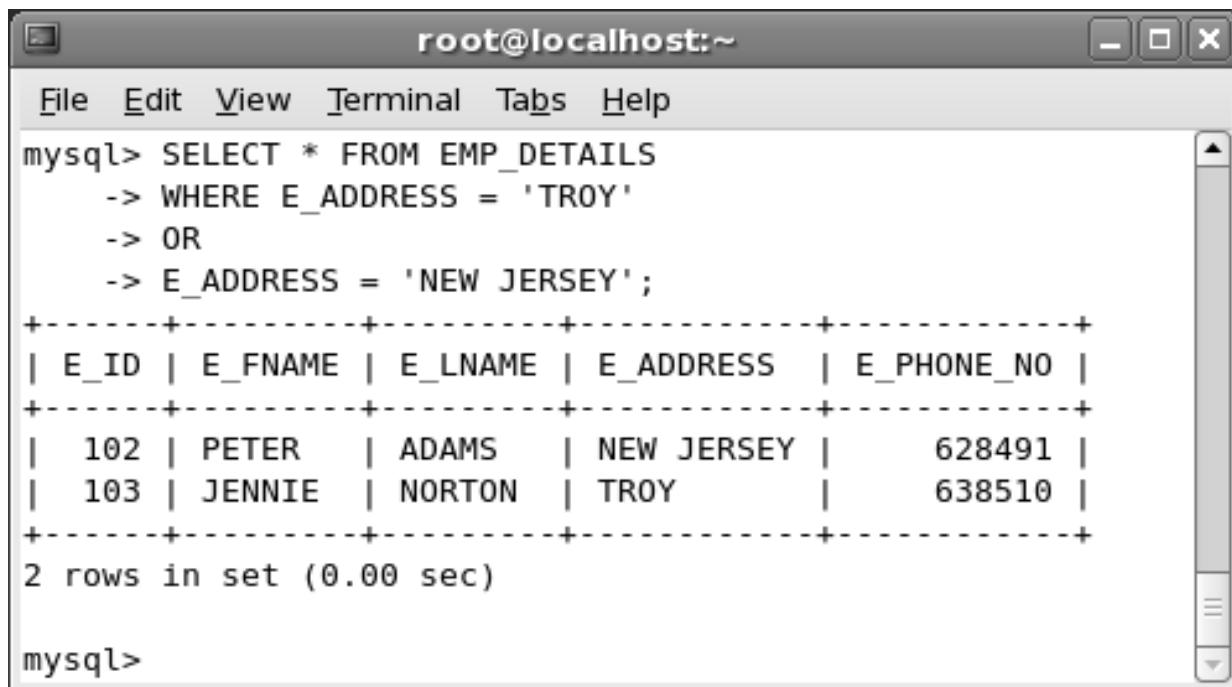
To view all the fields of the `EMP_DETAILS` table, where the address of the employee is `TROY` or the address of employee is `NEW JERSEY`, enter the following command at the command prompt:

```
SELECT * FROM EMP_DETAILS WHERE E_ADDRESS = 'TROY' OR E_ADDRESS = 'NEW JERSEY';
```

Figure 6.24 displays the output of the command.

Session 6

Implementing SQL Queries Using MySQL - I



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its execution:

```
mysql> SELECT * FROM EMP_DETAILS
-> WHERE E_ADDRESS = 'TROY'
-> OR
-> E_ADDRESS = 'NEW JERSEY';
+-----+-----+-----+-----+-----+
| E_ID | E_FNAME | E_LNAME | E_ADDRESS | E_PHONE_NO |
+-----+-----+-----+-----+-----+
| 102  | PETER   | ADAMS   | NEW JERSEY | 628491    |
| 103  | JENNIE  | NORTON  | TROY      | 638510    |
+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql>
```

The output shows two rows from the EMP_DETAILS table where either the address is 'TROY' or 'NEW JERSEY'. The columns are E_ID, E_FNAME, E_LNAME, E_ADDRESS, and E_PHONE_NO.

Figure 6.24: Displaying Records Using WHERE and OR Clause

6.4 Modifying Table Definitions Using ALTER Command

MySQL provides the `ALTER TABLE` command to modify the structure of a table. You can add or delete columns, rename columns or the table, create or destroy indexes, and modify the column type.

The `ALTER TABLE` command creates a temporary copy of the original table, on which the alteration is performed. MySQL alters the temporary copy of the table, deletes the original copy, and renames the temporary copy of the table. MySQL uses this feature as a security measure to prevent data loss in case the table modification fails.

The syntax for altering a table is:

```
ALTER [IGNORE] TABLE table_name alter_spec [, alter_spec...];
```

where,

`ALTER` – specifies to edit the object

`IGNORE` – checks for duplicate records on the index keys in the new table. If MySQL identifies a duplicate record on the unique keys, it returns only the first row and deletes the duplicates

`TABLE` – specifies the type of object to edit

Session 6

Implementing SQL Queries Using MySQL - I

table_name – specifies the name of the table

alter_spec – contains modification information

Note: If the [IGNORE] clause is not specified, MySQL aborts the copy operation if the index column contains duplicate records.

The syntax to add a column in the existing table is:

```
ALTER TABLE table_name ADD [COLUMN] create_definition [FIRST | AFTER column_name];
```

where,

table_name – specifies the name of the table to modify

ADD [COLUMN] – appends a new column to the table

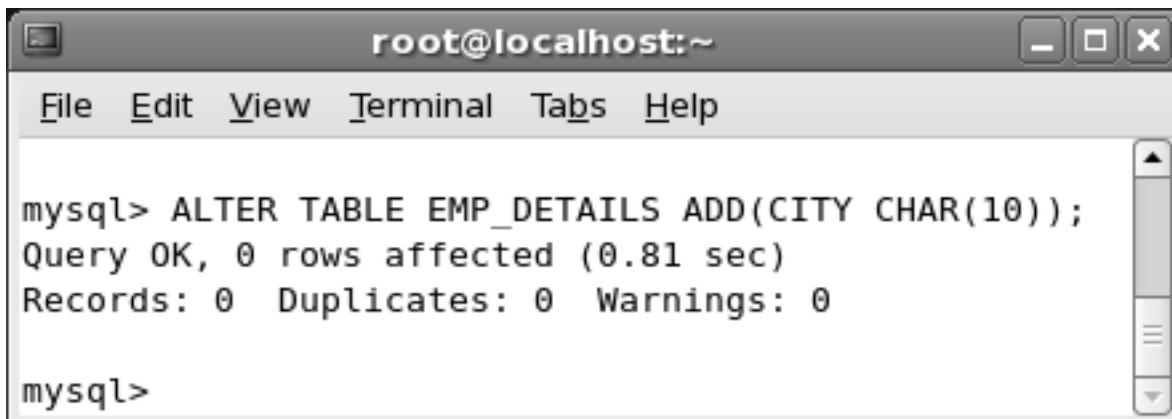
create_definition – defines the column type in a table creation

FIRST | AFTER column_name – specifies the location of the new column in the table

For example, to add a column CITY to the EMP_DETAILS table for entering the name of the city of employees, enter the following command at the command prompt:

```
ALTER TABLE EMP_DETAILS ADD (CITY CHAR (10));
```

Figure 6.25 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The main area of the terminal shows the following MySQL command and its execution:

```
mysql> ALTER TABLE EMP_DETAILS ADD(CITY CHAR(10));
Query OK, 0 rows affected (0.81 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
```

Figure 6.25: ALTER TABLE Command

Session 6

Implementing SQL Queries Using MySQL - I

The syntax to add an index key to a column of a table is:

```
ALTER TABLE table_name ADD INDEX [index_name] (index_column_name...);
```

where,

ALTER TABLE – specifies to modify the table

table_name – specifies the name of the table

ADD INDEX – appends an index to the table

index_name – specifies a name for the index

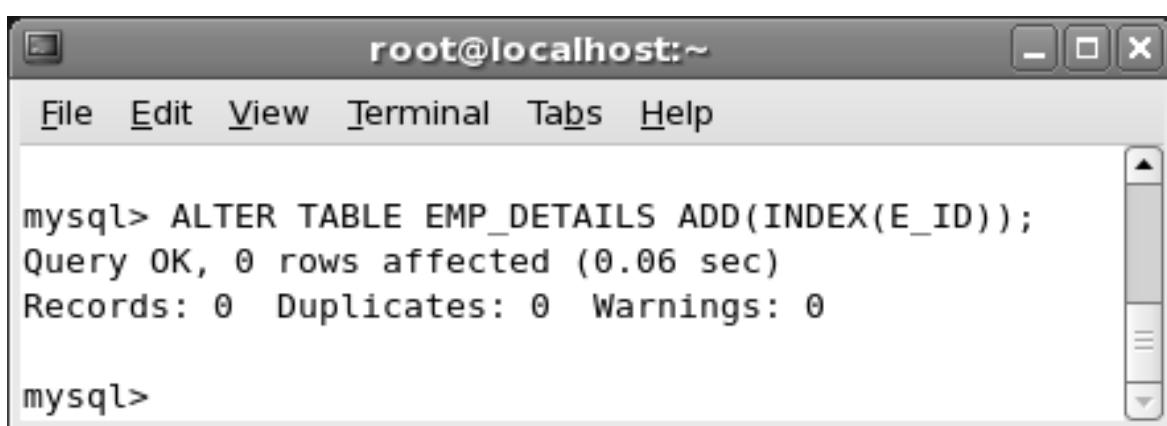
index_column_name – specifies the column in the table to index

This command adds an index to the table and indexes the specified column.

For example, to add an index on column E_ID on EMP_DETAILS table, enter the following command at the command prompt:

```
ALTER TABLE EMP_DETAILS ADD (INDEX (E_ID));
```

Figure 6.26 displays the output of the command.



A screenshot of a terminal window titled "root@localhost:~". The window has a standard Linux-style interface with a menu bar (File, Edit, View, Terminal, Tabs, Help) and a scroll bar on the right. The main area contains the following text:

```
mysql> ALTER TABLE EMP_DETAILS ADD(INDEX(E_ID));
Query OK, 0 rows affected (0.06 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
```

Figure 6.26: Adding an Index Key

Session 6

Implementing SQL Queries Using MySQL - I

The syntax to add a primary key to the column of a table is:

```
ALTER TABLE table_name ADD PRIMARY KEY (index_column_name...);
```

where,

ALTER TABLE – modifies the table structure

table_name – specifies the name of the table

ADD PRIMARY KEY – appends a primary key to the table

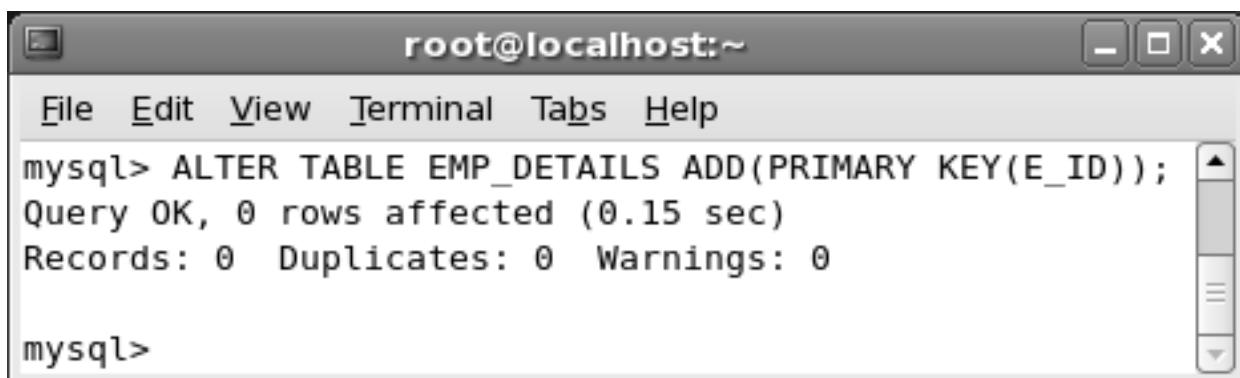
index_column_name – specifies the name of the column to use as an index for the primary key

This command adds a primary key to the specified indexed column in the table.

For example, to add a primary key on column E_ID on EMP_DETAILS table, enter the following command at the command prompt:

```
ALTER TABLE EMP_DETAILS ADD (PRIMARY KEY (E_ID));
```

Figure 6.27 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following text:

```
File Edit View Terminal Tabs Help
mysql> ALTER TABLE EMP_DETAILS ADD(PRIMARY KEY(E_ID));
Query OK, 0 rows affected (0.15 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
```

Figure 6.27: Adding Primary Key

To modify a column definition, use the following syntax:

```
ALTER TABLE table_name MODIFY [COLUMN] create_definition;
```

where,

ALTER TABLE – specifies to edit the table

Session 6

Implementing SQL Queries Using MySQL - I

table_name – specifies the name of the table

MODIFY [COLUMN] – specifies to change the column structure

create_definition – specifies the new rules for the column

For example, to modify DESIGNATION column from CHAR(50) to CHAR(20) data type of EMP_DEPARTMENT table, enter the following command at the command prompt:

```
ALTER TABLE EMP_DEPARTMENT MODIFY DESIGNATION CHAR(20);
```

Figure 6.28 displays the output of the command.

The screenshot shows a terminal window titled "root@localhost:~". The window has a standard Linux-style title bar with icons for minimize, maximize, and close. Below the title bar is a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The main area of the terminal contains the following MySQL session:

```
Database changed
mysql> ALTER TABLE EMP_DETAILS ADD(PRIMARY KEY(E_ID));
Query OK, 0 rows affected (0.15 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> ALTER TABLE EMP_DEPARTMENT MODIFY DESIGNATION CHAR(20);
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
```

Figure 6.28: Modifying a Designation Column

To drop a column from a table, use the following syntax:

```
ALTER TABLE table_name DROP [COLUMN] column_name;
```

where,

ALTER TABLE – modifies the table structure

table_name – specifies the name of the table to modify

Session 6

Implementing SQL Queries Using MySQL - I

DROP [COLUMN] – removes the column from the table

column_name – specifies the name of the column to remove from the table

This command removes a column from the table.

For example, to remove the column CITY from EMP_DETAILS table, enter the following command at the command prompt:

```
ALTER TABLE EMP_DETAILS DROP COLUMN CITY;
```

Figure 6.29 displays the output of the command.

The screenshot shows a terminal window titled "root@localhost:~". The window contains the following text:

```
root@localhost:~  
File Edit View Terminal Tabs Help  
Records: 0 Duplicates: 0 Warnings: 0  
mysql> ALTER TABLE EMP_DEPARTMENT MODIFY DESIGNATION CHAR(20);  
Query OK, 0 rows affected (0.01 sec)  
Records: 0 Duplicates: 0 Warnings: 0  
  
mysql> ALTER TABLE EMP_DETAILS DROP COLUMN CITY;  
Query OK, 0 rows affected (0.00 sec)  
Records: 0 Duplicates: 0 Warnings: 0  
  
mysql>
```

Figure 6.29: Deleting a Column

To drop a primary key of a column from the table, use the following syntax:

```
ALTER TABLE table_name DROP PRIMARY KEY;
```

where,

ALTER TABLE – edits the table structure

table_name – specifies the name of the table to modify

DROP PRIMARY KEY – removes the primary key from the table

Session 6

Implementing SQL Queries Using MySQL - I

Note: The `DROP PRIMARY KEY` option deletes the primary index. MySQL will delete the first indexed column, if a primary index is not defined in the table.

For example, to remove primary key constraint from `EMP_DETAILS` table, enter the following command at the command prompt:

```
ALTER TABLE EMP_DETAILS DROP PRIMARY KEY;
```

To change the name of a table, use the following syntax:

```
ALTER TABLE table_name RENAME new_table_name;
```

where,

`ALTER TABLE` – edits the table structure

`table_name` – specifies the name of the table to modify

`RENAME` – changes the name of the table

`new_table_name` – specifies the new name for the table

For example, to rename the table from `SALARY_DETAILS` to `EMP_SALARY`, enter the following command at the command prompt:

```
ALTER TABLE SALARY_DETAILS RENAME EMP_SALARY;
```

Figure 6.30 displays the output of the command.

Session 6

Implementing SQL Queries Using MySQL - I

Concepts

The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL session:

```
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> ALTER TABLE EMP_DETAILS DROP COLUMN CITY;
Query OK, 0 rows affected (0.00 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> ALTER TABLE SALARY_DETAILS RENAME EMP_SALARY;
Query OK, 0 rows affected (0.01 sec)

mysql>
```

Figure 6.30: Renaming a Table

Following are some of the points to be remembered while working with `ALTER TABLE` command:

- MySQL deletes the index values of columns when you drop columns from a table. MySQL also removes the index when you drop or delete all indexed columns.
- If `DESCRIBE table_name` displays that the column specification, modified using the `ALTER TABLE` command, has not changed, then it is possible that the modification is ignored because the size of the column to be changed may be greater than or less than the required size length. For example, you try to change `VARCHAR` column to `CHAR`, MySQL may not allow the modification to `CHAR` if the respective column length is not less than four.
- A single `ALTER TABLE` command can contain several `ADD`, `ALTER`, `DROP`, and `CHANGE` clauses.

Note: To use `ALTER TABLE`, you will require `SELECT`, `INSERT`, `DELETE`, `UPDATE`, `CREATE`, and `DROP` privileges on the table.

6.5 Deleting Table Definitions Using `DROP` Command

MySQL provides the `DROP TABLE` command to remove or delete tables from a database. This command removes table definition, data, indexes, triggers, constraints, and permissions for that table.

Note: The `DROP` command should be used carefully because it removes all the data from a table along with table definitions.

Session 6

Implementing SQL Queries Using MySQL - I

Concepts

The syntax for the `DROP` command is:

```
DROP TABLE [IF EXIST] table_name[table_name1,...] [RESTRICT | CASCADE];
```

where,

`table_name` – specifies the name of the table to delete

`IF EXIST` - prevents error occurrence while executing the command if the table does not exist

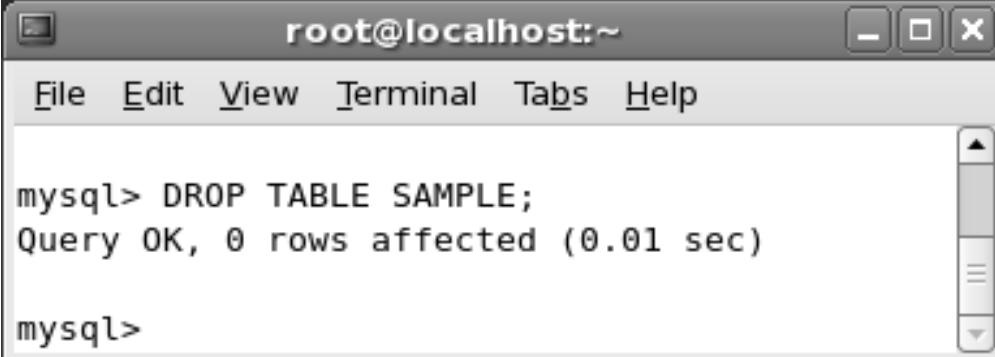
`RESTRICT` - specifies that if any dependencies exist, MySQL will not delete the table, if dependencies exist. If MySQL, encounters a dependency, it returns an error but does not delete the table.

`CASCADE` - specifies to remove dependencies before deleting the table

For example, to remove a table named `SAMPLE` from the `EMPLOYEE` database, enter the following command at the command prompt:

```
DROP TABLE SAMPLE;
```

Figure 6.31 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a standard Linux-style interface with a title bar, menu bar, and scroll bars. The terminal session shows the following text:
mysql> DROP TABLE SAMPLE;
Query OK, 0 rows affected (0.01 sec)
mysql>

Figure 6.31: Displays `DROP TABLE SAMPLE`

You can use the `IF EXIST` clause with `DROP` command if you are unsure about the existence of a table in the database.

Note: The `DROP TABLE` command cannot be used to delete tables that are referencing `FOREIGN KEY`. You will have to create a sample table to execute this command. Do not drop the existing tables from the database.



Summary

- MySQL provides the SHOW command to view the list of databases and tables on the server, displays the server status, and user account privileges.
- The ALTER command enables modification of the characteristics or attributes of a table or a database.
- The CHARACTER SET clause defines the default character set of a database.
- The default collation of a database can be defined using the COLLATE clause.
- The SELECT command is used to retrieve data from a table.
- The FROM clause in the SELECT command specifies the table name whose records are to be retrieved.
- The WHERE clause in the SELECT command specifies conditions for retrieving data from the table. MySQL displays only those records that satisfy the conditions in the WHERE clause.
- Conditional and logical operators can be used with SELECT command to retrieve data after satisfying data retrieval conditions.
- The DROP command removes or deletes a table from a database. This command removes all the data and the table definition from the database.
- MySQL provides the IF EXIST option in the DROP command to check the existence of the table before you delete it.

Session 6

Implementing SQL Queries Using MySQL - I



Check Your Progress

1. You can view the columns of a table using the _____ command.
 - a. SELECT
 - b. VIEW
 - c. SHOW
 - d. USE

2. MySQL displays only the first 100 characters of each process in the SHOW PROCESSLIST command, if you do not specify the _____ clause.
 - a. STATUS
 - b. VARIABLE
 - c. FULL
 - d. INDEX

3. Which of the following is similar to the SHOW TABLE command except that it displays detailed information about each table?
 - a. SHOW STATUS
 - b. SHOW TABLE STATUS
 - c. SHOW TABLES
 - d. SHOW VARIABLES

4. Identify the command that performs actions on temporary copy of original database.
 - a. MODIFY
 - b. ALTER

Session 6

Implementing SQL Queries Using MySQL - I



Check Your Progress

Concepts

- c. CHANGE
 - d. UPDATE
5. In which file of the database directory does MySQL store the global characteristics attributes of a database?
- a. tables_priv
 - b. db
 - c. user
 - d. db.opt
6. Which of the following commands will display all the records of EMPLOYEE table?
- a. SELECT ALL FROM EMPLOYEE;
 - b. SELECT COLUMNS FROM EMPLOYEE;
 - c. SELECT * FROM EMPLOYEE;
 - d. SELECT RECORDS FROM EMPLOYEE;
7. You will use the _____ option of the DROP command to display the dependencies of a table.
- a. IF EXIST
 - b. RESTRICT
 - c. CASCADE
 - d. DROP TABLE

Technowise



Are you a
TECHNO GEEK
looking for updates?

Login to

www.onlinevarsity.com

Objectives

At the end of this session, the student will be able to:

- *Modify the database.*
- *Use the SELECT command.*
- *Use the WHERE clause with the SELECT command.*
- *Modify the table structure.*
- *Delete a table from the database.*

The steps given in this session are detailed, comprehensive, and carefully thought through in order to meet the learning objectives and understand the command completely. Please follow the steps carefully.

Part I - For the first 1.5 hours:

Modifying the Database

The ALTER DATABASE command modifies the global characteristics or attributes of a database. These attributes are stored in the db.opt file of the database directory. For example, to modify the CHARACTER SET clause using the ALTER DATABASE command, execute the following steps.

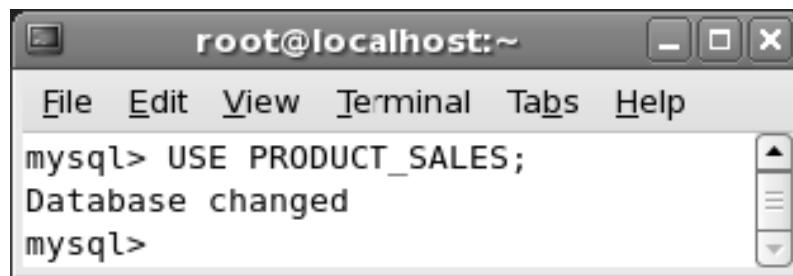
1. To activate the PRODUCT_SALES database, enter the following command at the command prompt:

```
USE PRODUCT_SALES;
```

Figure 7.1 displays the output of the command.

Session 7

Implementing SQL Queries Using MySQL - I (Lab)



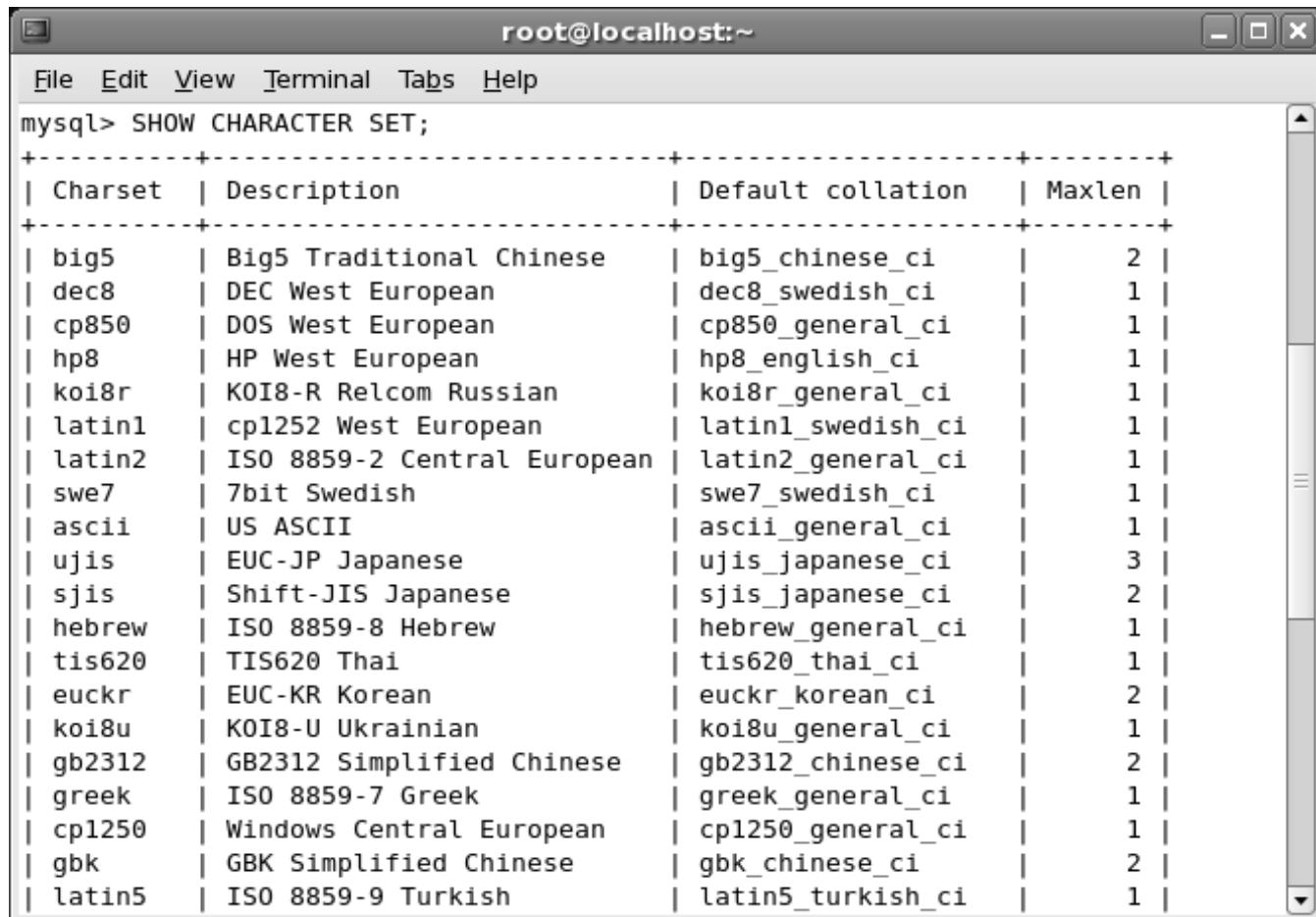
```
root@localhost:~ mysql> USE PRODUCT_SALES;
Database changed
mysql>
```

Figure 7.1: Activating PRODUCT_SALES Database

2. To view the default character set for the PRODCUT_SALES database, enter the following command at the command prompt:

```
SHOW CHARACTER SET;
```

Figure 7.2 displays the output of the command.



Charset	Description	Default collation	Maxlen
big5	Big5 Traditional Chinese	big5_chinese_ci	2
dec8	DEC West European	dec8_swedish_ci	1
cp850	DOS West European	cp850_general_ci	1
hp8	HP West European	hp8_english_ci	1
koi8r	KOI8-R Relcom Russian	koi8r_general_ci	1
latin1	cpi252 West European	latin1_swedish_ci	1
latin2	ISO 8859-2 Central European	latin2_general_ci	1
swe7	7bit Swedish	swe7_swedish_ci	1
ascii	US ASCII	ascii_general_ci	1
ujis	EUC-JP Japanese	ujis_japanese_ci	3
sjis	Shift-JIS Japanese	sjis_japanese_ci	2
hebrew	ISO 8859-8 Hebrew	hebrew_general_ci	1
tis620	TIS620 Thai	tis620_thai_ci	1
euckr	EUC-KR Korean	euckr_korean_ci	2
koi8u	KOI8-U Ukrainian	koi8u_general_ci	1
gb2312	GB2312 Simplified Chinese	gb2312_chinese_ci	2
greek	ISO 8859-7 Greek	greek_general_ci	1
cp1250	Windows Central European	cp1250_general_ci	1
gbk	GBK Simplified Chinese	gbk_chinese_ci	2
latin5	ISO 8859-9 Turkish	latin5_turkish_ci	1

Figure 7.2: Displaying Default Character Set

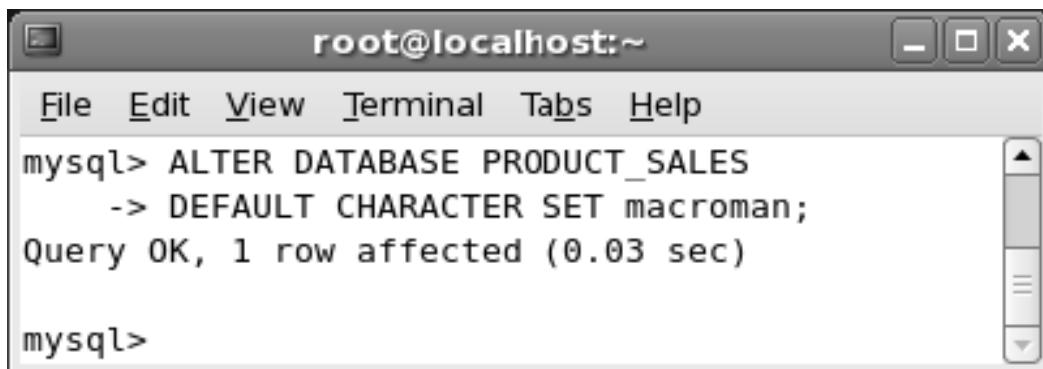
Session 7

Implementing SQL Queries Using MySQL - I (Lab)

3. To modify the character set of the PRODUCT_SALES database, enter the following command at the command prompt:

```
ALTER DATABASE PRODUCT_SALES DEFAULT CHARACTER SET macroman;
```

Figure 7.3 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area contains the following text:

```
mysql> ALTER DATABASE PRODUCT_SALES
      -> DEFAULT CHARACTER SET macroman;
Query OK, 1 row affected (0.03 sec)

mysql>
```

Figure 7.3: Altering the Database

Figure 7.3 indicates that the character set for the PRODUCT_SALES database has been modified.

Using SELECT Command

The SELECT command retrieves data from the tables in the database. The WHERE clause is used for retrieving records from the tables that satisfy the specified conditions. For example, to retrieve data from the BOOK_DETAILS table, execute the following steps.

1. To activate the LIBRARY database, enter the following command at the command prompt:

```
USE LIBRARY;
```

Figure 7.4 displays the output of the command.

Session 7

Implementing SQL Queries Using MySQL - I (Lab)

```
root@localhost:~  
File Edit View Terminal Tabs Help  
mysql> USE LIBRARY;  
Reading table information for completion of table and column names  
You can turn off this feature to get a quicker startup with -A  
Database changed  
mysql>
```

Figure 7.4: Activating Library Database

2. In order to view, records from a table, you must enter data into the tables. To enter data in the BOOK_DETAILS table, enter the following command at the command prompt:

```
INSERT INTO BOOK_DETAILS  
  
VALUES(101, 'PROGRAMMING WITH C', 'GODFRIED', 'THIRD', 'C PROGRAMMING');
```

Figure 7.5 displays the output of the command.

```
root@localhost:~  
File Edit View Terminal Tabs Help  
  
mysql> INSERT INTO BOOK_DETAILS  
-> VALUES  
-> (101,'PROGRAMMING WITH C','GODFRIED','THIRD','C PROGRAMMING');  
Query OK, 1 row affected (0.00 sec)  
  
mysql>
```

Figure 7.5: Adding Records to a Table

Similarly, add the remaining records to the BOOK_DETAILS table as listed in table 7.1

BOOK_ID	BOOK_NAME	AUTHOR	EDITION	DESCRIPTION
102	C++ PROGRAMMING	PETER NORTON	SECOND	PROGRAMMING WITH C++
103	UNIX	JOHN MACBILL	SECOND	UNIX OPERATING SYSTEM
104	LINUX	SCOTT MANN	FIRST	LINUX OPERATING SYSTEM
105	VISUAL BASIC	Black Book	FIRST	DESIGN USING VB

Table 7.1: Records for BOOK_DETAILS Table

Session 7

Implementing SQL Queries Using MySQL - I (Lab)

Add the records to the CATEGORY table as listed in table 7.2.

USER_ID	TYPE	FACILITY	ISSUE_STATUS
100	REGULAR	ISSUE 3 BOOKS AT A TIME	0
101	ONE DAY MEMBER	ISSUE 1 BOOK AT A TIME	0
102	REGULAR	ISSUE 3 BOOKS AT A TIME	0
103	REGULAR	ISSUE 3 BOOKS	0

Table 7.2: Records for CATEGORY Table

Add the records to the ISSUE_DETAILS table as listed in table 7.3.

USER_ID	BOOK_ID	BOOK_NAME	DATE_OF_ISSUE	DATE_OF_RETURN	FINE
100	101	PROGRAMMING WITH C	2010-06-05	2010-11-05	60.00
101	102	C++ PROGRAMMING	2010-04-04	2009-04-20	0.00
102	103	UNIX	2010-09-05	2010-10-09	40.00
103	104	LINUX	2010-01-01	2010-01-11	0.00

Table 7.3: Records for ISSUE_DETAILS Table

Add the records to the USER_DETAILS table as listed in table 7.4.

USER_ID	NAME	LOCATION	PHONE_NO
100	TOBY	MEXICO	45637
101	BINCY	MEXICO	45663
102	DENVER	MEXICO	45123
103	HELEN	MEXICO	45867

Table 7.4: Records for USER_DETAILS Table

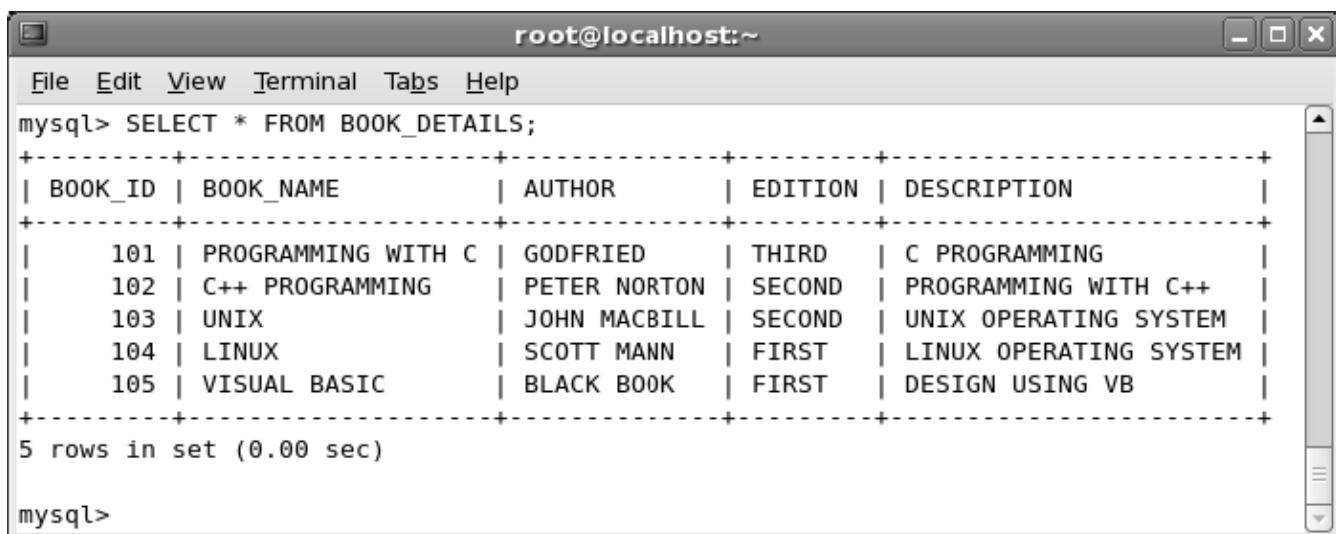
3. The librarian wants to view the list of all books available in the Book_Details table. To view all the records from the BOOK_DETAILS table, enter the following command at the command prompt:

```
SELECT * FROM BOOK_DETAILS;
```

Figure 7.6 displays the output of the command.

Session 7

Implementing SQL Queries Using MySQL - I (Lab)



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its output:

```
mysql> SELECT * FROM BOOK_DETAILS;
+----+-----+-----+-----+-----+
| BOOK_ID | BOOK_NAME | AUTHOR | EDITION | DESCRIPTION |
+----+-----+-----+-----+-----+
| 101 | PROGRAMMING WITH C | GODFRIED | THIRD | C PROGRAMMING |
| 102 | C++ PROGRAMMING | PETER NORTON | SECOND | PROGRAMMING WITH C++ |
| 103 | UNIX | JOHN MACBILL | SECOND | UNIX OPERATING SYSTEM |
| 104 | LINUX | SCOTT MANN | FIRST | LINUX OPERATING SYSTEM |
| 105 | VISUAL BASIC | BLACK BOOK | FIRST | DESIGN USING VB |
+----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql>
```

Figure 7.6: Displaying all Records from BOOK_DETAILS Table

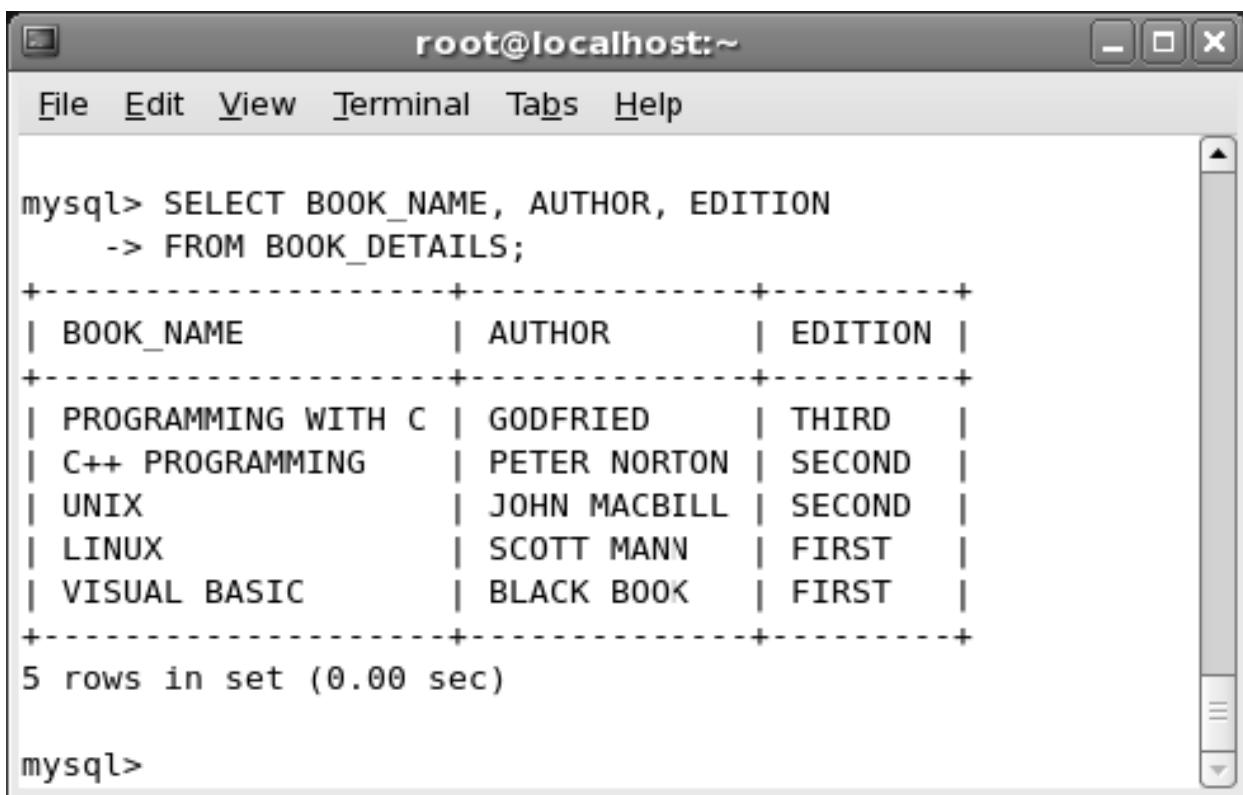
4. The librarian wants to view only the book name, author and the edition for books available in the Book_Details table. To view the BOOK_NAME, AUTHOR, and the EDITION fields from the BOOK_DETAILS table, enter the following command at the command prompt:

```
SELECT BOOK_NAME, AUTHOR, EDITION FROM BOOK_DETAILS;
```

Figure 7.7 displays the output of the command.

Session 7

Implementing SQL Queries Using MySQL - I (Lab)



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The main area displays the following MySQL command and its output:

```
mysql> SELECT BOOK_NAME, AUTHOR, EDITION  
-> FROM BOOK_DETAILS;  
+-----+-----+-----+  
| BOOK_NAME      | AUTHOR        | EDITION |  
+-----+-----+-----+  
| PROGRAMMING WITH C | GODFRIED      | THIRD    |  
| C++ PROGRAMMING   | PETER NORTON  | SECOND   |  
| UNIX            | JOHN MACBILL | SECOND   |  
| LINUX           | SCOTT MANN   | FIRST    |  
| VISUAL BASIC     | BLACK BOOK   | FIRST    |  
+-----+-----+-----+  
5 rows in set (0.00 sec)  
  
mysql>
```

A vertical scrollbar is visible on the right side of the terminal window.

Lab Guide

Figure 7.7: Displaying Specific Columns of the BOOK_DETAILS Table

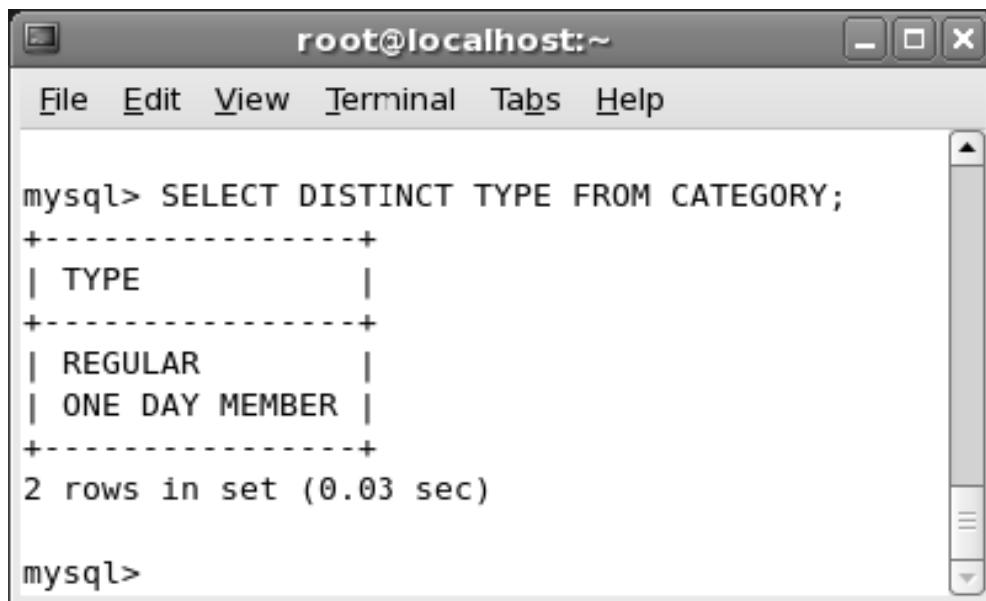
5. A student wants to be a member in the library. The student wants to know the different types of membership available in the library management system. The librarian wants to view the types of membership from the CATEGORY table. To view the distinct records from the CATEGORY table, depending on the column TYPE, enter the following command at the command prompt:

```
SELECT DISTINCT TYPE FROM CATEGORY;
```

Figure 7.8 displays the output of the command.

Session 7

Implementing SQL Queries Using MySQL - I (Lab)



A screenshot of a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area displays the following MySQL command and its output:

```
mysql> SELECT DISTINCT TYPE FROM CATEGORY;
+-----+
| TYPE |
+-----+
| REGULAR |
| ONE DAY MEMBER |
+-----+
2 rows in set (0.03 sec)

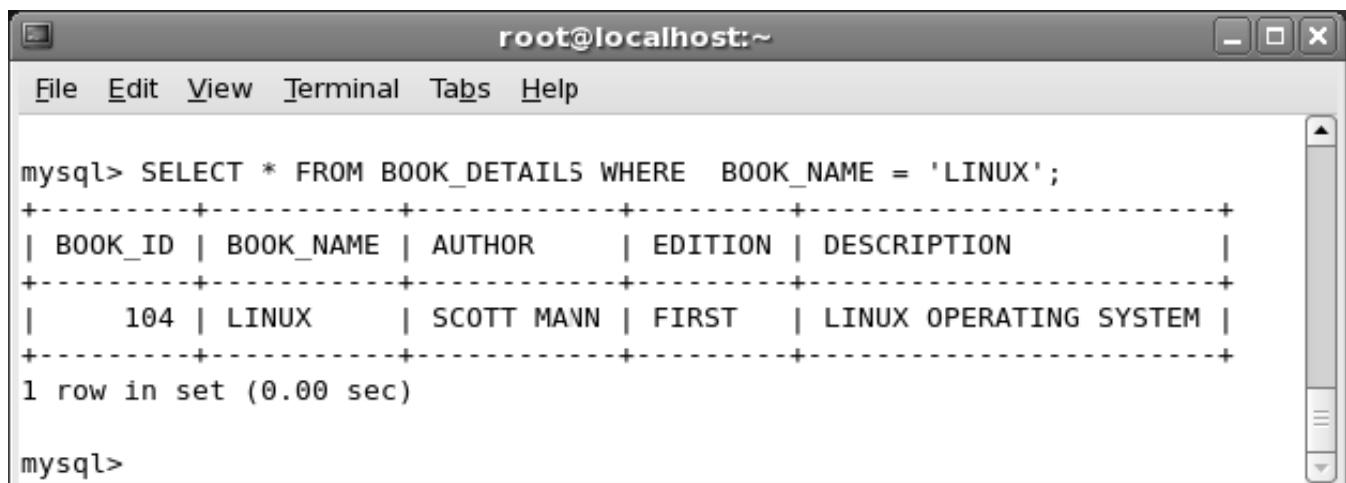
mysql>
```

Figure 7.8: Viewing Unique Membership types from CATEGORY table

6. The librarian has received a request from different students for Linux books. The librarian wants to know whether Linux books exist in the library. To view those records where the book name is LINUX, enter the following command at the command prompt:

```
SELECT * FROM BOOK_DETAILS WHERE BOOK_NAME = 'LINUX';
```

Figure 7.9 displays the output of the command.



A screenshot of a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area displays the following MySQL command and its output:

```
mysql> SELECT * FROM BOOK_DETAILS WHERE BOOK_NAME = 'LINUX';
+-----+-----+-----+-----+
| BOOK_ID | BOOK_NAME | AUTHOR      | EDITION | DESCRIPTION
+-----+-----+-----+-----+
|    104   | LINUX     | SCOTT MANN | FIRST   | LINUX OPERATING SYSTEM
+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 7.9: Displaying Records Where the Book_Name is LINUX

Session 7

Implementing SQL Queries Using MySQL - I (Lab)

7. To view the BOOK_ID, BOOK_NAME, and the EDITION fields of those books whose BOOK_ID is greater than 102, enter the following command at the command prompt:

```
SELECT BOOK_ID, BOOK_NAME, EDITION FROM BOOK_DETAILS WHERE  
BOOK_ID > 102;
```

Figure 7.10 displays the output of the command.

The screenshot shows a terminal window titled "root@localhost:~". The window contains the following text:

```
mysql> SELECT BOOK_ID,  
-> BOOK_NAME,  
-> EDITION  
-> FROM BOOK_DETAILS  
-> WHERE BOOK_ID > 102;  
+-----+-----+-----+  
| BOOK_ID | BOOK_NAME | EDITION |  
+-----+-----+-----+  
| 103    | UNIX      | SECOND   |  
| 104    | LINUX     | FIRST    |  
| 105    | VISUAL BASIC | FIRST    |  
+-----+-----+-----+  
3 rows in set (0.02 sec)  
  
mysql>
```

The terminal window has a standard window title bar with minimize, maximize, and close buttons. The main area is a text-based interface for a MySQL database session.

Figure 7.10: Displaying Records Where BOOK_ID > 102

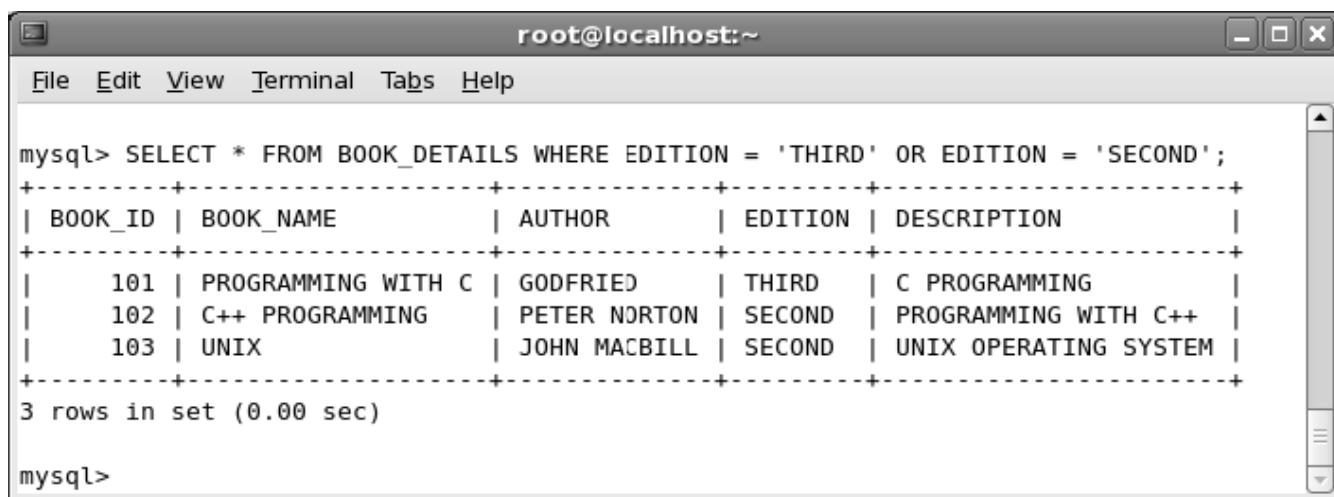
8. A student is preparing for examinations using different books. The student requires books that are marked SECOND or THIRD edition. The librarian wants to know if these books exist in the library. To view the second or third edition books from the BOOK_DETAILS table, enter the following command at the command prompt:

```
SELECT * FROM BOOK_DETAILS  
  
WHERE EDITION = 'THIRD' OR EDITION = 'SECOND';
```

Figure 7.11 displays the output of the command.

Session 7

Implementing SQL Queries Using MySQL - I (Lab)



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its output:

```
mysql> SELECT * FROM BOOK_DETAILS WHERE EDITION = 'THIRD' OR EDITION = 'SECOND';
+----+-----+-----+-----+
| BOOK_ID | BOOK_NAME | AUTHOR | EDITION | DESCRIPTION |
+----+-----+-----+-----+
| 101 | PROGRAMMING WITH C | GODFRIED | THIRD | C PROGRAMMING |
| 102 | C++ PROGRAMMING | PETER NORTON | SECOND | PROGRAMMING WITH C++ |
| 103 | UNIX | JOHN MACBILL | SECOND | UNIX OPERATING SYSTEM |
+----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql>
```

Figure 7.11: Displaying Records Where the Book Edition is Second or Third

In figure 7.11, two conditions have been specified in the WHERE clause using the OR operator.

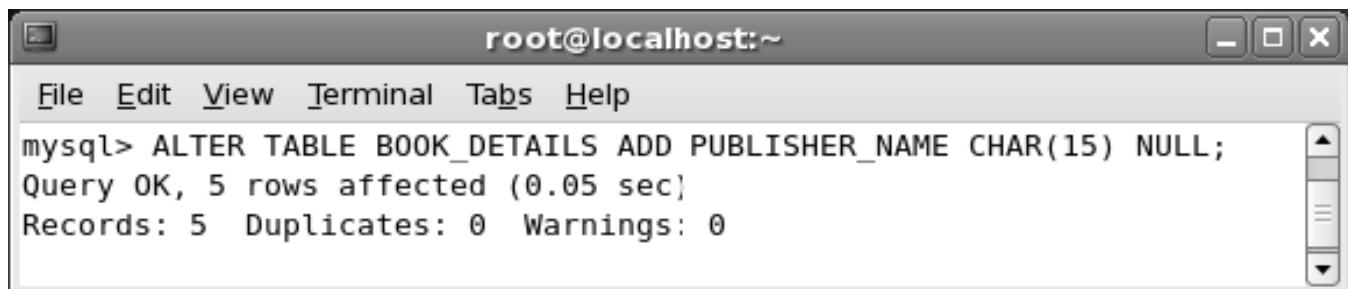
Modifying the Table

You can change the structure of a table with the ALTER command. This command enables you to add or delete columns or constraints, create or destroy indexes, change or modify the type of existing columns, rename columns or the table, and change the type of the table.

1. The librarian wants to sort the books for any subject based on publisher names. This column will be added to the BOOK_DETAILS table. To add a new column, PUBLISHER_NAME, to the BOOK_DETAILS table, enter the following command at the command prompt:

```
ALTER TABLE BOOK_DETAILS ADD PUBLISHER_NAME CHAR(15) NULL;
```

Figure 7.12 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its output:

```
mysql> ALTER TABLE BOOK_DETAILS ADD PUBLISHER_NAME CHAR(15) NULL;
Query OK, 5 rows affected (0.05 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

Figure 7.12: Adding a New Column Using ALTER command

Session 7

Implementing SQL Queries Using MySQL - I (Lab)

2. The librarian wants to ensure that the new column, PUBLISHER_NAME has been added to the BOOK_DETAILS table. To view the table structure after adding the PUBLISHER_NAME column, enter the following command at the command prompt:

```
DESC BOOK_DETAILS;
```

Figure 7.13 displays the output of the command.

The screenshot shows a terminal window titled "root@localhost:~". The window contains the following text:

```
File Edit View Terminal Tabs Help
mysql> DESC BOOK_DETAILS;
+-----+-----+-----+-----+-----+
| Field      | Type       | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| BOOK_ID    | int(4)     | YES  |     | NULL    |       |
| BOOK_NAME   | varchar(35) | YES  |     | NULL    |       |
| AUTHOR      | char(25)   | YES  |     | NULL    |       |
| EDITION     | char(15)   | YES  |     | NULL    |       |
| DESCRIPTION | text        | YES  |     | NULL    |       |
| PUBLISHER_NAME | char(15) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql>
```

Figure 7.13: Table Description after Adding a Column

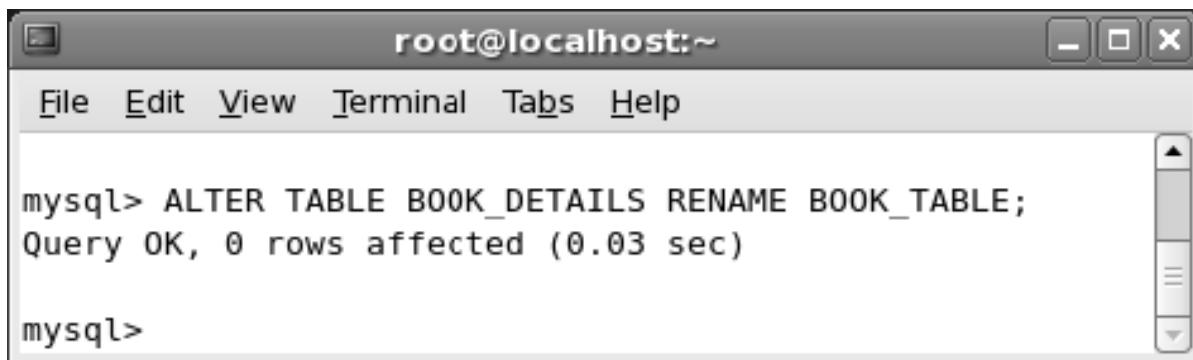
3. The IT department wants to change the name of the BOOK_DETAILS table to match their records. To rename the table from BOOK_DETAILS to BOOK_TABLE, enter the following command at the command prompt:

```
ALTER TABLE BOOK_DETAILS RENAME BOOK_TABLE;
```

Figure 7.14 displays the output of the command.

Session 7

Implementing SQL Queries Using MySQL - I (Lab)



```
mysql> ALTER TABLE BOOK_DETAILS RENAME BOOK_TABLE;
Query OK, 0 rows affected (0.03 sec)

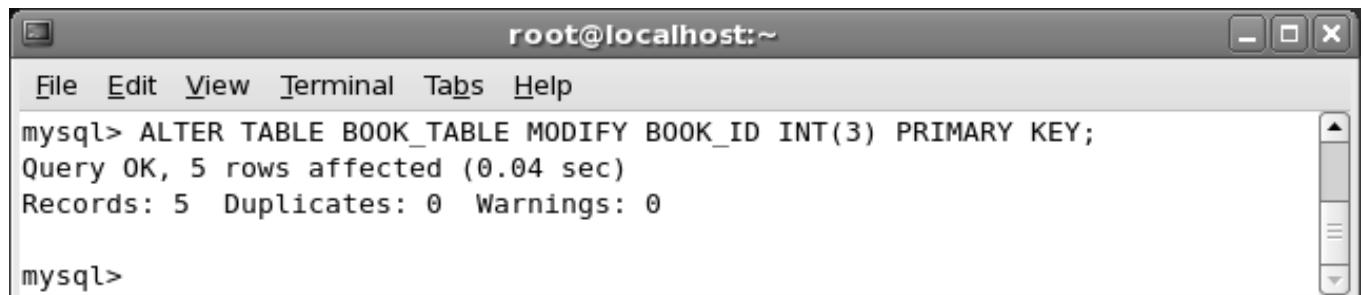
mysql>
```

Figure 7.14: Renaming the Table

4. A primary key can be defined to accept unique values. The librarian wants to use the primary key constraints in the table containing book details so that the name of the books does not duplicate in the table. To define a primary key on the BOOK_ID field in the BOOK_TABLE table, enter the following command at the command prompt:

```
ALTER TABLE BOOK_TABLE MODIFY BOOK_ID INT(3) PRIMARY KEY;
```

Figure 7.15 displays the output of the command.



```
mysql> ALTER TABLE BOOK_TABLE MODIFY BOOK_ID INT(3) PRIMARY KEY;
Query OK, 5 rows affected (0.04 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql>
```

Figure 7.15: Defining a Primary Key

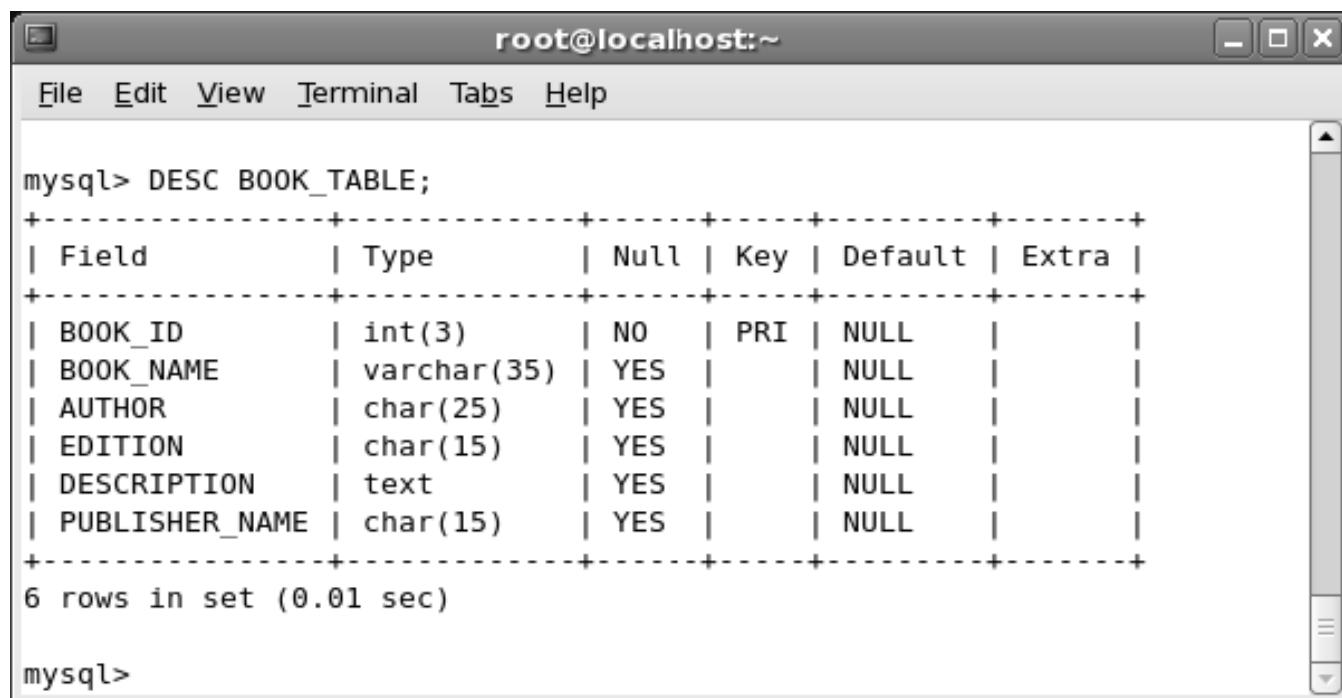
5. After defining the primary key constraint the librarian wants to view the table structure, enter the following command at the command prompt:

```
DESC BOOK_TABLE;
```

Figure 7.16 displays the output of the command.

Session 7

Implementing SQL Queries Using MySQL - I (Lab)



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its output:

```
mysql> DESC BOOK_TABLE;
+-----+-----+-----+-----+-----+
| Field      | Type       | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| BOOK_ID    | int(3)     | NO   | PRI | NULL    |       |
| BOOK_NAME  | varchar(35) | YES  |     | NULL    |       |
| AUTHOR     | char(25)   | YES  |     | NULL    |       |
| EDITION    | char(15)   | YES  |     | NULL    |       |
| DESCRIPTION | text        | YES  |     | NULL    |       |
| PUBLISHER_NAME | char(15) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+
6 rows in set (0.01 sec)

mysql>
```

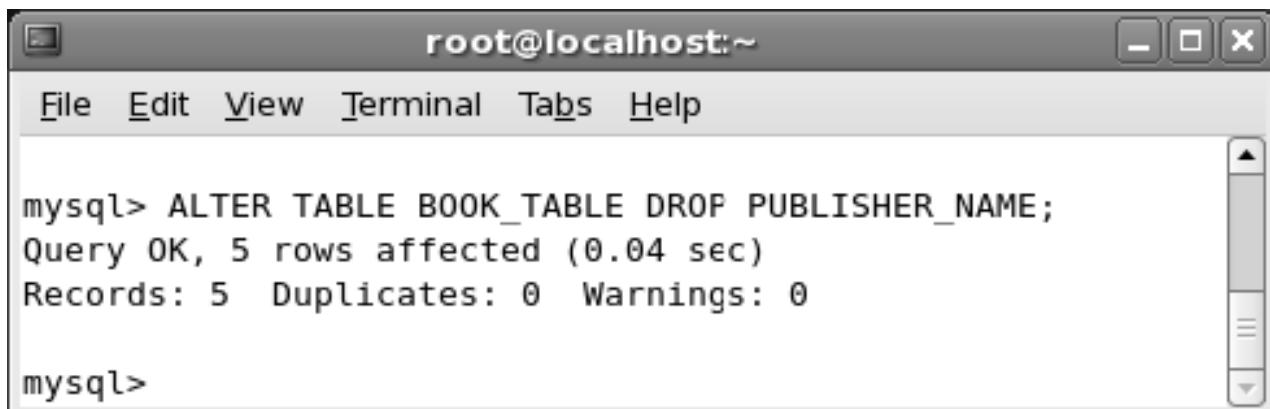
A vertical "Lab Guide" sidebar is visible on the right side of the terminal window.

Figure 7.16: Table Structure after Defining Primary Key

6. The librarian has identified problems to the BOOK_DETAILS table because of the addition of the new column PUBLISHER_NAME and wants to remove that column from the table. To delete the PUBLISHER_NAME column from the BOOK_TABLE, enter the following command at the command prompt:

```
ALTER TABLE BOOK_TABLE DROP PUBLISHER_NAME;
```

Figure 7.17 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its output:

```
mysql> ALTER TABLE BOOK_TABLE DROP PUBLISHER_NAME;
Query OK, 5 rows affected (0.04 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql>
```

Figure 7.17: Deleting a Column from the Table

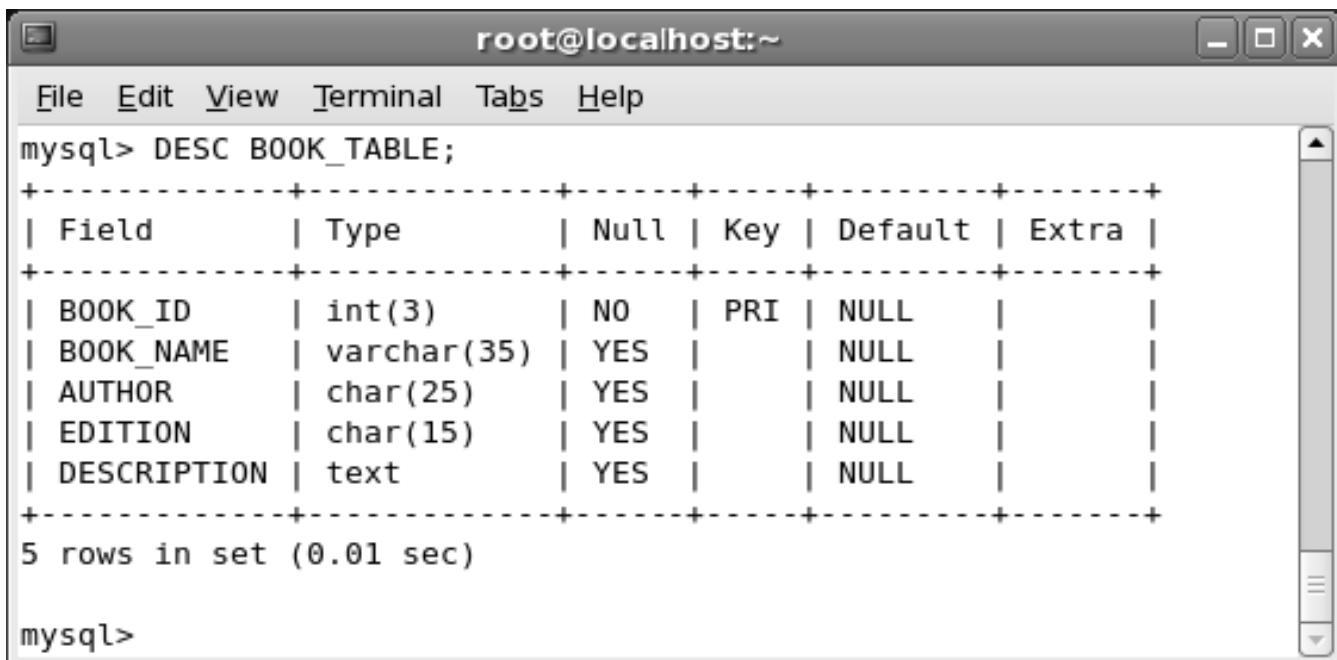
Session 7

Implementing SQL Queries Using MySQL - I (Lab)

7. After deleting the column, the librarian wants to ensure that the column no longer exists in the table. To view the structure of BOOK_TABLE after deleting the PUBLISHER_NAME column, enter the following command at the command prompt:

```
DESC BOOK_TABLE;
```

Figure 7.18 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a standard Linux-style title bar with icons for minimize, maximize, and close. Below the title bar is a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The main area of the terminal shows the MySQL command-line interface. A user has run the command "DESC BOOK_TABLE;". The output is a table describing the structure of the BOOK_TABLE. The columns are Field, Type, Null, Key, Default, and Extra. The table contains five rows, each corresponding to a column in the table: BOOK_ID, BOOK_NAME, AUTHOR, EDITION, and DESCRIPTION. The primary key is listed as PRI under the Key column. The Default column for all columns is set to NULL. The Extra column is empty. At the bottom of the terminal window, it says "5 rows in set (0.01 sec)".

Field	Type	Null	Key	Default	Extra
BOOK_ID	int(3)	NO	PRI	NULL	
BOOK_NAME	varchar(35)	YES		NULL	
AUTHOR	char(25)	YES		NULL	
EDITION	char(15)	YES		NULL	
DESCRIPTION	text	YES		NULL	

5 rows in set (0.01 sec)

mysql>

Figure 7.18: BOOK_TABLE Description after Modification

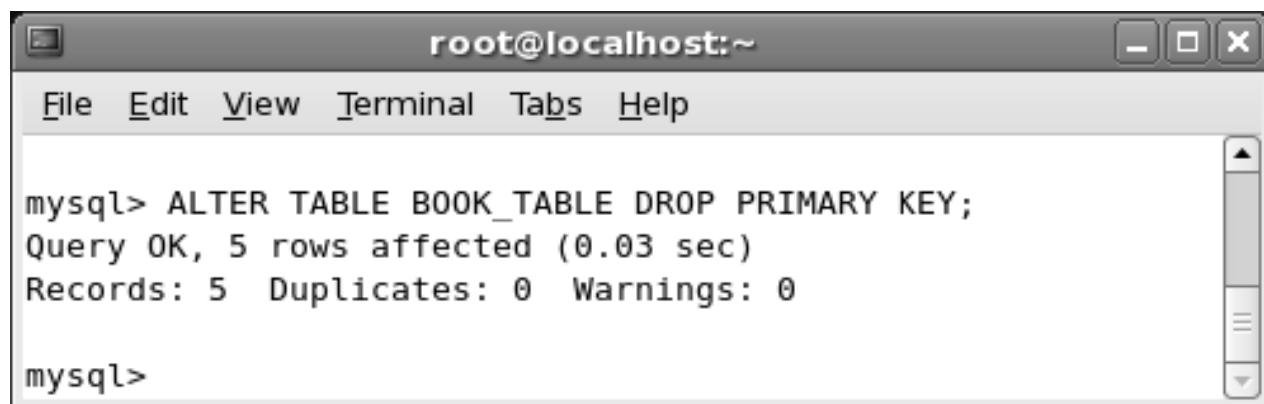
8. The librarian has procured new books on various subjects. The ID for these new books are not decided. An error is generated when the librarian attempts to add book details without the BOOK_ID in the BOOK_TABLE. To resolve this, the librarian has to remove the primary key of the table. To drop the primary key on the BOOK_ID column from BOOK_TABLE, enter the following command at the command prompt:

```
ALTER TABLE BOOK_TABLE DROP PRIMARY KEY;
```

Figure 7.19 displays the output of the command.

Session 7

Implementing SQL Queries Using MySQL - I (Lab)



The screenshot shows a terminal window titled "root@localhost:~". The menu bar includes "File", "Edit", "View", "Terminal", "Tabs", and "Help". The main area displays the following MySQL command and its output:

```
mysql> ALTER TABLE BOOK_TABLE DROP PRIMARY KEY;
Query OK, 5 rows affected (0.03 sec)
Records: 5  Duplicates: 0  Warnings: 0

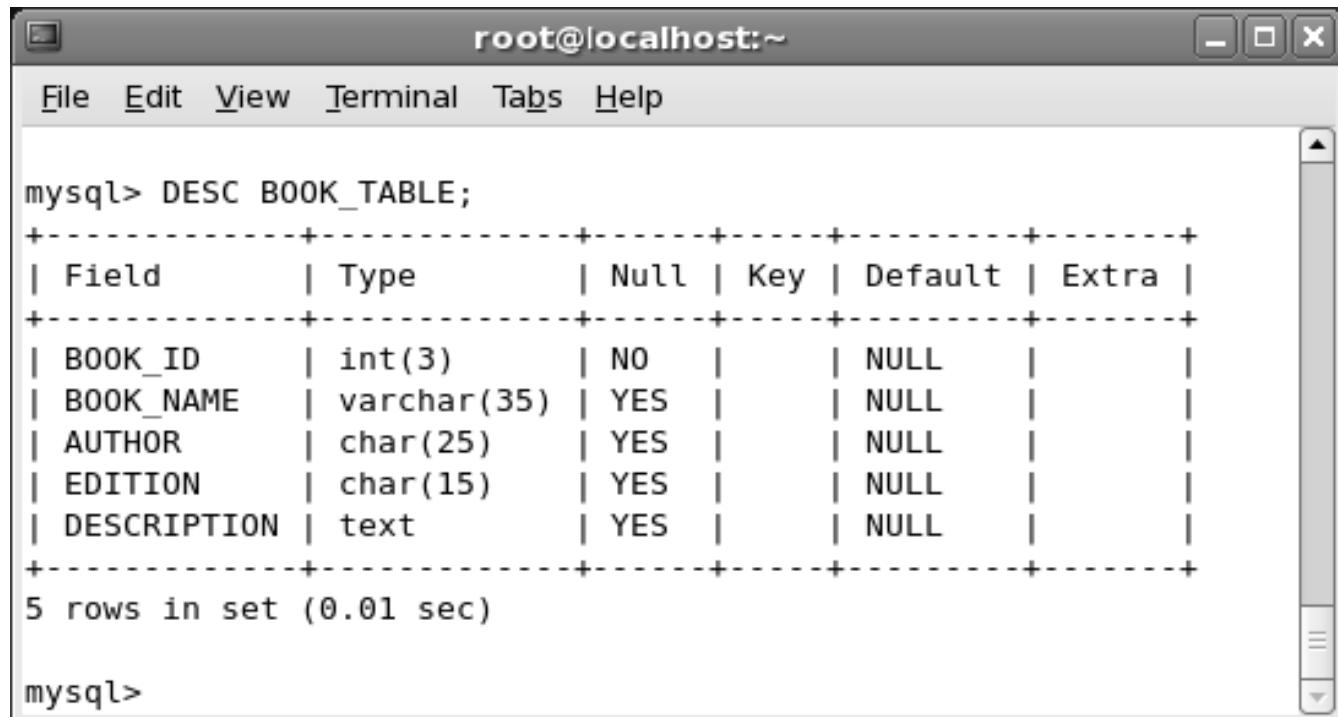
mysql>
```

Figure 7.19: Removing Primary key from the table

9. To add the new books to the database without any errors, the librarian wants to ensure that the primary key has been removed from the BOOK_ID column of the BOOK_TABLE. To view the structure of BOOK_TABLE after deleting the primary key, enter the following command at the command prompt:

```
DESC BOOK_TABLE;
```

Figure 7.20 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The menu bar includes "File", "Edit", "View", "Terminal", "Tabs", and "Help". The main area displays the following MySQL command and its output:

```
mysql> DESC BOOK_TABLE;
+-----+-----+-----+-----+-----+
| Field      | Type       | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| BOOK_ID    | int(3)     | NO   | YES  | NULL    |          |
| BOOK_NAME   | varchar(35) | YES  |      | NULL    |          |
| AUTHOR      | char(25)    | YES  |      | NULL    |          |
| EDITION     | char(15)    | YES  |      | NULL    |          |
| DESCRIPTION | text        | YES  |      | NULL    |          |
+-----+-----+-----+-----+-----+
5 rows in set (0.01 sec)

mysql>
```

Figure 7.20: BOOK_TABLE Structure after Deleting Primary Key

Session 7

Implementing SQL Queries Using MySQL - I (Lab)

Figure 7.20 displays that the primary key has been deleted from the table structure.

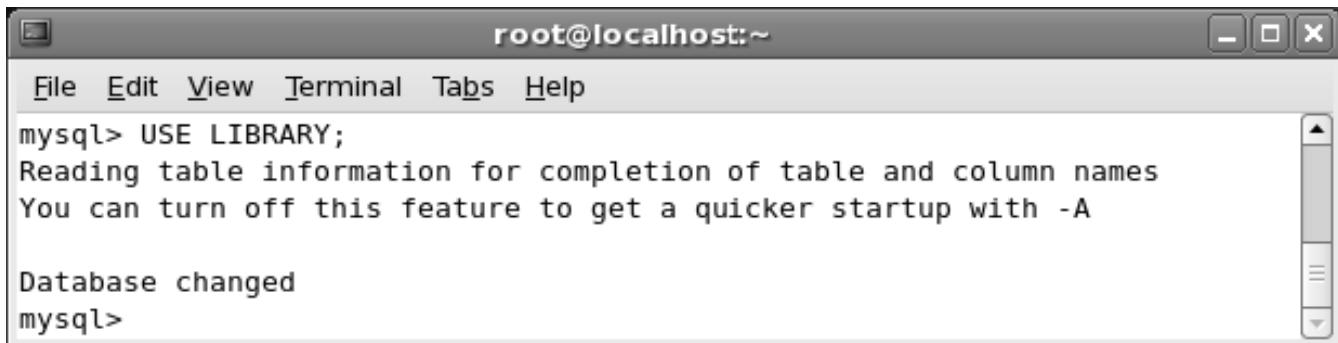
Using DROP Command

You can use the `DROP` command not only to remove a table from the database but also to remove keys, fields, and database from the server.

1. To reactivate the `LIBRARY` database, enter the following command at the command prompt:

```
USE LIBRARY;
```

Figure 7.21 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a standard Linux-style interface with a title bar, menu bar (File, Edit, View, Terminal, Tabs, Help), and a scrollable text area. In the text area, the user has entered the command "USE LIBRARY;". The terminal responds with "Reading table information for completion of table and column names" and "You can turn off this feature to get a quicker startup with -A". Below this, it says "Database changed" and ends with "mysql>".

Figure 7.21: Reactivating LIBRARY Database

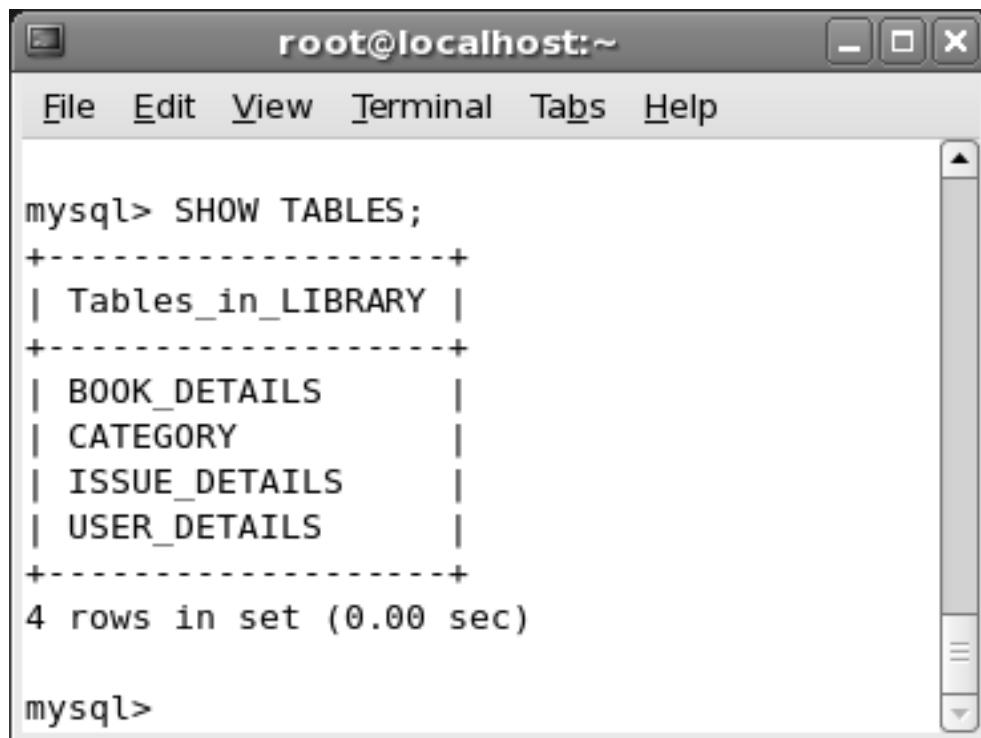
2. To view the list of tables in the `LIBRARY` database, enter the following command at the command prompt:

```
SHOW TABLES;
```

Figure 7.22 displays the output of the command.

Session 7

Implementing SQL Queries Using MySQL - I (Lab)



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The main area displays the following MySQL command and its output:

```
mysql> SHOW TABLES;
+-----+
| Tables_in_LIBRARY |
+-----+
| BOOK_DETAILS      |
| CATEGORY          |
| ISSUE_DETAILS     |
| USER_DETAILS      |
+-----+
4 rows in set (0.00 sec)

mysql>
```

Lab Guide

Figure 7.22: Displaying List of Tables Under LIBRARY Database

3. The librarian is learning to delete unused tables from the database. Unused tables cannot be deleted from the database at random. In order to learn, how to delete a table, the librarian will first create a sample table TRIAL and then use the DROP TABLE command to delete it. To create a table TRIAL with the structure listed in table 7.5, enter the following command at the command prompt:

```
CREATE TABLE TRIAL (Sample_ID INT(3), Sample_Name Char(5));
```

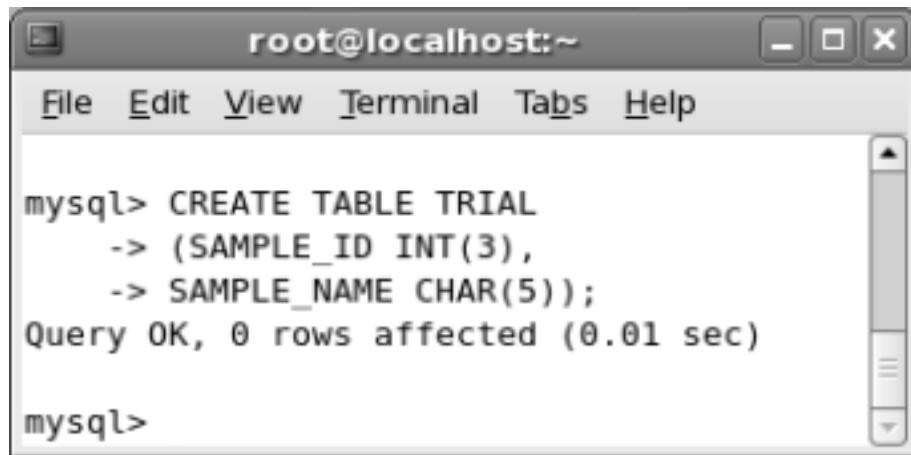
Figure 7.23 displays the output of the command.

Field	Data Type
Sample_ID	INT(3)
Sample_Name	Char(5)

Table 7.5: Trial Table Structure

Session 7

Implementing SQL Queries Using MySQL - I (Lab)



```
root@localhost:~ File Edit View Terminal Tabs Help mysql> CREATE TABLE TRIAL
-> (SAMPLE_ID INT(3),
-> SAMPLE_NAME CHAR(5));
Query OK, 0 rows affected (0.01 sec)

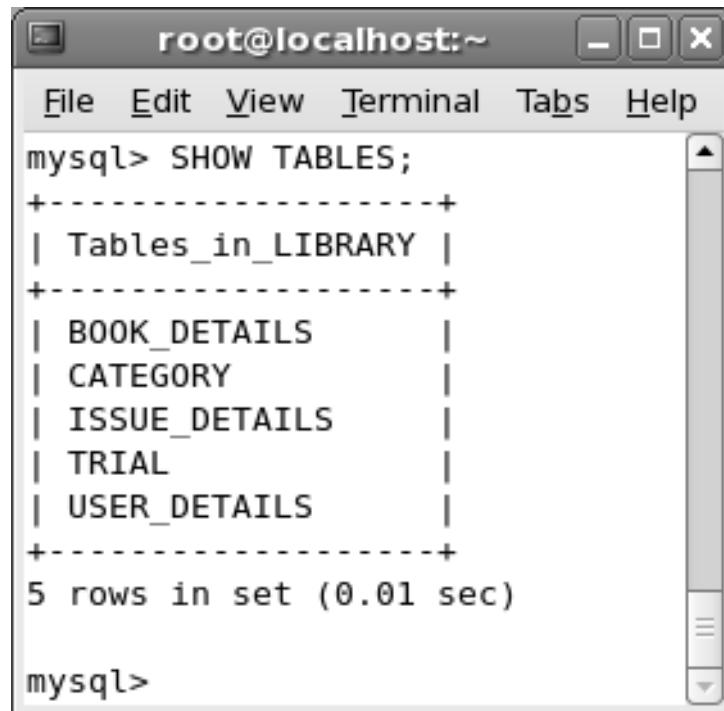
mysql>
```

Figure 7.23: TRIAL Table Creation

4. The librarian wants to ensure that the table TRIAL has been created in the LIBRARY database. To view the list of tables in the LIBRARY database, enter the following command at the command prompt:

```
SHOW TABLES;
```

Figure 7.24 displays the output of the command.



```
root@localhost:~ File Edit View Terminal Tabs Help mysql> SHOW TABLES;
+-----+
| Tables_in_LIBRARY |
+-----+
| BOOK_DETAILS
| CATEGORY
| ISSUE_DETAILS
| TRIAL
| USER_DETAILS
+-----+
5 rows in set (0.01 sec)

mysql>
```

Figure 7.24: Displaying Tables In LIBRARY Database

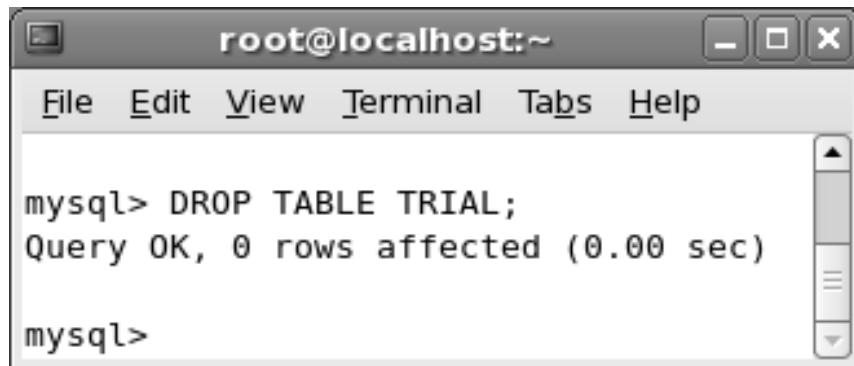
Session 7

Implementing SQL Queries Using MySQL - I (Lab)

5. To drop the TRIAL table, enter the following command at the command prompt:

```
DROP TABLE TRIAL;
```

Figure 7.25 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area contains the following text:

```
mysql> DROP TABLE TRIAL;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

Lab Guide

Figure 7.25: Deletion of TRIAL Table from the LIBRARY Database

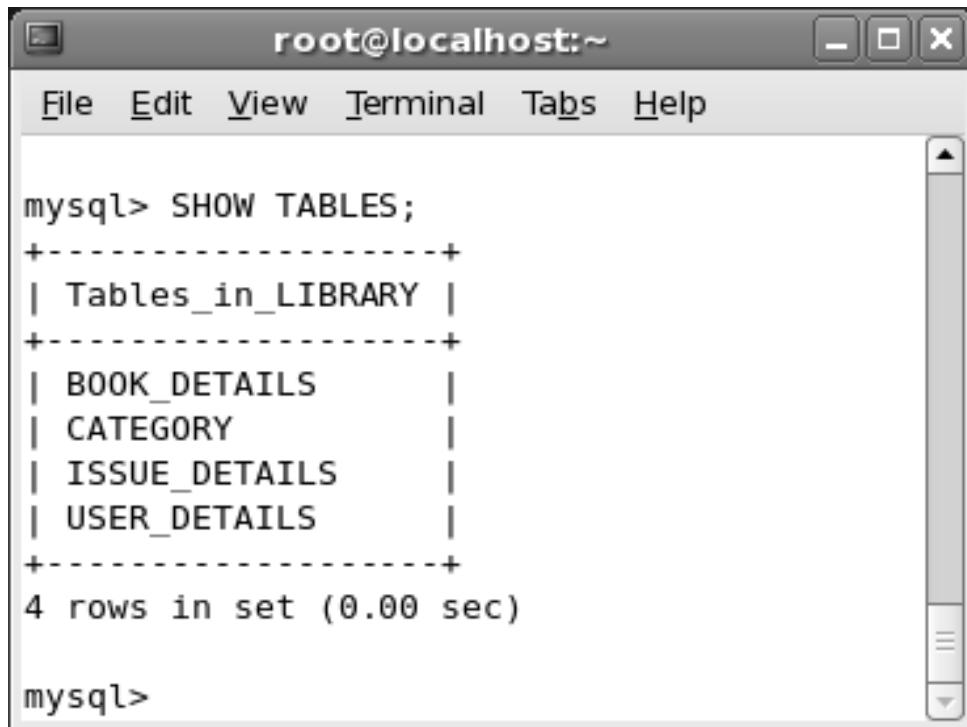
6. The librarian wants to ensure that the table TRIAL has been removed from the LIBRARY database To view the list of tables in the LIBRARY database after removing the TRIAL table, enter the following command at the command prompt:

```
SHOW TABLES;
```

Figure 7.26 displays the output of the command.

Session 7

Implementing SQL Queries Using MySQL - I (Lab)



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area displays the following SQL command and its results:

```
mysql> SHOW TABLES;
+-----+
| Tables_in_LIBRARY |
+-----+
| BOOK_DETAILS      |
| CATEGORY          |
| ISSUE_DETAILS     |
| USER_DETAILS      |
+-----+
4 rows in set (0.00 sec)

mysql>
```

The results show four tables: BOOK_DETAILS, CATEGORY, ISSUE_DETAILS, and USER_DETAILS. The message "4 rows in set (0.00 sec)" indicates the query completed quickly.

Figure 7.26: Tables in the LIBRARY Database after Removing TRIAL Table

Figure 7.26 indicates that the TRIAL table does not appear in the list of tables.

Session 7

Implementing SQL Queries Using MySQL - I (Lab)



Do It Yourself

1. Create a Computer_Course database and then create a Course_Table in the Computer_Course database with the following structure:

Field Name	Data Type
COURSE_NAME	VARCHAR (20)
DURATION	INT (5)
START_DATE	DATE
FEES	DECIMAL (5, 2)

2. View the list of character sets.
3. Modify the character set of Computer_Course database.
4. Create a PRODUCT_SALES database and a STOCK table having the following table structure:

Field Name	Data Type
Product_Code	INT (4)
Product_Name	CHAR (15)
Description	TEXT
Cost Price	DECIMAL (8, 2)

5. Enter the following records in the STOCK table:

Product_Code	Product_Name	Description	Cost Price
101	Bags	Leather Bags	150
102	Books	Notepads	10
103	Pens	Fine Tip Pens	20

6. Display all the records of the PRODUCT_SALES table.
7. Display Product_Code, Product_Name, and Description fields of the table.
8. Display records where the cost price is less than 25.
9. Display records where the Product_Name = Books.
10. Add a new field Quantity in the table.

Session 7

Implementing SQL Queries Using MySQL - I (Lab)



Do It Yourself

11. Add the primary key constraint to the `Product_Code` field.
12. Rename the table to `PRODUCT_DETAILS`.
13. Remove the `DESCRIPTION` field from the table.

Objectives

At the end of this session, the student will be able to:

- *Explain the use of keys in a table*
- *Explain the use of indexes in a table*
- *Explain modification of tables*
- *Explain the use of the ORDER BY command*
- *Explain the use of the GROUP BY command*

8.1 Introduction

MySQL stores records in tables. You not only add data to the table after you create it but you must also regularly update the data stored in the database. The update operation may require replacing certain rows in a table. In addition, you will also have to delete data from the table that you no longer require. Besides performing the additions, modifications, and deletions of records, you can also define an index on a column to enhance the speed of data search and sort operations in a database. To organize the data efficiently in tables and for performing update and delete operations on tables, you need to have certain columns designated as keys. A primary key in the table uniquely identifies a record. A primary key does not allow duplication of records. An index specifies the location of the record in a database.

In this session, you will learn to create an index. You will also identify the different types of keys. In addition, you will learn to modify, group, and sort the data in a table.

8.2 Working with Keys and Indexes

Columns with key definitions have a unique value for each row. Indexing a column will help to enhance the speed of data search and sort operations on the database. MySQL creates an index file and searches for data according to the index file when you index a column.

8.2.1 Defining Keys

Keys are columns that uniquely identify information present in a table. Some of the different types of keys include primary key, composite key, and foreign key.

Session 8

Implementing SQL Queries Using MySQL - II

➤ Primary key

A primary key is used to uniquely identify each row in a table. It can consist of one or more fields on a table. A primary key specifies that there cannot be a column in a table that contains two similar values. The primary key does not allow blank or `NULL` values for the column. All the key columns are declared as `NOT NULL`.

You must remember the following rules while defining a primary key:

- Contains a value and cannot contain a null value
- Is unique for each record

You can define a primary key when you create the table. The syntax for defining a primary key while creating a table is:

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name col_name column  
definition PRIMARY KEY (fieldname);
```

where,

`CREATE` – adds an object to the database

`TABLE` – adds a table to the database

`tbl_name` – specifies the name for the table to add to the database

`PRIMARY KEY` – creates a primary key on the specified column

`fieldname` – specifies the name of the column to define the primary key

MySQL provides different ways to define a primary key for a table. You can define the primary key on a table while creating it. You can also modify an existing table structure and define a primary key using the `ALTER TABLE` command. In addition, you can modify the existing primary key on a table and specify conditions for the primary key. However, you will have to remember the rules for a primary key while defining or modifying it.

For example, you have already created a primary key for the `EMP_DETAILS` table. You will now modify this key as `NOT NULL` and `AUTO_INCREMENT` so that this primary key does not accept or contain a `NULL` value.

To modify the primary key as `NOT NULL` and `AUTO_INCREMENT`, enter the following command at the command prompt:

Session 8

Implementing SQL Queries Using MySQL - II

```
ALTER TABLE EMP_DETAILS MODIFY E_ID INT(3) NOT NULL PRIMARY KEY AUTO_INCREMENT;
```

Figure 8.1 displays the output of the command.

The screenshot shows a terminal window titled "root@localhost:~". The window contains the following text:

```
File Edit View Terminal Tabs Help
mysql> ALTER TABLE EMP_DETAILS
      -> MODIFY E_ID INT(3) NOT NULL
      -> PRIMARY KEY AUTO_INCREMENT;
Query OK, 5 rows affected (0.02 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql>
```

The terminal window has a standard window title bar with minimize, maximize, and close buttons. The main area is a scrollable text box displaying the MySQL command and its execution results.

Concepts

Figure 8.1: Defining a Primary Key

➤ **Composite Key**

When you use more than one column to define a primary key, then this type of a primary key is known as a composite key. The rules for defining a composite key are the same as the primary key.

➤ **Foreign Key**

A foreign key is used to establish relationship between two tables. A foreign key is a column in a table that has a corresponding primary key column in a different table. In other words, a foreign key in a table is used for referencing a primary key present in another table. The main goal of foreign key is to maintain referential integrity of the data.

In other words, it can be said that it is a database constraint that refers to a column in another table for reference. You can define a column as a foreign key only if both the tables have `InnoDB` as their storage engine. By default, MySQL assigns the `MyISAM` storage engine when you create a table. However, you can change the storage engine of the table using the `ALTER TABLE` command.

Session 8

Implementing SQL Queries Using MySQL - II

The advantages of using InnoDB as the storage engine are:

- Cancels INSERT and UPDATE commands that attempt to add records to the child table without having a corresponding record in the parent table
- Allows the foreign key to reference a group of columns in the parent table

To modify the storage engine of the EMP_DETAILS to InnoDB, enter the following command at the command prompt:

```
ALTER TABLE EMP_DETAILS ENGINE=INNODB;
```

Similarly, to modify the EMP_SALARY storage engine to InnoDB, enter the following command at the command prompt:

```
ALTER TABLE EMP_SALARY ENGINE=INNODB;
```

After modifying the storage engine, you can now define the foreign key.

The syntax to define a foreign key is:

```
ALTER TABLE tbl_name ADD [Constraint symbol] FOREIGN KEY [index_name](index_col_name,...) REFERENCES table_name (index_col_name,...) [ON DELETE {CASCADE|SET NULL|NO ACTION|RESTRICT}] [ON UPDATE {CASCADE|SET NULL|NO ACTION|RESTRICT}];
```

where,

ALTER TABLE – modifies the table structure

tbl_name – specifies the name of the table to edit

ADD – appends an object to the table structure

FOREIGN KEY – defines the foreign key to the column

index_name – specifies the name of the index

index_col_name – specifies the name of the column defined as index

REFERENCES – defines the relationship with a primary key of another table

Session 8

Implementing SQL Queries Using MySQL - II

The options for defining a foreign key are listed in table 8.1.

Concepts

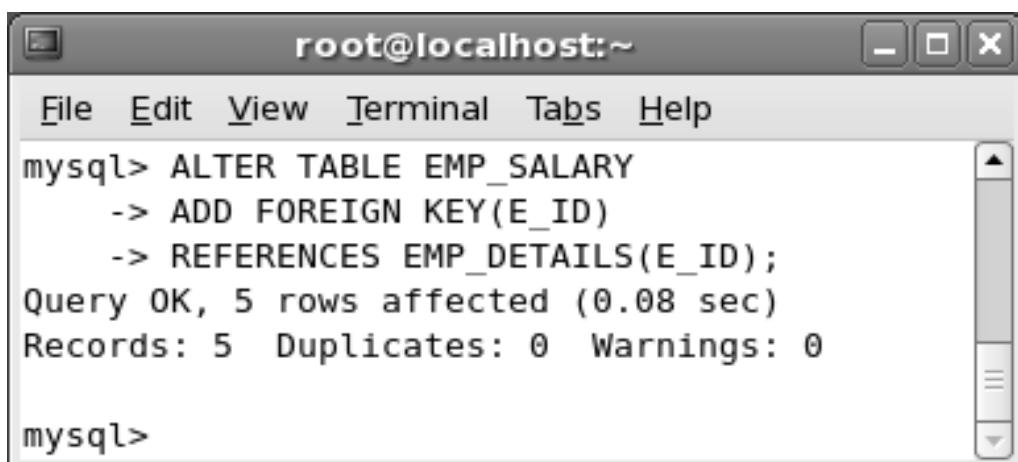
Option	Description
ON DELETE	Deletes the child row if the related parent row is deleted
ON UPDATE	Modifies the child row if the related parent row is modified
CASCADE	Delete or update the row in the parent table and automatically delete or update corresponding rows in the child table
SET NULL	Deletes or modifies the row in the parent table and sets the corresponding row in the child table to NULL provided NOT NULL is not specified in the foreign key column
NO ACTION	Prevents a delete or update to the primary key if a foreign key value exists
RESTRICT	Prevents the deletion or updation of row in the parent table if a dependent child row exists

Table 8.1: Foreign Key Options

To define a foreign key, enter the following command at the command prompt:

```
ALTER TABLE EMP_SALARY
ADD FOREIGN KEY(E_ID)
REFERENCES EMP_DETAILS(E_ID);
```

Figure 8.2 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area contains the following MySQL command and its output:

```
mysql> ALTER TABLE EMP_SALARY
      -> ADD FOREIGN KEY(E_ID)
      -> REFERENCES EMP_DETAILS(E_ID);
Query OK, 5 rows affected (0.08 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql>
```

Figure 8.2: Defining a Foreign Key

8.2.2 Creating Indexes

You can use indexes to improve the search characteristics of the database. For example, if you have a table with thousands of rows of data then searching for a particular row would take time. Indexing enables the search operation on a database to be faster. MySQL creates separate files to store and track indexes when you define an index for a table.

MySQL allows upto 64 indexes for each table. Each index allows upto 16 columns to be included. You can define a multicolumn index when you have to search on the same set of multiple columns. Indexing improves the speed of data retrieval. Indexing also has certain disadvantages. The process of altering data in a database becomes slow. The process such as update, insert, and delete require more time to execute. MySQL requires this additional time to record the changes in the index file also.

In MySQL, indexes can be created either during the initial Data Definition Language (DDL) to create the table or during later operations. You can index the field that:

- Is used in the WHERE clause of query
- Is used in ORDER BY clause of the query
- Is having a unique value

If you do not specify an index key, MySQL automatically indexes the primary key.

The syntax to create an index for a table is:

```
CREATE INDEX index_name ON tablename (column1,column2,..., columnN);
```

where,

CREATE INDEX – appends an index to the table

index_name – specifies the name for the index

tablename – specifies the name of the table to create the index

column1 – specifies the names of the columns to be indexed in the table

Note: MySQL enables you to create a unique index and eliminate the duplicate values. You can use the UNIQUE clause to define an unique index.

To create an index named INDEX1 on the columns ID and NAME while creating a SAMPLE table, enter the following command at the command prompt:

Session 8

Implementing SQL Queries Using MySQL - II

Concepts

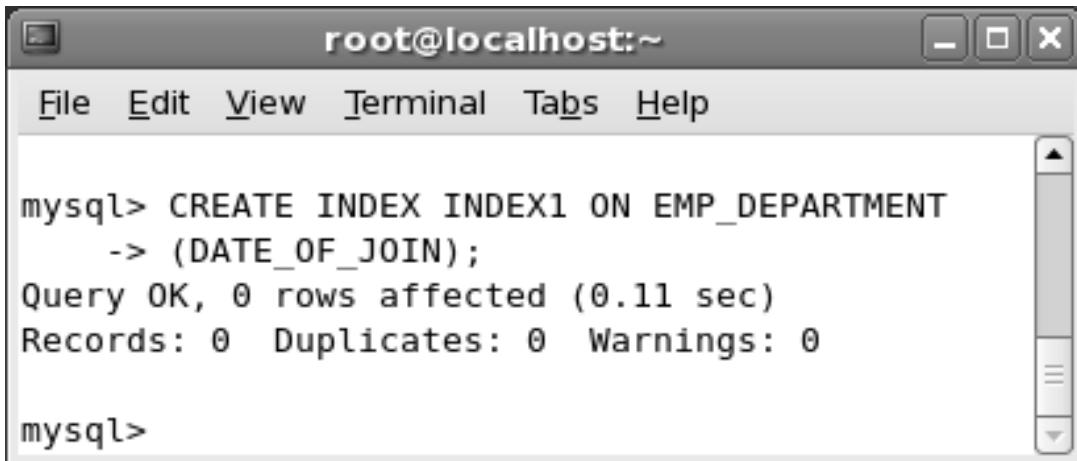
```
CREATE TABLE SAMPLE (ID INT(2) NOT NULL, NAME CHAR(10), INDEX INDEX1(ID, NAME),  
UNIQUE INDEX(ID));
```

You have already created a table. You will now create an index on the DATE_OF_JOIN field in the EMP_DEPARTMENT table to search data.

To create an index, INDEX1 on EMP_DEPARTMENT, enter the following command at the command prompt:

```
CREATE INDEX INDEX1 ON EMP_DEPARTMENT(DATE_OF_JOIN);
```

Figure 8.3 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a standard OS X-style title bar with minimize, maximize, and close buttons. The menu bar includes "File", "Edit", "View", "Terminal", "Tabs", and "Help". The main pane displays MySQL command-line output:

```
mysql> CREATE INDEX INDEX1 ON EMP_DEPARTMENT  
      -> (DATE_OF_JOIN);  
Query OK, 0 rows affected (0.11 sec)  
Records: 0  Duplicates: 0  Warnings: 0  
  
mysql>
```

Figure 8.3: Creating an Index

8.3 Manipulating Tables

MySQL only defines the structure of the table when you create a table. This table structure does not contain any data. You will have to add data into the table after creating it. You can also replace or modify the data after adding it to the table. In addition, you can delete data from a table if you do not require the data.

8.3.1 Inserting Data into a Table

You are required to add data to a table after creating it. MySQL provides the `INSERT` command to add new records into a table. MySQL appends the newly inserted record after the last record in the table if the table already contains records.

Session 8

Implementing SQL Queries Using MySQL - II

The syntax to insert data into a table is:

```
INSERT [LOW_PRIORITY | DELAYED] [IGNORE] [INTO] tbl_name [(col_name,...)]
VALUES ({expression | DEFAULT},...),(...),...
[ON DUPLICATE KEY UPDATE col_name=expression,...];
```

where,

`INSERT` – adds new row to an existing table

`tbl_name` – specifies the name of the table where MySQL will append the row

`col_name` – specifies the name of the column where MySQL will add the data

`VALUES` – defines the data that will be added to the table

The options for the `INSERT` command are listed in table 8.2.

Option	Description
<code>LOW_PRIORITY</code>	Postpones addition of records till all operations (reading) from the client have completed
<code>DELAYED</code>	Delays the insertion against all incoming <code>SELECT</code> statements
<code>IGNORE</code>	Overrides the error conditions for columns and adds the record to the table
<code>INTO</code>	Specifies the name of the table into which the columns are to be inserted
<code>col_name</code>	Specifies the name of the columns to insert values
<code>expression</code>	Includes a mathematical expression for a column
<code>DEFAULT</code>	Stores the default value into the specified columns
<code>ON DUPLICATE KEY UPDATE col_name=expression</code>	Updates the existing record if the addition of a new record creates a duplicate value in the <code>PRIMARY KEY</code> column

Table 8.2: INSERT Command Options

You can specify an expression for a column in the `INSERT` command. The condition to be satisfied is that the value for the column on which you build the expression should already be present.

MySQL allows you to insert values for specific column names. All other columns, if not specified, will contain a `NULL` value. If you have created columns to contain default data, then you can use the `DEFAULT` clause of the `INSERT` command to specify a particular column name and the value for that column. An error occurs if the column does not have a default value.

Session 8

Implementing SQL Queries Using MySQL - II

Note: If you have not set a default value for a column and the column is NOT NULL, then that column must have a value in the INSERT statement.

To insert rows in EMP_DEPARTMENT, enter the following command at the command prompt:

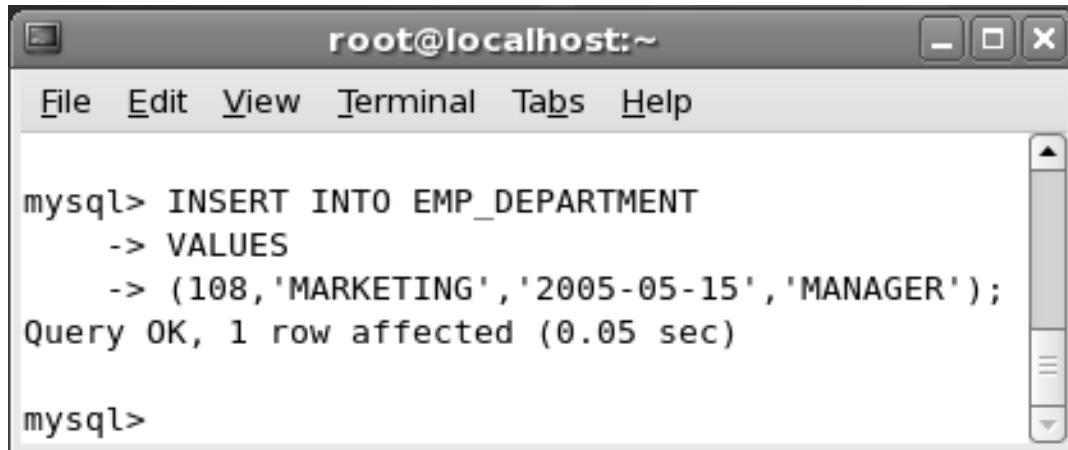
```
INSERT INTO EMP_DEPARTMENT (E_ID, D_NAME, DATE_OF_JOIN, DESIGNATION) VALUES  
(108, 'MARKETING', '2000-05-15', 'MANAGER');
```

Note: The number of columns and the number of values specified inside the parentheses with or without quotes must match when you insert data into a table. MySQL will return an error if these numbers do not match.

You can also enter the following command at the command prompt:

```
INSERT INTO EMP_DEPARTMENT VALUES (108, 'MARKETING', '2000-05-15',  
'MANAGER');
```

This command will work only if values for all the columns are specified following the VALUES keyword. Figure 8.4 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area contains the following MySQL command and its output:

```
mysql> INSERT INTO EMP_DEPARTMENT  
-> VALUES  
-> (108, 'MARKETING', '2005-05-15', 'MANAGER');  
Query OK, 1 row affected (0.05 sec)  
  
mysql>
```

Figure 8.4: Inserting Data in DEPARTMENT Table

Figure 8.4 displays a message that specifies the authenticity or correctness of the query and the number of rows affected by the INSERT query.

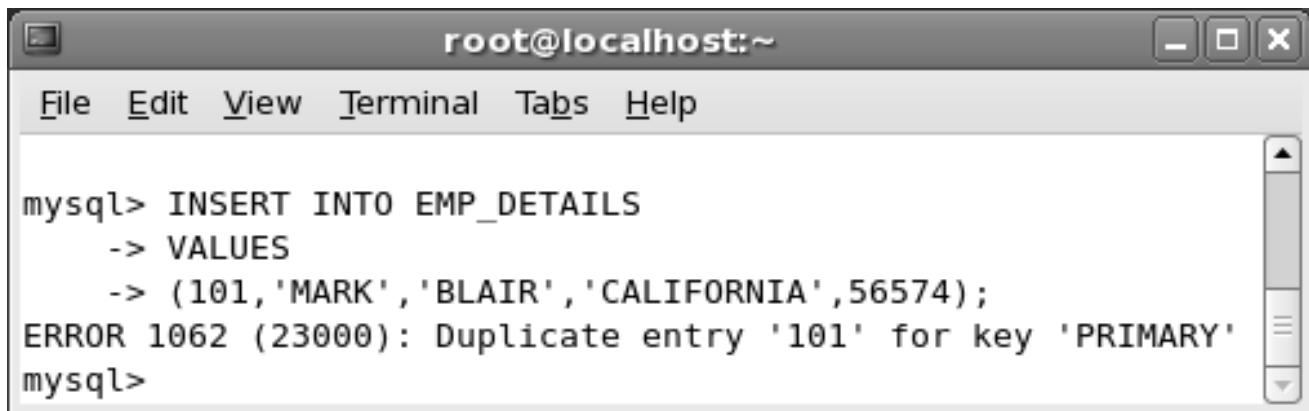
You had earlier defined the primary key of EMP_DEPARTMENT table as NOT NULL and AUTO_INCREMENT. Now you can try to insert a row having the E_ID that already exists, by entering the following command at the command prompt:

```
INSERT INTO EMP_DETAILS VALUES (101, 'MARK', 'BLAIR', 'CALIFORNIA', 56574);
```

Session 8

Implementing SQL Queries Using MySQL - II

Figure 8.5 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The menu bar includes "File", "Edit", "View", "Terminal", "Tabs", and "Help". The main area contains the following MySQL command and its output:

```
mysql> INSERT INTO EMP_DETAILS  
-> VALUES  
-> (101, 'MARK', 'BLAIR', 'CALIFORNIA', 56574);  
ERROR 1062 (23000): Duplicate entry '101' for key 'PRIMARY'  
mysql>
```

Figure 8.5: Inserting Data in EMP_DETAILS Table

Figure 8.5 displays an error message stating that MySQL does not allow a duplicate entry for the primary key.

A different form of the `INSERT` command enables you to set the column to a specific value. You can specify the column name and the values together. The syntax is as follows:

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE] [INTO] tbl_name  
SET col_name = {expression | DEFAULT},... [ON DUPLICATE KEY UPDATE col_  
name=expression,...];
```

where,

`INSERT` – adds data to the table

`tbl_name` – specifies the name of the table

`SET` – specifies the column names explicitly to which the values will be added

To insert rows in `EMP_DEPARTMENT`, enter the following command at the command prompt:

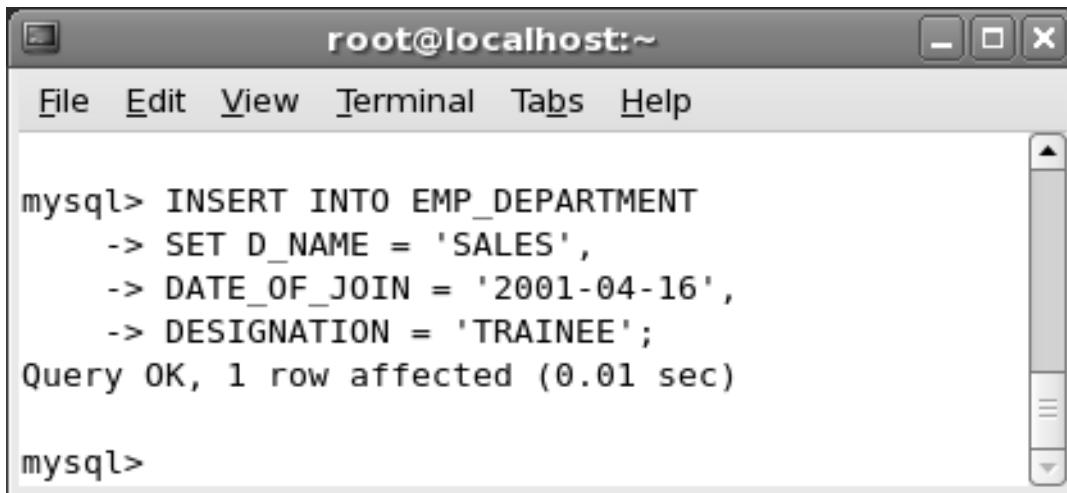
```
INSERT INTO EMP_DEPARTMENT SET D_NAME = 'SALES', DATE_OF_JOIN = '2001-04-  
16', DESIGNATION= 'TRAINEE';
```

Figure 8.6 displays the output of the command.

Session 8

Implementing SQL Queries Using MySQL - II

Concepts



The screenshot shows a terminal window titled "root@localhost:~". The menu bar includes "File", "Edit", "View", "Terminal", "Tabs", and "Help". The main area displays the following MySQL command and its output:

```
mysql> INSERT INTO EMP_DEPARTMENT
-> SET D_NAME = 'SALES',
-> DATE_OF_JOIN = '2001-04-16',
-> DESIGNATION = 'TRAINEE';
Query OK, 1 row affected (0.01 sec)

mysql>
```

Figure 8.6: Inserting Data in EMP_DEPARTMENT Table

MySQL also provides options to copy data across tables. The `INSERT` command enables you to add data to a table returned from a `SELECT` query.

```
INSERT [LOW_PRIORITY | HIGH_PRIORITY] [IGNORE] [INTO] tbl_name [(col_name,...)] SELECT ...;
```

where,

`INSERT` – adds data into the table

`tbl_name` – specifies the name of the table to add data

`col_name` – specifies the name of the column

`SELECT` – defines the data to be added in the table specified in the `tbl_name` clause. The data to be added will be retrieved using the conditions specified in the `SELECT` clause

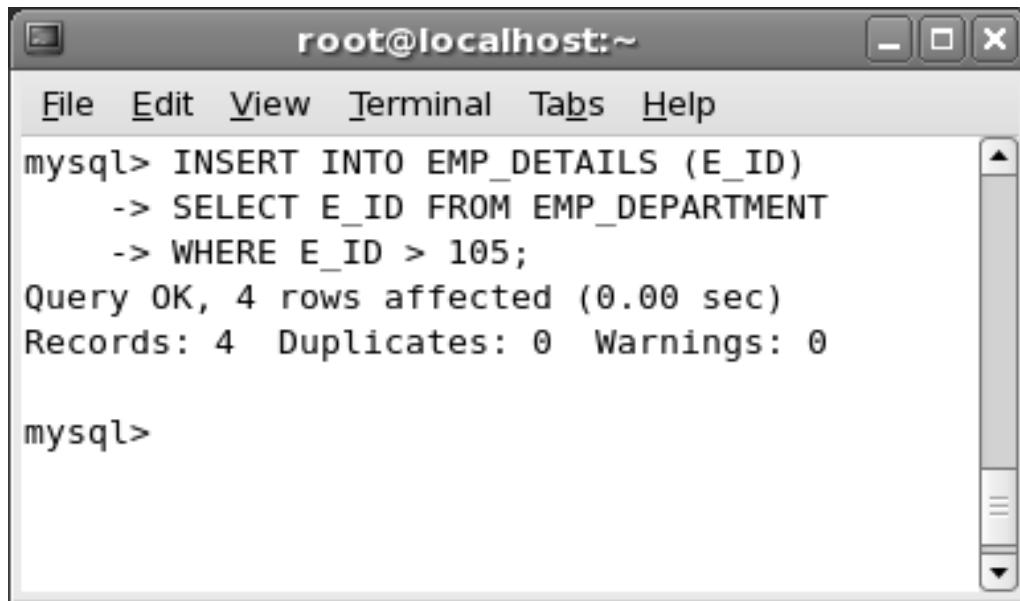
To insert `E_ID` in the `EMP_DETAILS` table where the `E_ID` is retrieved from the `EMP_DEPARTMENT` table using the `SELECT` command, enter the following command at the command prompt:

```
INSERT INTO EMP_DETAILS (E_ID)
SELECT E_ID FROM EMP_DEPARTMENT
WHERE E_ID > 105;
```

Figure 8.7 displays the output of the command.

Session 8

Implementing SQL Queries Using MySQL - II



```
root@localhost:~ File Edit View Terminal Tabs Help mysql> INSERT INTO EMP_DETAILS (E_ID)
-> SELECT E_ID FROM EMP_DEPARTMENT
-> WHERE E_ID > 105;
Query OK, 4 rows affected (0.00 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql>
```

Figure 8.7: Inserting Data into a Table Using `INSERT INTO...SELECT` Command

Figure 8.7 displays the number of rows that have been copied into the `EMP_DETAILS` table from the `EMP_DEPARTMENT` table. In the output, Records specify the number of rows processed by the query, Duplicates indicates the number of rows that could not be inserted and Warnings indicates the number of attempts that was undertaken to insert column values.

8.3.2 Updating Data in a Table

You may require changing or modifying data in a table. MySQL provides the `UPDATE` command to change or modify data within a table.

The syntax to update a table is:

```
UPDATE [LOW_PRIORITY] [IGNORE] tbl_name SET column=value, ... [WHERE clause] [LIMIT n];
```

where,

`UPDATE` – modifies the column data

`tbl_name` – specifies the name of the table to modify

`SET` – defines the value to change

`column` – specifies the name of the column

Session 8

Implementing SQL Queries Using MySQL - II

value – defines the new value to be entered for the column

The options for the UPDATE command are listed in table 8.3.

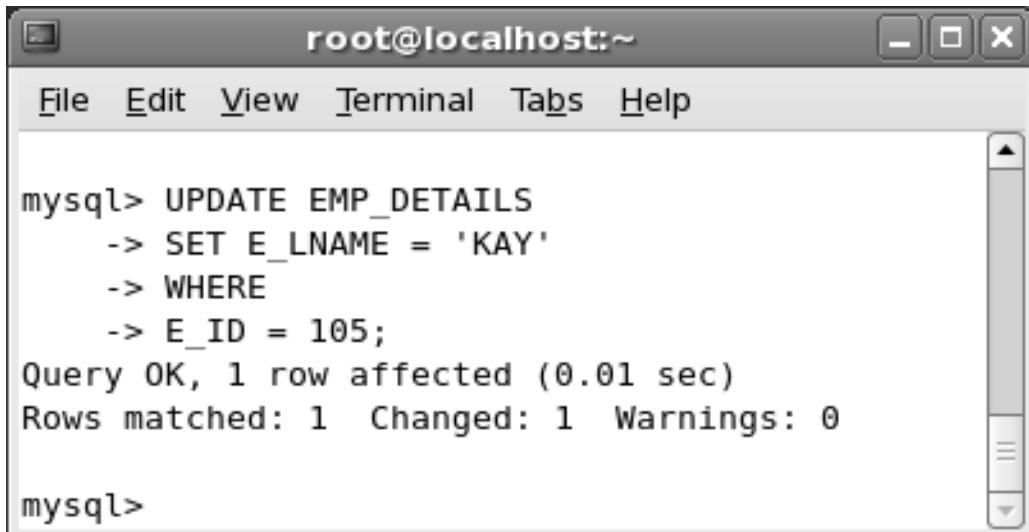
Option	Description
LOW_PRIORITY	Assigns a low priority to the update against all other commands
IGNORE	Updates a row even if there is an error
WHERE	Specifies the rows to be updated based on the specific condition
LIMIT	Specifies a limit on the number of rows that can be updated

Table 8.3: UPDATE Command Options

To update a row in EMP_DETAILS table, enter the following command at the command prompt:

```
UPDATE EMP_DETAILS SET E_LNAME = 'KAY' WHERE E_ID = 105;
```

Figure 8.8 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area contains the following MySQL command and its output:

```
mysql> UPDATE EMP_DETAILS
      -> SET E_LNAME = 'KAY'
      -> WHERE
      -> E_ID = 105;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql>
```

Figure 8.8: Updating a Table

Figure 8.8 displays the result of the UPDATE command. The UPDATE command returns the number of row that was changed. When you include the ORDER BY clause in an UPDATE command, MySQL will update the rows in the order specified in the ORDER BY clause. You can include the ORDER BY clause in an UPDATE command to modify records in a table sorted on specific conditions.

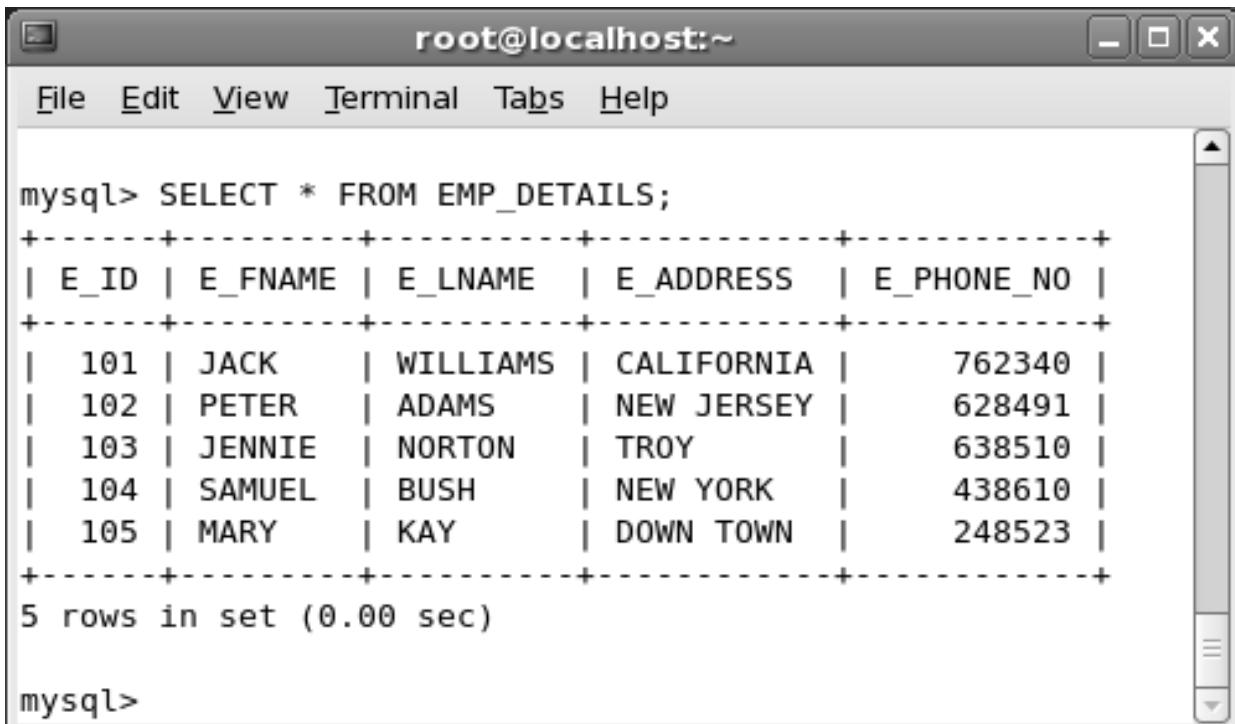
You can now use the SELECT command to view the updated data in the table. To view the updated results, enter the following command at the command prompt:

Session 8

Implementing SQL Queries Using MySQL - II

```
SELECT * FROM EMP_DETAILS;
```

Figure 8.9 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL session:

```
mysql> SELECT * FROM EMP_DETAILS;
+-----+-----+-----+-----+-----+
| E_ID | E_FNAME | E_LNAME | E_ADDRESS | E_PHONE_NO |
+-----+-----+-----+-----+-----+
| 101  | JACK    | WILLIAMS | CALIFORNIA | 762340   |
| 102  | PETER   | ADAMS    | NEW JERSEY  | 628491   |
| 103  | JENNIE  | NORTON   | TROY        | 638510   |
| 104  | SAMUEL  | BUSH     | NEW YORK   | 438610   |
| 105  | MARY    | KAY      | DOWN TOWN  | 248523   |
+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql>
```

Figure 8.9: Viewing Data after Updating

In figure 8.9, you can observe the change in data in the row as result of the UPDATE command.

8.3.3 Replacing Data in a Table

MySQL provides the REPLACE command to substitute data in the table. The REPLACE command is similar to the INSERT command, except that it replaces the old value of a column with a new value. The REPLACE command will delete the old row only if the new row satisfies the conditions of the column definitions. If the value in the REPLACE command does not match the existing record then MySQL executes it as an INSERT statement.

The syntax for replacing data in a table is:

```
REPLACE [DELAYED|LOW_PRIORITY] INTO tbl_name [(column,...)] VALUES  
(value,...);
```

where,

REPLACE – substitutes the column data

Session 8

Implementing SQL Queries Using MySQL - II

tbl_name – specifies the name of the table to modify

value – defines the new value for the column

The options of the REPLACE command are listed in table 8.4.

Concepts

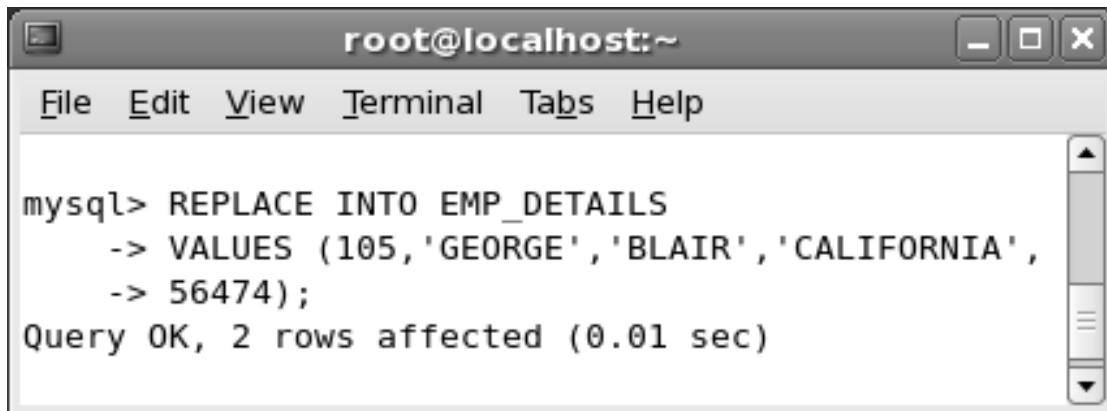
Option	Description
DELAYED	Postpones the execution of the command until no client is accessing the table
LOW_PRIORITY	Specifies the rows to be inserted into a buffer
column	Specifies the column names for which the values are to be replaced
value	Defines the new values for the columns

Table 8.4: REPLACE Command Options

To replace a row in EMP_DETAILS, enter the following command at the command prompt:

```
REPLACE INTO EMP_DETAILS  
VALUES (105,'GEORGE','BLAIR', 'CALIFORNIA',56474);
```

Figure 8.10 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a standard Linux-style title bar with icons for minimize, maximize, and close. Below the title bar is a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The main area of the terminal shows the MySQL command-line interface. A user has entered the following command:

```
mysql> REPLACE INTO EMP_DETAILS  
-> VALUES (105,'GEORGE','BLAIR', 'CALIFORNIA',  
-> 56474);  
Query OK, 2 rows affected (0.01 sec)
```

The terminal window also features scroll bars on the right side.

Figure 8.10: Replacing data into EMP_DETAILS Table

MySQL displays the number of rows affected by the REPLACE query. This includes the number of old rows that have been deleted and the number of new rows that have been added to the table. If MySQL returns the number of rows affected as one for a single row modification, it means that a row has been added and existing rows have not been deleted. When the number of rows affected is more than one, it means that an existing record has been deleted and a new record has been added to the table.

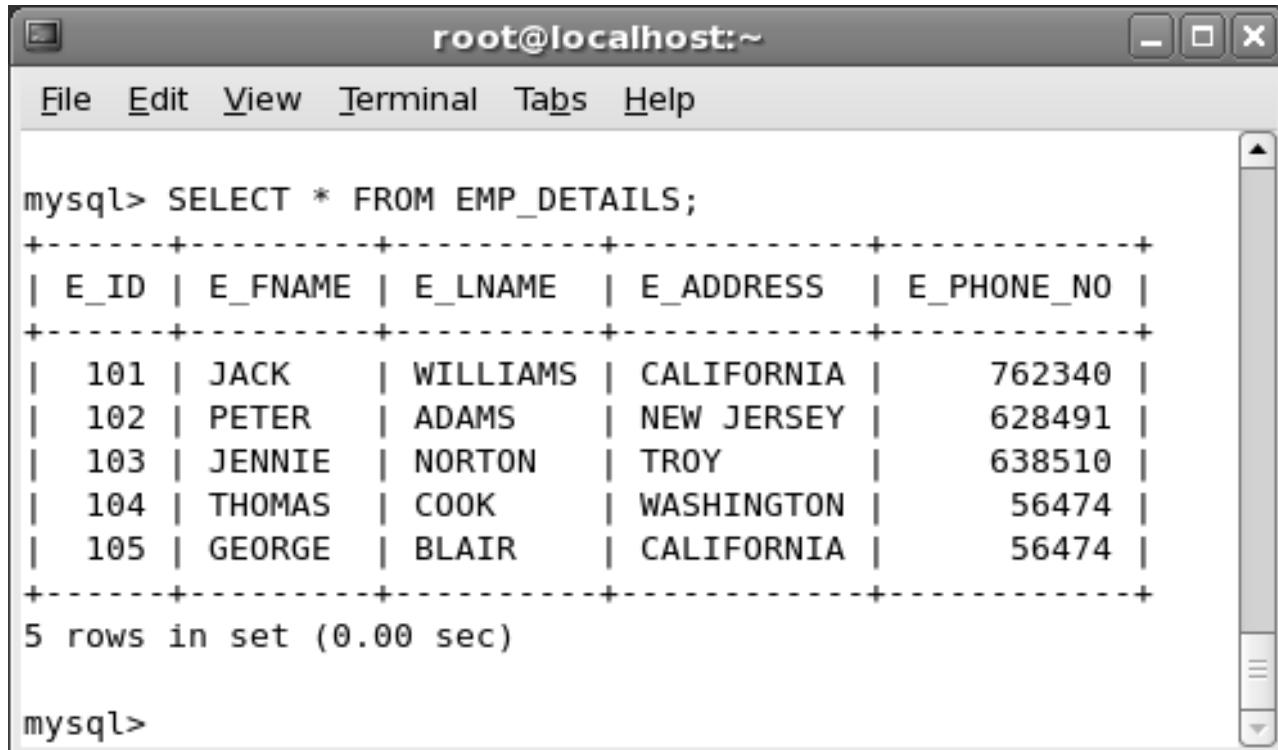
To view the data in the EMP_DETAILS table, enter the following command at the command prompt:

Session 8

Implementing SQL Queries Using MySQL - II

```
SELECT * FROM EMP_DETAILS;
```

Figure 8.11 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its output:

```
mysql> SELECT * FROM EMP_DETAILS;
+-----+-----+-----+-----+-----+
| E_ID | E_FNAME | E_LNAME | E_ADDRESS | E_PHONE_NO |
+-----+-----+-----+-----+-----+
| 101  | JACK    | WILLIAMS | CALIFORNIA | 762340   |
| 102  | PETER   | ADAMS    | NEW JERSEY  | 628491   |
| 103  | JENNIE  | NORTON   | TROY        | 638510   |
| 104  | THOMAS  | COOK     | WASHINGTON  | 56474    |
| 105  | GEORGE  | BLAIR    | CALIFORNIA | 56474    |
+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql>
```

Figure 8.11: Viewing Data After REPLACE command

Figure 8.11 displays the change in row data as a result of the REPLACE command.

You can also use the SELECT query within a REPLACE command. The syntax to include a SELECT query within a REPLACE command is:

```
REPLACE [DELAYED|LOW_PRIORITY] INTO tbl_name [(column,...)] SELECT select_clause;
```

where,

REPLACE – substitutes the column data

tbl_name – specifies the name of the table to modify

SELECT – executes the select query specified in the clause

Session 8

Implementing SQL Queries Using MySQL - II

The `REPLACE` command also enables you to specify the column names and the values together. MySQL assigns a `NULL` value to the columns if you fail to specify a value in the `REPLACE` command.

Concepts

```
REPLACE [DELAYED|LOW_PRIORITY] INTO tbl_name SET column=value,  
column=value, ...;
```

where,

`REPLACE` – substitutes the column data

`tbl_name` – specifies the name of the table to modify

`SET` – defines the new value to be entered in the column

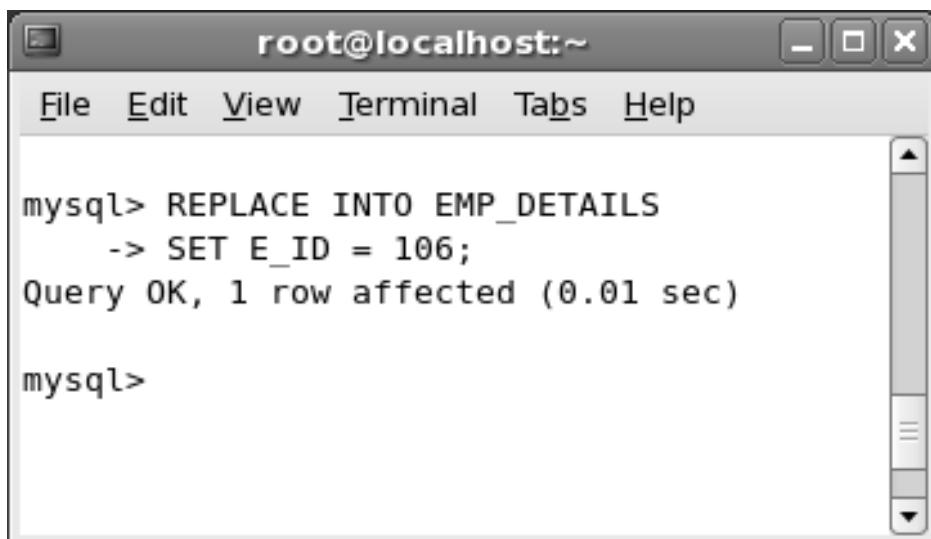
`column` – specifies the name of the column

`value` – defines the new value for the specified column

For example, to add a new record in the `EMP_DETAILS` table, where you only specify the `E_ID`, enter the following command at the command prompt:

```
REPLACE INTO EMP_DETAILS  
SET E_ID = 106;
```

Figure 8.12 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The main area contains the following MySQL session:

```
mysql> REPLACE INTO EMP_DETAILS  
-> SET E_ID = 106;  
Query OK, 1 row affected (0.01 sec)  
  
mysql>
```

Figure 8.12: Adding Data into Table using `REPLACE` Command

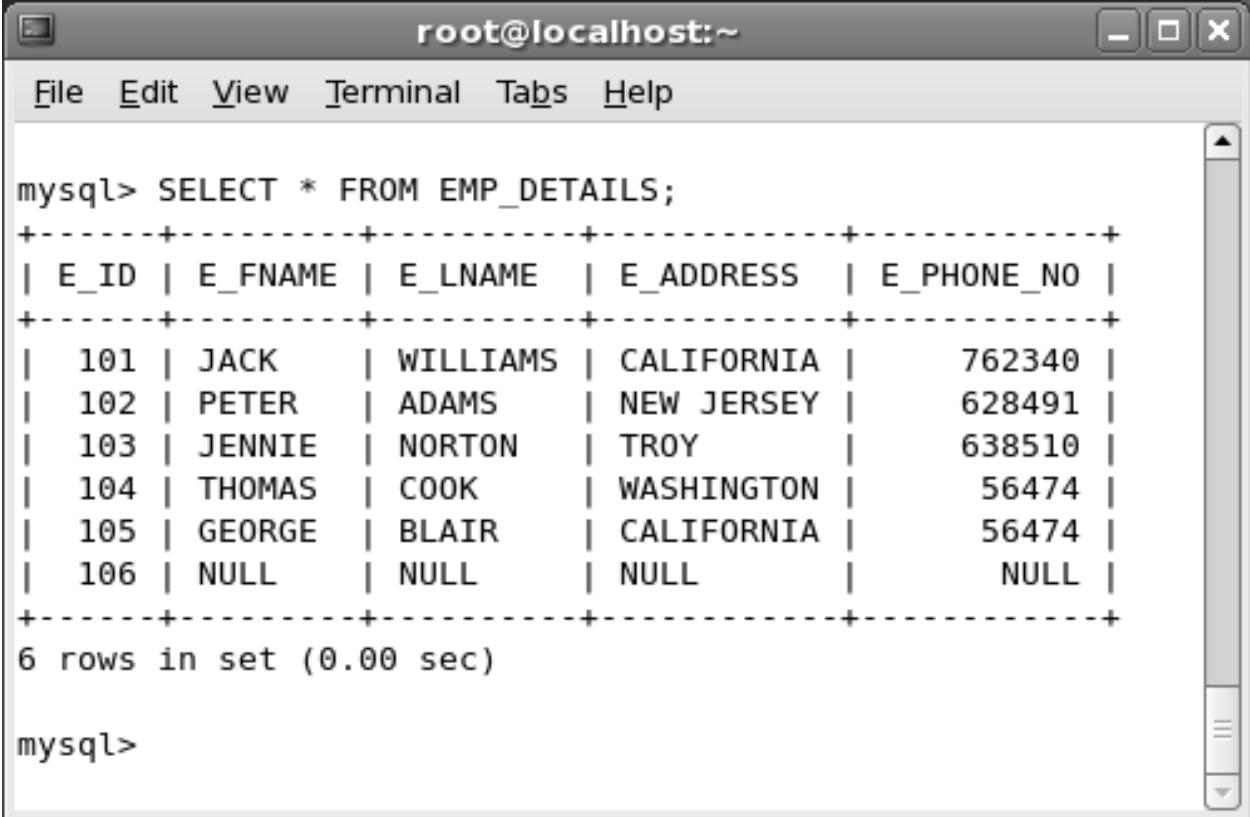
Session 8

Implementing SQL Queries Using MySQL - II

Figure 8.12 displays the results of the REPLACE query. In this command, you have specified a value only for the E_ID field. MySQL will assign NULL values to the other columns in the table for this record. To view the NULL values for this record, enter the following command at the command prompt:

```
SELECT * FROM EMP_DETAILS;
```

Figure 8.13 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL session:

```
mysql> SELECT * FROM EMP_DETAILS;
+-----+-----+-----+-----+-----+
| E_ID | E_FNAME | E_LNAME | E_ADDRESS | E_PHONE_NO |
+-----+-----+-----+-----+-----+
| 101  | JACK    | WILLIAMS | CALIFORNIA | 762340    |
| 102  | PETER   | ADAMS    | NEW JERSEY | 628491    |
| 103  | JENNIE  | NORTON   | TROY       | 638510    |
| 104  | THOMAS  | COOK     | WASHINGTON | 56474     |
| 105  | GEORGE  | BLAIR    | CALIFORNIA | 56474     |
| 106  | NULL    | NULL     | NULL       | NULL      |
+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql>
```

Figure 8.13: Viewing NULL Data after REPLACE

8.3.4 Deleting Data from a Table

MySQL stores data in a table in the form of rows. You may require deleting old and obsolete data from the table. MySQL provides the `DELETE` command to remove rows from a table. You can choose to remove selected rows of a table or the entire table.

The syntax to delete rows from a table is:

```
DELETE [LOW_PRIORITY|QUICK] [IGNORE] FROM tbl_name [WHERE clause] [ORDER BY column,...] [LIMIT n];
```

Session 8

Implementing SQL Queries Using MySQL - II

where,

DELETE – removes data from the table

tbl_name – specifies the name of the table from which rows are to be deleted

The options for the DELETE command are provided in table 8.5.

Concepts

Option	Description
LOW_PRIORITY	Postpones deletion until all other operations on the table have been completed by the client such as reading
QUICK	Suspends the merging of indexes during execution
IGNORE	Ignores error while deleting a row
WHERE	Deletes rows that satisfy the specified conditions
ORDER BY	Deletes data in the order specified
LIMIT n	Specifies the maximum number of rows to be deleted in a single attempt

Table 8.5: DELETE Command Options

If you use the DELETE command without the WHERE clause, the entire table will be deleted. If you specify the WHERE clause in the DELETE command, the system prompts a message displaying the number of rows that will be deleted. If no WHERE clause is used, all the rows are deleted. MySQL returns a '0' value, when the entire table is deleted because MySQL cannot identify the number of rows to be deleted.

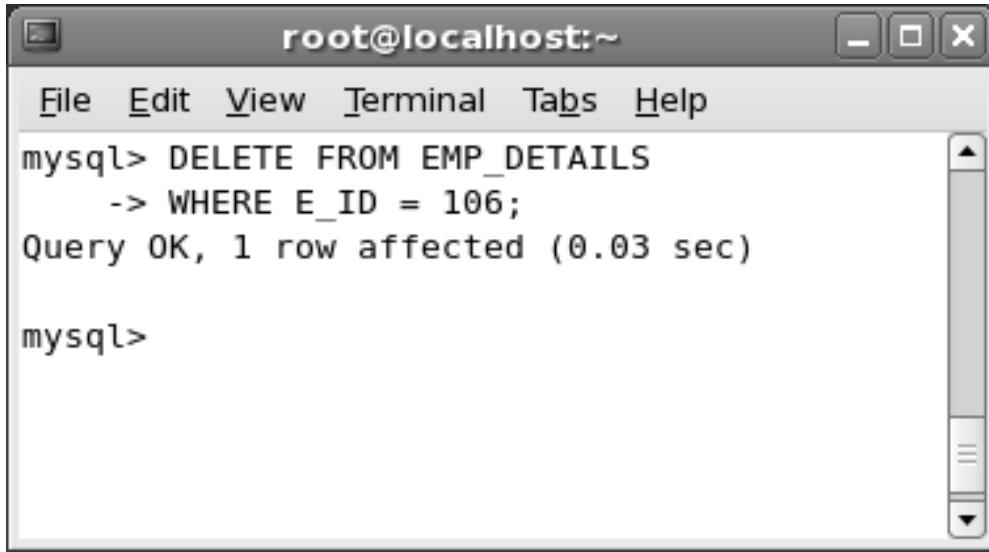
Note: The DELETE command will only remove all the data from the table. It does not remove the table structure. The DELETE command without the WHERE clause will delete all the files associated with the table except for the file that contains the table structure.

To delete a row from the EMP_DETAILS table, enter the following command at the command prompt:

```
DELETE FROM EMP_DETAILS WHERE E_ID=106;
```

Figure 8.14 displays the output of the command.

Session 8



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The main area contains the following MySQL command and its output:

```
mysql> DELETE FROM EMP_DETAILS  
      -> WHERE E_ID = 106;  
Query OK, 1 row affected (0.03 sec)  
  
mysql>
```

Figure 8.14: Deleting Data from a Table

MySQL displays a message containing the number of rows affected by the execution of the `DELETE` query.

8.4 Using the ORDER BY Clause

You use the `SELECT` command to retrieve data from a database. MySQL displays the `SELECT` command result in the order in which the data is inserted in the table.

You can also sort the output on more than one column in ascending or descending order. You will use the `ORDER BY` clause with `SELECT` command to sort the data. The use of `DISTINCT` keyword causes only one row of data to be displayed for every group of rows that is identical.

The syntax for `ORDER BY` is:

```
SELECT column_name FROM tbl_name ORDER BY column [ASC|DESC] [,column2[ASC|DESC],...];
```

where,

`SELECT` – retrieves data from the table

`column_name` – specifies the name for the column to retrieve data from

`tbl_name` – specifies the name of the table that contains the column and data

Session 8

Implementing SQL Queries Using MySQL - II

ORDER BY – sorts the column data in the specified order

column – specifies the name of the column

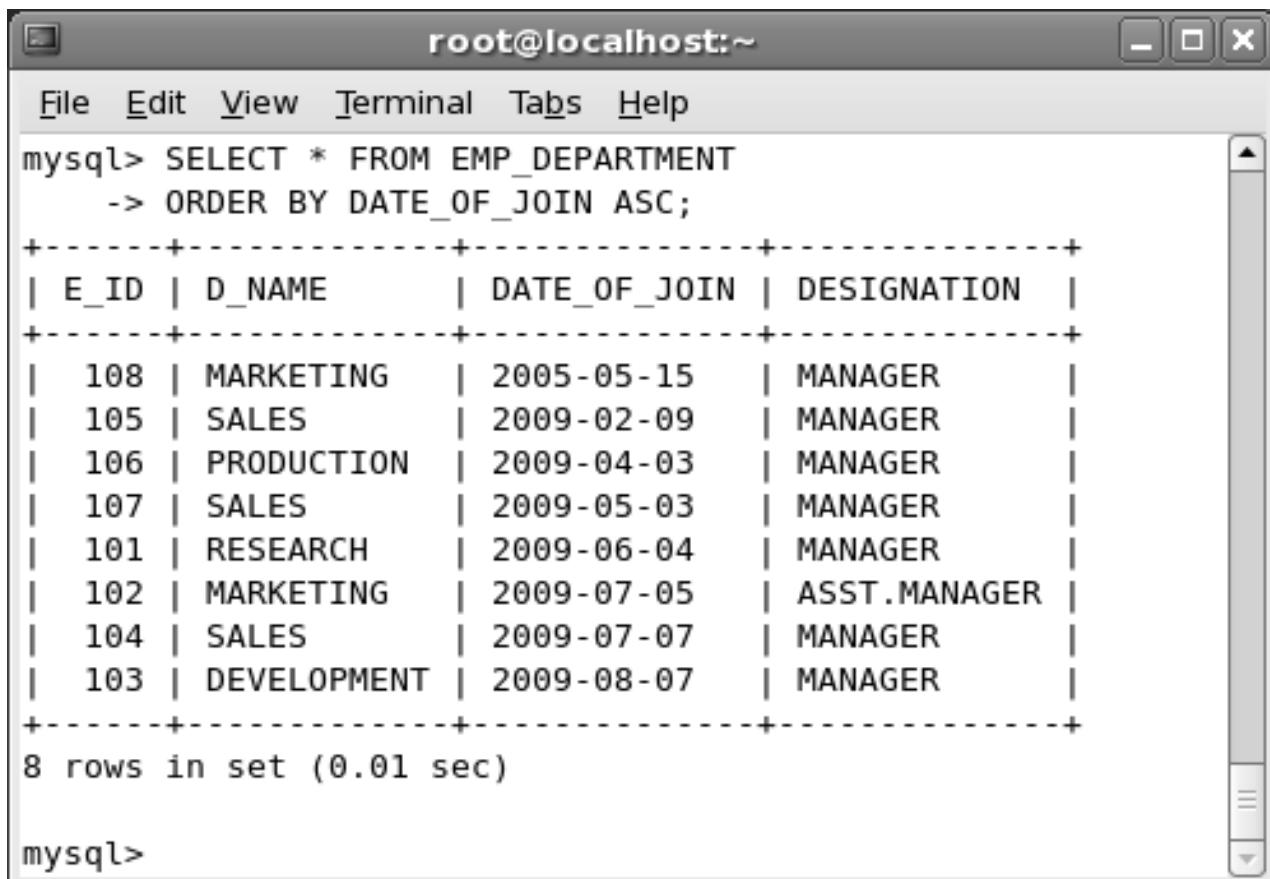
ASC | DESC – specifies the sort direction for data. MySQL supports two types of sorting. Ascending order sorts the data from lowest to highest whereas Descending order sorts the data from highest to lowest

To organize the output of EMP_DEPARTMENT table on the DATE_OF_JOIN field in an ascending order, enter the following command at the command prompt:

```
SELECT * FROM EMP_DEPARTMENT ORDER BY DATE_OF_JOIN ASC;
```

Figure 8.15 displays the output of the command.

Concepts



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following text:

```
File Edit View Terminal Tabs Help
mysql> SELECT * FROM EMP_DEPARTMENT
-> ORDER BY DATE_OF_JOIN ASC;
+-----+-----+-----+
| E_ID | D_NAME      | DATE_OF_JOIN | DESIGNATION |
+-----+-----+-----+
| 108  | MARKETING   | 2005-05-15  | MANAGER     |
| 105  | SALES        | 2009-02-09  | MANAGER     |
| 106  | PRODUCTION   | 2009-04-03  | MANAGER     |
| 107  | SALES        | 2009-05-03  | MANAGER     |
| 101  | RESEARCH     | 2009-06-04  | MANAGER     |
| 102  | MARKETING   | 2009-07-05  | ASST.MANAGER|
| 104  | SALES        | 2009-07-07  | MANAGER     |
| 103  | DEVELOPMENT  | 2009-08-07  | MANAGER     |
+-----+-----+-----+
8 rows in set (0.01 sec)

mysql>
```

Figure 8.15: Organizing Data

You will now sort the data in the EMP_DETAILS table on the E_FNAME and E_LNAME columns. Also, you will sort data from the E_FNAME column in the descending order.

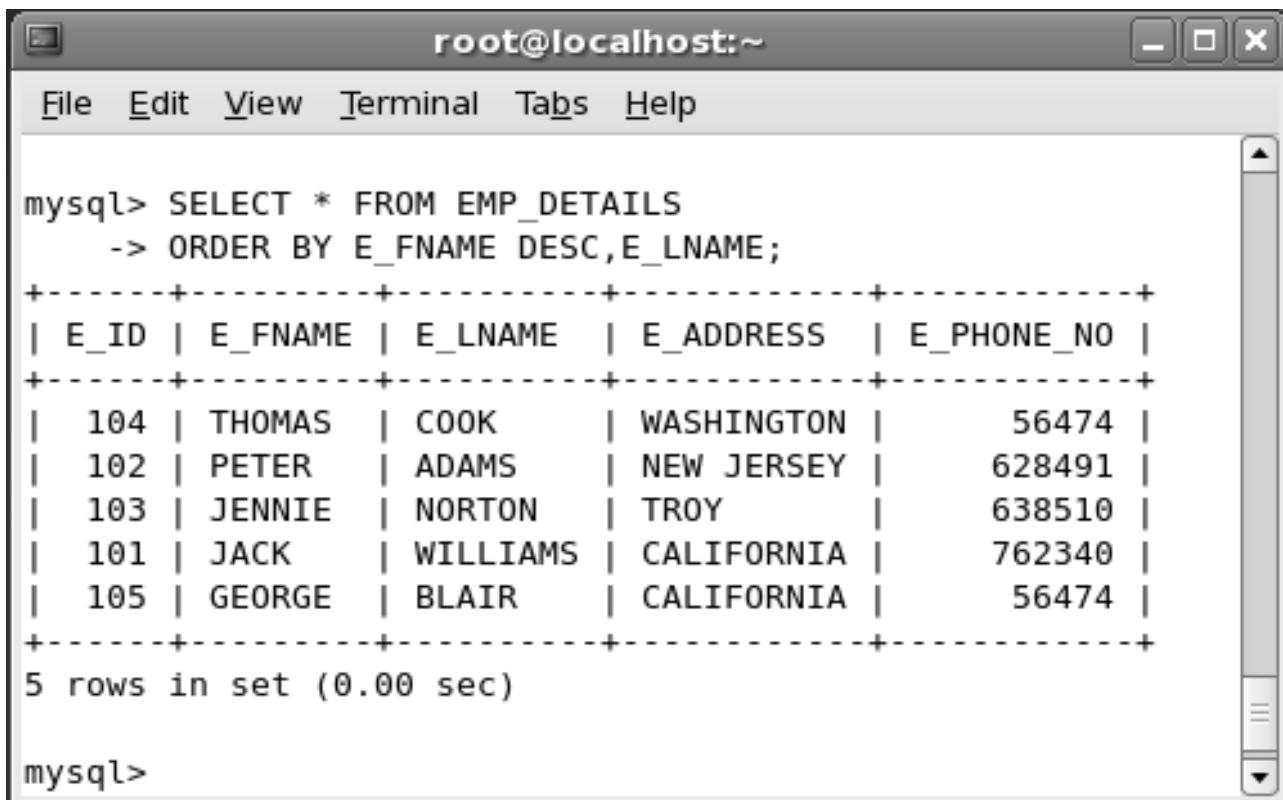
Session 8

Implementing SQL Queries Using MySQL - II

To sort the output of `EMP_DETAILS` table on the `E_FNAME` field in the descending order and then by `E_LNAME` field, enter the following command at the command prompt:

```
SELECT * FROM EMP_DETAILS ORDER BY E_FNAME DESC, E_LNAME;
```

Figure 8.16 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following text:

```
mysql> SELECT * FROM EMP_DETAILS
      -> ORDER BY E_FNAME DESC,E_LNAME;
+-----+-----+-----+-----+
| E_ID | E_FNAME | E_LNAME | E_ADDRESS | E_PHONE_NO |
+-----+-----+-----+-----+
| 104 | THOMAS | COOK    | WASHINGTON | 56474      |
| 102 | PETER   | ADAMS   | NEW JERSEY | 628491     |
| 103 | JENNIE  | NORTON  | TROY       | 638510     |
| 101 | JACK    | WILLIAMS | CALIFORNIA | 762340     |
| 105 | GEORGE  | BLAIR   | CALIFORNIA | 56474      |
+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql>
```

Figure 8.16: Ordering the data by `E_FNAME` and `E_LNAME`

8.5 Using the GROUP BY Clause

You may require to group data from columns. MySQL provides the `GROUP BY` command to group rows with similar values for a specific column into a single row. This will enable you to execute tasks on these grouped rows.

The syntax for grouping data is:

```
SELECT column_name FROM tbl_name GROUP BY column [,column2...];
```

Session 8

Implementing SQL Queries Using MySQL - II

where,

SELECT – retrieves data from the table

column_name – specifies the column name to retrieve data from

tbl_name – specifies the name of the table that contains the column and data

GROUP BY – categorizes the data

column – specifies the name of the column in the table

You can group the employees based on their designations from the EMP_DEPARTMENT table on the DESIGNATION column.

To group the output of EMP_DEPARTMENT table on the DESIGNATION field, enter the following command at the command prompt:

```
SELECT * FROM EMP_DEPARTMENT GROUP BY DESIGNATION;
```

Figure 8.17 displays the output of the command.

The screenshot shows a terminal window titled "root@localhost:~". The window has a standard Linux-style title bar with icons for minimize, maximize, and close. Below the title bar is a menu bar with options: File, Edit, View, Terminal, Tabs, Help. The main area of the terminal shows the MySQL command-line interface. The user has entered the following SQL query:

```
mysql> SELECT * FROM EMP_DEPARTMENT
-> GROUP BY DESIGNATION;
```

After executing the query, the terminal displays the results in a tabular format:

E_ID	D_NAME	DATE_OF_JOIN	DESIGNATION
102	MARKETING	2009-07-05	ASST.MANAGER
101	RESEARCH	2009-06-04	MANAGER

Below the table, the message "2 rows in set (0.02 sec)" is displayed. The MySQL prompt "mysql>" appears again at the bottom of the terminal window.

Figure 8.17: GROUP BY Clause



Summary

- Keys help to identify a record in the table.
- A primary key uniquely identifies the records in a table.
- A foreign key in a table must have a corresponding primary key in a different table.
- A primary key and a foreign key can be defined while creating a table.
- The `ALTER TABLE` command enables to define keys in a table.
- The `INSERT` command allows you to add a row in a table.
- The `UPDATE` command allows you to modify the values present in the columns of a table.
- MySQL enables you to substitute values in a column using the `REPLACE` command.
- MySQL provides the `DELETE` command to remove unwanted or obsolete rows in a table.
- The `ORDER BY` clause can be used with the `SELECT` command to sort the data.
- Grouping categorizes rows with similar values for a specific column into a single row to execute combined data processing operations.

Session 8

Implementing SQL Queries Using MySQL - II



Check Your Progress

Concepts

1. While creating a foreign key _____ option prevents the deletion of parent row if a dependent child row exists.
 - a. RESTRICT
 - b. NO ACTION
 - c. ON DELETE
 - d. ON UPDATE

2. The _____ option of INSERT command postpones the addition of records unless all the clients finish reading from the table.
 - a. LOW_PRIORITY
 - b. IGNORE
 - c. DELAYED
 - d. DEFAULT

3. If you use the _____ option with the UPDATE command then the conflicting rows will not be updated.
 - a. IGNORE
 - b. WHERE
 - c. LIMIT
 - d. LOW_PRIORITY

4. The _____ option of the DELETE command suspends the merging of indexes.
 - a. LOW_PRIORITY
 - b. QUICK



Check Your Progress

- c. LIMIT
 - d. ORDER BY
5. Grouping combines rows with _____ in order to operate on them together.
- a. values for a specific column into multiple rows
 - b. values for a specific row
 - c. values for a specific column into a single row
 - d. values for specific columns into a single column

Objectives

At the end of this session, the student will be able to:

- *Define keys and indexes.*
- *Use the INSERT, ALTER, UPDATE, and DELETE statements.*
- *Use the GROUP BY command.*
- *Use the ORDER BY command.*

The steps given in this session are detailed, comprehensive, and carefully thought-through in order to meet the learning objectives and understand the tool completely. Please follow the steps carefully.

Part I - For the first 1.5 hours:

Working with Keys and Indexes

A key is the unique identifier in a table. A column that uniquely identifies a record is defined as the primary key. You index a column on which you wish to search the data frequently.

1. To activate the LIBRARY database, enter the following command at the command prompt:

```
USE LIBRARY;
```

Figure 9.1 displays the output of the command.

Session 9

Implementing SQL Queries Using MySQL-II (Lab)

```
root@localhost:~  
File Edit View Terminal Tabs Help  
  
mysql> USE LIBRARY;  
Reading table information for completion of table and column names  
You can turn off this feature to get a quicker startup with -A  
  
Database changed  
mysql>
```

Figure 9.1: Activating the Library Database

2. The librarian has identified two new members who have the same User IDs. This is causing problems while issuing books. To resolve this problem, the librarian will define a primary key on the USER_ID field in the USER_DETAILS table. To modify the USER_ID field as the primary key in the USER_DETAILS table, enter the following command at the command prompt.

```
ALTER TABLE USER_DETAILS MODIFY USER_ID INT(3)  
NOT NULL PRIMARY KEY AUTO_INCREMENT;
```

Figure 9.2 displays the output of the command.

```
root@localhost:~  
File Edit View Terminal Tabs Help  
  
mysql> ALTER TABLE USER_DETAILS  
-> MODIFY USER_ID INT(3)  
-> NOT NULL PRIMARY KEY  
-> AUTO_INCREMENT;  
Query OK, 0 rows affected (0.00 sec)  
Records: 0 Duplicates: 0 Warnings: 0  
  
mysql>
```

Figure 9.2: Defining the Primary key of USER_DETAILS Table

Session 9

Implementing SQL Queries Using MySQL-II (Lab)

Now, create an index for the CATEGORY table to enhance sorting and searching operations.

3. The time taken to search and retrieve data from the CATEGORY table is taking longer than required. An index can be defined on the USER_ID column of the CATEGORY table to resolve the problem. To create an index key named INDEX1 on the USER_ID field of the CATEGORY table, enter the following command at the command prompt:

```
CREATE INDEX INDEX1 ON CATEGORY(USER_ID);
```

Figure 9.3 displays the output of the command.

```
root@localhost:~
File Edit View Terminal Tabs Help
mysql> CREATE INDEX INDEX1 ON CATEGORY(USER_ID);
Query OK, 0 rows affected (0.08 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

Figure 9.3: Creating Index on the USER_ID Field

4. The librarian wants to add information about a new member in the USER_DETAILS table. To insert a record into the USER_DETAILS table, enter the following command at the command prompt:

```
INSERT INTO USER_DETAILS VALUES(100, 'TOBY', 'MEXICO', 45637);
```

Figure 9.4 displays the output of the command.

```
root@localhost:~
File Edit View Terminal Tabs Help
mysql> INSERT INTO USER_DETAILS VALUES(100, 'TOBY', 'MEXICO', 45637);
Query OK, 1 row affected (0.00 sec)

mysql>
```

Figure 9.4: Inserting Data

A primary key can be defined as AUTO_INCREMENT. This feature automatically generates a new unique value for the record when NULL value is specified in the input. Now, insert a NULL value for the primary key and view the result.

Session 9

Implementing SQL Queries Using MySQL-II (Lab)

5. The librarian is adding a user to the database. The librarian requires the system to generate the USER_ID for the user. To insert a record into the USER_DETAILS table, enter the following command at the command prompt:

```
INSERT INTO USER_DETAILS VALUES(NULL, 'BINYC', 'MEXICO', 45663);
```

Figure 9.5 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area contains the following text:

```
mysql> INSERT INTO USER_DETAILS VALUES(NULL, 'BINYC', 'MEXICO', 45663);
Query OK, 1 row affected (0.04 sec)

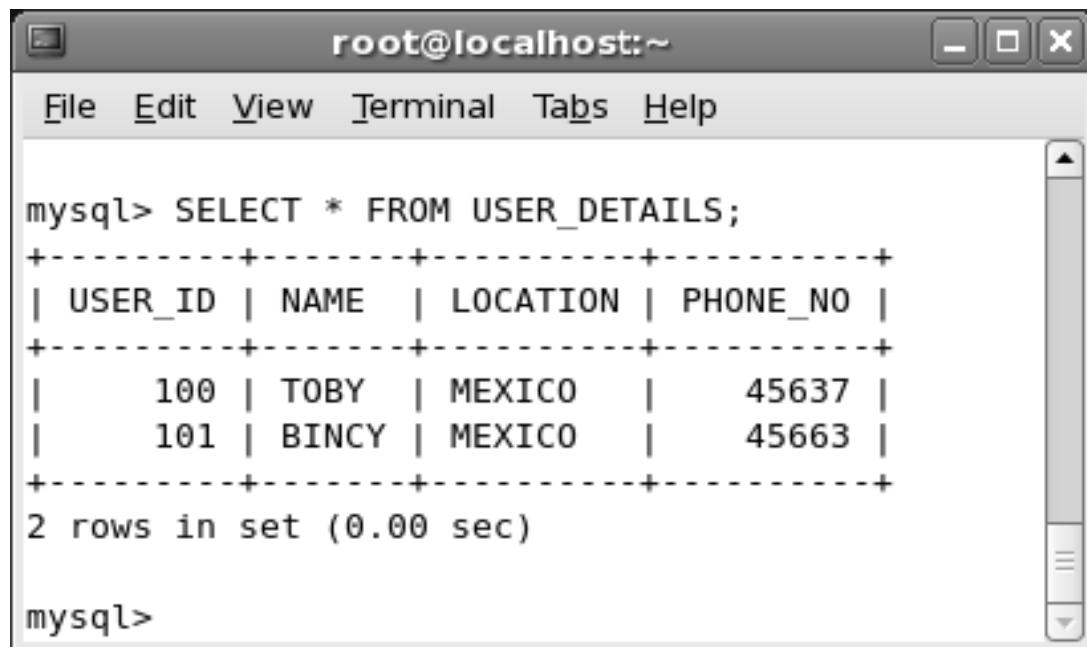
mysql>
```

Figure 9.5: Inserting Data into USER_DETAILS Table

6. After adding the new user, the librarian wants to check if the system has automatically generated the correct USER_ID. To view the changes after inserting a new record into the USER_DETAILS table, enter the following command at the command prompt:

```
SELECT * FROM USER_DETAILS;
```

Figure 9.6 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area contains the following text:

```
mysql> SELECT * FROM USER_DETAILS;
+-----+-----+-----+-----+
| USER_ID | NAME   | LOCATION | PHONE_NO |
+-----+-----+-----+-----+
|     100 | TOBY   | MEXICO  |      45637 |
|     101 | BINYC  | MEXICO  |      45663 |
+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql>
```

Figure 9.6: Viewing Data After Adding Rows

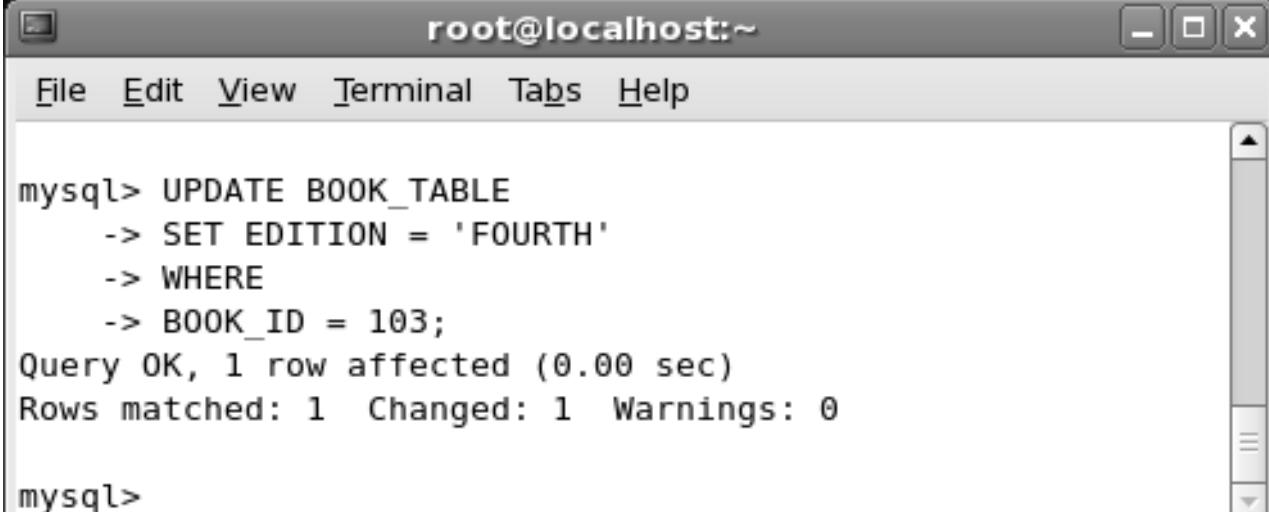
Session 9

Implementing SQL Queries Using MySQL-II (Lab)

7. The librarian has entered incorrect edition information for a book and wants to update this information in the BOOK_TABLE table. To update a record from the BOOK_TABLE table, enter the following command at the command prompt:

```
UPDATE BOOK_TABLE SET EDITION = 'FOURTH' WHERE BOOK_ID=103;
```

Figure 9.7 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following text:

```
mysql> UPDATE BOOK_TABLE
      -> SET EDITION = 'FOURTH'
      -> WHERE
      -> BOOK_ID = 103;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql>
```

Figure 9.7: Updating the BOOK_TABLE Table

8. The librarian wants to ensure that the information about the edition has been updated. To view the changes after updating the record, enter the following command at the command prompt:

```
SELECT * FROM BOOK_TABLE;
```

Figure 9.8 displays the output of the command.

Session 9

Implementing SQL Queries Using MySQL-II (Lab)

```
root@localhost:~
File Edit View Terminal Tabs Help
mysql> SELECT * FROM BOOK_TABLE;
+-----+-----+-----+-----+
| BOOK_ID | BOOK_NAME | AUTHOR | EDITION | DESCRIPTION |
+-----+-----+-----+-----+
| 101 | PROGRAMMING WITH C | GODFRIED | THIRD | C PROGRAMMING |
| 102 | C++ PROGRAMMING | PETER NORTON | SECOND | PROGRAMMING WITH C++ |
| 103 | UNIX | JOHN MACBILL | FOURTH | UNIX OPERATING SYSTEM |
| 104 | LINUX | SCOTT MANN | FIRST | LINUX OPERATING SYSTEM |
| 105 | VISUAL BASIC | BLACK BOOK | FIRST | DESIGN USING VB |
+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql>
```

Figure 9.8: Viewing the BOOK_TABLE Table after Updating

- The librarian wants to provide additional facilities for a member. The member requires three books at a time. The librarian wants to replace this information in the facility column of the CATEGORY table. To replace the information in the CATEGORY table, enter the following command at the command prompt:

```
REPLACE INTO CATEGORY(USER_ID, TYPE, FACILITY, ISSUE_STATUS)
VALUES(103, 'REGULAR', 'ISSUE 3 BOOKS', 'YES');
```

Figure 9.9 displays the output of the command.

```
root@localhost:~
File Edit View Terminal Tabs Help
mysql> REPLACE INTO CATEGORY(USER_ID, TYPE, FACILITY, ISSUE_STATUS)
-> VALUES(103, 'REGULAR', 'ISSUE 3 BOOKS', 'YES');
Query OK, 1 row affected, 1 warning (0.01 sec)

mysql>
```

Figure 9.9: Replacing Data in CATEGORY Table

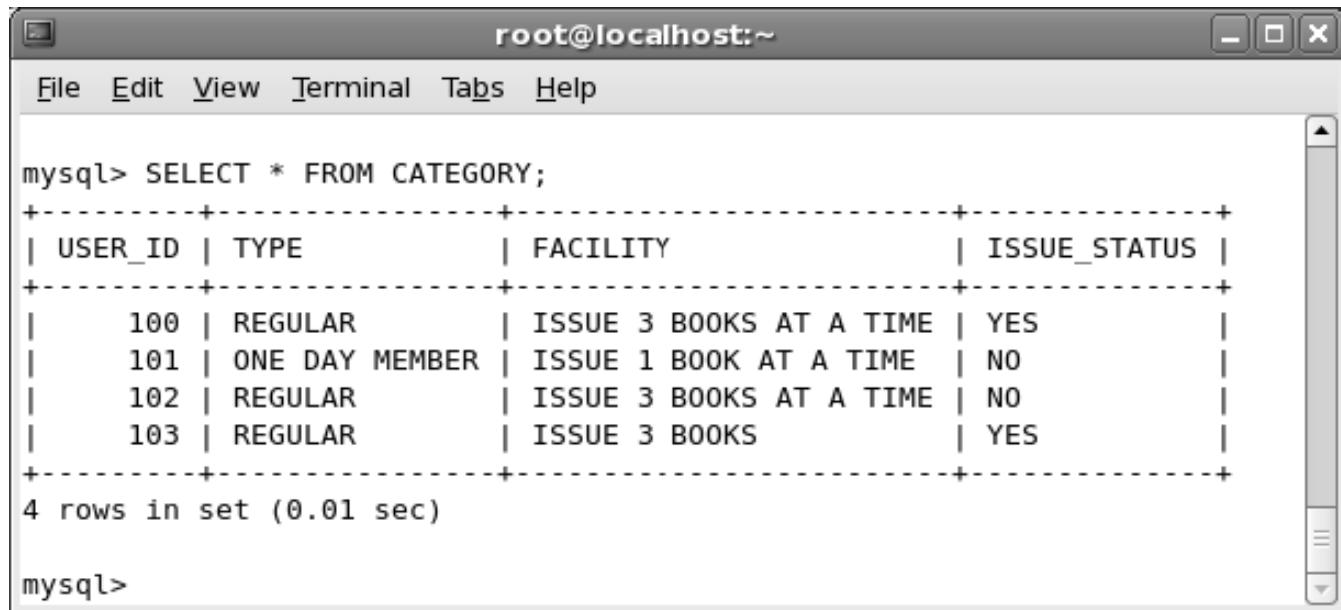
- After replacing the information, the librarian wants to view the updated information. To view the changes after replacing a record in the CATEGORY table, enter the following command at the command prompt:

Session 9

Implementing SQL Queries Using MySQL-II (Lab)

```
SELECT * FROM CATEGORY;
```

Figure 9.10 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area displays the following MySQL command and its output:

```
mysql> SELECT * FROM CATEGORY;
+-----+-----+-----+
| USER_ID | TYPE            | FACILITY          | ISSUE_STATUS |
+-----+-----+-----+
| 100    | REGULAR         | ISSUE 3 BOOKS AT A TIME | YES
| 101    | ONE DAY MEMBER | ISSUE 1 BOOK AT A TIME | NO
| 102    | REGULAR         | ISSUE 3 BOOKS AT A TIME | NO
| 103    | REGULAR         | ISSUE 3 BOOKS          | YES
+-----+-----+-----+
4 rows in set (0.01 sec)

mysql>
```

The output shows four rows of data from the CATEGORY table. The columns are USER_ID, TYPE, FACILITY, and ISSUE_STATUS. The data includes regular users who can issue 3 books at a time and one-day members who can issue 1 book at a time. The ISSUE_STATUS column indicates whether the facility is available (YES) or not (NO).

Figure 9.10: Viewing Data After Replace

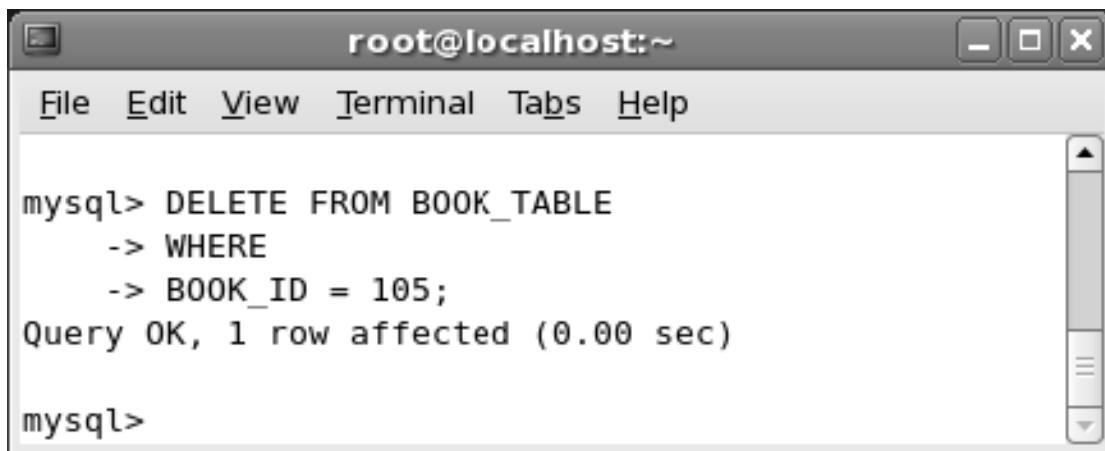
11. A book has been lost and the librarian is unable to trace it. Therefore, the librarian wants to delete that book from the table. To delete a record from the BOOK_TABLE table, enter the following command at the command prompt:

```
DELETE FROM BOOK_TABLE WHERE BOOK_ID=105;
```

Figure 9.11 displays the output of the command.

Session 9

Implementing SQL Queries Using MySQL-II (Lab)



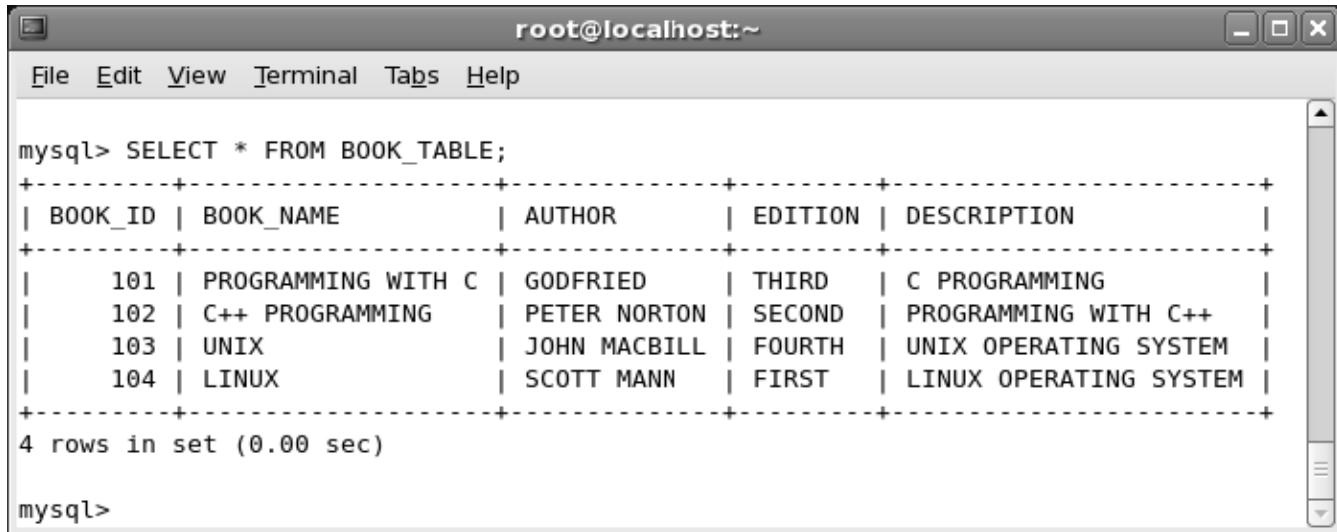
```
root@localhost:~  
File Edit View Terminal Tabs Help  
  
mysql> DELETE FROM BOOK_TABLE  
-> WHERE  
-> BOOK_ID = 105;  
Query OK, 1 row affected (0.00 sec)  
  
mysql>
```

Figure 9.11: Deleting Data from the BOOK_TABLE Table

12. The librarian wants to ensure that the book has been removed from the database. To view the changes after deleting a record from the BOOK_TABLE table, enter the following command at the command prompt:

```
SELECT * FROM BOOK_TABLE;
```

Figure 9.12 displays the output of the command.



```
root@localhost:~  
File Edit View Terminal Tabs Help  
  
mysql> SELECT * FROM BOOK_TABLE;  
+-----+-----+-----+-----+  
| BOOK_ID | BOOK_NAME | AUTHOR | EDITION | DESCRIPTION |  
+-----+-----+-----+-----+  
| 101 | PROGRAMMING WITH C | GODFRIED | THIRD | C PROGRAMMING |  
| 102 | C++ PROGRAMMING | PETER NORTON | SECOND | PROGRAMMING WITH C++ |  
| 103 | UNIX | JOHN MACBILL | FOURTH | UNIX OPERATING SYSTEM |  
| 104 | LINUX | SCOTT MANN | FIRST | LINUX OPERATING SYSTEM |  
+-----+-----+-----+-----+  
4 rows in set (0.00 sec)  
  
mysql>
```

Figure 9.12: Viewing Data After Deletion

Using the ORDER BY Command

You can use the ORDER BY option with the SELECT command to arrange the output in an ascending or descending order.

Session 9

Implementing SQL Queries Using MySQL-II (Lab)

1. The librarian wants to view all the users in the database. This information requires the data to be sorted on the USER_ID field. To arrange the records on the USER_ID field in the CATEGORY table, enter the following command at the command prompt:

```
SELECT * FROM CATEGORY ORDER BY USER_ID;
```

Figure 9.13 displays the output of the command.

The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its execution results:

```
mysql> SELECT * FROM CATEGORY
-> ORDER BY
-> USER_ID;
+-----+-----+-----+
| USER_ID | TYPE      | FACILITY           | ISSUE_STATUS |
+-----+-----+-----+
| 100    | REGULAR   | ISSUE 3 BOOKS AT A TIME | YES
| 101    | ONE DAY MEMBER | ISSUE 1 BOOK AT A TIME | NO
| 102    | REGULAR   | ISSUE 3 BOOKS AT A TIME | NO
| 103    | REGULAR   | ISSUE 3 BOOKS          | YES
+-----+-----+-----+
4 rows in set (0.00 sec)

mysql>
```

The results show four rows of data from the CATEGORY table, ordered by USER_ID. The columns are USER_ID, TYPE, FACILITY, and ISSUE_STATUS. The data includes entries for REGULAR and ONE DAY MEMBER types, along with their respective facility issue rules and status.

Figure 9.13: Sorting Data in the CATEGORY Table

Organizing the Output Using the GROUP BY Command

Grouping enables you to group rows with matching values for a specific column into a single row, allowing you to operate on them together.

1. The librarian wants to view records from the CATEGORY table by grouping them on the column TYPE. To view records by grouping them on the column TYPE from the CATEGORY table, enter the following command at the command prompt:

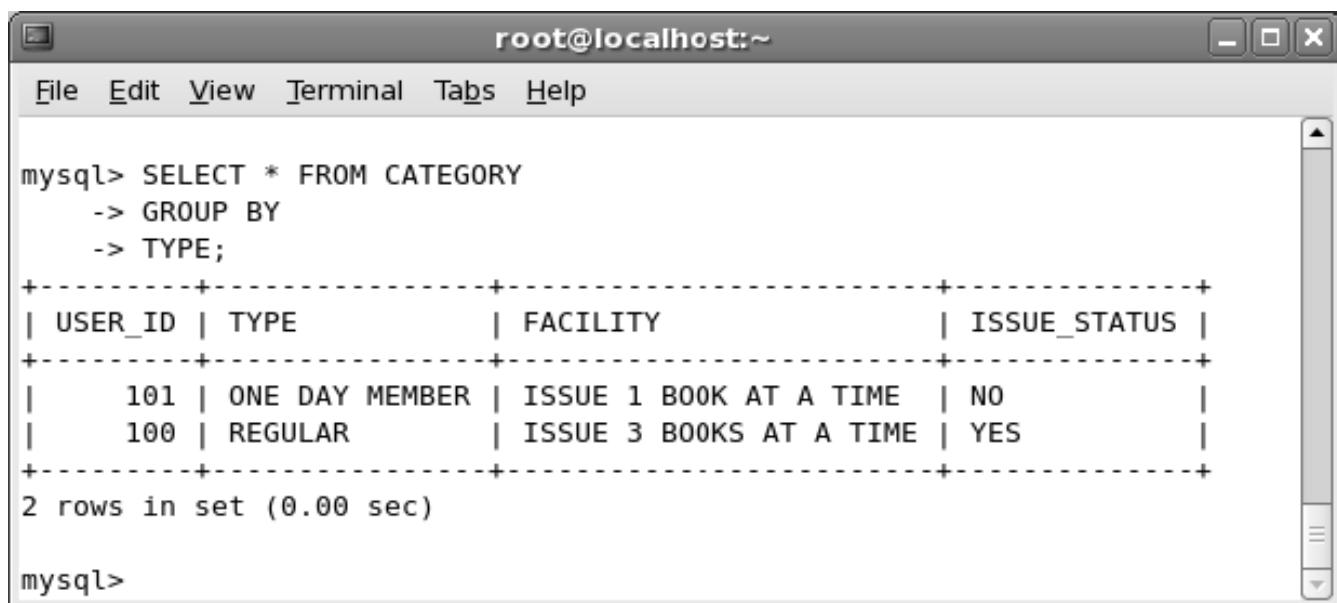
```
SELECT * FROM CATEGORY GROUP BY TYPE;
```

Figure 9.14 displays the output of the command.

Session 9

Implementing SQL Queries Using MySQL-II (Lab)

Lab Guide



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The main area displays the following MySQL query output:

```
mysql> SELECT * FROM CATEGORY
-> GROUP BY
-> TYPE;
+-----+-----+-----+
| USER_ID | TYPE           | FACILITY          | ISSUE_STATUS |
+-----+-----+-----+
|    101  | ONE DAY MEMBER | ISSUE 1 BOOK AT A TIME | NO
|    100  | REGULAR        | ISSUE 3 BOOKS AT A TIME | YES
+-----+-----+-----+
2 rows in set (0.00 sec)

mysql>
```

The output shows two rows of data from the "CATEGORY" table, grouped by the "TYPE" field. The columns are "USER_ID", "TYPE", "FACILITY", and "ISSUE_STATUS".

Figure 9.14: Grouping Data on a Field

Session 9

Implementing SQL Queries Using MySQL-II (Lab)



Do It Yourself

1. Create a primary key on the `Product_Code` column for the `Sales_Details` table.
2. Create an index named `IND1` on the `Date_Of_Manufacture` column for the `Stock_Details` table.
3. Insert the following records in the `Sales_Details` table:

Product_Code	Selling Price	Qty	Units Sold	Month	Location
101	200	1500	1450	Jan	California
102	100	5000	3000	Jan	New York
103	20	2900	2000	Jan	Los Angeles
104	56	120	9	Jan	Sydney
101	200	1000	500	Jan	New York

4. Insert the following records in the `Stock_Details` table:

Product_Code	Product_Name	Description	Cost Price	Date_of_Manufacture
101	Bags	Leather Bags	150	2003-10-05
102	Books	Notepads	10	2003-12-05
103	Pens	Fine Tip Pens	20	2003-01-09
104	Staplers	Staplers	40	2003-10-05
105	Clips	Paper Clips	100	2003-10-05

5. Replace the following records in the `Sales_Details` table:

Product_Code	Selling Price	Qty	Units Sold	Month	Location
103	20	2900	2000	Jan	Los Angeles

6. Update the following record in the Product table:

Product_Code	Selling Price	Qty	Units Sold	Month	Location
103	20	2900	2500	Jan	Los Angeles

7. Delete the following record from the table:

Product_Code	Selling Price	Qty	Units Sold	Month	Location
104	56	120	9	Jan	Sydney

Session 9

Implementing SQL Queries Using MySQL-II (Lab)

Lab Guide



Do It Yourself

8. Group the Stock_Details table on the Date_Of_Manufacture column.
9. Sort the data in the Stock_Details table on the Date_Of_Manufacture column.

Objectives

At the end of this session, the student will be able to:

- *Explain the different types of table joins in MySQL*
- *Explain the use of Equi-Join*
- *Explain the use of Inner Join*
- *Explain the use of Outer-Join*
- *Explain the use of Self-Join*
- *Explain the use of multiple SELECT queries in a single SELECT query*
- *Explain the use of UNION with the query*

10.1 Introduction

Data is often spread across multiple tables. Sometimes, it becomes necessary to retrieve data from one table based on the information present in another table. For this, you need to join the tables. MySQL provides commands such as `JOIN`, `UNION`, and subqueries to join multiple tables.

In this session, you will learn about the various types of joins and how to use them. In addition, you will also learn to execute multiple queries and the `UNION` clause.

10.2 Joining Tables in MySQL

Combining two or more records of different tables from the same database, into one comprehensive structure, is known as joining of tables. The purpose of joining tables is to retrieve data from multiple tables into a single result without repeating the data in the tables. Joining tables enables ease of manipulation, increases the speed of access, and reduces data redundancy.

The concept of joining tables is to display a single table that will be derived from the base of all related tables linked to each other with a common attribute. You can join the tables either using the `WHERE` clause with the `SELECT` command or by using the `JOIN` keyword.

Session 10

10.3 Using Equi-Join

In Equi-Join, MySQL combines records from two or more tables based on a common column. An Equi-Join displays only those records where a match is found in both the tables. The syntax for Equi-Join is:

```
SELECT column_name,...[*] FROM table1 [as alias],table2 [as alias] WHERE  
(join_condition);
```

where,

`SELECT` – retrieves data from the specified column and the table

`column_name` – specifies the name of the column to retrieve from the table

`table1` – specifies the name of the table that contains the columns

`as alias` – specifies the reference name for the table

`WHERE` – specifies the conditions that need to be satisfied before retrieving data

`join_condition` – compares data from the tables as specified in this clause

You can specify the `join_condition` using any one of the following clauses:

- `ON` clause - specifies the conditional expression that can be used in a `WHERE` clause. If there is no matching record present in the specified table according to the query, then the row with all columns set to `NULL` will be used.
- `USING` clause - specifies the common attribute of the joining tables.

For example, to display the first name, department name, and designation of all employees from the `EMP_DETAILS` and the `EMP_DEPARTMENT` tables of `EMPLOYEE` database enter the following command at the command prompt:

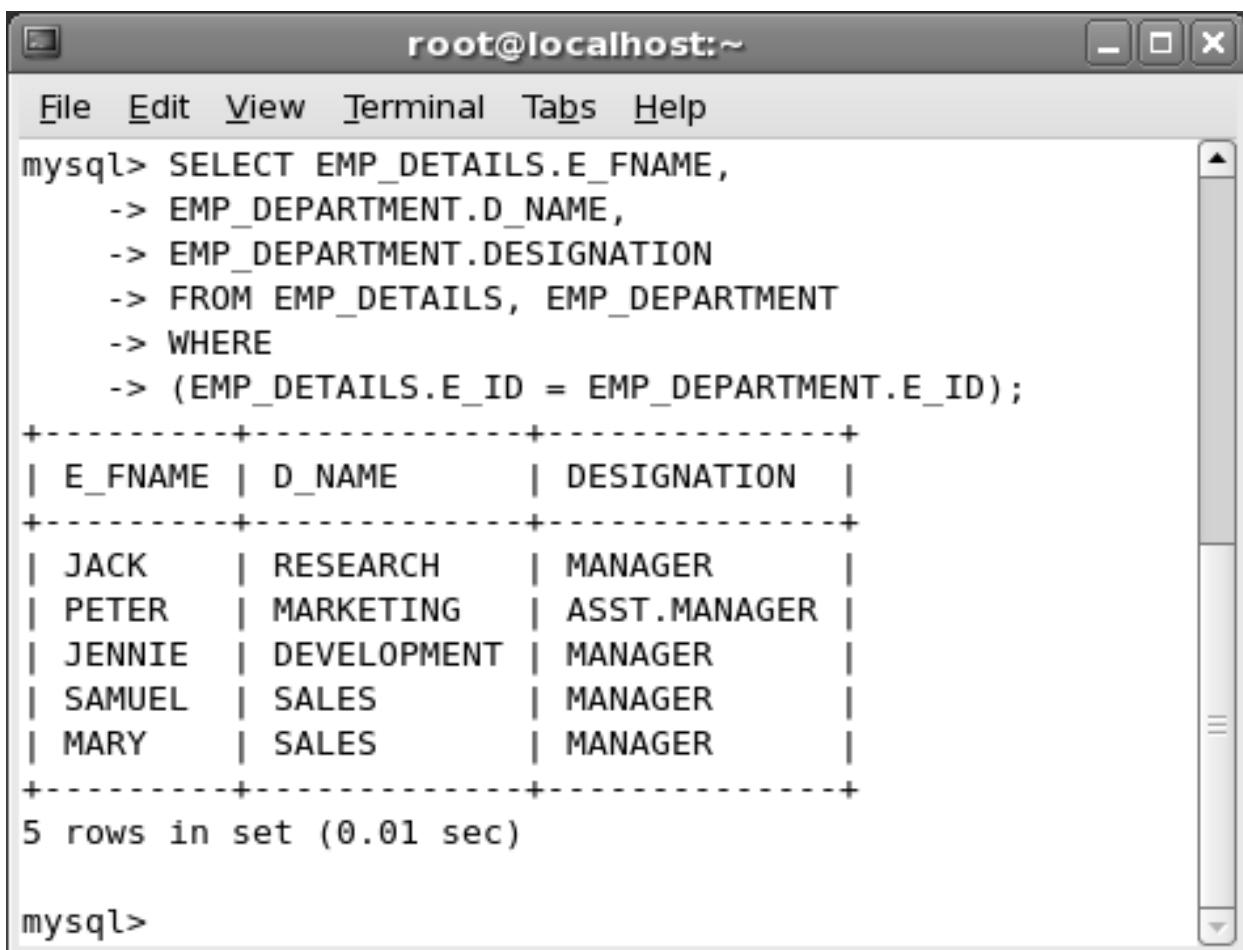
```
SELECT EMP_DETAILS.E_FNAME, EMP_DEPARTMENT.D_NAME, EMP_DEPARTMENT.DESIGNATION  
FROM EMP_DETAILS, EMP_DEPARTMENT WHERE (EMP_DETAILS.E_ID = EMP_DEPARTMENT.  
E_ID);
```

Figure 10.1 displays the output of the command.

Session 10

Using Joins

Concepts



The screenshot shows a terminal window titled "root@localhost:~". The window contains a MySQL command-line session. The user has run a query to join two tables, "EMP_DETAILS" and "EMP_DEPARTMENT", on their common attribute "E_ID". The output displays five rows of data, each consisting of an employee's first name, department name, and designation.

```
root@localhost:~ mysql> SELECT EMP_DETAILS.E_FNAME,
   -> EMP_DEPARTMENT.D_NAME,
   -> EMP_DEPARTMENT.DESIGNATION
   -> FROM EMP_DETAILS, EMP_DEPARTMENT
   -> WHERE
   -> (EMP_DETAILS.E_ID = EMP_DEPARTMENT.E_ID);
+-----+-----+-----+
| E_FNAME | D_NAME      | DESIGNATION |
+-----+-----+-----+
| JACK    | RESEARCH    | MANAGER     |
| PETER   | MARKETING   | ASST.MANAGER|
| JENNIE  | DEVELOPMENT | MANAGER     |
| SAMUEL  | SALES       | MANAGER     |
| MARY    | SALES       | MANAGER     |
+-----+-----+-----+
5 rows in set (0.01 sec)

mysql>
```

Figure 10.1: Joining tables with Equi-Join

Figure 10.1 displays all the employee names with their corresponding department names and designations. Here `E_ID` is a common attribute or column for both the tables.

10.4 Using INNER JOIN

The most common join operation used in applications is an `INNER JOIN`. An `INNER JOIN` creates a virtual table by combining the fields of both tables that satisfy the query for both the tables. The output of the `INNER JOIN` is same as that of the `Equi-join`. The only difference is in the syntax.

The syntax for inner join is:

```
SELECT column_name1, column_name2, column_name3 FROM table_name1 INNER JOIN
table_name2 ON table_name1.primary_keyfield = table_name2.foreign_keyfield;
```

Session 10

Using Joins

where,

Concepts

SELECT – retrieves data from the table

column_name – specifies the name of the column

table_name – specifies the name of the table that contain data

INNER JOIN – creates a virtual table by combining the fields from both tables that satisfy the query

table_name1.primary_keyfield = table_name2.foreign_keyfield – specifies the relationship between the tables

For example, to display the first names, departments, and designations of all employees from EMP_DETAILS and EMP_DEPARTMENT tables from EMPLOYEE database, enter the following command at the command prompt:

```
SELECT      EMP_DETAILS.E_FNAME,      EMP_DEPARTMENT.D_NAME,      EMP_DEPARTMENT.
DESIGNATION FROM EMP_DETAILS INNER JOIN EMP_DEPARTMENT ON EMP_DETAILS.E_ID =
EMP_DEPARTMENT.E_ID;
```

Figure 10.2 displays the output of the command.

Session 10

Using Joins

Concepts

The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its execution results:

```
mysql> SELECT EMP_DETAILS.E_FNAME,
   -> EMP_DEPARTMENT.D_NAME,
   -> EMP_DEPARTMENT.DESIGNATION
   -> FROM
   -> EMP_DETAILS INNER JOIN EMP_DEPARTMENT
   -> ON EMP_DETAILS.E_ID = EMP_DEPARTMENT.E_ID;
+-----+-----+-----+
| E_FNAME | D_NAME      | DESIGNATION |
+-----+-----+-----+
| JACK    | RESEARCH    | MANAGER     |
| PETER   | MARKETING   | ASST.MANAGER|
| JENNIE  | DEVELOPMENT | MANAGER     |
| SAMUEL  | SALES       | MANAGER     |
| MARY    | SALES       | MANAGER     |
+-----+-----+-----+
5 rows in set (0.00 sec)

mysql>
```

The output displays five rows of data, each consisting of an employee's first name, their department name, and their designation. All employees listed are managers.

Figure 10.2: Performing Inner Join

Figure 10.2 displays all the employee names with their corresponding department names and designations using `INNER JOIN` between both the tables. Here, `INNER JOIN` is used for `Equi-Join`.

You can join numeric fields together, as long as they are of same data type such as `AutoNumber` or `Long`. In case of non-numeric data, the fields must be of same type and length and should contain same kind of data.

In the `ON` clause, you can use relational operators such as `=`, `<`, `>`, `<=`, `>=`, or `<>`. In the example, MySQL displays the first names and the designations of those employees whose `E_ID` of the `EMP_DETAILS` table matches with `E_ID` of the `EMP_DEPARTMENT` table. The relational operator, `=`, checks for equality of `E_ID` of both the tables.

To display the designations of those employees whose `E_ID` of `EMP_DETAILS` table does not match with the `E_ID` of `EMP_DEPARTMENT` table, enter the following command at the command prompt:

```
SELECT DISTINCT EMP_DETAILS.E_ID,EMP_DETAILS.E_FNAME, EMP_DEPARTMENT.
```

Session 10

Using Joins

```
DESIGNATION FROM EMP_DETAILS INNER JOIN EMP_DEPARTMENT ON EMP_DETAILS.E_ID
<> EMP_DEPARTMENT.E_ID;
```

Figure 10.3 displays the output of the command.

The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its execution results:

```
mysql> SELECT DISTINCT
    -> EMP_DETAILS.E_ID,
    -> EMP_DETAILS.E_FNAME,
    -> EMP_DEPARTMENT.DESIGNATION
    -> FROM EMP_DETAILS
    -> INNER JOIN
    -> EMP_DEPARTMENT ON
    -> EMP_DETAILS.E_ID <> EMP_DEPARTMENT.E_ID;
+-----+-----+
| E_ID | E_FNAME | DESIGNATION |
+-----+-----+
| 101  | JACK    | ASST.MANAGER |
| 101  | JACK    | MANAGER      |
| 102  | PETER   | MANAGER      |
| 103  | JENNIE  | MANAGER      |
| 103  | JENNIE  | ASST.MANAGER |
| 104  | SAMUEL  | MANAGER      |
| 104  | SAMUEL  | ASST.MANAGER |
| 105  | MARY    | MANAGER      |
| 105  | MARY    | ASST.MANAGER |
| 106  | BOB     | MANAGER      |
| 106  | BOB     | ASST.MANAGER |
| 107  | GEORGE  | MANAGER      |
| 107  | GEORGE  | ASST.MANAGER |
+-----+-----+
13 rows in set (0.00 sec)
```

Figure 10.3: INNER JOIN

You can also use the AND or the OR operator with INNER JOIN command. For example, to view all employees whose E_ID values are same and who live in California city, enter the following command at

Session 10

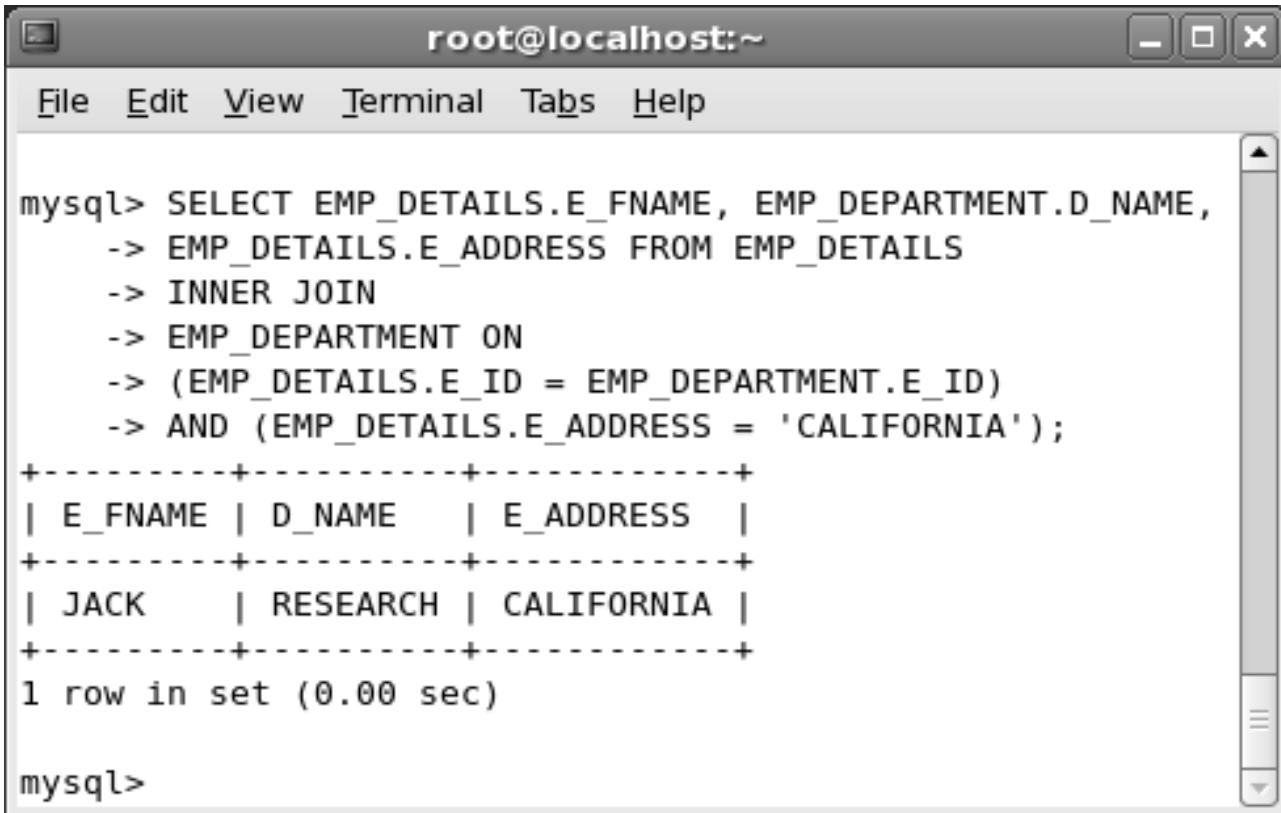
Using Joins

Concepts

the command prompt:

```
SELECT EMP_DETAILS.E_FNAME, EMP_DEPARTMENT.D_NAME, EMP_DETAILS.E_ADDRESS FROM  
EMP_DETAILS INNER JOIN EMP_DEPARTMENT ON (EMP_DETAILS.E_ID = EMP_DEPARTMENT.  
E_ID) AND (EMP_DETAILS.E_ADDRESS = 'CALIFORNIA');
```

Figure 10.4 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its output:

```
mysql> SELECT EMP_DETAILS.E_FNAME, EMP_DEPARTMENT.D_NAME,  
-> EMP_DETAILS.E_ADDRESS FROM EMP_DETAILS  
-> INNER JOIN  
-> EMP_DEPARTMENT ON  
-> (EMP_DETAILS.E_ID = EMP_DEPARTMENT.E_ID)  
-> AND (EMP_DETAILS.E_ADDRESS = 'CALIFORNIA');  
+-----+-----+-----+  
| E_FNAME | D_NAME | E_ADDRESS |  
+-----+-----+-----+  
| JACK    | RESEARCH | CALIFORNIA |  
+-----+-----+-----+  
1 row in set (0.00 sec)  
  
mysql>
```

The output shows a single row of data: JACK in the E_FNAME column, RESEARCH in the D_NAME column, and CALIFORNIA in the E_ADDRESS column.

Figure 10.4: Joining tables Using INNER JOIN

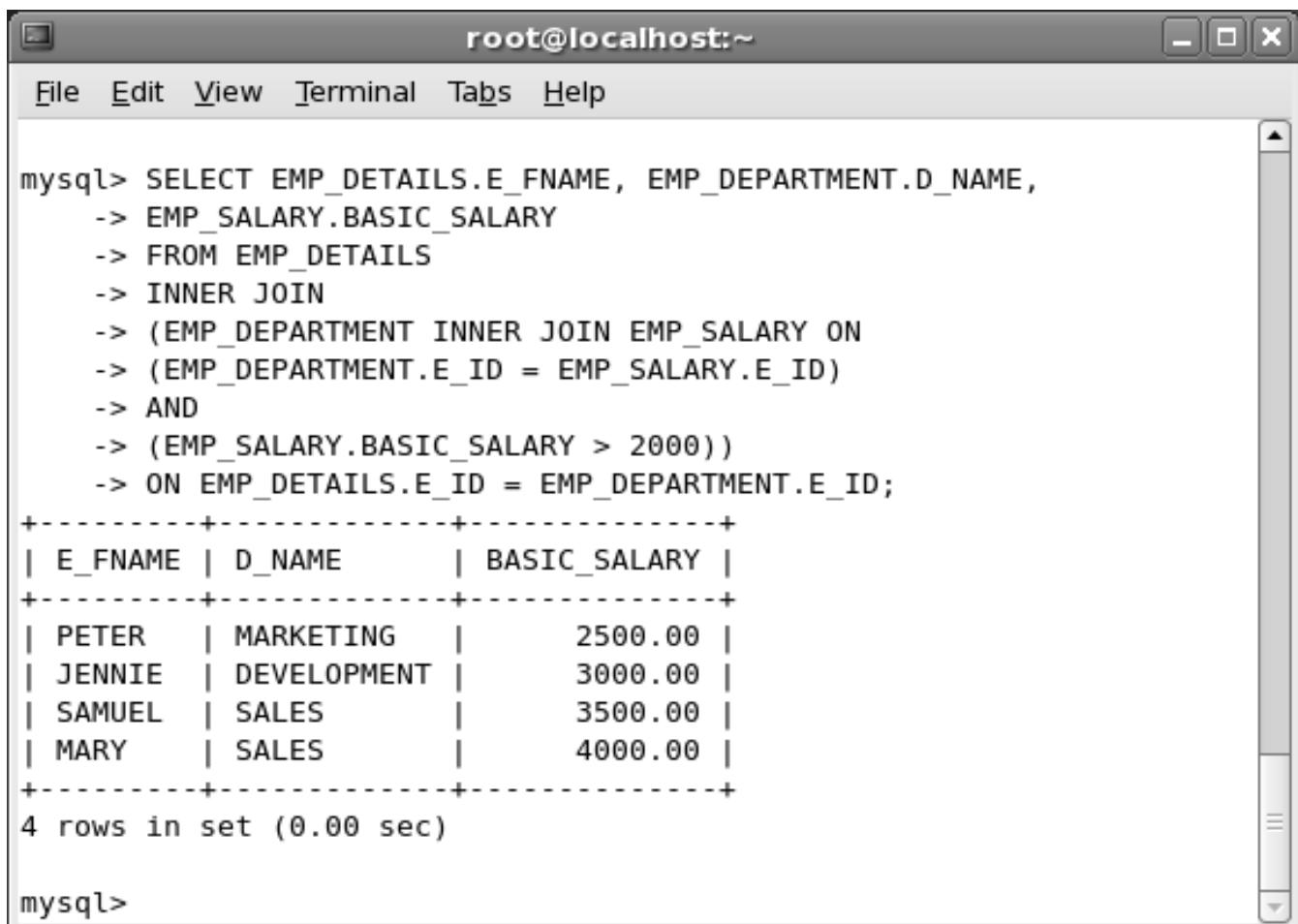
Similarly, you can combine records from three or more tables and display the output in a single result set. For example, to view all employees who have a basic salary greater than \$2,000, you will retrieve data from three tables (EMP_DETAILS, EMP_DEPARTMENT, and EMP_SALARY) and combine them into a single result set.

To join data from three tables, enter the following command at the command prompt:

```
SELECT EMP_DETAILS.E_FNAME, EMP_DEPARTMENT.D_NAME, EMP_SALARY.BASIC_SALARY  
FROM EMP_DETAILS INNER JOIN (EMP_DEPARTMENT INNER JOIN EMP_SALARY ON (EMP_  
DEPARTMENT.E_ID = EMP_SALARY.E_ID) AND (EMP_SALARY.BASIC_SALARY > 2000)) ON  
EMP_DETAILS.E_ID = EMP_DEPARTMENT.E_ID;
```

Session 10

Figure 10.5 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL query and its execution results:

```
mysql> SELECT EMP_DETAILS.E_FNAME, EMP_DEPARTMENT.D_NAME,
-> EMP_SALARY.BASIC_SALARY
-> FROM EMP_DETAILS
-> INNER JOIN
-> (EMP_DEPARTMENT INNER JOIN EMP_SALARY ON
-> (EMP_DEPARTMENT.E_ID = EMP_SALARY.E_ID)
-> AND
-> (EMP_SALARY.BASIC_SALARY > 2000))
-> ON EMP_DETAILS.E_ID = EMP_DEPARTMENT.E_ID;
+-----+-----+
| E_FNAME | D_NAME      | BASIC_SALARY |
+-----+-----+
| PETER   | MARKETING   | 2500.00     |
| JENNIE  | DEVELOPMENT | 3000.00     |
| SAMUEL  | SALES       | 3500.00     |
| MARY    | SALES       | 4000.00     |
+-----+-----+
4 rows in set (0.00 sec)

mysql>
```

The results show four employees: PETER, JENNIE, SAMUEL, and MARY, each assigned to a department (MARKETING, DEVELOPMENT, SALES, SALES) and having basic salaries of 2500.00, 3000.00, 3500.00, and 4000.00 respectively, all of which are greater than \$2000.

Figure 10.5: Joining three tables Using INNER JOIN

Figure 10.5 displays the employee details of only those employees whose `E_ID` from the `EMP_DETAILS` table matches with the `E_ID` of the `EMP_DEPARTMENT` and `EMP_SALARY` tables and whose basic salary is greater than \$2000.

To view employee details using table aliases, enter the following command at the command prompt:

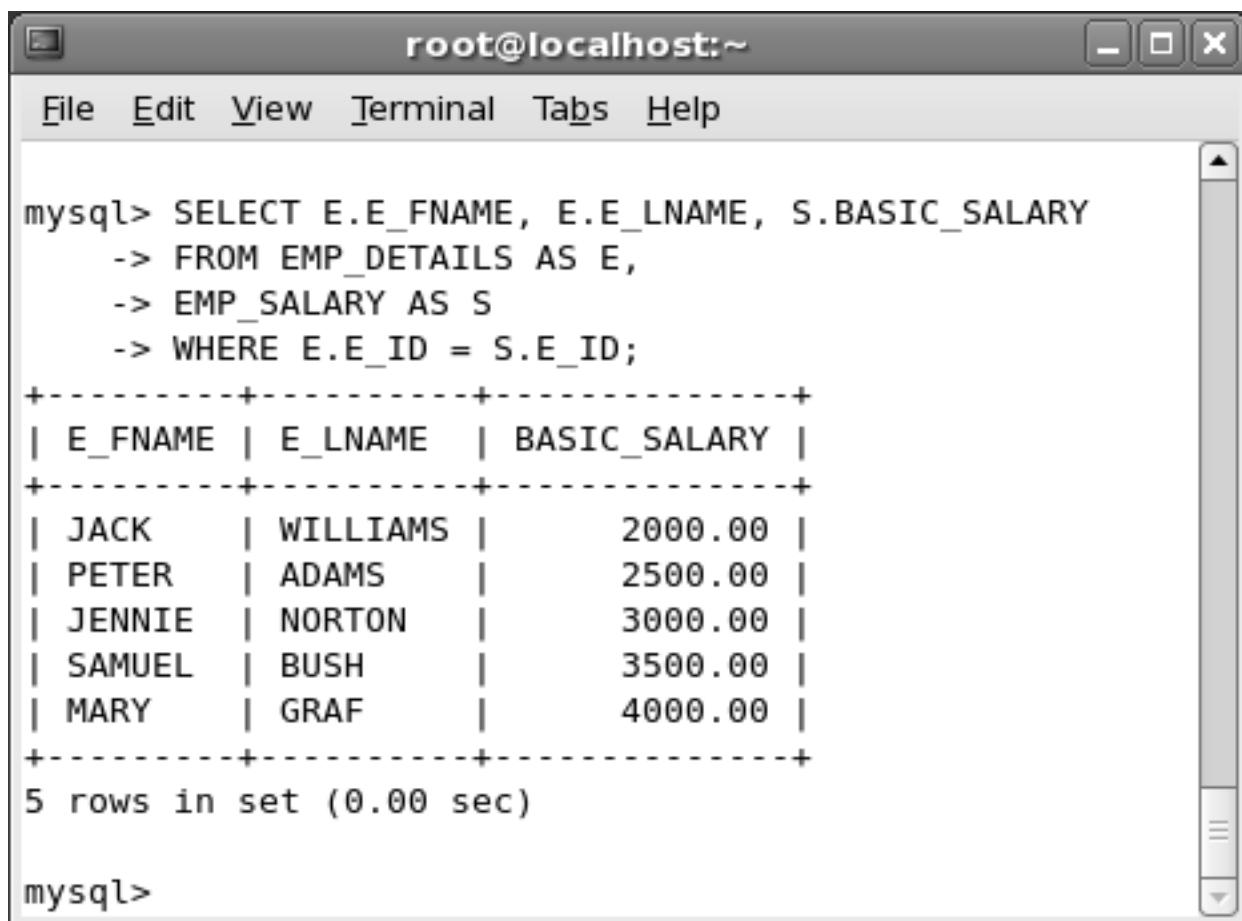
```
SELECT E.E_FNAME, E.E_LNAME, S.BASIC_SALARY FROM EMP_DETAILS AS E, EMP_SALARY
AS S WHERE E.E_ID = S.E_ID;
```

Figure 10.6 displays the output of the command.

Session 10

Using Joins

Concepts



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its execution:

```
mysql> SELECT E.E_FNAME, E.E_LNAME, S.BASIC_SALARY
-> FROM EMP_DETAILS AS E,
-> EMP_SALARY AS S
-> WHERE E.E_ID = S.E_ID;
+-----+-----+
| E_FNAME | E_LNAME | BASIC_SALARY |
+-----+-----+
| JACK    | WILLIAMS |      2000.00 |
| PETER   | ADAMS    |      2500.00 |
| JENNIE  | NORTON   |      3000.00 |
| SAMUEL  | BUSH     |      3500.00 |
| MARY    | GRAF     |      4000.00 |
+-----+-----+
5 rows in set (0.00 sec)

mysql>
```

The output displays five rows of data from the joined tables, showing employee names and basic salary.

Figure 10.6: Joining Two Tables Using Aliases

In figure 10.6, the AS clause is used to refer to aliases for `EMP_DETAILS` and `EMP_SALARY` tables. The alias used for `EMP_DETAILS` table is `E` and for `EMP_SALARY` table is `S`.

MySQL also supports multi-table delete statements using the `DELETE` command. You can specify to remove data from multiple tables in the `DELETE` command. The syntax to delete matching rows from two tables is:

```
DELETE [LOW_PRIORITY|QUICK] [IGNORE] tbl_name[.*] [,tbl_name[.*]]... FROM
table_references [WHERE where_definition];
```

where,

`DELETE` – removes data from the table

`tbl_name[.*]` – specifies the names of the table that contain data to delete

Session 10

table_references – specifies the tables that are linked with a key

WHERE – defines the condition to satisfy before deleting records from the table

This command deletes matching rows from the tables that are listed before the FROM clause.

You can also include the FROM and USING clause with the DELETE command to remove matching rows from tables. The syntax to delete matching rows with DELETE command having FROM and USING clause is:

```
DELETE [LOW PRIORITY|QUICK] [IGNORE] FROM tbl_name[.*][,tbl_name[.*]...]USING  
table_references [WHERE where_definition];
```

where,

DELETE – removes data from the table

tbl_name[.*] – specifies the names of the table that contain data to delete

USING – compares data between tables and deletes the matching records that satisfy the conditions specified in the WHERE clause

table_references – specifies the tables that are linked with a key

WHERE – defines the condition to satisfy before deleting records from the table

This command deletes matching rows from the tables specified in the FROM clause and before the USING clause.

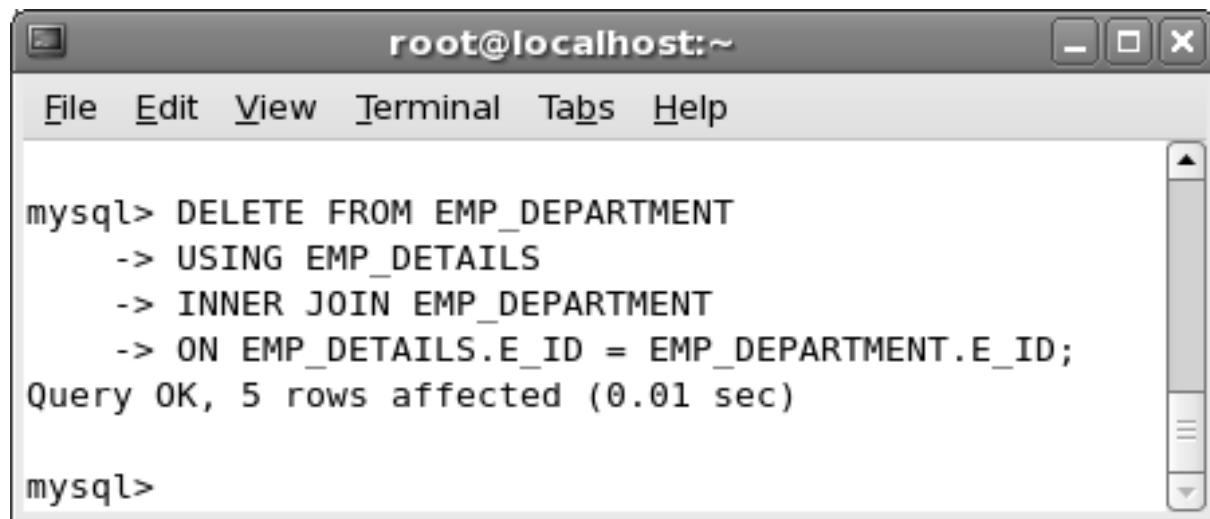
For example, to delete records from the EMP_DEPARTMENT table that have matching values with EMP_DETAILS table, enter the following command at the command prompt:

```
DELETE FROM EMP_DEPARTMENT  
USING EMP_DETAILS  
INNER JOIN EMP_DEPARTMENT  
ON EMP_DETAILS.E_ID = EMP_DEPARTMENT.E_ID;
```

Figure 10.7 displays the output of the command.

Session 10

Using Joins



The screenshot shows a terminal window titled "root@localhost:~". The menu bar includes "File", "Edit", "View", "Terminal", "Tabs", and "Help". The main area displays the following MySQL command and its execution:

```
mysql> DELETE FROM EMP_DEPARTMENT
-> USING EMP_DETAILS
-> INNER JOIN EMP_DEPARTMENT
-> ON EMP_DETAILS.E_ID = EMP_DEPARTMENT.E_ID;
Query OK, 5 rows affected (0.01 sec)

mysql>
```

Concepts

Figure 10.7: Deleting Matching Records after Comparison

Figure 10.7 displays the deletion of records from the `EMP_DEPARTMENT` table.

Note: You will have to add data to the `EMP_DEPARTMENT` table using the `INSERT` command after executing this command because the `DELETE` command removes all rows from `EMP_DEPARTMENT` table that have matching `E_ID` with `EMP_DETAILS` table.

10.5 Using OUTER JOIN

In the previous section, you used the `INNER JOIN` to view records that were common between two or more tables. An `OUTER JOIN` displays rows of the tables that may not have any matching value in the other tables to appear in the result set. To use an `OUTER JOIN`, the tables must have one or more columns in common.

The two types of **OUTER JOIN** are as follows:

- **LEFT JOIN** – is also known as **LEFT OUTER JOIN**. It displays all the records from the table specified on the left of the `OUTER JOIN` operator in the result set, with or without any matching records from the table specified on the right. If no matching record is found in any of the rows from the table placed on the right, then the corresponding columns of that row will contain `NULL` values.

The syntax for `LEFT OUTER JOIN` is:

```
SELECT COLUMN1, COLUMN2 FROM TABLE1
LEFT OUTER JOIN TABLE2 ON (JOIN CONDITION);
```

Session 10

where,

SELECT – retrieves data from columns in the table

COLUMN1 , ... – specifies the name of the column to retrieve data from

LEFT OUTER JOIN - displays all rows from the table specified on the left of the OUTER JOIN operator in the result set, with or without any matching record from the table specified on the right

TABLE1 – specifies the name of the table

JOIN CONDITION – specifies the conditions to retrieve data from tables

Table 10.1 lists the JOIN conditions that can be specified in the ON clause.

Join Condition	Description
BETWEEN	Compares values that lie in between a specific range
COMPARISON	Compares two values using operators such as =, >, <, >=, <=, <>
EXIST	Determines whether a value exist in the given table
IN	Determines whether a value exist in the list of values or a table
IS NULL	Compares a value with an empty or NULL value
LIKE	Compares one value with another
SOME/ANY/ALL	Performs quantified comparisons

Table 10.1: JOIN Conditions in the ON Clause

You can also specify the join conditions using the WHERE and HAVING clauses.

To view all the records from EMP_DETAILS and EMP_DEPARTMENT tables by joining them using LEFT OUTER JOIN, enter the following command at the command prompt:

```
SELECT EMP_DETAILS.E_ID, EMP_DEPARTMENT.D_NAME, EMP_DEPARTMENT.DATE_OF_
JOIN FROM EMP_DETAILS LEFT OUTER JOIN EMP_DEPARTMENT ON EMP_DETAILS.E_ID
= EMP_DEPARTMENT.E_ID;
```

Note: The term OUTER in the LEFT OUTER JOIN clause is optional.

Figure 10.8 displays the output of the command.

Session 10

Using Joins

Concepts

The screenshot shows a terminal window titled "root@localhost:~". The menu bar includes File, Edit, View, Terminal, Tabs, and Help. The terminal window displays the following MySQL query and its output:

```
mysql> SELECT EMP_DETAILS.E_ID, EMP_DEPARTMENT.D_NAME,
-> EMP_DEPARTMENT.DATE_OF_JOIN FROM EMP_DETAILS
-> LEFT OUTER JOIN
-> EMP_DEPARTMENT ON
-> EMP_DETAILS.E_ID = EMP_DEPARTMENT.E_ID;
+-----+-----+
| E_ID | D_NAME      | DATE_OF_JOIN |
+-----+-----+
| 101  | RESEARCH    | 0000-00-00   |
| 102  | MARKETING   | 0000-00-00   |
| 103  | DEVELOPMENT | 0000-00-00   |
| 104  | SALES       | 0000-00-00   |
| 105  | SALES       | 0000-00-00   |
+-----+-----+
5 rows in set (0.00 sec)

mysql>
```

The output shows five rows of data from the joined tables. The first four rows have non-null values in the DATE_OF_JOIN column, while the last row has a null value.

Figure 10.8: Joining Tables Using LEFT OUTER JOIN

In figure 10.8, `EMP_DETAILS` table is joined with `EMP_DEPARTMENT` table with the help of `E_ID` as the common attribute.

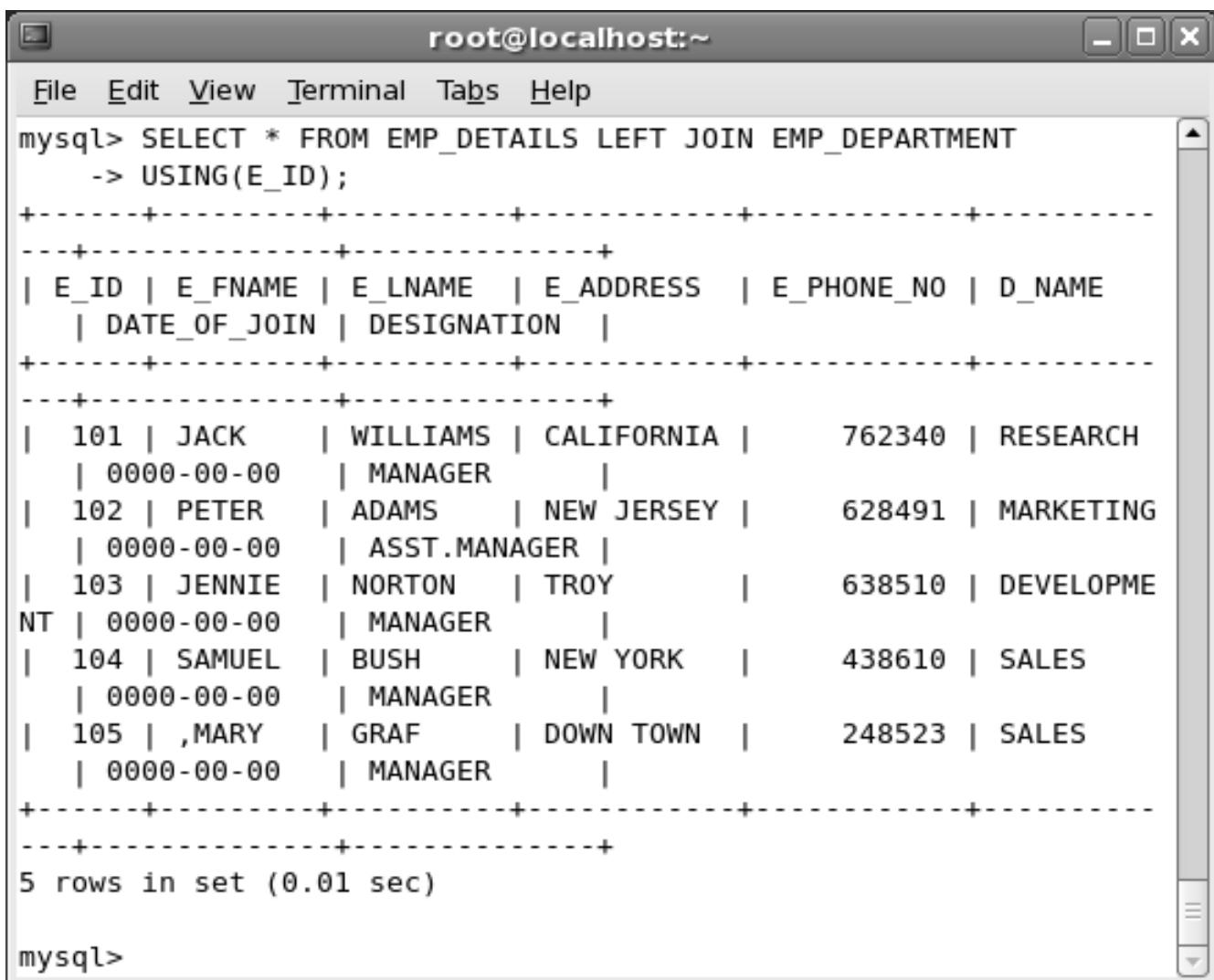
You can specify conditions in a join with the `ON` and `USING` clauses in the `SELECT` command. The `ON` clause specifies the conditions to join tables. The `USING` clause specifies the requirement for matching columns to be present in both the tables.

To view all the records from `EMP_DETAILS` and `EMP_DEPARTMENT` tables by joining them with the `USING` clause, enter the following command at the command prompt:

```
SELECT * FROM EMP_DETAILS LEFT JOIN EMP_DEPARTMENT USING(E_ID);
```

Figure 10.9 displays the output of the command.

Session 10



The screenshot shows a terminal window titled "root@localhost:~". The MySQL command line interface is running. The user has entered a query to perform a LEFT JOIN between two tables: "EMP_DETAILS" and "EMP_DEPARTMENT". The query uses the "USING" clause to join the tables based on the common attribute "E_ID". The output displays the results of the join, showing five rows of data. The columns in the result set are: E_ID, E_FNAME, E_LNAME, E_ADDRESS, E_PHONE_NO, and D_NAME. The D_NAME column is explicitly labeled as "RESEARCH", "MARKETING", "DEVELOPME", "SALES", and "SALES" respectively. The "DATE_OF_JOIN" and "DESIGNATION" columns are also present in the result set.

```
mysql> SELECT * FROM EMP_DETAILS LEFT JOIN EMP_DEPARTMENT
-> USING(E_ID);
+-----+-----+-----+-----+-----+-----+
| E_ID | E_FNAME | E_LNAME | E_ADDRESS | E_PHONE_NO | D_NAME
| DATE_OF_JOIN | DESIGNATION |
+-----+-----+-----+-----+-----+
| 101 | JACK | WILLIAMS | CALIFORNIA | 762340 | RESEARCH
| 0000-00-00 | MANAGER |
| 102 | PETER | ADAMS | NEW JERSEY | 628491 | MARKETING
| 0000-00-00 | ASST.MANAGER |
| 103 | JENNIE | NORTON | TROY | 638510 | DEVELOPME
| NT | 0000-00-00 | MANAGER |
| 104 | SAMUEL | BUSH | NEW YORK | 438610 | SALES
| 0000-00-00 | MANAGER |
| 105 | MARY | GRAF | DOWN TOWN | 248523 | SALES
| 0000-00-00 | MANAGER |
+-----+-----+-----+-----+-----+
-----+
5 rows in set (0.01 sec)

mysql>
```

Figure 10.9: Joining Tables Using LEFT JOIN with USING clause

Figure 10.9 displays the records of `EMP_DETAILS` table that corresponds with `E_ID` of `EMP_DEPARTMENT` table. Both the tables are joined with `E_ID` as a common attribute and the `USING` clause is used for specifying a condition.

The syntax for joining three tables using the `LEFT OUTER JOIN` is:

```
SELECT COLUMN1, COLUMN2 FROM TABLE1
LEFT OUTER JOIN TABLE2 ON (JOIN CONDITION)
LEFT OUTER JOIN TABLE3 ON (JOIN CONDITION);
```

Session 10

Using Joins

where,

SELECT – retrieves data from the table

COLUMN1, ... – specifies the name of the column to be retrieved

TABLE1 – specifies the name of the table that contain the columns

LEFT OUTER JOIN – compares the records from tables using the conditions specified

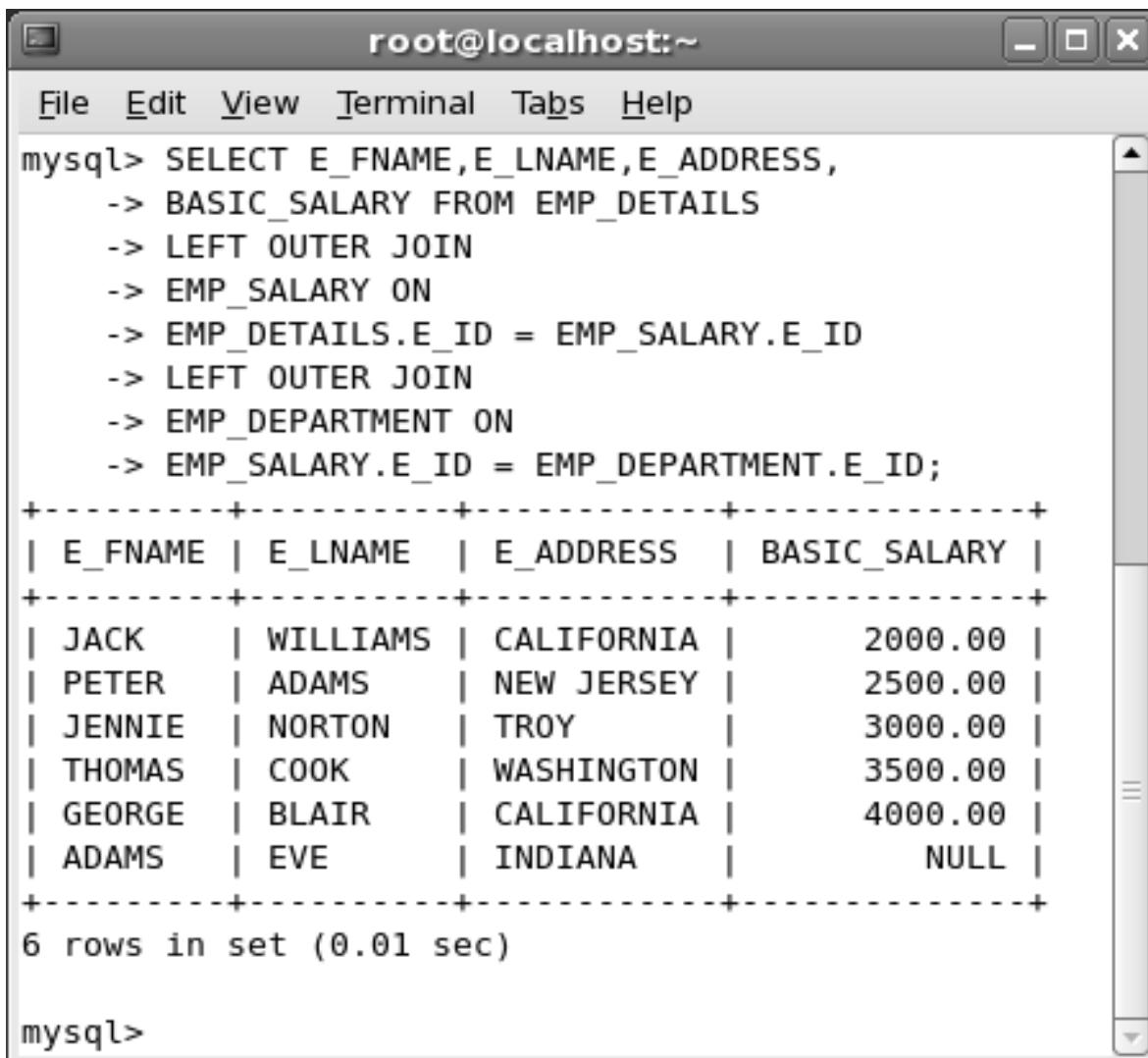
JOIN CONDITION – contains the clauses to satisfy, before retrieving data from the tables

For example, to view all the records from EMP_DETAILS, EMP_SALARY, and EMP_DEPARTMENT tables by joining them with the common attribute E_ID, enter the following command at the command prompt:

```
SELECT E_FNAME,E_LNAME,E_ADDRESS,  
BASIC_SALARY FROM EMP_DETAILS  
LEFT OUTER JOIN  
EMP_SALARY ON  
EMP_DETAILS.E_ID = EMP_SALARY.E_ID  
LEFT OUTER JOIN  
EMP_DEPARTMENT ON  
EMP_SALARY.E_ID = EMP_DEPARTMENT.E_ID;
```

Figure 10.10 displays the output of the command.

Session 10



The screenshot shows a terminal window titled "root@localhost:~". The window contains a MySQL command-line session. The user has run a query to join three tables: EMP_DETAILS, EMP_SALARY, and EMP_DEPARTMENT. The query uses LEFT OUTER JOINs to combine the tables based on the common attribute E_ID. The resulting output is a table with columns E_FNAME, E_LNAME, E_ADDRESS, and BASIC_SALARY, displaying 6 rows of data.

E_FNAME	E_LNAME	E_ADDRESS	BASIC_SALARY
JACK	WILLIAMS	CALIFORNIA	2000.00
PETER	ADAMS	NEW JERSEY	2500.00
JENNIE	NORTON	TROY	3000.00
THOMAS	COOK	WASHINGTON	3500.00
GEORGE	BLAIR	CALIFORNIA	4000.00
ADAMS	EVE	INDIANA	NULL

6 rows in set (0.01 sec)

mysql>

Figure 10.10: Joining three tables Using LEFT OUTER JOIN

In figure 10.10, the `EMP_DETAILS` table is joined with the `EMP_SALARY` table and the `EMP_SALARY` table is joined with the `EMP_DEPARTMENT` table. All the three tables have `E_ID` as a common attribute for joining the tables.

- **RIGHT OUTER JOIN:** displays all records from the table specified on the right of the `OUTER JOIN` operator in the result set, with or without any matching records from the table specified on the left. If there is no matching record found in any of the row from the table placed on the left, then the corresponding columns of that row will contain `NULL` values.

The syntax for RIGHT OUTER JOIN is:

```
SELECT COLUMN1, COLUMN2 FROM TABLE1  
RIGHT OUTER JOIN TABLE2 ON (JOIN CONDITION);
```

Session 10

Using Joins

where,

SELECT – retrieves data from the table

COLUMN1, ... – specifies the name for the column to retrieve data from

TABLE1 – specifies the name of the table that contains the columns

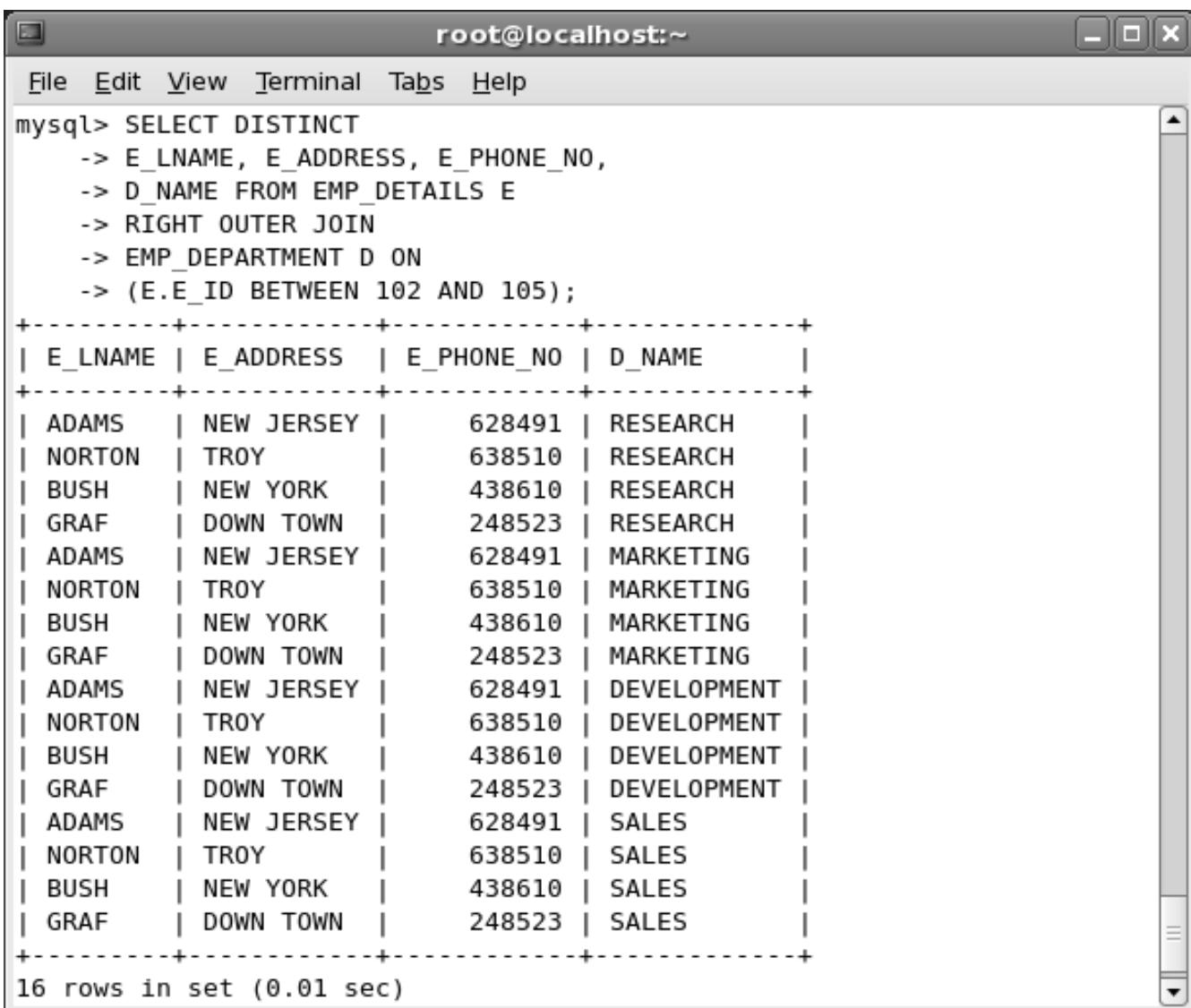
RIGHT OUTER JOIN – specifies the comparison type to fetch data from the tables

For example, to display E_LNAME, E_ADDRESS, and E_PHONE_NO from EMP_DETAILS table where the employee ids range from 102 to 105, you will compare the values of EMP_DEPARTMENT table with the EMP_DETAILS table and display the output.

```
SELECT DISTINCT E_LNAME,E_ADDRESS,E_PHONE_NO,D_NAME
FROM EMP_DETAILS E
RIGHT OUTER JOIN
EMP_DEPARTMENT D ON
(E.E_ID BETWEEN 102 AND 105);
```

Figure 10.11 displays the output of the command.

Session 10



The screenshot shows a terminal window titled "root@localhost:~". The window contains a MySQL command and its execution results. The command is:

```
mysql> SELECT DISTINCT
    -> E_LNAME, E_ADDRESS, E_PHONE_NO,
    -> D_NAME FROM EMP_DETAILS E
    -> RIGHT OUTER JOIN
    -> EMP_DEPARTMENT D ON
    -> (E.E_ID BETWEEN 102 AND 105);
```

The result set is displayed as a table:

E_LNAME	E_ADDRESS	E_PHONE_NO	D_NAME
ADAMS	NEW JERSEY	628491	RESEARCH
NORTON	TROY	638510	RESEARCH
BUSH	NEW YORK	438610	RESEARCH
GRAF	DOWN TOWN	248523	RESEARCH
ADAMS	NEW JERSEY	628491	MARKETING
NORTON	TROY	638510	MARKETING
BUSH	NEW YORK	438610	MARKETING
GRAF	DOWN TOWN	248523	MARKETING
ADAMS	NEW JERSEY	628491	DEVELOPMENT
NORTON	TROY	638510	DEVELOPMENT
BUSH	NEW YORK	438610	DEVELOPMENT
GRAF	DOWN TOWN	248523	DEVELOPMENT
ADAMS	NEW JERSEY	628491	SALES
NORTON	TROY	638510	SALES
BUSH	NEW YORK	438610	SALES
GRAF	DOWN TOWN	248523	SALES

16 rows in set (0.01 sec)

Figure 10.11: Displaying Details Using Right Outer Join

For example, to display E_LNAME, E_ADDRESS, and E_PHONE_NO from EMP_DETAILS table where the employee ids are greater than 102, perform a join operation of the EMP_DEPARTMENT table with the EMP_DETAILS table.

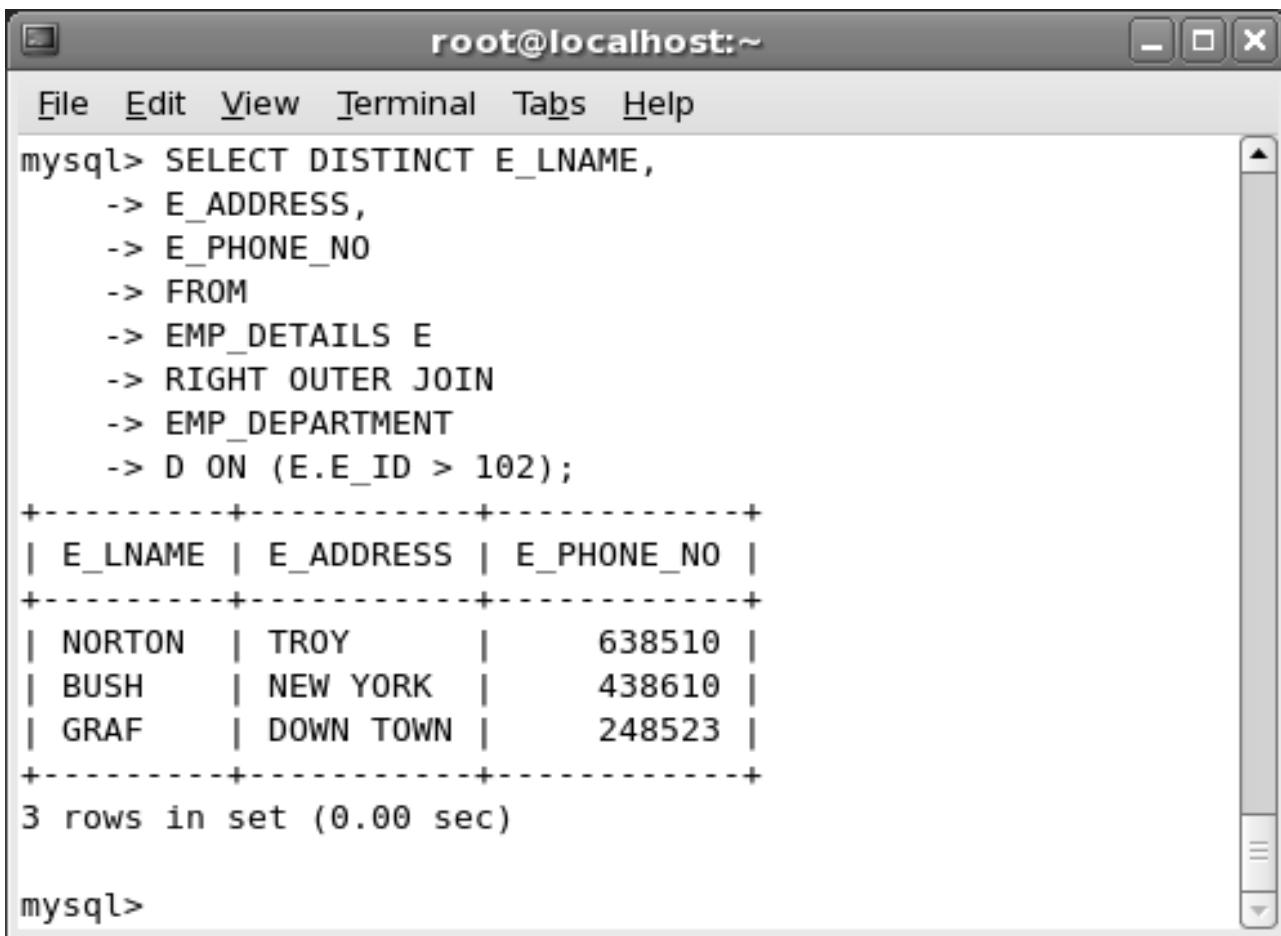
```
SELECT DISTINCT E_LNAME,E_ADDRESS,E_PHONE_NO FROM
EMP_DETAILS E
RIGHT OUTER JOIN EMP_DEPARTMENT D ON (E.E_ID > 102);
```

Figure 10.12 displays the output of the command.

Session 10

Using Joins

Concepts



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL query and its results:

```
mysql> SELECT DISTINCT E_LNAME,
   -> E_ADDRESS,
   -> E_PHONE_NO
   -> FROM
   -> EMP_DETAILS E
   -> RIGHT OUTER JOIN
   -> EMP_DEPARTMENT
   -> D ON (E.E_ID > 102);
+-----+-----+
| E_LNAME | E_ADDRESS | E_PHONE_NO |
+-----+-----+
| NORTON  | TROY      | 638510    |
| BUSH    | NEW YORK   | 438610    |
| GRAF    | DOWN TOWN  | 248523    |
+-----+-----+
3 rows in set (0.00 sec)

mysql>
```

The results show three rows of data from the join between EMP_DETAILS and EMP_DEPARTMENT tables.

Figure 10.12: Displaying Unique Values Using RIGHT OUTER JOIN

You can make a comparison using the SUBSTRING function. A SUBSTRING function extracts string or character literal from a column of a table, specified in the column reference. The FROM clause of SUBSTRING function specifies the character position from where the extracted string should start. The FOR clause of the function specifies the length of the extracted string.

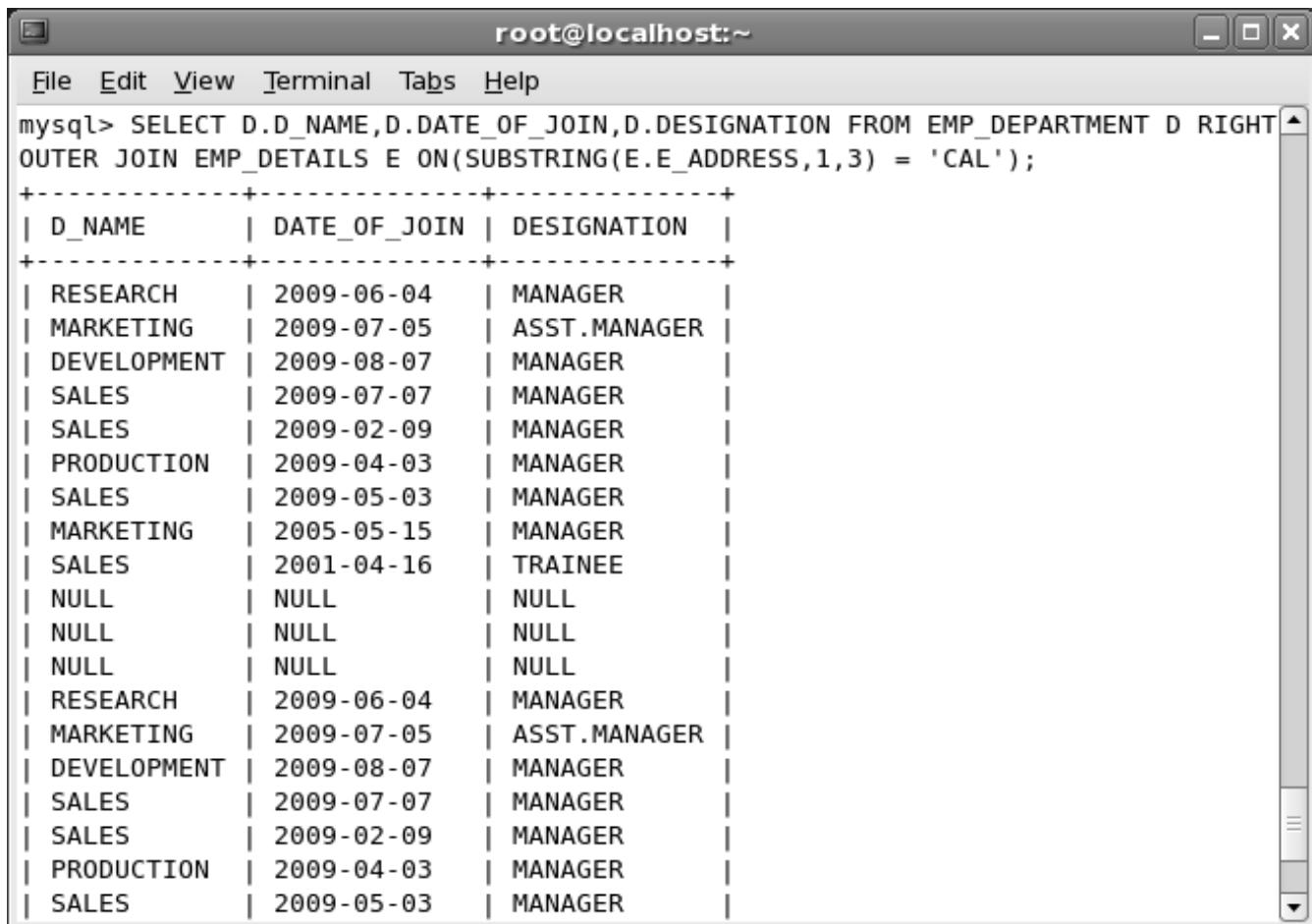
Similarly, to use the SUBSTRING function with the RIGHT OUTER JOIN, enter the following command at the command prompt:

```
SELECT D.D_NAME,
D.DATE_OF_JOIN,
D.DESIGNATION
FROM EMP_DEPARTMENT D
RIGHT OUTER JOIN
EMP_DETAILS E
ON(SUBSTRING(E.E_ADDRESS,1,3) = 'CAL');
```

Session 10

Using Joins

Figure 10.13 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window contains a MySQL command and its execution results. The command is:

```
mysql> SELECT D.D_NAME,D.DATE_OF_JOIN,D.DESIGNATION FROM EMP_DEPARTMENT D RIGHT  
OUTER JOIN EMP_DETAILS E ON(SUBSTRING(E.E_ADDRESS,1,3) = 'CAL');
```

The resulting table has three columns: D_NAME, DATE_OF_JOIN, and DESIGNATION. The data is as follows:

D_NAME	DATE_OF_JOIN	DESIGNATION
RESEARCH	2009-06-04	MANAGER
MARKETING	2009-07-05	ASST.MANAGER
DEVELOPMENT	2009-08-07	MANAGER
SALES	2009-07-07	MANAGER
SALES	2009-02-09	MANAGER
PRODUCTION	2009-04-03	MANAGER
SALES	2009-05-03	MANAGER
MARKETING	2005-05-15	MANAGER
SALES	2001-04-16	TRAINEE
NULL	NULL	NULL
NULL	NULL	NULL
NULL	NULL	NULL
RESEARCH	2009-06-04	MANAGER
MARKETING	2009-07-05	ASST.MANAGER
DEVELOPMENT	2009-08-07	MANAGER
SALES	2009-07-07	MANAGER
SALES	2009-02-09	MANAGER
PRODUCTION	2009-04-03	MANAGER
SALES	2009-05-03	MANAGER

Figure 10.13: Joining tables Using RIGHT OUTER JOIN

10.6 Using Self-Join

A Self-Join statement joins or compares a table with itself. It means that self-join compares records of a column with the other.

The syntax for Self Join is:

```
SELECT DISTINCT COLUMN1, COLUMN2 FROM TABLE WHERE (JOIN CONDITION);
```

where,

SELECT – retrieves data from the table

Session 10

Using Joins

DISTINCT – displays unique values

COLUMN1, ... – specifies the name for the column to retrieve data from

TABLE – specifies the name of the table that contains the columns

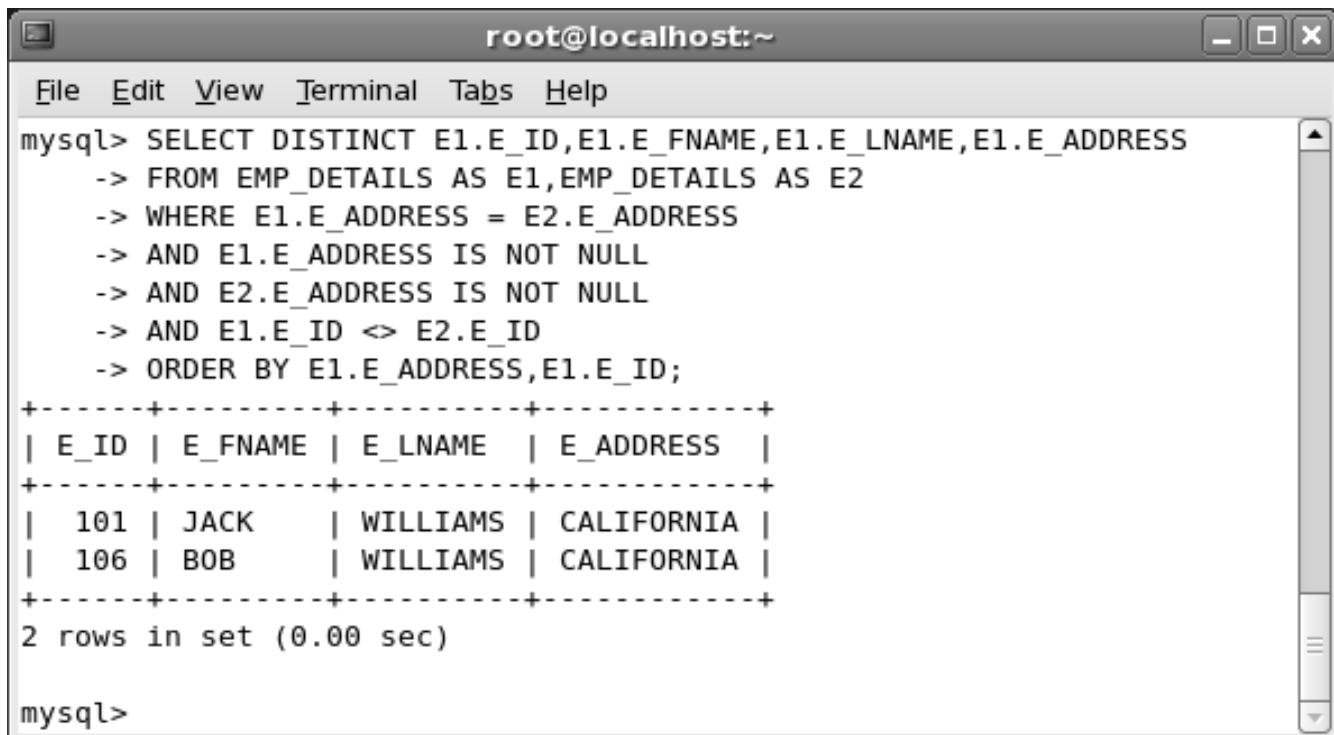
Concepts

Note: The DISTINCT clause ignores duplicates and lists unique records.

For example, to view all the employees, who are located in the same city, enter the following command at the command prompt:

```
SELECT DISTINCT E1.E_ID, E1.E_FNAME, E1.E_LNAME, E1.E_ADDRESS  
FROM EMP_DETAILS AS E1, EMP_DETAILS AS E2  
WHERE E1.E_ADDRESS = E2.E_ADDRESS  
AND E1.E_ADDRESS IS NOT NULL  
AND E2.E_ADDRESS IS NOT NULL  
AND E1.E_ID <> E2.E_ID  
ORDER BY E1.E_ADDRESS, E1.E_ID;
```

Figure 10.14 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL session:

```
File Edit View Terminal Tabs Help  
mysql> SELECT DISTINCT E1.E_ID,E1.E_FNAME,E1.E_LNAME,E1.E_ADDRESS  
-> FROM EMP_DETAILS AS E1,EMP_DETAILS AS E2  
-> WHERE E1.E_ADDRESS = E2.E_ADDRESS  
-> AND E1.E_ADDRESS IS NOT NULL  
-> AND E2.E_ADDRESS IS NOT NULL  
-> AND E1.E_ID <> E2.E_ID  
-> ORDER BY E1.E_ADDRESS,E1.E_ID;  
+-----+-----+-----+  
| E_ID | E_FNAME | E_LNAME | E_ADDRESS |  
+-----+-----+-----+  
| 101 | JACK    | WILLIAMS | CALIFORNIA |  
| 106 | BOB     | WILLIAMS | CALIFORNIA |  
+-----+-----+-----+  
2 rows in set (0.00 sec)  
  
mysql>
```

Figure 10.14: Performing Self-Join

Session 10

Using Joins

In figure 10.14, two different aliases for one table are used to compare the values of the `E_ADDRESS` field with itself.

10.7 Using Subqueries and UNIONS

A query used inside another select query is known as subquery. You will use the `UNION` command when you want to combine tables having same fields into a single result using the `SELECT` command.

10.7.1 Subqueries

When a `SELECT` query is placed inside another `SELECT` query, then the inner `SELECT` query is said to be a subquery. The syntax for the subquery is:

```
SELECT [*] column_name... FROM table_name1 WHERE column_name IN  
(SELECT column_name FROM table_name2);
```

where,

`SELECT` – retrieves data from the table

`*` - retrieves all the records from the table

`column_name` – specifies the name of the column

`IN` – compares data from both the tables

`table_name1` – specifies the name of the table

Consider an example, where you want to view the details of all the employees from all the departments. In this case, you will compare the employee `ID` in the `EMP_DEPARTMENT` and the `EMP_DETAILS` table. You will display the results only if the `E_ID` from `EMP_DEPARTMENT` matches with `E_ID` of `EMP_DETAILS`. To view all records from `EMP_DETAILS` and `EMP_DEPARTMENT` tables using subqueries, enter the following command at the command prompt:

```
SELECT * FROM EMP_DETAILS WHERE E_ID IN (SELECT E_ID FROM EMP_DEPARTMENT);
```

Figure 10.15 displays the output of the command.

Session 10

Using Joins

Concepts

The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL session:

```
mysql> SELECT * FROM EMP_DETAILS
   -> WHERE E_ID IN
   -> (SELECT E_ID FROM EMP_DEPARTMENT);
+-----+-----+-----+-----+-----+
| E_ID | E_FNAME | E_LNAME | E_ADDRESS | E_PHONE_NO |
+-----+-----+-----+-----+-----+
| 101  | JACK    | WILLIAMS | CALIFORNIA | 762340      |
| 102  | PETER   | ADAMS    | NEW JERSEY  | 628491      |
| 103  | JENNIE  | NORTON   | TROY        | 638510      |
| 104  | SAMUEL  | BUSH     | NEW YORK   | 438610      |
| 105  | MARY    | GRAF     | DOWN TOWN  | 248523      |
+-----+-----+-----+-----+-----+
5 rows in set (0.03 sec)

mysql>
```

Figure 10.15: Retrieve Data Using Subqueries

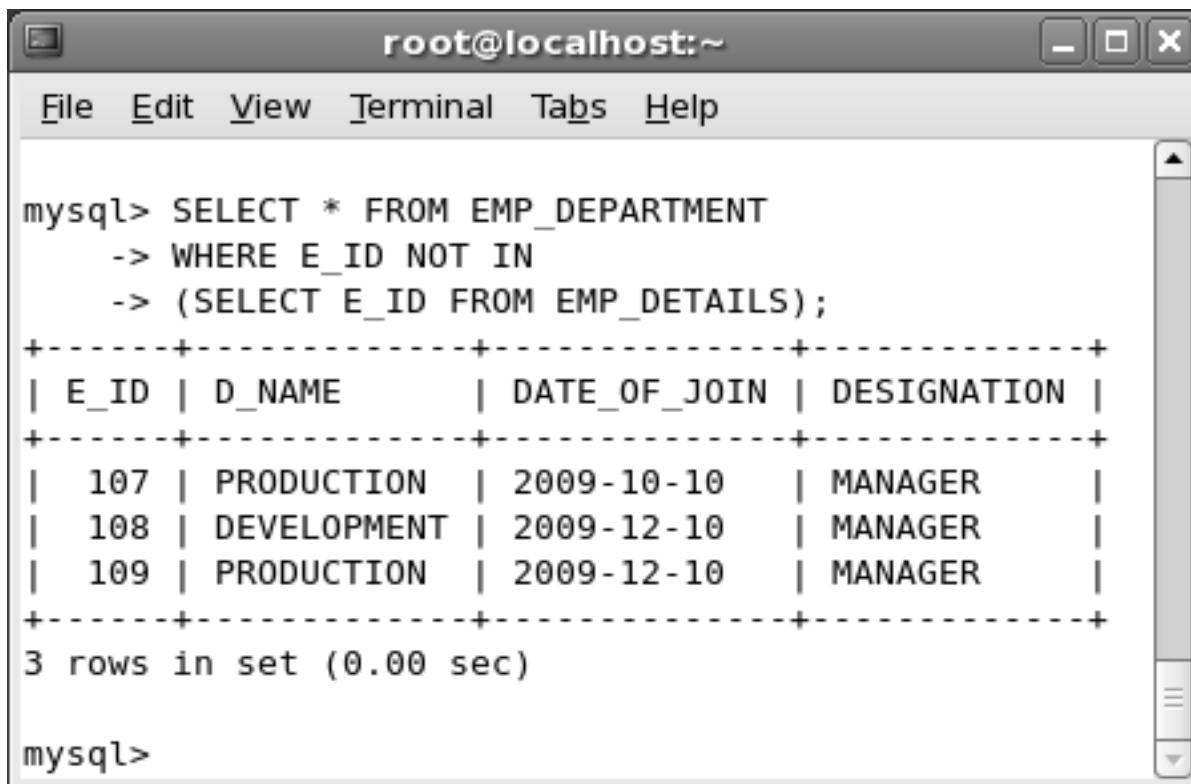
In figure 10.15, the `IN` operator is used for comparing the employee IDs from both the tables.

You can use the `NOT IN` operator to display the unmatched values of tables. For example, to view `E_ID` of `EMP_DEPARMENT` table that do not match with the `E_ID` of `EMP_DETAILS` table, enter the following command at the command prompt:

```
SELECT * FROM EMP_DEPARTMENT
WHERE E_ID NOT IN
(SELECT E_ID FROM EMP_DETAILS);
```

Figure 10.16 displays the output of the command.

Session 10



The screenshot shows a terminal window titled "root@localhost:~". The menu bar includes File, Edit, View, Terminal, Tabs, and Help. The command line shows:

```
mysql> SELECT * FROM EMP_DEPARTMENT
-> WHERE E_ID NOT IN
-> (SELECT E_ID FROM EMP_DETAILS);
+-----+-----+-----+
| E_ID | D_NAME      | DATE_OF_JOIN | DESIGNATION |
+-----+-----+-----+
| 107  | PRODUCTION   | 2009-10-10   | MANAGER     |
| 108  | DEVELOPMENT   | 2009-12-10   | MANAGER     |
| 109  | PRODUCTION   | 2009-12-10   | MANAGER     |
+-----+-----+-----+
3 rows in set (0.00 sec)

mysql>
```

The output displays three rows of data from the EMP_DEPARTMENT table where the E_ID does not exist in the EMP_DETAILS table.

Figure 10.16: Retrieving Unmatched Data Using Subqueries

Figure 10.16 displays all the `E_ID` from the `EMP_DEPARTMENT` table that does not match with the `E_ID` of `EMP_DETAILS` table.

MySQL also supports the use of `SELECT` statement within the `REPLACE` and `INSERT` queries. It means that when a `SELECT` query is used within an `INSERT` or `REPLACE` query, then that `SELECT` query is said to be subquery of `INSERT` or `REPLACE` query.

For example, to insert department names of all employees from `EMP_DEPARTMENT` table to `EMP_SALARY` table, enter the following command at the command prompt:

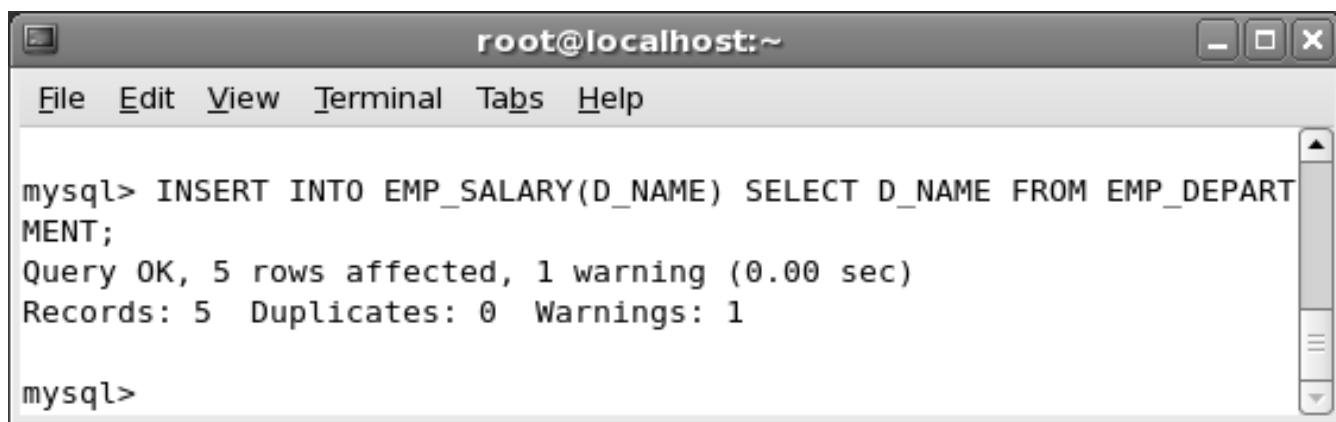
```
INSERT INTO EMP_SALARY(D_NAME) SELECT D_NAME FROM EMP_DEPARTMENT;
```

Figure 10.17 displays the output of the command.

Session 10

Using Joins

Concepts



The screenshot shows a terminal window titled "root@localhost:~". The menu bar includes "File", "Edit", "View", "Terminal", "Tabs", and "Help". The main area displays the following MySQL command and its execution:

```
mysql> INSERT INTO EMP_SALARY(D_NAME) SELECT D_NAME FROM EMP_DEPARTMENT;
Query OK, 5 rows affected, 1 warning (0.00 sec)
Records: 5  Duplicates: 0  Warnings: 1

mysql>
```

Figure 10.17: Using Subquery in INSERT Statement

10.7.2 Unions

A Union is used to combine many result sets used with `SELECT` command into a single result set. The syntax for the `UNION` command is:

```
SELECT * FROM TABLE1
UNION [ALL | DISTINCT]
SELECT * FROM TABLE2
UNION [ALL | DISTINCT];
```

where,

`SELECT` – retrieves data from the table

`*` - retrieves all data from the table

`TABLE1` – specifies the name of the table that contains data

`UNION` – combines query results into a single result

The columns specified in the `SELECT` command should be of the same type. MySQL will use the column names from the first `SELECT` query as the column names for the results returned.

MySQL provides the `ALL` keyword to display matching rows from all the `SELECT` statements. If this keyword is not used, only unique rows are displayed.

In other words, if you do not use the keyword `ALL` in the `UNION` clause, all returned rows would be unique. If you specify `ALL`, then you will get all matching rows from all the `SELECT` statements.

Session 10

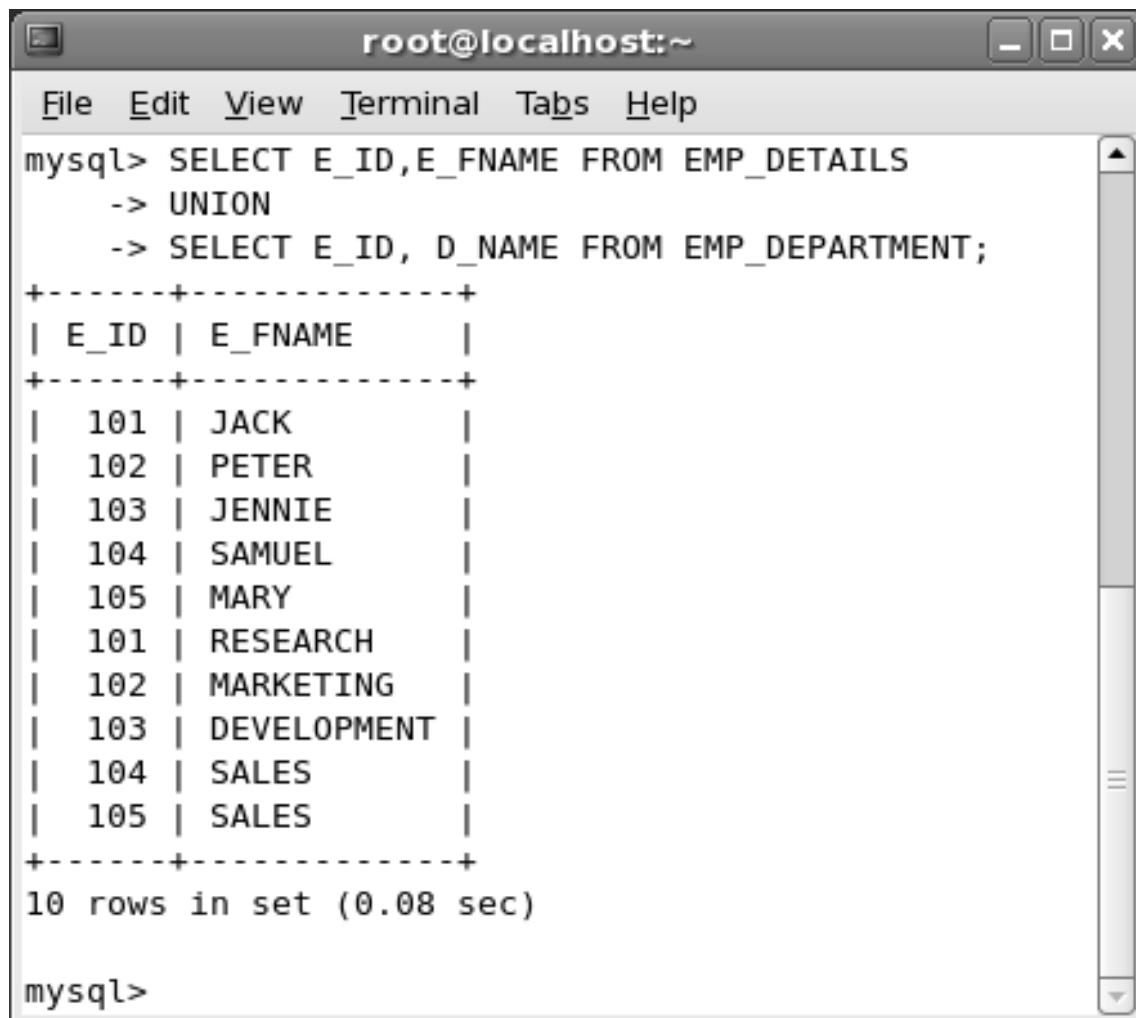
Using Joins

Note: The DISTINCT keyword is an optional keyword.

For example, to view E_ID and E_FNAME from EMP_DETAILS table and E_ID and D_NAME from EMP_DEPARTMENT table, enter the following command at the command prompt:

```
SELECT E_ID, E_FNAME FROM EMP_DETAILS
UNION
SELECT E_ID, D_NAME FROM EMP_DEPARTMENT;
```

Figure 10.18 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL session:

```
mysql> SELECT E_ID,E_FNAME FROM EMP_DETAILS
-> UNION
-> SELECT E_ID, D_NAME FROM EMP_DEPARTMENT;
+-----+
| E_ID | E_FNAME      |
+-----+
| 101  | JACK         |
| 102  | PETER        |
| 103  | JENNIE       |
| 104  | SAMUEL       |
| 105  | MARY          |
| 101  | RESEARCH     |
| 102  | MARKETING    |
| 103  | DEVELOPMENT  |
| 104  | SALES         |
| 105  | SALES         |
+-----+
10 rows in set (0.08 sec)

mysql>
```

The output shows a table with two columns: E_ID and E_FNAME. The data is a combination of records from the EMP_DETAILS table and the EMP_DEPARTMENT table, separated by a UNION operator. The records are listed in the order they appear in the query, starting with records from EMP_DETAILS and then records from EMP_DEPARTMENT.

Figure 10.18: Using UNION for Joining Tables

The ORDER BY clause can also be used to sort the entire UNION results. The ORDER BY clause used to sort the results of the UNION must be included in the last SQL statement. The SELECT statements must

Session 10

Using Joins

Concepts

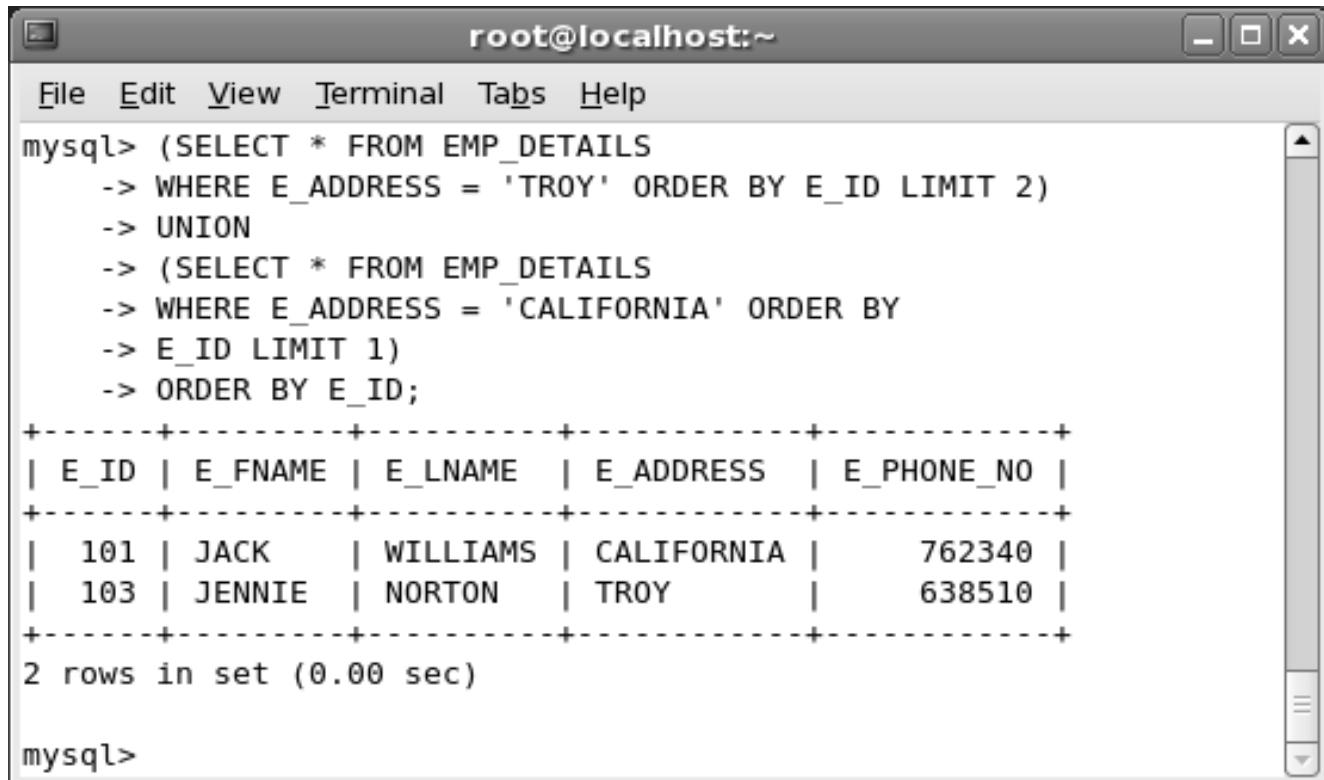
be included within parenthesis.

MySQL supports the `LIMIT` clause to restrict the number of records returned by the query. The restriction for the number of rows must be specified in one or two numeric values. These values must be in the form of non-negative integer constants. The first value in the argument defines the first row to return and the second value defines the maximum number of rows to return.

For example, to sort the records retrieved from the `EMP_DETAILS` table where the employee addresses are California and Troy, enter the following command at the command prompt:

```
(SELECT * FROM EMP_DETAILS WHERE E_ADDRESS = 'TROY' ORDER BY  
E_ID LIMIT 2)  
UNION  
(SELECT * FROM EMP_DETAILS WHERE E_ADDRESS = 'CALIFORNIA' ORDER BY  
E_ID LIMIT 1) ORDER BY E_ID;
```

Figure 10.19 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL session:

```
mysql> (SELECT * FROM EMP_DETAILS  
-> WHERE E_ADDRESS = 'TROY' ORDER BY E_ID LIMIT 2)  
-> UNION  
-> (SELECT * FROM EMP_DETAILS  
-> WHERE E_ADDRESS = 'CALIFORNIA' ORDER BY  
-> E_ID LIMIT 1)  
-> ORDER BY E_ID;  
+----+----+----+----+  
| E_ID | E_FNAME | E_LNAME | E_ADDRESS | E_PHONE_NO |  
+----+----+----+----+  
| 101 | JACK    | WILLIAMS | CALIFORNIA | 762340    |  
| 103 | JENNIE  | NORTON   | TROY       | 638510    |  
+----+----+----+----+  
2 rows in set (0.00 sec)  
  
mysql>
```

Figure 10.19: Using ORDER BY with UNION clause

Figure 10.19 displays the output that has been restricted using the `LIMIT` clause.



Summary

- Joining of tables means combining two or more records from different tables of the same database into one comprehensive structure.
- Joining of tables is performed either using `WHERE` clause with the `SELECT` command or using the `JOIN` keyword. This will enhance manipulation, increase access speed, and reduce data redundancy.
- In an Equi-Join, the comparison is made between two columns that contain similar values.
- The `ON` clause specifies the conditional expression to be used in a `WHERE` clause. It also supports the use of relational operators such as `=`, `<`, `>`, `<=`, `>=`, or `<>`.
- The `USING` clause can be used in a join only when the tables have matching columns.
- Numeric fields can be joined together, as long as they are of same data type such as `AutoNumber` or `Long`. Incase of non-numeric data, the fields must be of same type and length, and should contain same kind of data.
- An `INNER JOIN` creates a virtual table by combining the fields of both tables that satisfy the query for both the tables. The `AND` or the `OR` operators can also be used with `INNER JOIN` command.
- An `OUTER JOIN` is used to join two tables, a source and joining table, which have one or more columns in common. The rows of the tables that may not have any matching value in the other tables are also displayed in the result set.
- MySQL supports two types of `OUTER JOINS`: `LEFT OUTER JOIN` and `RIGHT OUTER JOIN`.
- The `LEFT OUTER JOIN` displays all rows from the table specified on the left of the `OUTER JOIN` operator in the result set, with or without any matching record with the table specified on the right.
- The `RIGHT OUTER JOIN` displays all rows from the table specified on the right of the `OUTER JOIN` operator in the result set, with or without any matching record with the table specified on the left.
- When using the `OUTER JOIN`, if there is no match found in any of the rows from the table placed on the right, then the corresponding columns of that row will contain `NULL` values.



Summary

- A Self-Join is a query that is used to join or compare a table to itself.
- The DISTINCT clause lists unique records from tables.
- A query used inside another select query is known as subquery.
- The UNION clause can be used to combine results from different SELECT commands.
- The ORDER BY clause must be used with a parenthesis to sort results displayed by the UNION option.
- The LIMIT clause restricts the number of results displayed by the query



Check Your Progress

1. A comparison is made between two columns that carry same values are termed as _____.
 - a. Self-Join
 - b. Equi-Join
 - c. Left Join
 - d. Sub-Select

2. When there is no match in the columns of the joining table with the source table, then those columns contain _____ values.
 - a. Equal
 - b. Different
 - c. Null
 - d. Single

3. When a select query is used inside another select query then the second select query is known as _____.
 - a. Multiple query
 - b. Subquery
 - c. Join query
 - d. Single query

4. A _____ function is used to extract string or _____ from a column of a table, specified in the column reference.
 - a. IN, Numeric
 - b. SUB-STRING, Character literal



Check Your Progress

- c. EXIST, Decimal
 - d. LIKE, Boolean
5. Which of the following specifies the conditional expression that can be used in a WHERE clause?
- a. USING
 - b. FROM
 - c. ON
 - d. HAVING
6. Which of the following clause determine the existence of values in the list or the table?
- a. BETWEEN
 - b. LIKE
 - c. EXIST
 - d. IN
7. Which of the following options displays all the rows from all the tables specified in the FROM clause in the result set, with or without any match with the specified table?
- a. LEFT OUTER JOIN
 - b. RIGHT OUTER JOIN
 - c. SELF JOIN
 - d. INNER JOIN



Login to www.onlinevarsity.com

Objectives

At the end of this session, the student will be able to:

- *Use different types of JOINS.*
- *Use a subquery.*
- *Use the UNION clause.*

The steps given in this session are detailed, comprehensive, and carefully thought-through in order to meet the learning objectives and understand the tool completely. Please follow the steps carefully.

Part I - For the first 1.5 hours:

Using various types of JOINS

Tables are joined either using the WHERE clause with the SELECT command or by using the JOIN keyword.

1. In order to display all the details from BOOK_TABLE and ISSUE_DETAILS table based on the common field BOOK_ID, enter the following command at the command prompt:

```
SELECT BOOK_TABLE.BOOK_NAME, BOOK_TABLE.AUTHOR,  
ISSUE_DETAILS.DATE_OF_ISSUE  
FROM BOOK_TABLE, ISSUE_DETAILS  
WHERE BOOK_TABLE.BOOK_ID = ISSUE_DETAILS.BOOK_ID;
```

Figure 11.1 displays the output of the command.

Session 11

Using Joins (Lab)

The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL query and its results:

```
mysql> SELECT BOOK_TABLE.BOOK_NAME, BOOK_TABLE.AUTHOR,
-> ISSUE_DETAILS.DATE_OF_ISSUE
-> FROM BOOK_TABLE, ISSUE_DETAILS
-> WHERE
-> BOOK_TABLE.BOOK_ID = ISSUE_DETAILS.BOOK_ID;
+-----+-----+-----+
| BOOK_NAME      | AUTHOR        | DATE_OF_ISSUE |
+-----+-----+-----+
| PROGRAMMING WITH C | GODFRIED     | 2010-06-05   |
| C++ PROGRAMMING    | PETER NORTON  | 2010-04-04   |
| UNIX            | JOHN MACBILL  | 2010-09-05   |
| LINUX           | SCOTT MANN    | 2010-01-01   |
+-----+-----+-----+
4 rows in set (0.00 sec)

mysql>
```

The results show four rows of data from the joined tables, mapping book names and authors to their issue dates.

Figure 11.1: Joining Two Tables

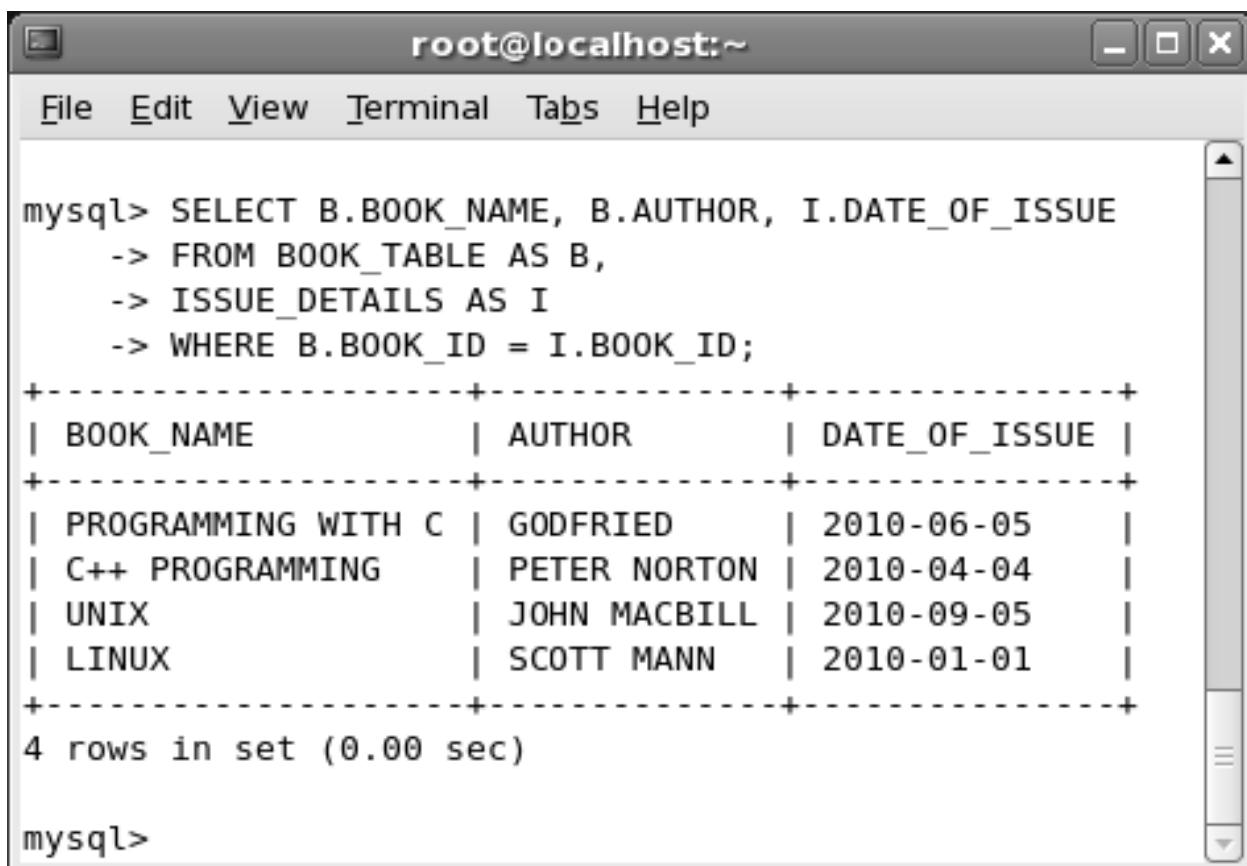
2. In order to display the BOOK_NAME, AUTHOR, and the DATE_OF_ISSUE from the BOOK_TABLE and ISSUE_DETAILS table, using alias names for the tables and BOOK_ID as the common field, enter the following command at the command prompt:

```
SELECT B.BOOK_NAME, B.AUTHOR, I.DATE_OF_ISSUE
FROM BOOK_TABLE AS B, ISSUE_DETAILS AS I
WHERE B.BOOK_ID = I.BOOK_ID;
```

Figure 11.2 displays the output of the command.

Session 11

Using Joins (Lab)



The screenshot shows a terminal window titled "root@localhost:~". The menu bar includes File, Edit, View, Terminal, Tabs, and Help. The main area displays a MySQL command and its output:

```
mysql> SELECT B.BOOK_NAME, B.AUTHOR, I.DATE_OF_ISSUE
-> FROM BOOK_TABLE AS B,
-> ISSUE_DETAILS AS I
-> WHERE B.BOOK_ID = I.BOOK_ID;
+-----+-----+-----+
| BOOK_NAME      | AUTHOR        | DATE_OF_ISSUE |
+-----+-----+-----+
| PROGRAMMING WITH C | GODFRIED      | 2010-06-05   |
| C++ PROGRAMMING | PETER NORTON | 2010-04-04   |
| UNIX           | JOHN MACBILL | 2010-09-05   |
| LINUX          | SCOTT MANN    | 2010-01-01   |
+-----+-----+-----+
4 rows in set (0.00 sec)

mysql>
```

A vertical scrollbar is visible on the right side of the terminal window.

Lab Guide

Figure 11.2: Joining Two Tables Using Alias

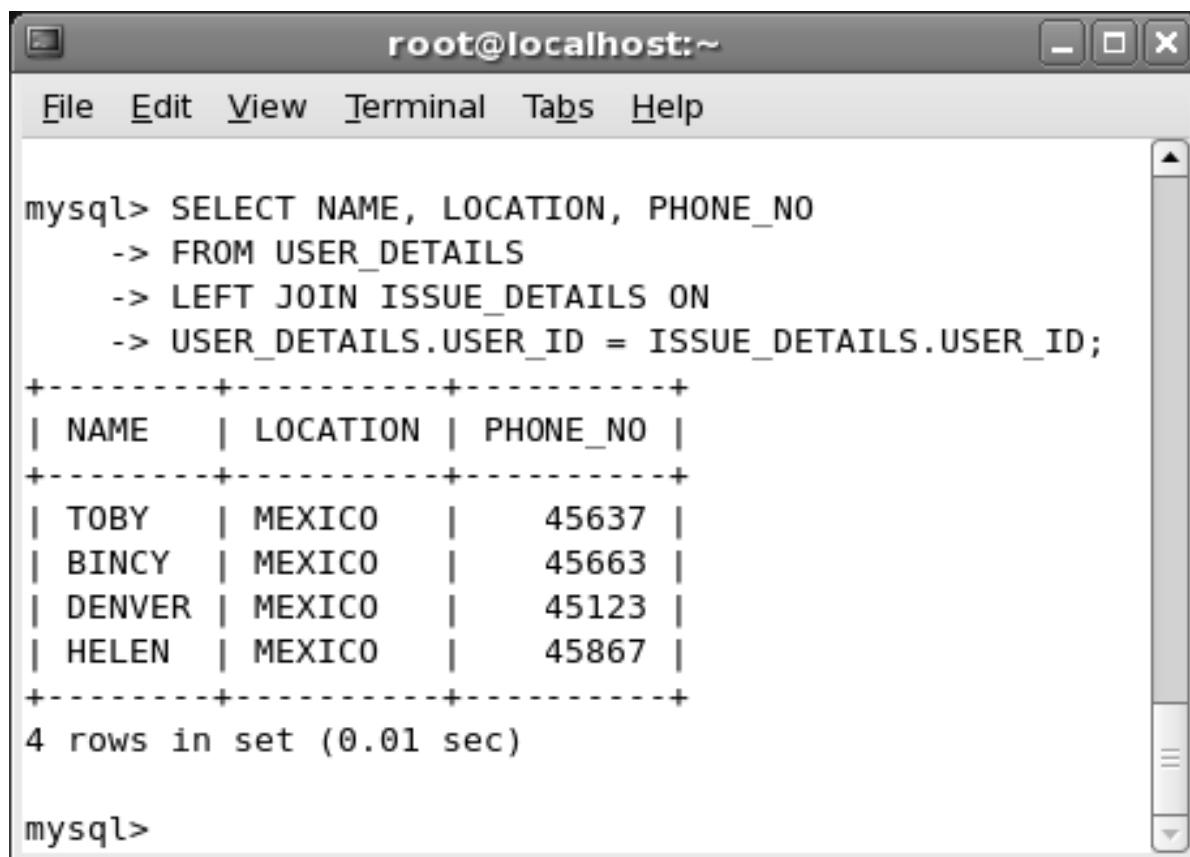
3. In order to compare and view the name, location, and phone numbers of books issued to members, you need to use LEFT JOIN and ON clause with USER ID as the common field. To do so, enter the following command at the command prompt:

```
SELECT NAME, LOCATION, PHONE_NO FROM USER_DETAILS
LEFT JOIN ISSUE_DETAILS ON
USER_DETAILS.USER_ID = ISSUE_DETAILS.USER_ID;
```

Figure 11.3 displays the output of the command.

Session 11

Using Joins (Lab)



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL query and its output:

```
mysql> SELECT NAME, LOCATION, PHONE_NO
-> FROM USER_DETAILS
-> LEFT JOIN ISSUE_DETAILS ON
-> USER_DETAILS.USER_ID = ISSUE_DETAILS.USER_ID;
+-----+-----+-----+
| NAME   | LOCATION | PHONE_NO |
+-----+-----+-----+
| TOBY    | MEXICO   | 45637    |
| BINCY   | MEXICO   | 45663    |
| DENVER  | MEXICO   | 45123    |
| HELEN   | MEXICO   | 45867    |
+-----+-----+-----+
4 rows in set (0.01 sec)

mysql>
```

The output shows four rows of data from the joined tables.

Figure 11.3: Joining Two Tables Using LEFT JOIN with ON Clause

4. In order to compare and view the name, location, and phone numbers of members who have been issued books, you need to use LEFT JOIN and USING clause with USER_ID as the common field. To do so, enter the following command at the command prompt:

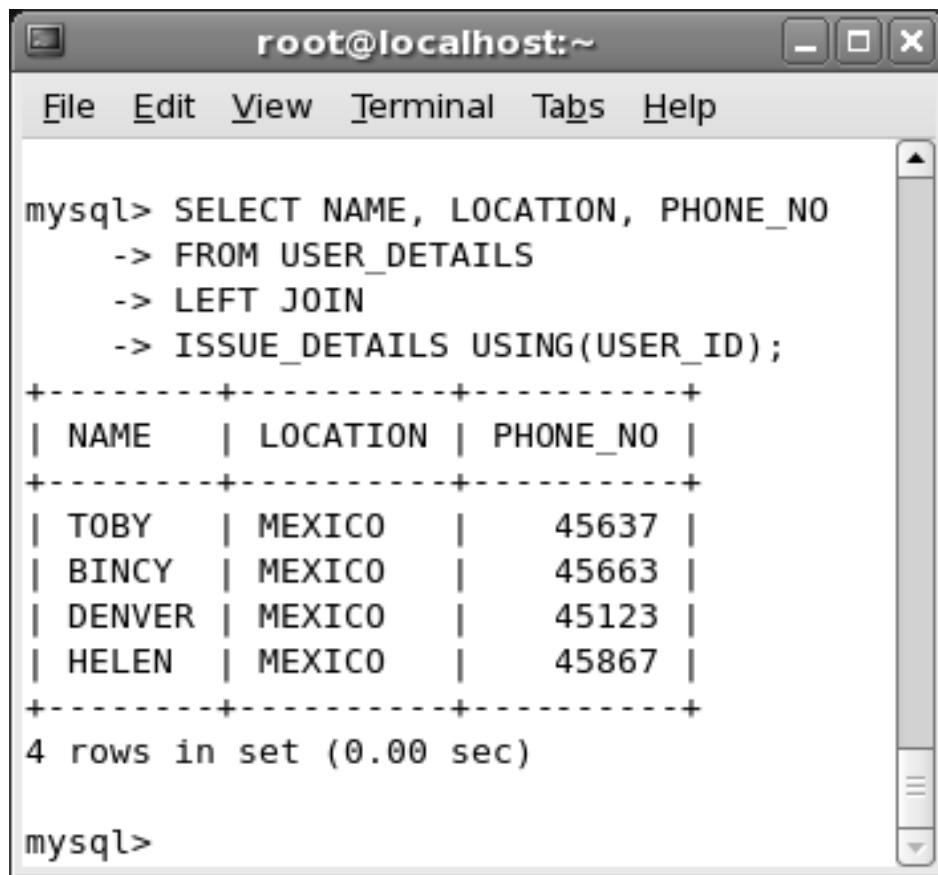
```
SELECT NAME, LOCATION, PHONE_NO
FROM USER_DETAILS
LEFT JOIN
ISSUE_DETAILS USING(USER_ID);
```

Figure 11.4 displays the output of the command.

Session 11

Using Joins (Lab)

Lab Guide



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its execution:

```
mysql> SELECT NAME, LOCATION, PHONE_NO
-> FROM USER_DETAILS
-> LEFT JOIN
-> ISSUE_DETAILS USING(USER_ID);
+-----+-----+
| NAME   | LOCATION | PHONE_NO |
+-----+-----+
| TOBY    | MEXICO   | 45637    |
| BINCY   | MEXICO   | 45663    |
| DENVER  | MEXICO   | 45123    |
| HELEN   | MEXICO   | 45867    |
+-----+-----+
4 rows in set (0.00 sec)

mysql>
```

The command performs a left join between the `USER_DETAILS` and `ISSUE_DETAILS` tables using the `USER_ID` column. The resulting table has columns `NAME`, `LOCATION`, and `PHONE_NO`. The output shows four rows of data.

Figure 11.4: Joining Two Tables Using LEFT JOIN with USING Clause

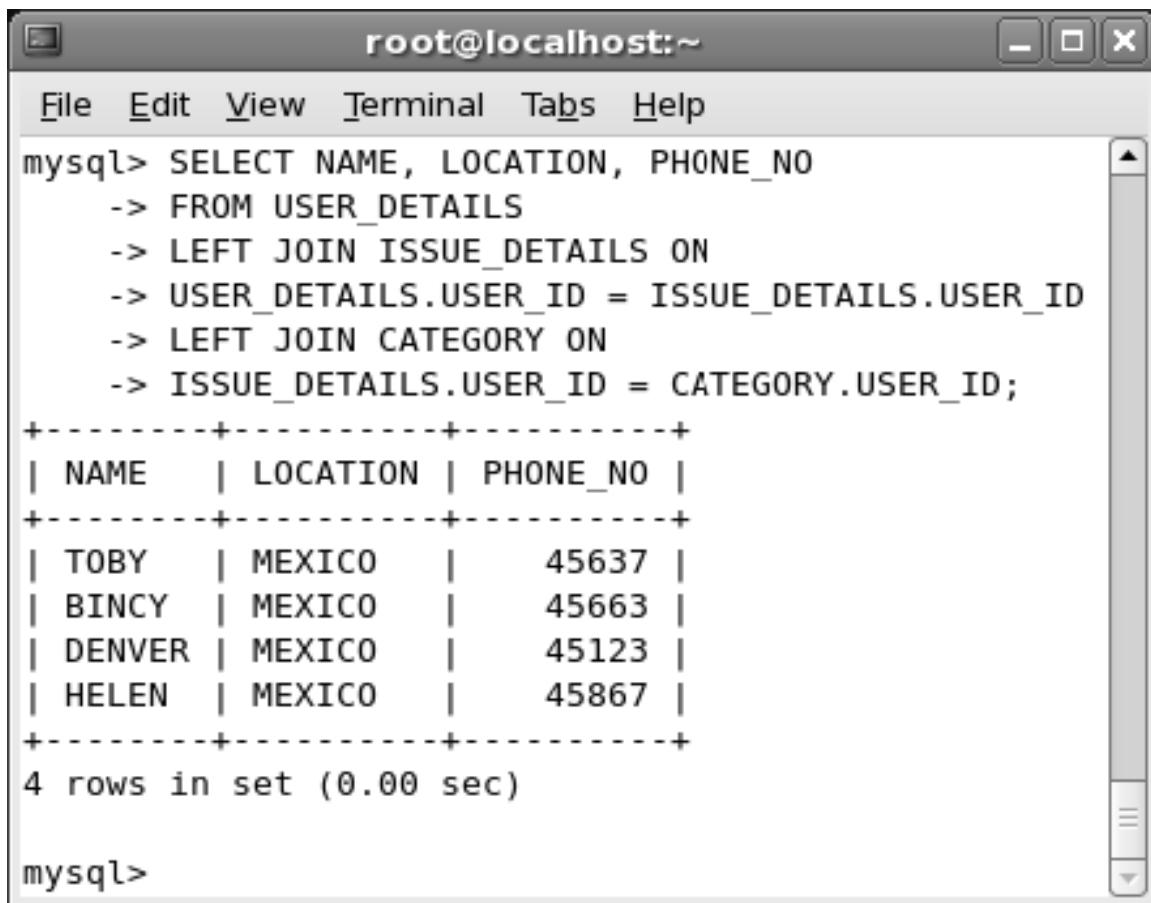
5. In order to display the name, phone number, and location from `USER_DETAILS`, `ISSUE_DETAILS`, and the `CATEGORY` tables you need to use LEFT JOIN with ON clause. To do so, enter the following command at the command prompt:

```
SELECT NAME, LOCATION, PHONE_NO FROM USER_DETAILS
LEFT JOIN ISSUE_DETAILS ON
USER_DETAILS.USER_ID = ISSUE_DETAILS.USER_ID
LEFT JOIN CATEGORY ON
ISSUE_DETAILS.USER_ID = CATEGORY.USER_ID;
```

Figure 11.5 displays the output of the command.

Session 11

Using Joins (Lab)



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL query and its output:

```
mysql> SELECT NAME, LOCATION, PHONE_NO
   -> FROM USER_DETAILS
   -> LEFT JOIN ISSUE_DETAILS ON
   -> USER_DETAILS.USER_ID = ISSUE_DETAILS.USER_ID
   -> LEFT JOIN CATEGORY ON
   -> ISSUE_DETAILS.USER_ID = CATEGORY.USER_ID;
+-----+-----+-----+
| NAME  | LOCATION | PHONE_NO |
+-----+-----+-----+
| TOBY   | MEXICO   | 45637    |
| BINYC  | MEXICO   | 45663    |
| DENVER | MEXICO   | 45123    |
| HELEN  | MEXICO   | 45867    |
+-----+-----+-----+
4 rows in set (0.00 sec)

mysql>
```

The output shows four rows of data with columns: NAME, LOCATION, and PHONE_NO. All four rows have the same LOCATION value (MEXICO) and PHONE_NO values (45637, 45663, 45123, 45867). The NAME values are TOBY, BINYC, DENVER, and HELEN respectively.

Figure 11.5: Joining Three Tables Using LEFT JOIN with ON Clause

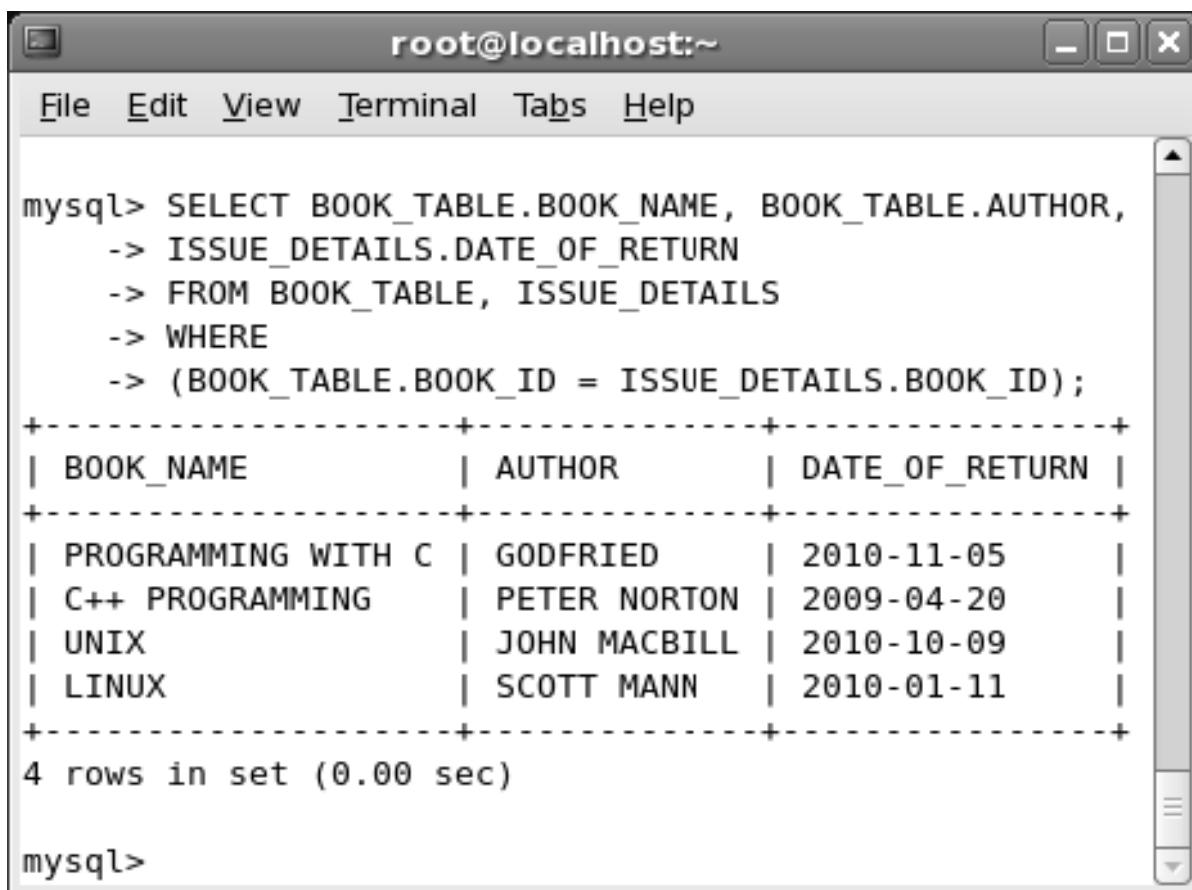
6. In order to display the book name, author, and return date from BOOK_TABLE and ISSUE_TABLE using equi-join with BOOK_ID as the common field, enter the following command at the command prompt:

```
SELECT BOOK_TABLE.BOOK_NAME, BOOK_TABLE.AUTHOR,
       ISSUE_DETAILS.DATE_OF_RETURN
  FROM BOOK_TABLE, ISSUE_DETAILS
 WHERE (BOOK_TABLE.BOOK_ID = ISSUE_DETAILS.BOOK_ID);
```

Figure 11.6 displays the output of the command.

Session 11

Using Joins (Lab)



```
root@localhost:~ File Edit View Terminal Tabs Help mysql> SELECT BOOK_TABLE.BOOK_NAME, BOOK_TABLE.AUTHOR,
-> ISSUE_DETAILS.DATE_OF_RETURN
-> FROM BOOK_TABLE, ISSUE_DETAILS
-> WHERE
-> (BOOK_TABLE.BOOK_ID = ISSUE_DETAILS.BOOK_ID);
+-----+-----+-----+
| BOOK_NAME      | AUTHOR        | DATE_OF_RETURN |
+-----+-----+-----+
| PROGRAMMING WITH C | GODFRIED      | 2010-11-05
| C++ PROGRAMMING    | PETER NORTON   | 2009-04-20
| UNIX            | JOHN MACBILL   | 2010-10-09
| LINUX           | SCOTT MANN     | 2010-01-11
+-----+-----+-----+
4 rows in set (0.00 sec)

mysql>
```

Lab Guide

Figure 11.6: Joining Two Tables Using Equi-Join

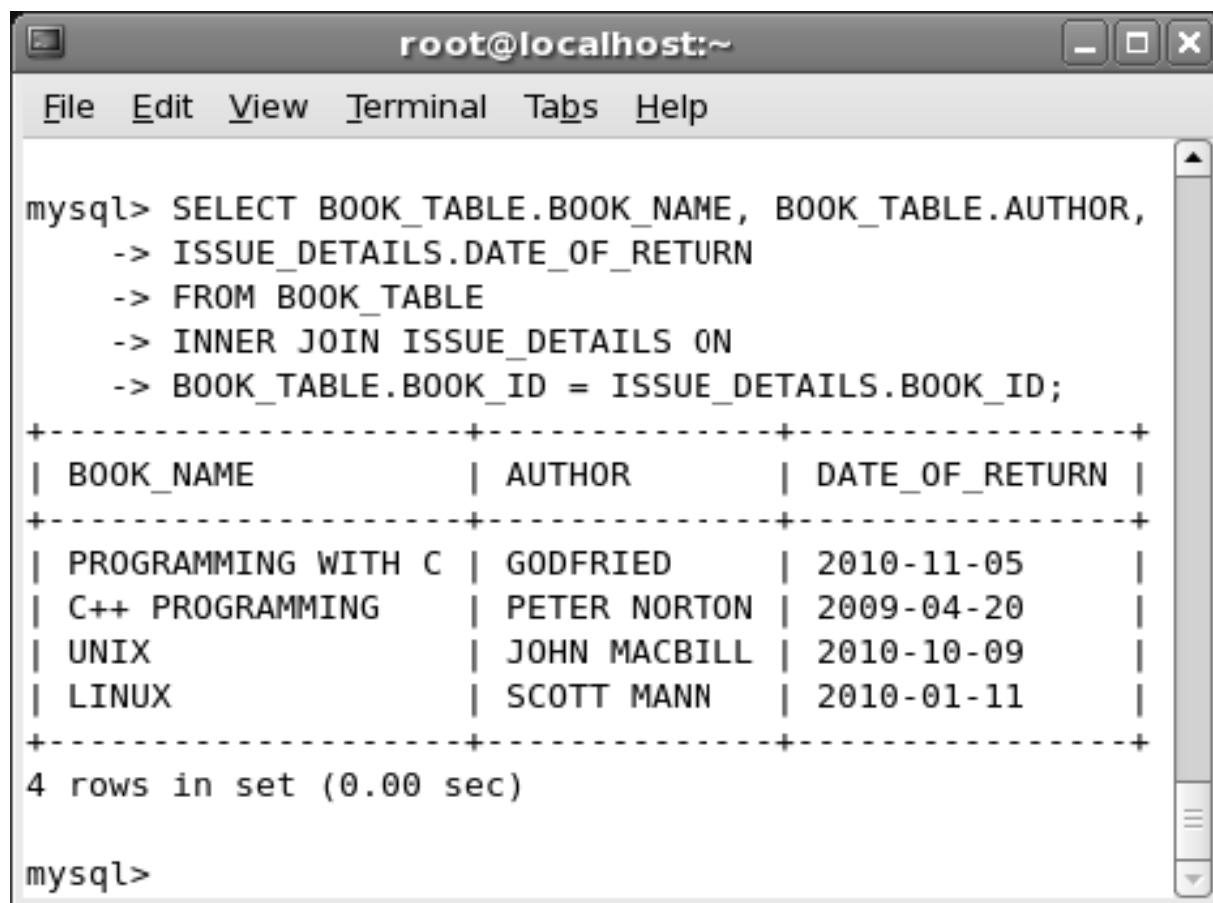
7. In order to view the return date for a book, book name, and the author from BOOK_TABLE and ISSUE DETAILS table using INNER JOIN, enter the following command at the command prompt:

```
SELECT BOOK_TABLE.BOOK_NAME, BOOK_TABLE.AUTHOR,
ISSUE_DETAILS.DATE_OF_RETURN
FROM BOOK_TABLE
INNER JOIN ISSUE_DETAILS ON
BOOK_TABLE.BOOK_ID = ISSUE_DETAILS.BOOK_ID;
```

Figure 11.7 displays the output of the command.

Session 11

Using Joins (Lab)



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL query and its results:

```
mysql> SELECT BOOK_TABLE.BOOK_NAME, BOOK_TABLE.AUTHOR,
-> ISSUE_DETAILS.DATE_OF_RETURN
-> FROM BOOK_TABLE
-> INNER JOIN ISSUE_DETAILS ON
-> BOOK_TABLE.BOOK_ID = ISSUE_DETAILS.BOOK_ID;
+-----+-----+-----+
| BOOK_NAME      | AUTHOR        | DATE_OF_RETURN |
+-----+-----+-----+
| PROGRAMMING WITH C | GODFRIED      | 2010-11-05    |
| C++ PROGRAMMING   | PETER NORTON  | 2009-04-20    |
| UNIX            | JOHN MACBILL  | 2010-10-09    |
| LINUX           | SCOTT MANN    | 2010-01-11    |
+-----+-----+-----+
4 rows in set (0.00 sec)

mysql>
```

The output shows four rows of data from the joined tables. The columns are BOOK_NAME, AUTHOR, and DATE_OF_RETURN.

Figure 11.7: Joining Two Tables Using INNER JOIN with ON Clause

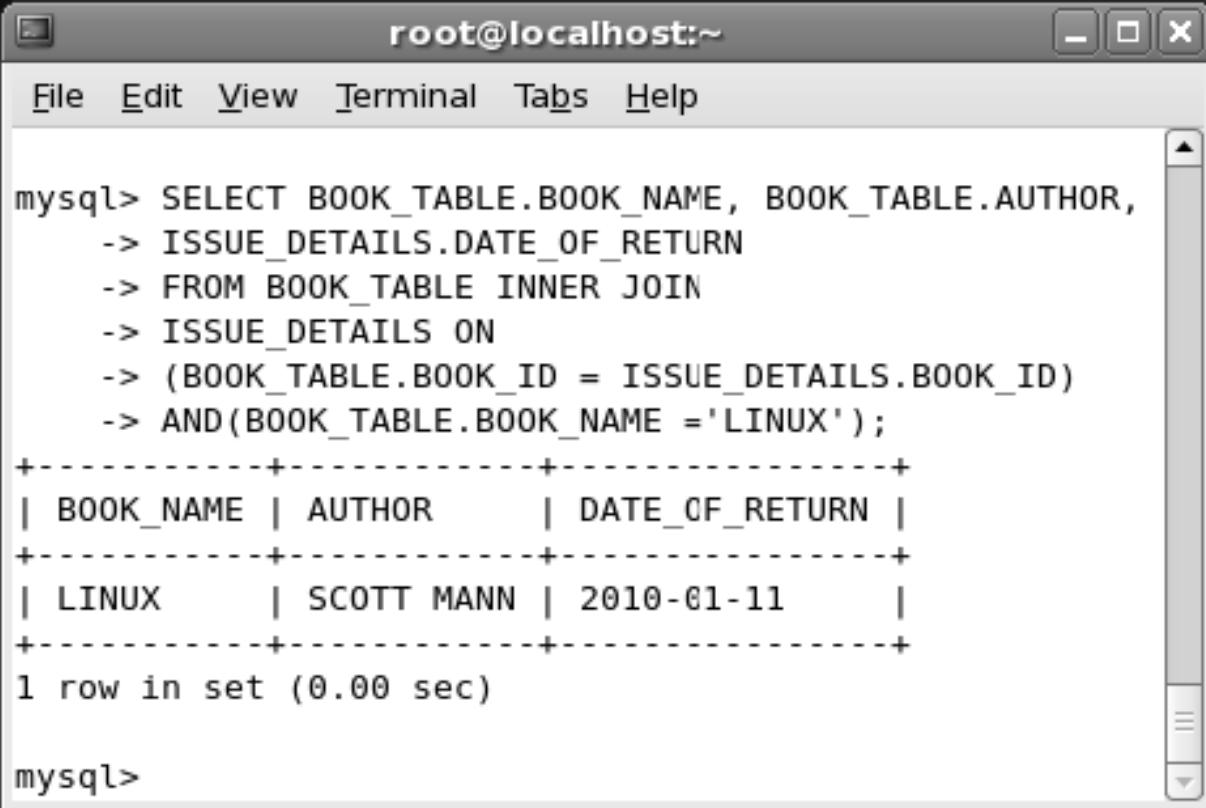
8. In order to display the book name, author, and date of return from the BOOK_TABLE and ISSUE_DETAILS table for Linux books. To display these details using INNER JOIN with the ON clause with BOOK_ID as the common field, enter the following command at the command prompt:

```
SELECT BOOK_TABLE.BOOK_NAME, BOOK_TABLE.AUTHOR,
       ISSUE_DETAILS.DATE_OF_RETURN
  FROM BOOK_TABLE INNER JOIN
       ISSUE_DETAILS ON
          (BOOK_TABLE.BOOK_ID = ISSUE_DETAILS.BOOK_ID)
     AND (BOOK_TABLE.BOOK_NAME = 'LINUX');
```

Figure 11.8 displays the output of the command.

Session 11

Using Joins (Lab)



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL query and its output:

```
mysql> SELECT BOOK_TABLE.BOOK_NAME, BOOK_TABLE.AUTHOR,
-> ISSUE_DETAILS.DATE_OF_RETURN
-> FROM BOOK_TABLE INNER JOIN
-> ISSUE_DETAILS ON
-> (BOOK_TABLE.BOOK_ID = ISSUE_DETAILS.BOOK_ID)
-> AND(BOOK_TABLE.BOOK_NAME = 'LINUX');
+-----+-----+-----+
| BOOK_NAME | AUTHOR      | DATE_OF_RETURN |
+-----+-----+-----+
| LINUX     | SCOTT MANN   | 2010-01-11    |
+-----+-----+-----+
1 row in set (0.00 sec)

mysql>
```

A vertical scroll bar is visible on the right side of the terminal window.

Lab Guide

Figure 11.8: Joining Two Tables Using INNER JOIN

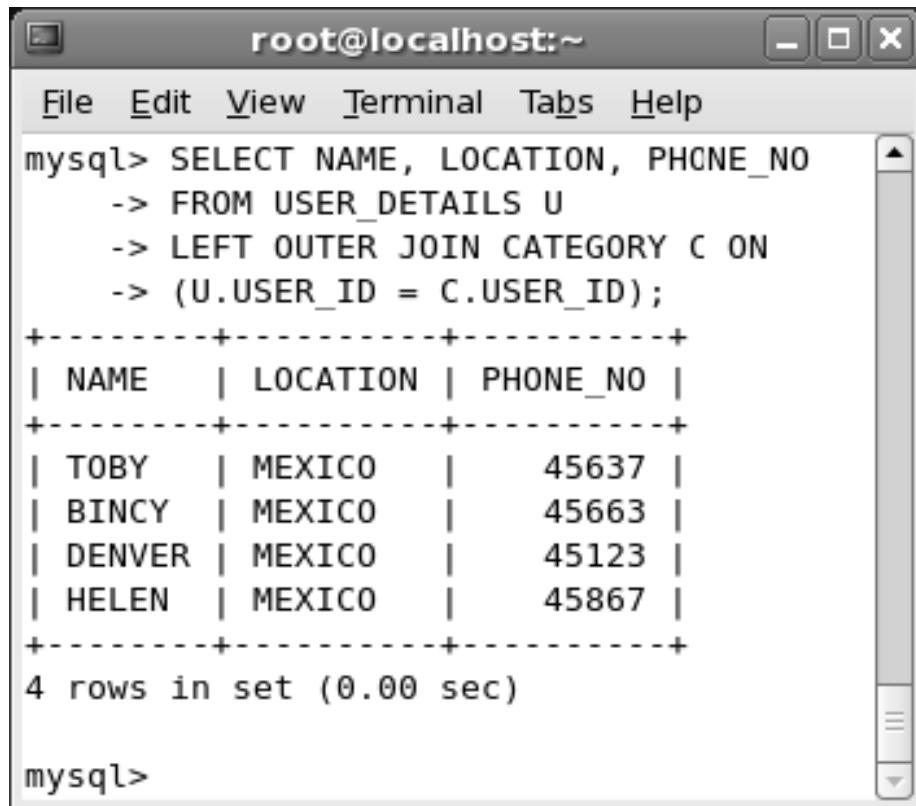
9. In order to view the user details, such as name, location, and phone number of registered members, using LEFT OUTER JOIN and USER_ID as the common field, enter the following command at the command prompt:

```
SELECT NAME, LOCATION, PHONE_NO FROM USER_DETAILS U
LEFT OUTER JOIN CATEGORY C ON
(U.USER_ID = C.USER_ID);
```

Figure 11.9 displays the output of the command.

Session 11

Using Joins (Lab)



The screenshot shows a terminal window titled "root@localhost:~". The MySQL command line interface is running. A query is being executed to select columns NAME, LOCATION, and PHONE_NO from the USER_DETAILS table (alias U), performing a LEFT OUTER JOIN with the CATEGORY table (alias C) on the condition U.USER_ID = C.USER_ID. The result set contains four rows, each with the same location (MEXICO) and different phone numbers (45637, 45663, 45123, 45867). The command concludes with a prompt mysql>.

```
mysql> SELECT NAME, LOCATION, PHONE_NO
   -> FROM USER_DETAILS U
   -> LEFT OUTER JOIN CATEGORY C ON
   -> (U.USER_ID = C.USER_ID);
+-----+-----+-----+
| NAME    | LOCATION | PHONE_NO |
+-----+-----+-----+
| TOBY    | MEXICO  | 45637   |
| BINCY   | MEXICO  | 45663   |
| DENVER  | MEXICO  | 45123   |
| HELEN   | MEXICO  | 45867   |
+-----+-----+-----+
4 rows in set (0.00 sec)

mysql>
```

Figure 11.9: Joining Two Tables Using the LEFT OUTER JOIN

Using Subqueries and UNION

A query used within a `SELECT` query is called a subquery. You can use the `UNION` clause to combine the results of multiple `SELECT` statements into a single result set.

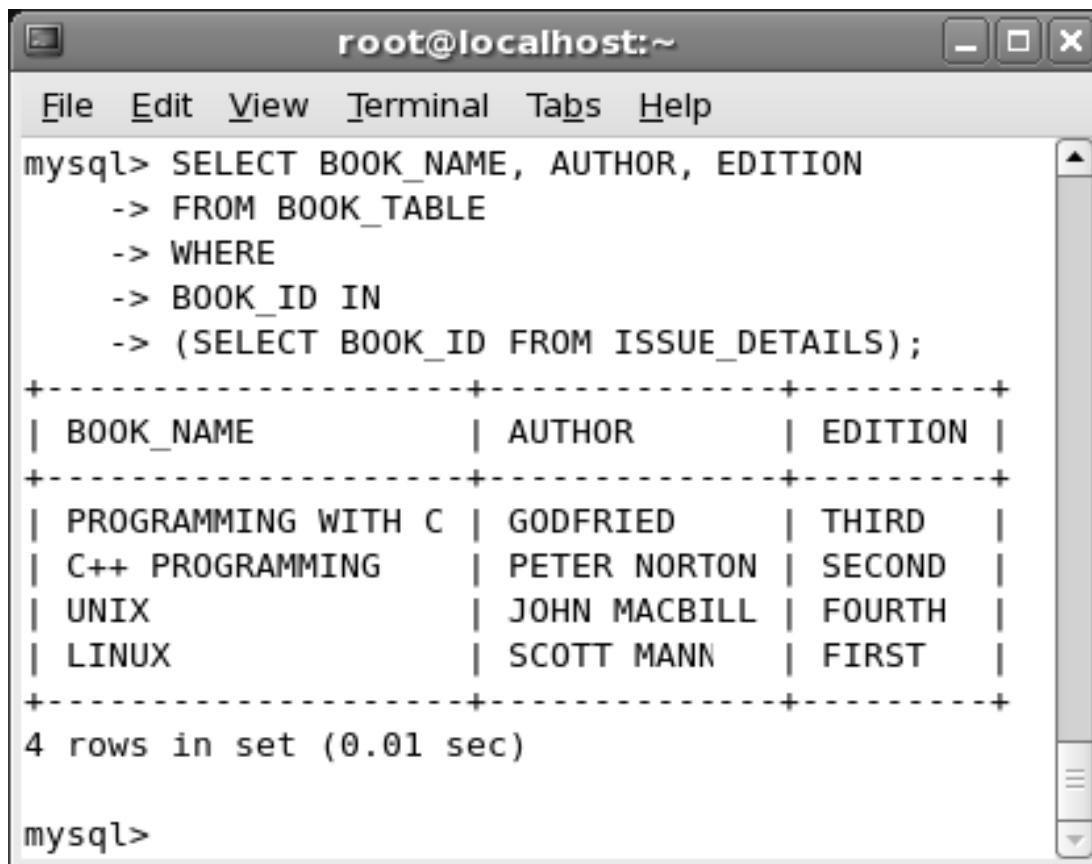
1. To display the book name, author, and edition for all the books that have been issued to the users, you need to create a subquery between the `BOOK_TABLE` and `ISSUE_DETAILS` table. To view the records, enter the following command at the command prompt:

```
SELECT BOOK_NAME, AUTHOR, EDITION
FROM BOOK_TABLE
WHERE BOOK_ID IN(SELECT BOOK_ID FROM ISSUE_DETAILS);
```

Figure 11.10 displays the output of the command.

Session 11

Using Joins (Lab)



The screenshot shows a terminal window titled "root@localhost:~". The MySQL command line interface is running. The user has entered a query to select book names, authors, and editions from the BOOK_TABLE where the book ID is present in the ISSUE_DETAILS table. The output shows four rows of data.

```
mysql> SELECT BOOK_NAME, AUTHOR, EDITION
    -> FROM BOOK_TABLE
    -> WHERE
    -> BOOK_ID IN
    -> (SELECT BOOK_ID FROM ISSUE_DETAILS);
+-----+-----+-----+
| BOOK_NAME      | AUTHOR        | EDITION      |
+-----+-----+-----+
| PROGRAMMING WITH C | GODFRIED      | THIRD        |
| C++ PROGRAMMING   | PETER NORTON  | SECOND       |
| UNIX            | JOHN MACBILL  | FOURTH       |
| LINUX           | SCOTT MANN    | FIRST        |
+-----+-----+-----+
4 rows in set (0.01 sec)

mysql>
```

BOOK_NAME	AUTHOR	EDITION
PROGRAMMING WITH C	GODFRIED	THIRD
C++ PROGRAMMING	PETER NORTON	SECOND
UNIX	JOHN MACBILL	FOURTH
LINUX	SCOTT MANN	FIRST

Lab Guide

Figure 11.10: Joining Two Tables Using Subquery

2. In order to combine the result of two SELECT queries and display books from the BOOK_TABLE table using UNION clause and a condition specified using the BOOK_ID, enter the following command at the command prompt:

```
(SELECT * FROM BOOK_TABLE WHERE BOOK_ID = 101
    ORDER BY BOOK_ID LIMIT 3)

UNION

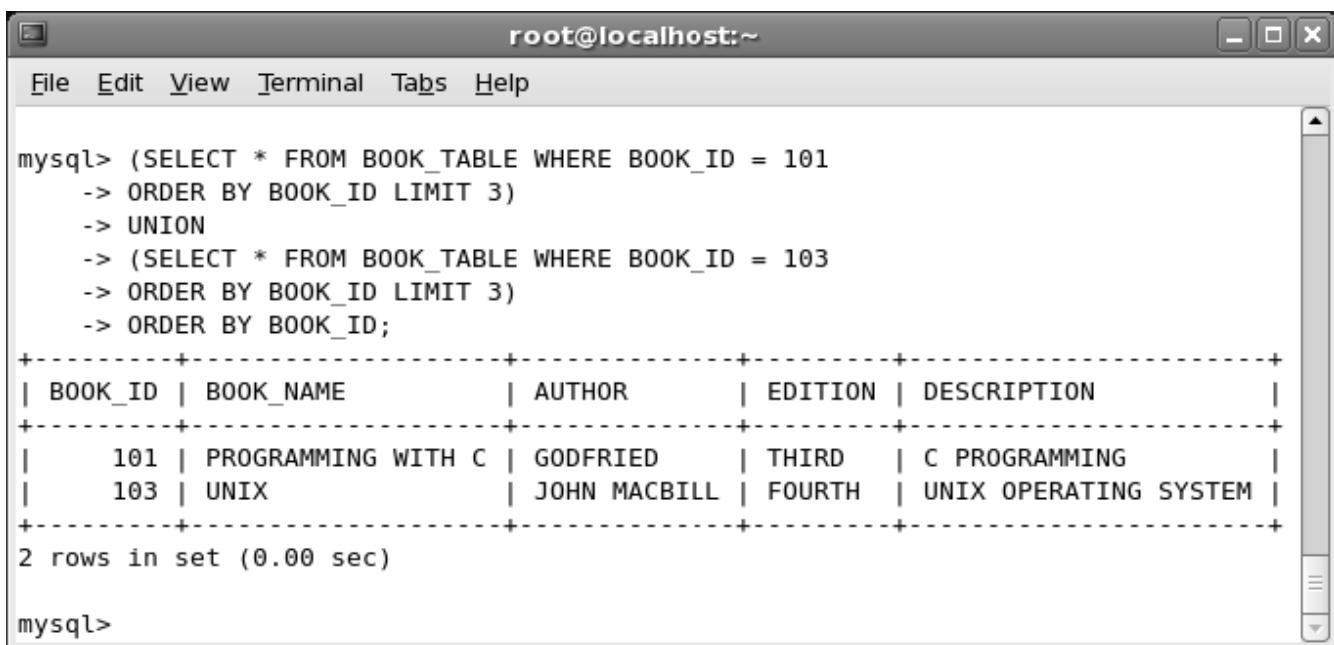
(SELECT * FROM BOOK_TABLE WHERE BOOK_ID = 103
    ORDER BY BOOK_ID LIMIT 3)

ORDER BY BOOK_ID;
```

Figure 11.11 displays the output of the command.

Session 11

Using Joins (Lab)



The screenshot shows a terminal window titled "root@localhost:~". The menu bar includes File, Edit, View, Terminal, Tabs, and Help. The terminal window displays the following MySQL query and its results:

```
mysql> (SELECT * FROM BOOK_TABLE WHERE BOOK_ID = 101
-> ORDER BY BOOK_ID LIMIT 3)
-> UNION
-> (SELECT * FROM BOOK_TABLE WHERE BOOK_ID = 103
-> ORDER BY BOOK_ID LIMIT 3)
-> ORDER BY BOOK_ID;
+-----+-----+-----+-----+
| BOOK_ID | BOOK_NAME        | AUTHOR      | EDITION | DESCRIPTION
+-----+-----+-----+-----+
|    101  | PROGRAMMING WITH C | GODFRIED    | THIRD    | C PROGRAMMING
|    103  | UNIX              | JOHN MACBILL | FOURTH   | UNIX OPERATING SYSTEM
+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql>
```

The query uses the UNION clause to combine two SELECT statements from the BOOK_TABLE. The first statement selects rows where BOOK_ID is 101, ordered by BOOK_ID. The second statement selects rows where BOOK_ID is 103, also ordered by BOOK_ID. The final result is ordered by BOOK_ID. The output shows two rows: one for book ID 101 (Programming with C) and one for book ID 103 (UNIX).

Figure 11.11: Joining Two Tables Using UNION Clause

Session 11

Using Joins (Lab)



Do It Yourself

1. Select the PRODUCT_SALES database.
2. Join the Stock_Details table and Sales_Details tables with the help of the common field, Product_Code.

Stock_Details table:

Product_Code	Product_Name	Description	Cost Price	Date_Of_Manufacture
101	Bags	Leather Bags	150	2003-10-05
102	Books	Notepads	10	2003-12-05
103	Pens	Fine Tip Pens	20	2003-01-09
104	56	120	9	2003-10-05

Sales_Details table:

Product_Code	Selling Price	Qty	Units Sold	Month	Location
101	200	1500	1450	Jan	California
102	100	5000	3000	Jan	New York
103	20	2900	2000	Jan	Los Angeles
104	56	120	9	Jan	Sydney
101	200	1000	500	Jan	New York

3. Display Product_Name, Description, Cost Price, Selling Price, Qty, and Location field values from Stock_Details and Sales_Details table using table reference.
4. Display Product_Name, Description, Cost Price, Selling Price, and Qty field values from Stock_Details and Sales_Details table using table alias names.
5. Display Product_Name, Description, Cost Price, and Date_Of_Manufacture field values from Stock_Details and Sales_Details table. Display all the records from the table on the left with or without matching records on the right using the ON clause.
6. Display Product_Name, Description, Cost Price, and Date_Of_Manufacture field values from Stock_Details and Sales_Details table. Display all the records from the table on the left with or without matching records on the right with USING clause.
7. Display Product_Name, Description, Cost Price, Selling Price, and Qty field values from Stock_Details and Sales_Details table using Inner Join.

Session 11

Using Joins (Lab)



Do It Yourself

8. Display Product_Name, Description, Cost Price, and Date_Of_Manufacture field values from Stock_Details and Sales_Details table using the Left Outer Join.
9. Display Product_Name, Description, Cost Price, and Date_Of_Manufacture field values from Stock_Details and Sales_Details table. Display the records by using a SELECT query within another SELECT query.
10. Display Product_Name, Description, Cost Price, and Date_Of_Manufacture field values from Stock_Details and Sales_Details table using the UNION clause.

Objectives

At the end of this session, the student will be able to:

- *Use the Aggregate functions in MySQL.*
- *Use the Mathematical functions in MySQL.*

12.1 Introduction

Functions are independent programs that work on the parameter passed and returns a value depending on the parameter. Parameters are also known as arguments. You can use MySQL functions within a query to operate on single or multiple columns of a table.

In this session, you will learn about the basics of using functions. You will use different types of functions, such as aggregate and mathematical functions. In addition, you will also learn the use of functions in the GROUP BY and HAVING clause.

12.2 Basics of Using Functions

The functions used in MySQL are similar to any other programming language. In MySQL, you can use functions along with the SELECT command in two different ways.

- **Value to be retrieved** - In this form, the function is used with column names. Consider the following query:

```
SELECT E_ID, E_FNAME, LENGTH(E_FNAME) FROM EMP_DETAILS WHERE E_ID=101;
```

This query returns the E_ID and E_FNAME columns from the EMP_DETAILS table with the length of E_FNAME.

- **Part of a WHERE clause** - In this form, the function is used in the WHERE clause. The value specified is compared for each row in the table. Consider the following query:

```
SELECT E_ID, D_NAME FROM EMP_SALARY HAVING AVG(BASIC_SAL)<8000;
```

12.3 Using Aggregate Functions in MySQL

Aggregate functions operate on a group of values and return a single value as the final result. Some

Session 12

Using Basic Functions in MySQL - I

aggregate functions work with the GROUP BY and HAVING clause. The GROUP BY clause is used to group rows having similar values for a specific column into a single row. Group functions do not accept NULL values. The HAVING clause defines the result set based on a set of calculations. You can use the GROUP BY and HAVING clause with the SELECT statement to retrieve data that satisfies the specified conditions defined in the HAVING clause. The functions discussed in this section illustrate the use of functions in a GROUP BY and HAVING clause.

12.3.1 AVG Function

The AVG function returns the mean value of the argument. The syntax for using this function is:

```
SELECT COLUMN_NAME FROM TABLE_NAME HAVING AVG(expression);
```

For example, to obtain the E_ID and D_NAME of the employee whose average basic salary is less than 8000, enter the following command at the command prompt:

```
SELECT E_ID, D_NAME FROM EMP_SALARY HAVING AVG(BASIC_SAL)<8000;
```

Figure 12.1 displays the output of the command.

```
root@localhost:~
```

```
File Edit View Terminal Tabs Help
```

```
mysql> SELECT E_ID, D_NAME
-> FROM EMP_SALARY
-> HAVING
-> AVG(BASIC_SAL)<8000;
+-----+
| E_ID | D_NAME   |
+-----+
| 101  | RESEARCH |
+-----+
1 row in set (0.00 sec)
```

```
mysql>
```

Figure 12.1: AVG Function

Session 12

Using Basic Functions in MySQL - I

Concepts

12.3.2 BITWISE AND Function

The `BITWISE AND` function works with two arguments that have equal length. `BITWISE AND` converts the arguments into binary format and compares every bit of the two arguments.

The syntax for this function is:

```
SELECT BIT1 & BIT2;
```

where,

`BIT1, BIT2` – defines the operands

To find out the `BITWISE AND` for 29 and 15, enter the following command at the command prompt:

```
SELECT 29 & 15;
```

Figure 12.2 displays the output of the command.

The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area contains the following MySQL command and its output:

```
mysql> SELECT 29 & 15;
+-----+
| 29 & 15 |
+-----+
|      13 |
+-----+
1 row in set (0.00 sec)

mysql>
```

The output shows the result of the bitwise AND operation between 29 and 15, which is 13.

Figure 12.2: BIT AND Function

12.3.3 COUNT Function

The `COUNT` function returns the total number of non-NULL values of the expression specified as the argument. The syntax for obtaining the count of any expression is:

Session 12

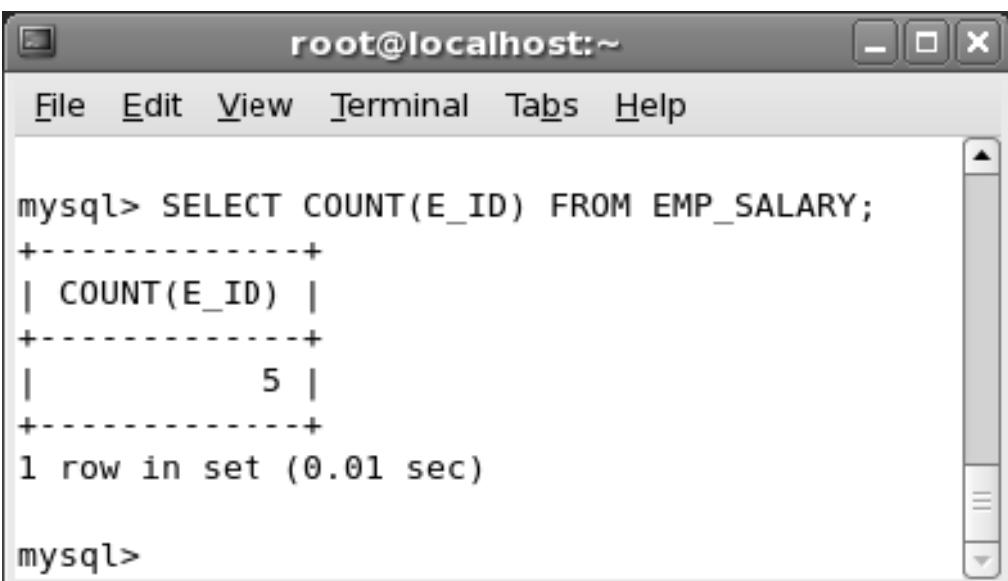
Using Basic Functions in MySQL - I

```
SELECT COUNT(expression) FROM TABLE_NAME;
```

For example, to obtain the count of the `E_ID` column from the `EMP_SALARY` table, enter the following command at the command prompt:

```
SELECT COUNT(E_ID) FROM EMP_SALARY;
```

Figure 12.3 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a standard OS X-style title bar with icons for minimize, maximize, and close. Below the title bar is a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The main area of the terminal contains the following text:

```
mysql> SELECT COUNT(E_ID) FROM EMP_SALARY;
+-----+
| COUNT(E_ID) |
+-----+
|      5      |
+-----+
1 row in set (0.01 sec)

mysql>
```

Figure 12.3: COUNT Function

12.3.4 COUNT(DISTINCT) Function

The `COUNT(DISTINCT)` function returns the number of unique values in a column. You can retrieve the number of different combinations of unique values in two or more columns of the same table. If there are no matching values in any of the columns, then the `COUNT(DISTINCT)` function returns 0. The syntax for using this function is:

```
SELECT COUNT(DISTINCT expression) FROM TABLE_NAME;
```

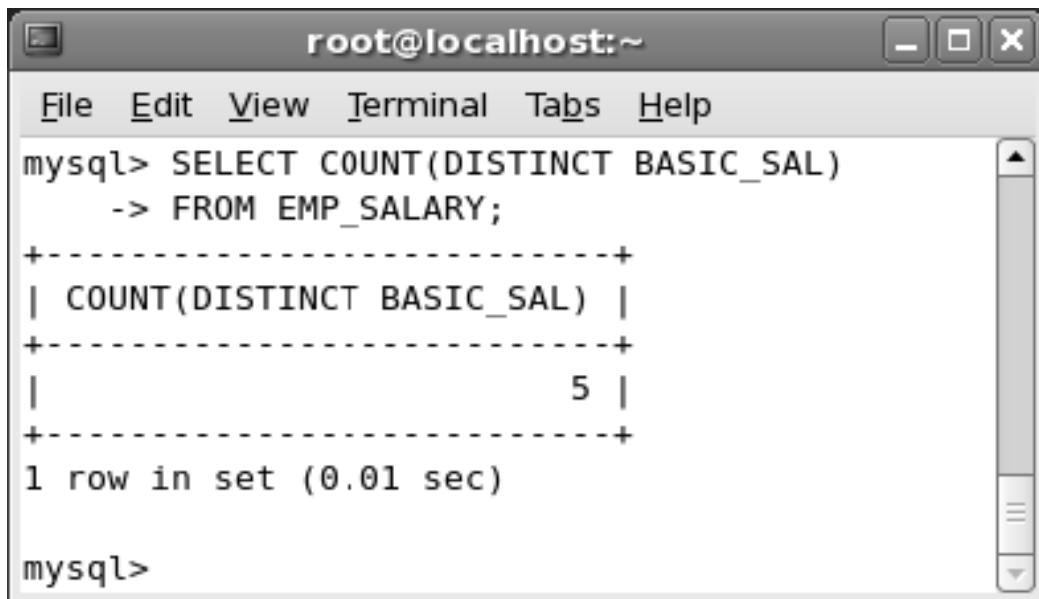
For example, to obtain the number of employees who have different basic salary, enter the following command at the command prompt:

```
SELECT COUNT(DISTINCT BASIC_SAL) FROM EMP_SALARY;
```

Session 12

Using Basic Functions in MySQL - I

Figure 12.4 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its output:

```
mysql> SELECT COUNT(DISTINCT BASIC_SAL)
   -> FROM EMP_SALARY;
+-----+
| COUNT(DISTINCT BASIC_SAL) |
+-----+
|                      5 |
+-----+
1 row in set (0.01 sec)

mysql>
```

The output shows that there are 5 distinct values in the BASIC_SAL column of the EMP_SALARY table.

Concepts

Figure 12.4: COUNT (DISTINCT) Function

12.3.5 GROUP_CONCAT Function

The GROUP_CONCAT function joins the unique values of the argument and returns them as a string. This function returns NULL if the argument has no non-NULL values. To eliminate duplicate values, you can use the DISTINCT clause. The syntax for using this function is:

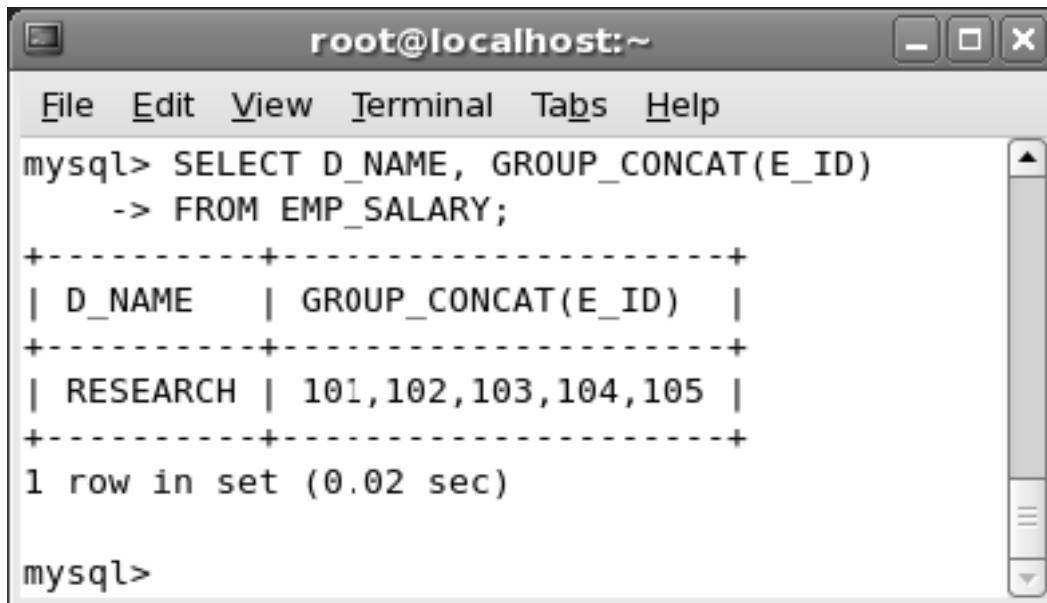
```
SELECT GROUP_CONCAT(expression) FROM TABLE_NAME;
```

For example, to display all the E_ID from the EMP_SALARY table that have RESEARCH as the D_NAME, enter the following command at the command prompt:

```
SELECT D_NAME, GROUP_CONCAT(E_ID) FROM EMP_SALARY;
```

Figure 12.5 displays the output of the command.

Session 12



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its output:

```
mysql> SELECT D_NAME, GROUP_CONCAT(E_ID)
   -> FROM EMP_SALARY;
+-----+-----+
| D_NAME | GROUP_CONCAT(E_ID) |
+-----+-----+
| RESEARCH | 101,102,103,104,105 |
+-----+-----+
1 row in set (0.02 sec)

mysql>
```

The output shows a single row from the EMP_SALARY table where the department name is "RESEARCH" and the concatenated employee IDs are "101,102,103,104,105".

Figure 12.5: GROUP_CONCAT Function

12.3.6 MAX Function

The MAX function returns the greatest value of the expression. The syntax for obtaining the greatest value is:

```
SELECT COLUMN_NAME FROM TABLE_NAME HAVING MAX(expression);
```

For example, display the maximum gross salary earned by any of the employee.

```
SELECT MAX(GROSS_SAL) FROM EMP_SALARY;
```

Figure 12.6 displays the output of the command.

Session 12

Using Basic Functions in MySQL - I

Concepts

The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its output:

```
mysql> SELECT MAX(GROSS_SAL) FROM EMP_SALARY;
+-----+
| MAX(GROSS_SAL) |
+-----+
|      5500.00 |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 12.6: MAX Function

12.3.7 STD Function

The `STD` function returns the standard deviation of the argument. Standard deviation specifies the variation of a population from its mean value. The output of this function is `NULL` if there are no rows satisfying the given query. The syntax to use this function is:

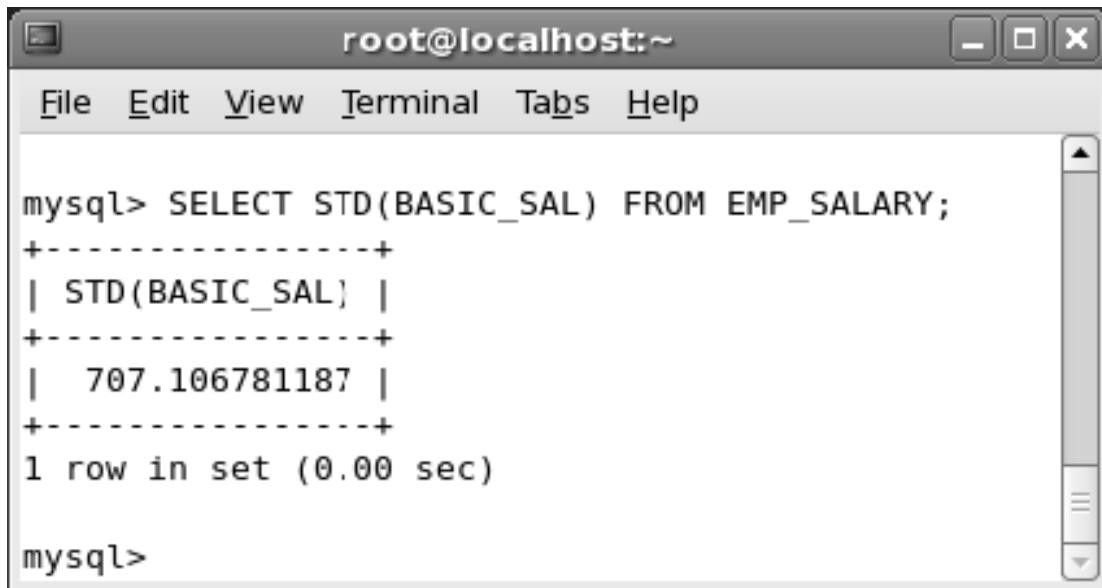
```
SELECT STD(expression) FROM TABLE_NAME;
```

For example, to obtain the standard deviation of the basic salary of all of the employees, enter the following command at the command prompt:

```
SELECT STD(BASIC_SAL) FROM EMP_SALARY;
```

Figure 12.7 displays the output of the command.

Session 12



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The main area displays the following MySQL command and its output:

```
mysql> SELECT STD(BASIC_SAL) FROM EMP_SALARY;
+-----+
| STD(BASIC_SAL) |
+-----+
| 707.106781187 |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 12.7: STD Function

12.3.8 SUM Function

The `SUM` function adds the values specified in the expression. The syntax for obtaining the sum is:

```
SELECT SUM(expression) FROM TABLE_NAME;
```

For example, to obtain the sum of `HRA` of all the employees, enter the following command at the command prompt:

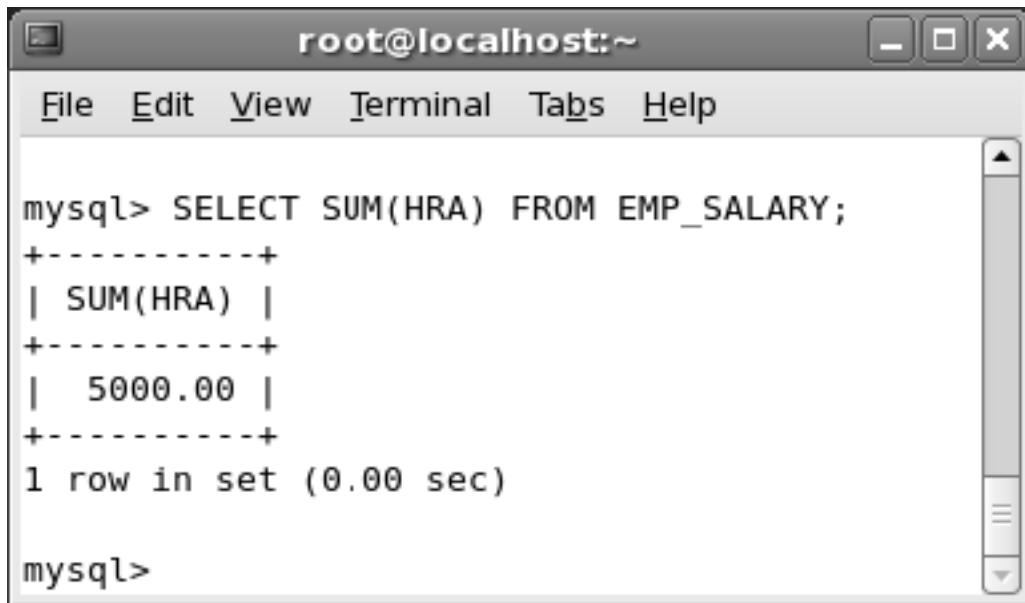
```
SELECT SUM(HRA) FROM EMP_SALARY;
```

Figure 12.8 displays the output of the command.

Session 12

Using Basic Functions in MySQL - I

Concepts



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area displays the following MySQL command and its result:

```
mysql> SELECT SUM(HRA) FROM EMP_SALARY;
+-----+
| SUM(HRA) |
+-----+
| 5000.00 |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 12.8: SUM Function

12.3.9 VARIANCE Function

The VARIANCE function returns the variance of the values of the argument. The syntax for obtaining the variance is:

```
SELECT VARIANCE(expression) FROM TABLE_NAME;
```

To obtain the variance of gross salary of all the employees, enter the following command at the command prompt:

```
SELECT VARIANCE(GROSS_SAL) FROM EMP_SALARY;
```

Figure 12.9 displays the output of the command.

Session 12

Using Basic Functions in MySQL - I

The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its output:

```
mysql> SELECT VARIANCE(GROSS_SAL) FROM EMP_SALARY;
+-----+
| VARIANCE(GROSS_SAL) |
+-----+
|      500000.000000 |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 12.9: VARIANCE Function

Table 12.1 describes additional AGGREGATE functions supported by MySQL:

Name	Description	Example
BITWISE OR	<p>The BITWISE OR function works with two arguments of equal length. BITWISE OR compares every bit of the two arguments. This function returns 0 only if both the bits being compared equal to 0, else it returns 1. The syntax for this function is:</p> <pre>SELECT BIT1 BIT2;</pre> <p>where,</p> <p>BIT1, BIT2 – specify the values for the operands</p>	<p>To calculate the BITWISE OR for 30 and 12, enter the following command at the command prompt:</p> <pre>SELECT 30 12;</pre> <p>The output of the command is:</p> <p>30</p>
BITWISE XOR	<p>The BITWISE XOR function works with two arguments of equal length. BITWISE XOR compares every bit of the two arguments. This function returns 0 in the output if there are no matching rows in the comparison. The syntax for this function is:</p>	<p>To find out the BITWISE XOR for 5 and 2, enter the following command at the command prompt:</p> <pre>SELECT 5 ^ 2;</pre>

Session 12

Using Basic Functions in MySQL - I

Concepts

Name	Description	Example
BITWISE XOR	<pre>SELECT BIT1 ^ BIT2;</pre> <p>where,</p> <p>BIT1, BIT2 – specifies a value for the operands</p>	The output of the command is: 7
MIN	<p>The MIN function returns the smallest value of the expression. The syntax for using this function is:</p> <pre>SELECT COLUMN_NAME FROM TABLE_NAME HAVING MIN(expression);</pre>	For example, display the minimum gross salary earned by any of the employee. <pre>SELECT MIN(GROSS_SAL) FROM EMP_SALARY;</pre> <p>The output of this function is:</p> <p>3500.00</p>
STDDEV	<p>The STDDEV function also returns the standard deviation of the values of the argument. The output of this function is NULL if there are no rows satisfying the given query. The syntax for obtaining the standard deviation is:</p> <pre>SELECT STDDEV(expression) FROM TABLE_NAME;</pre>	For example, to obtain the standard deviation of the basic salary of all the employees, enter the following command at the command prompt: <pre>SELECT STDDEV(BASIC_SAL) FROM EMP_SALARY;</pre> <p>The output of the function is:</p> <p>707.106781</p>
STDDEV_POP	<p>The STDDEV_POP function calculates the population standard deviation and returns the square root of the variance. The syntax for this function is:</p> <pre>SELECT STDDEV_POP(expression) FROM TABLE_NAME;</pre>	For example, to obtain the standard deviation of the gross salary of all the employees, enter the following command at the command prompt: <pre>SELECT STDDEV_POP(GROSS_SAL) FROM EMP_SALARY;</pre> <p>The output of the function is:</p> <p>707.106781</p>
STDDEV_SAMP	<p>The STDDEV_SAMP function returns the sample standard deviation of the argument. Sample standard deviation is</p>	For example, to obtain the sample standard deviation of the gross salary of all the employees, enter the following

Session 12

Name	Description	Example
STDDEV_SAMP	applicable only to a sample, that is, a part of an entire population. The syntax for using this function is: SELECT STDDEV_SAMP(expression) FROM TABLE_NAME;	command at the command prompt: SELECT STDDEV_SAMP(GROSS_SAL) FROM EMP_SALARY; The output of the function is: 790.569415
VAR_POP	The VAR_POP function returns the standard variance of the argument. This function accepts the rows as the argument and defines the number of rows as the denominator. The syntax for using this function is: SELECT VAR_POP(expression) FROM TABLE_NAME;	For example, to obtain the variance of gross salary of all the employees, enter the following command at the command prompt: SELECT VAR_POP(GROSS_SAL) FROM EMP_SALARY; The output of this function is: 500000.00000
VAR_SAMP	The VAR_SAMP function returns the sample variance of the argument. The denominator returned by this function is the number of rows minus one. The syntax for using this function is: SELECT VAR_SAMP(expression) FROM TABLE_NAME;	For example, to obtain the variance of gross salary of all the employees, enter the following command at the command prompt: SELECT VAR_SAMP(GROSS_SAL) FROM EMP_SALARY; The output of this function is: 625000.00000

Table 12.1: Additional AGGREGATE Functions in MySQL

12.4 Using Mathematical Functions in MySQL

MySQL provides mathematical functions that process the data provided as an argument and return a numerical value. All the mathematical functions will return a `NULL` value if an error occurs.

12.4.1 ABS Function

The `ABS` function returns the absolute value of the argument. In mathematics, the absolute value of a

Session 12

Using Basic Functions in MySQL - I

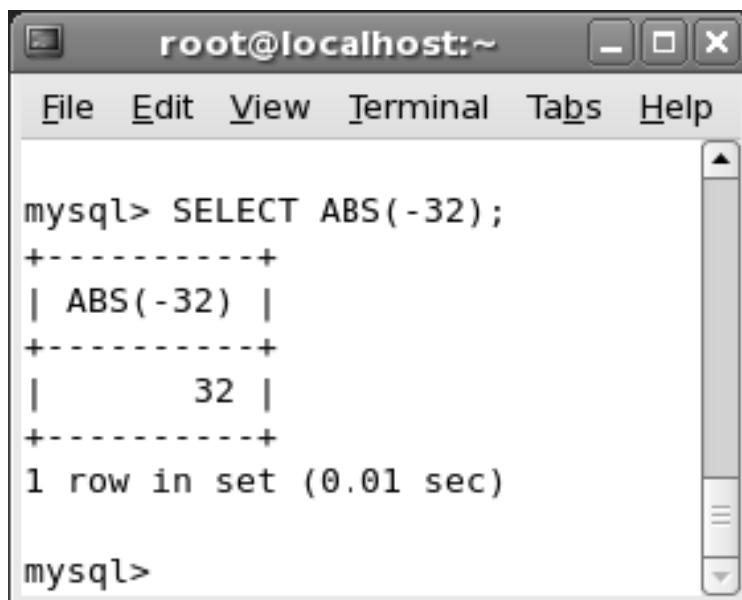
number is its distance from the origin and is therefore, never negative. The syntax for using this function is:

```
SELECT ABS(expression);
```

For example, to calculate the absolute value of -32, enter the following command at the command prompt:

```
SELECT ABS(-32);
```

Figure 12.10 displays the output of the command.



```
root@localhost:~ 
File Edit View Terminal Tabs Help 
mysql> SELECT ABS( -32 );
+-----+
| ABS(-32) |
+-----+
|      32   |
+-----+
1 row in set (0.01 sec)

mysql>
```

Figure 12.10: ABS Function

12.4.2 ACOS Function

The ACOS function returns the arc cosine of the specified argument. The arc cosine function returns the angle, expressed in radians, of the argument whose cosine is specified. The syntax to obtain the arc cosine of a function is:

```
SELECT ACOS(expression);
```

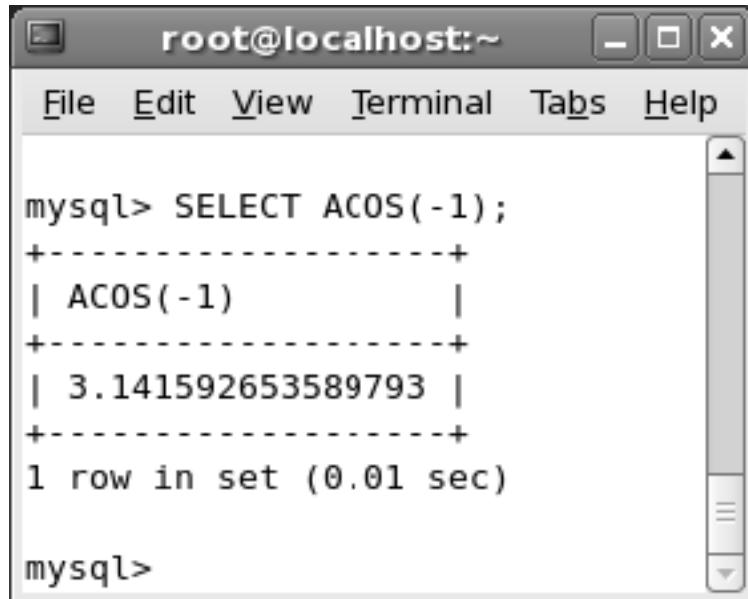
For example, to calculate the arc cosine value of 1, enter the following command at the command prompt:

Session 12

Using Basic Functions in MySQL - I

```
SELECT ACOS(-1);
```

Figure 12.11 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area contains the following MySQL command and its output:

```
mysql> SELECT ACOS(-1);
+-----+
| ACOS(-1) |
+-----+
| 3.141592653589793 |
+-----+
1 row in set (0.01 sec)

mysql>
```

Figure 12.11: ACOS Function

12.4.3 CEILING Function

The CEILING function returns the smallest integer value greater than the argument. The syntax for using this function is:

```
SELECT CEILING(X);
```

To use the CEILING function on 5.56, enter the following command at the command prompt:

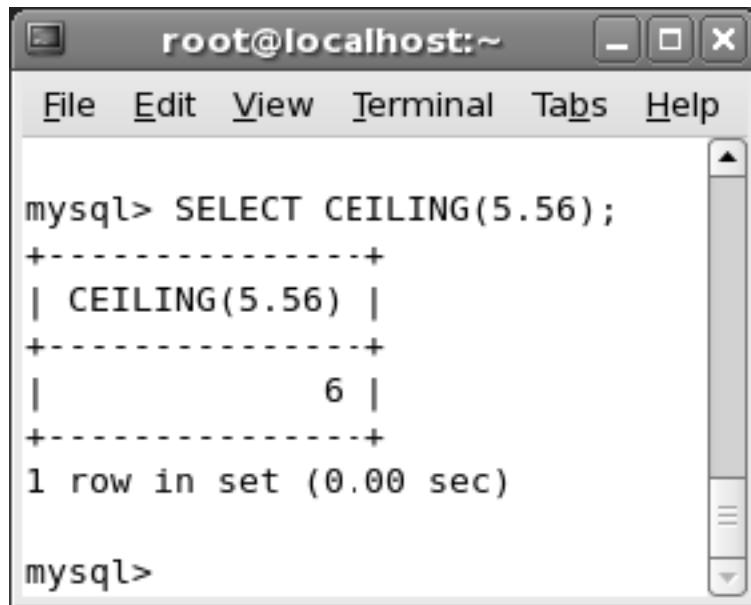
```
SELECT CEILING(5.56);
```

Figure 12.12 displays the output of the command.

Session 12

Using Basic Functions in MySQL - I

Concepts



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The main area displays the following MySQL command and its result:

```
mysql> SELECT CEILING(5.56);
+-----+
| CEILING(5.56) |
+-----+
|          6   |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 12.12: CEILING Function

12.4.4 CONV Function

The `CONV` function changes a number from one base to another. The output is either generated as a string or as `NULL` if any argument is `NULL`. The minimum and maximum base values are 2 and 36 respectively. You can perform case-insensitive comparisons with this function. The syntax to use the `CONV` function is:

```
SELECT CONV(N, from_base, to_base);
```

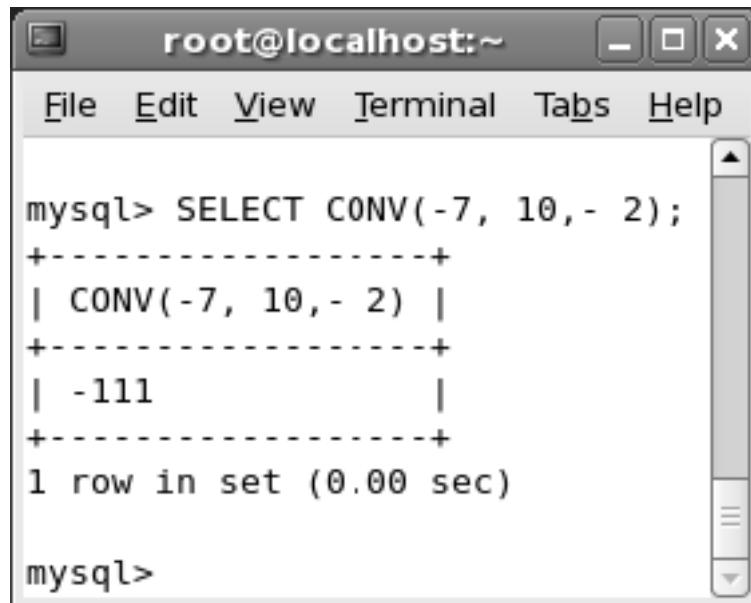
For example, to convert `-7` from base `10` to base `-2`, enter the following command at the command prompt:

```
SELECT CONV(-7, 10, -2);
```

Figure 12.13 displays the output of the command.

Session 12

Using Basic Functions in MySQL - I



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area displays the following MySQL command and its output:

```
mysql> SELECT CONV(-7, 10,- 2);
+-----+
| CONV(-7, 10,- 2) |
+-----+
| -111           |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 12.13: CONV Function

12.4.5 COS Function

The **COS** function returns the cosine of an argument. The argument specified in the function should be in radians. The syntax for obtaining the cosine of an argument X is:

```
SELECT COS(X);
```

For example, to obtain the cosine value of $\pi/6$, enter the following command at the command prompt:

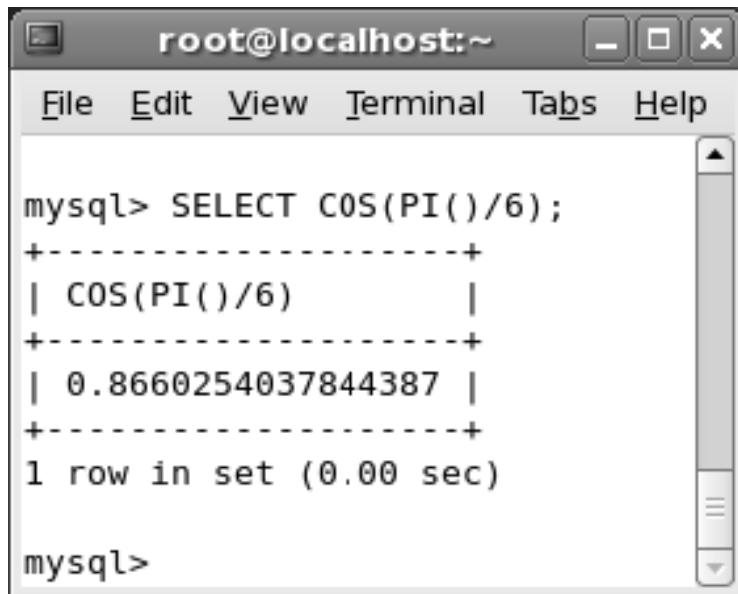
```
SELECT COS(PI()/6);
```

Figure 12.14 displays the output of the command.

Session 12

Using Basic Functions in MySQL - I

Concepts



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The main area displays the following MySQL command and its result:

```
mysql> SELECT COS(PI()/6);
+-----+
| COS(PI()/6) |
+-----+
| 0.8660254037844387 |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 12.14: COS Function

12.4.6 CRC Function

The `CRC` function calculates a cyclic redundancy check value. It returns a 32-bit unsigned value. It returns a `NULL` value if a `NULL` argument is specified. The argument must be a string. The function converts the input to a string and computes the cyclic redundancy check value. The syntax to calculate the cyclic redundancy value of an argument X is:

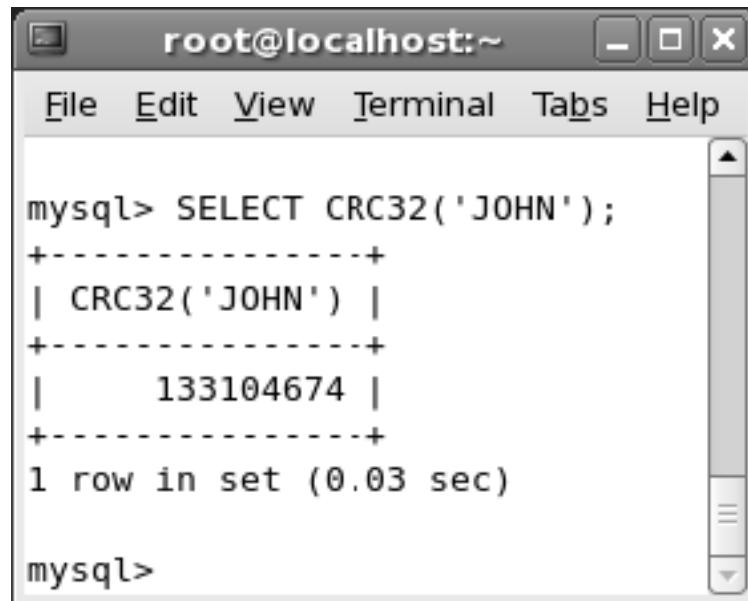
```
SELECT CRC32 (X) ;
```

For example, to calculate the cyclic redundancy check value for the string `JOHN`, enter the following command at the command prompt:

```
SELECT CRC32 ('JOHN') ;
```

Figure 12.15 displays the output of the command.

Session 12



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its output:

```
mysql> SELECT CRC32('JOHN');
+-----+
| CRC32('JOHN') |
+-----+
|      133104674 |
+-----+
1 row in set (0.03 sec)

mysql>
```

Figure 12.15: CRC Function

12.4.7 COT Function

The `COT` function returns the cotangent of the argument. Cotangent of an argument is the ratio of the adjacent side to the opposite side. The syntax to use this function is:

```
SELECT COT(expression);
```

To calculate the cotangent of 45, enter the following command at the command prompt:

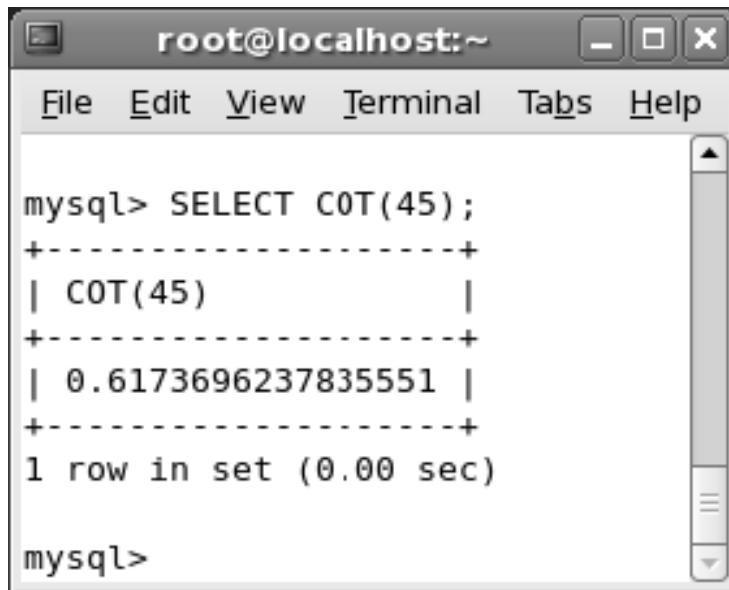
```
SELECT COT(45);
```

Figure 12.16 displays the output of the command.

Session 12

Using Basic Functions in MySQL - I

Concepts



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area displays the following MySQL command and its result:

```
mysql> SELECT COT(45);
+-----+
| COT(45)           |
+-----+
| 0.6173696237835551 |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 12.16: COT Function

12.4.8 DEGREES Function

The DEGREES function converts the specified argument from radians to degrees. The syntax to use this function is:

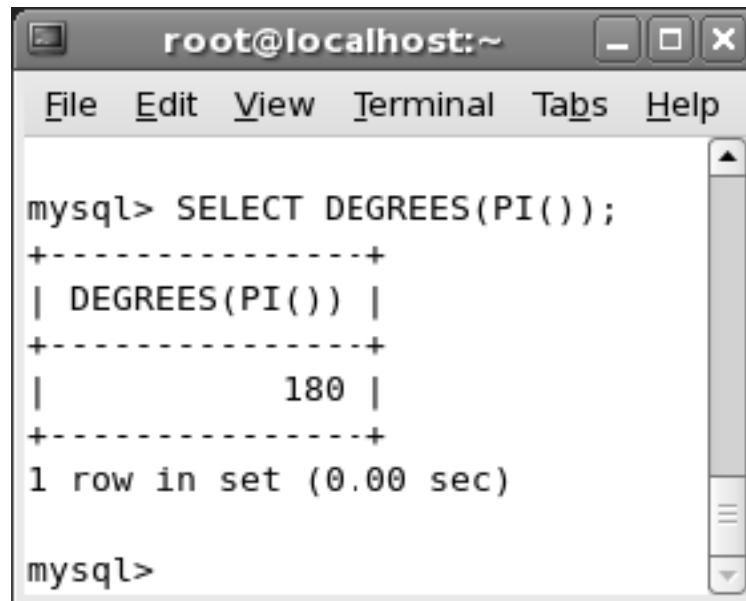
```
SELECT DEGREES(expression);
```

For example, to obtain the value of pi in degrees, enter the following command at the command prompt:

```
SELECT DEGREES(PI());
```

Figure 12.17 displays the output of the command.

Session 12



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its output:

```
mysql> SELECT DEGREES(PI());
+-----+
| DEGREES(PI()) |
+-----+
|          180 |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 12.17: DEGREES Function

12.4.9 EXP Function

The `EXP` function returns the exponential value of the given argument. The syntax for obtaining the exponential value of an argument is:

```
SELECT EXP (X) ;
```

To calculate the result of e raised to 5, enter the following command at the command prompt:

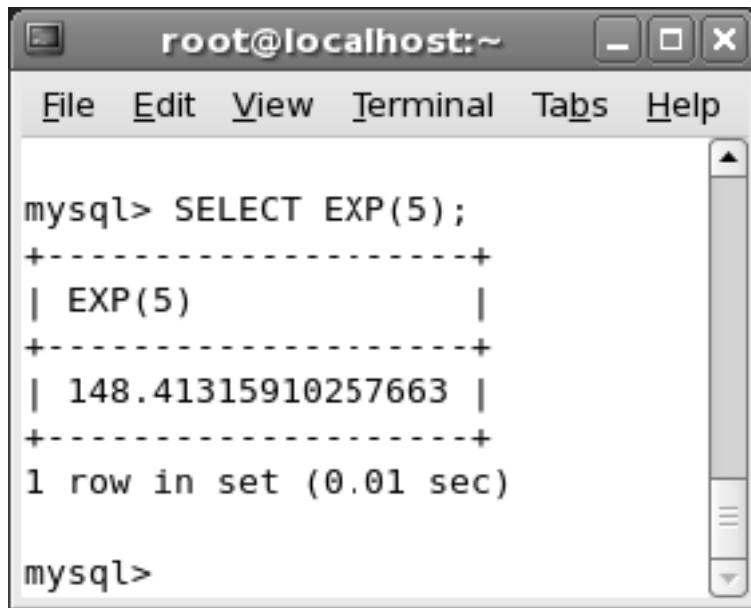
```
SELECT EXP (5) ;
```

Figure 12.18 displays the output of the command.

Session 12

Using Basic Functions in MySQL - I

Concepts



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area displays the following MySQL command and its result:

```
mysql> SELECT EXP(5);
+-----+
| EXP(5)           |
+-----+
| 148.41315910257663 |
+-----+
1 row in set (0.01 sec)

mysql>
```

Figure 12.18: EXP Function

12.4.10 FLOOR Function

The `FLOOR` function returns the largest value less than or equal to the given numeric value. The syntax for obtaining the largest value not greater than the argument specified is:

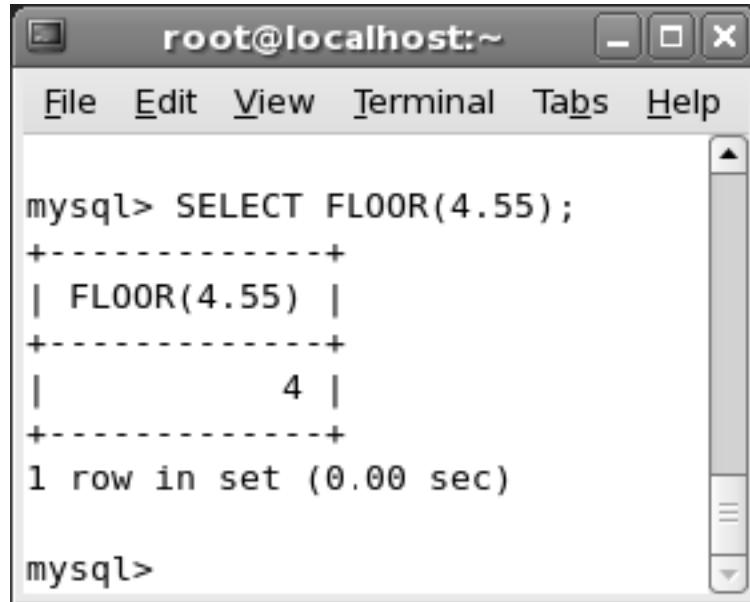
```
SELECT FLOOR(X);
```

To apply the `FLOOR` function on `4.55`, enter the following command at the command prompt:

```
SELECT FLOOR(4.55);
```

Figure 12.19 displays the output of the command.

Session 12



A screenshot of a terminal window titled "root@localhost:~". The window contains the following MySQL command and its output:

```
mysql> SELECT FLOOR(4.55);
+-----+
| FLOOR(4.55) |
+-----+
|          4   |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 12.19: FLOOR Function

12.4.11 LOG Function

The `LOG` function returns the natural logarithm of the argument. The syntax to calculate the logarithm value for a single argument is:

```
SELECT LOG (X);
```

For example, to calculate the logarithm of 100, enter the following command at the command prompt:

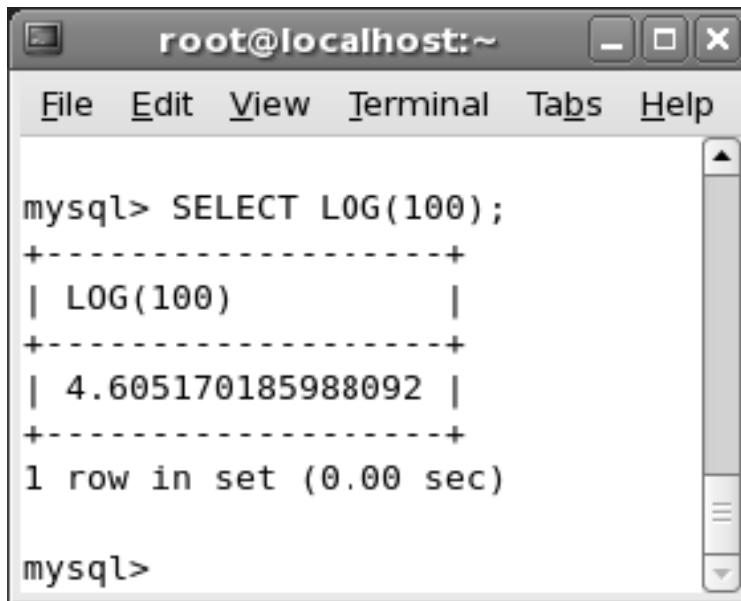
```
SELECT LOG(100);
```

Figure 12.20 displays the output of the command.

Session 12

Using Basic Functions in MySQL - I

Concepts



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area displays the following MySQL command and its result:

```
mysql> SELECT LOG(100);
+-----+
| LOG(100) |
+-----+
| 4.605170185988092 |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 12.20: LOG Function

The `LOG` function allows you to specify two values as arguments. Consider an example, where two arguments `X` and `Y` are specified in the `LOG` function. This function returns the logarithm of `X` for an arbitrary base `Y`. The syntax for specifying two arguments in `LOG` function is:

```
LOG(X, Y);
```

Note that `LOG(X, Y)` is equivalent to `Log(Y) / Log(X)`.

12.4.12 MOD Function

The `MOD` function operates on two numeric arguments. The output of the function is the remainder after dividing the number by the divisor. The syntax for obtaining the remainder is:

```
SELECT MOD(X, Y);
```

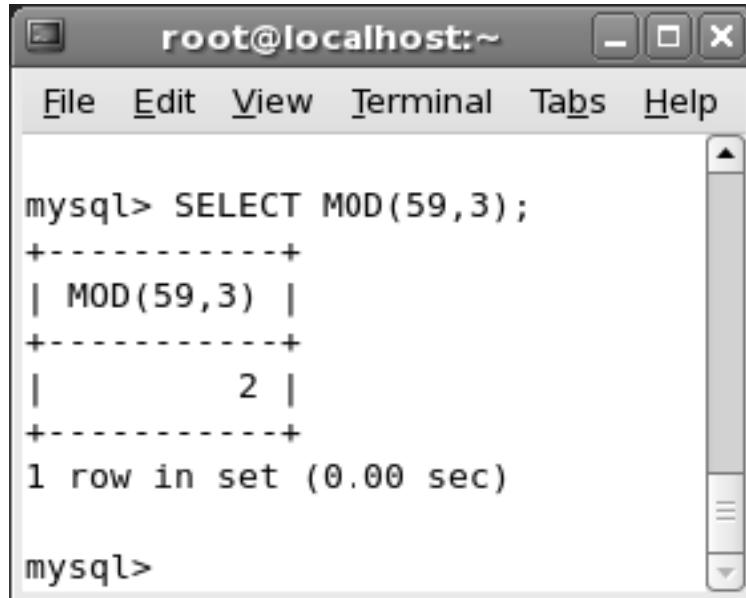
This function returns the remainder of `X` divided by `Y`.

For example, to obtain the remainder of 59 divided by 3, enter the following command at the command prompt:

```
SELECT MOD(59, 3);
```

Figure 12.21 displays the output of the command.

Session 12



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area displays the following MySQL command and its result:

```
mysql> SELECT MOD(59,3);
+-----+
| MOD(59,3) |
+-----+
|          2 |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 12.21: MOD Function

12.4.13 OCT Function

The OCT function returns the octal value of a number. Octal is a numbering system that has a base of 8 and each digit in the number is represented using only the numerals 0-7. The syntax for using this function is:

```
SELECT OCT(expression);
```

For example, to obtain the octal value of 12, enter the following command at the command prompt:

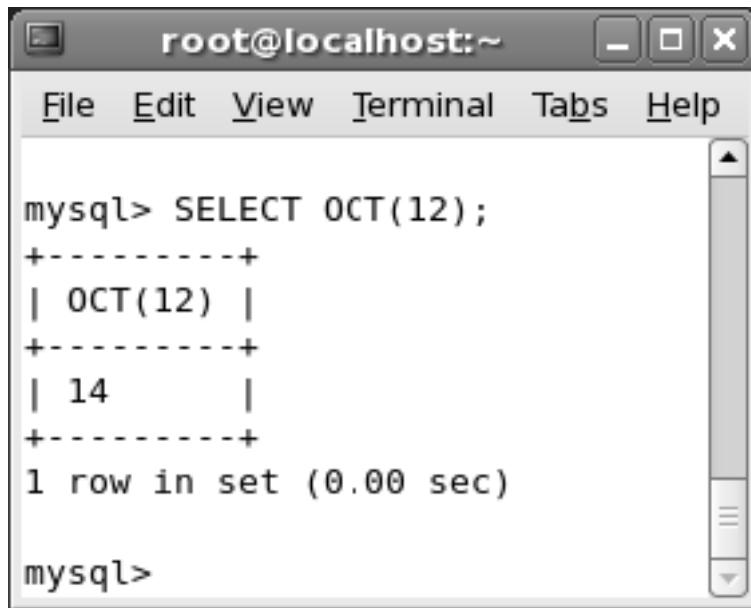
```
SELECT OCT(12);
```

Figure 12.22 displays the output of the command.

Session 12

Using Basic Functions in MySQL - I

Concepts



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area displays the following MySQL command and its result:

```
mysql> SELECT OCT(12);
+-----+
| OCT(12) |
+-----+
| 14      |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 12.22: OCT Function

12.4.14 PI Function

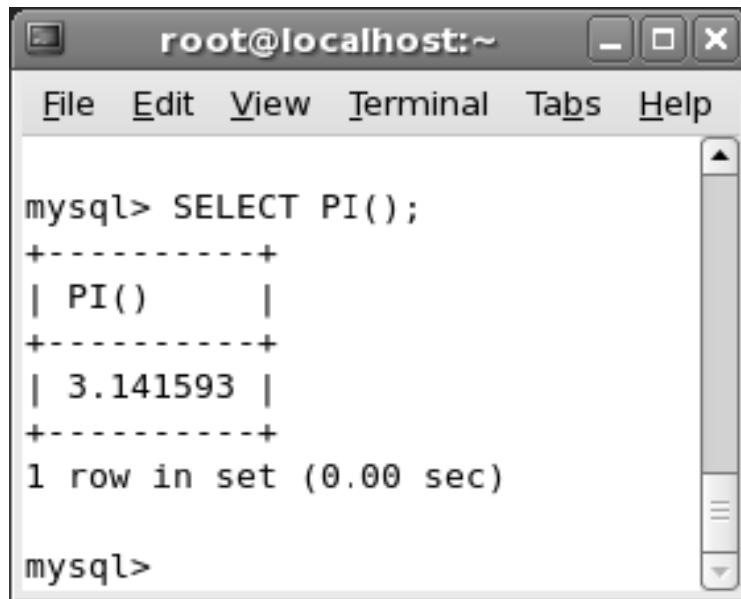
The `PI` function returns the value of the mathematical constant, `pi`. The syntax for using this function is:

```
SELECT PI();
```

To view the output of this function, enter the following command at the command prompt:

```
SELECT PI();
```

Figure 12.23 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area displays the following MySQL command and its result:

```
mysql> SELECT PI();
+-----+
| PI() |
+-----+
| 3.141593 |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 12.23: PI Function

12.4.15 POWER Function

The **POWER** function allows you to enter two arguments. This function raises the first argument to the power of the second one. The syntax for using this function is:

```
SELECT POWER(X,Y);
```

This command returns the value of X raised to the power of Y .

For example, to calculate the result of 24 raised to 6 , enter the following command at the command prompt:

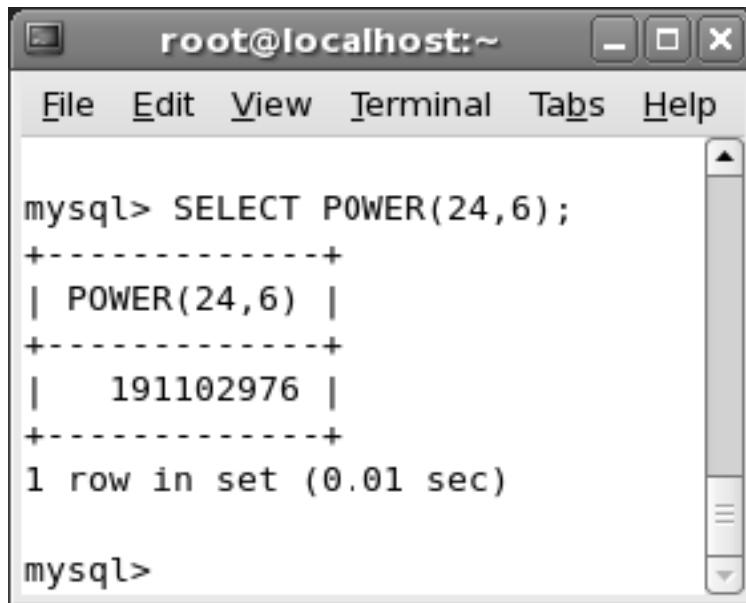
```
SELECT POWER(24,6);
```

Figure 12.24 displays the output of the command.

Session 12

Using Basic Functions in MySQL - I

Concepts



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The main area displays the following MySQL query and its result:

```
mysql> SELECT POWER(24,6);
+-----+
| POWER(24,6) |
+-----+
| 191102976 |
+-----+
1 row in set (0.01 sec)

mysql>
```

Figure 12.24: POWER Function

The alternative syntax for the `POWER` function is:

```
SELECT POW(X,Y);
```

12.4.16 RADIANS Function

The `RADIANS` function converts the specified argument from degrees to radians. The syntax to use this function is:

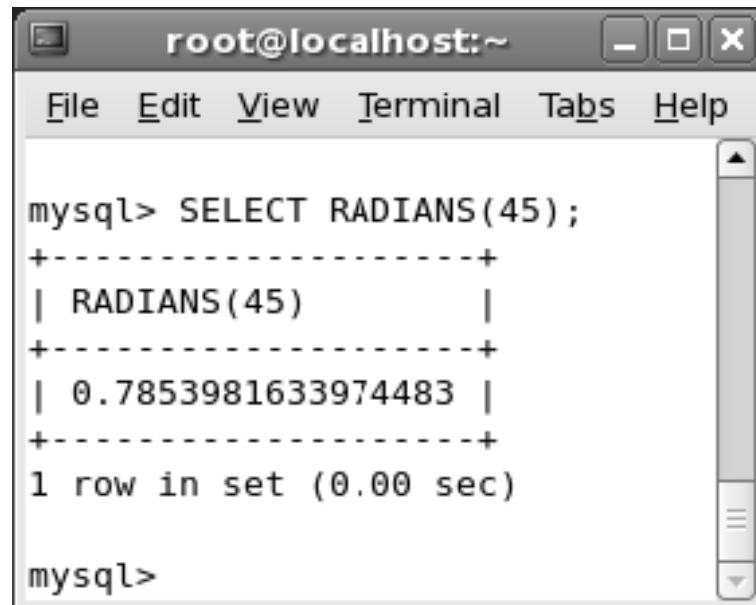
```
SELECT RADIANS(expression);
```

For example, to calculate the value of 45 in radians, enter the following command at the command prompt:

```
SELECT RADIANS(45);
```

Figure 12.25 displays the output of the command.

Session 12



A screenshot of a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area displays the following MySQL command and its result:

```
mysql> SELECT RADIANS(45);
+-----+
| RADIANS(45) |
+-----+
| 0.7853981633974483 |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 12.25: RADIANS Function

12.4.17 SIGN Function

The `SIGN` function specifies if the specified argument is negative or non-negative. It returns a `-1`, `0`, or `1` depending upon the argument, whether it is positive, zero, or negative respectively. The syntax for obtaining the sign of an argument is:

```
SELECT SIGN(X);
```

For example, to calculate the sign of `-900`, enter the following command at the command prompt:

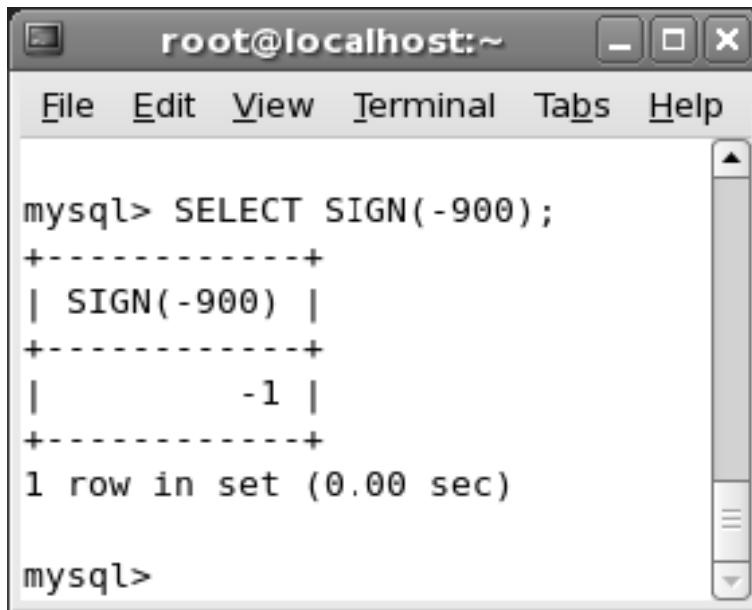
```
SELECT SIGN(-900);
```

Figure 12.26 displays the output of the command.

Session 12

Using Basic Functions in MySQL - I

Concepts



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The main area displays the following MySQL command and its output:

```
mysql> SELECT SIGN(-900);
+-----+
| SIGN(-900) |
+-----+
|          -1 |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 12.26: SIGN Function

12.4.18 SQRT Function

The `SQRT` function returns the square root of the argument. The function returns the square root as a non-negative integer. The syntax to calculate the square root of an argument is:

```
SELECT SQRT(X);
```

To calculate the square root of 2500, enter the following command at the command prompt:

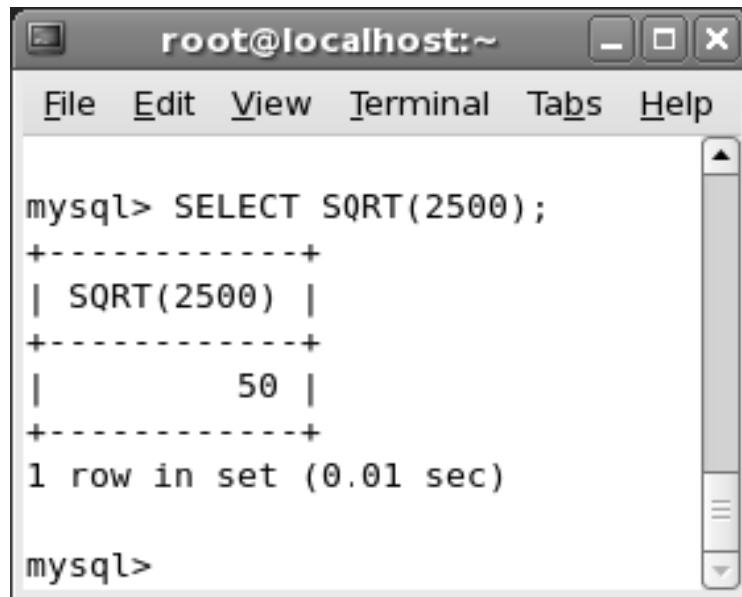
```
SELECT SQRT(2500);
```

Figure 12.27 displays the output of the command.

Session 12

Using Basic Functions in MySQL - I

Concepts



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area displays the following MySQL command and its result:

```
mysql> SELECT SQRT(2500);
+-----+
| SQRT(2500) |
+-----+
|      50   |
+-----+
1 row in set (0.01 sec)

mysql>
```

Figure 12.27: SQRT Function

12.4.19 TRUNCATE Function

The TRUNCATE function trims the given argument to the specified number of decimal places. The syntax to use this function is:

```
SELECT TRUNCATE(x, d);
```

where,

x – specifies the value for the argument

d – specifies the number of decimal places

If d = 0, then the number has no decimal part. If d is negative, then d digits to the left of the decimal point are made zero.

To truncate 2.4567 to two decimal places, enter the following command at the command prompt:

```
SELECT TRUNCATE(2.4567, 2);
```

Figure 12.28 displays the output of the command.

Session 12

Using Basic Functions in MySQL - I

Concepts

The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area displays the following MySQL command and its output:

```
mysql> SELECT TRUNCATE(2.4567,2);
+-----+
| TRUNCATE(2.4567,2) |
+-----+
|          2.45 |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 12.28: TRUNCATE Function

Table 12.2 describes additional Mathematical functions supported by MySQL:

Name	Description	Example
ASIN	The ASIN function returns the arc sine of the specified argument. The arc sine function returns the angle, expressed in radians, of the argument whose sine is specified. The syntax to obtain the arc sine of a function is: SELECT ASIN(expression);	For example, to obtain the arcsine value of 1, enter the following command at the command prompt: SELECT ASIN(-1); The output of this function is: -1.5707963267948966
LOG10	The LOG10 function returns the base-10 logarithm of the argument. The syntax to use this function is: SELECT LOG10(expression);	For example, to obtain the logarithm of 100 to the base 10, enter the following command at the command prompt: SELECT LOG10(100); The output of this function is: 2

Session 12

Name	Description	Example
LOG2	The LOG2 function returns the base-2 logarithm of the argument. The syntax to use this function is: SELECT LOG2(expression);	For example, to obtain the logarithm of 16 to the base 2, enter the following command at the command prompt: SELECT LOG2(16); The output of this function is: 4
SIN	The SIN function returns the sine value of the argument. You should specify the argument in radians. The syntax to use this function is: SELECT SIN(expression);	For example, to calculate the sin value of pi/2, enter the following command at the command prompt: SELECT SIN(PI()/2); The output of this function is: 1
TAN	The TAN function returns the tangent value of the argument passed. You must specify the argument in radians. The syntax to use this function is: SELECT TAN(expression);	For example, to obtain the tan value of pi/4, enter the following command at the command prompt: SELECT TAN(PI()/4); The output of this function is: 0.9999999999999999

Table 12.2: Additional Mathematical Functions in MySQL



Summary

- In SQL queries, functions can be used in place of the column name and in the WHERE clause.
- The GROUP BY clause can be used with functions to group the data on a specific constraint.
- GROUP BY functions are used to obtain the average value of the argument, count the number of values in a given column, calculate the variance and standard deviation of a given set of values, and calculate the maximum or minimum value from a given set of numbers.
- Examples of functions that can be used with the GROUP BY clause are AVG, COUNT, COUNT(DISTINCT), and SUM.
- Mathematical functions are used to operate on numbers. They obtain the trigonometric values such as SIN, COS, TAN, ATAN, and ACOS. In addition, you can use these functions to obtain the square root, logarithmic value, exponential value of a number, and the modulus value.
- Mathematical functions return a NULL value if an error occurs while processing. Some examples of mathematical functions are ABS, ACOS, ASIN, EXP, LOG, and SQRT.



Check Your Progress

1. The output of `CEIL(7.86)` would be _____.
 - a. 7
 - b. 8
 - c. 0.86
 - d. 0.06

2. Which function returns the number of distinct non-NULL values in a given column?
 - a. COUNT
 - b. COUNT(DISTINCT)
 - c. MAX
 - d. LOCATE

3. The output of `FLOOR(9.85)` would be _____.
 - a. 9
 - b. 10
 - c. 8
 - d. 0.85

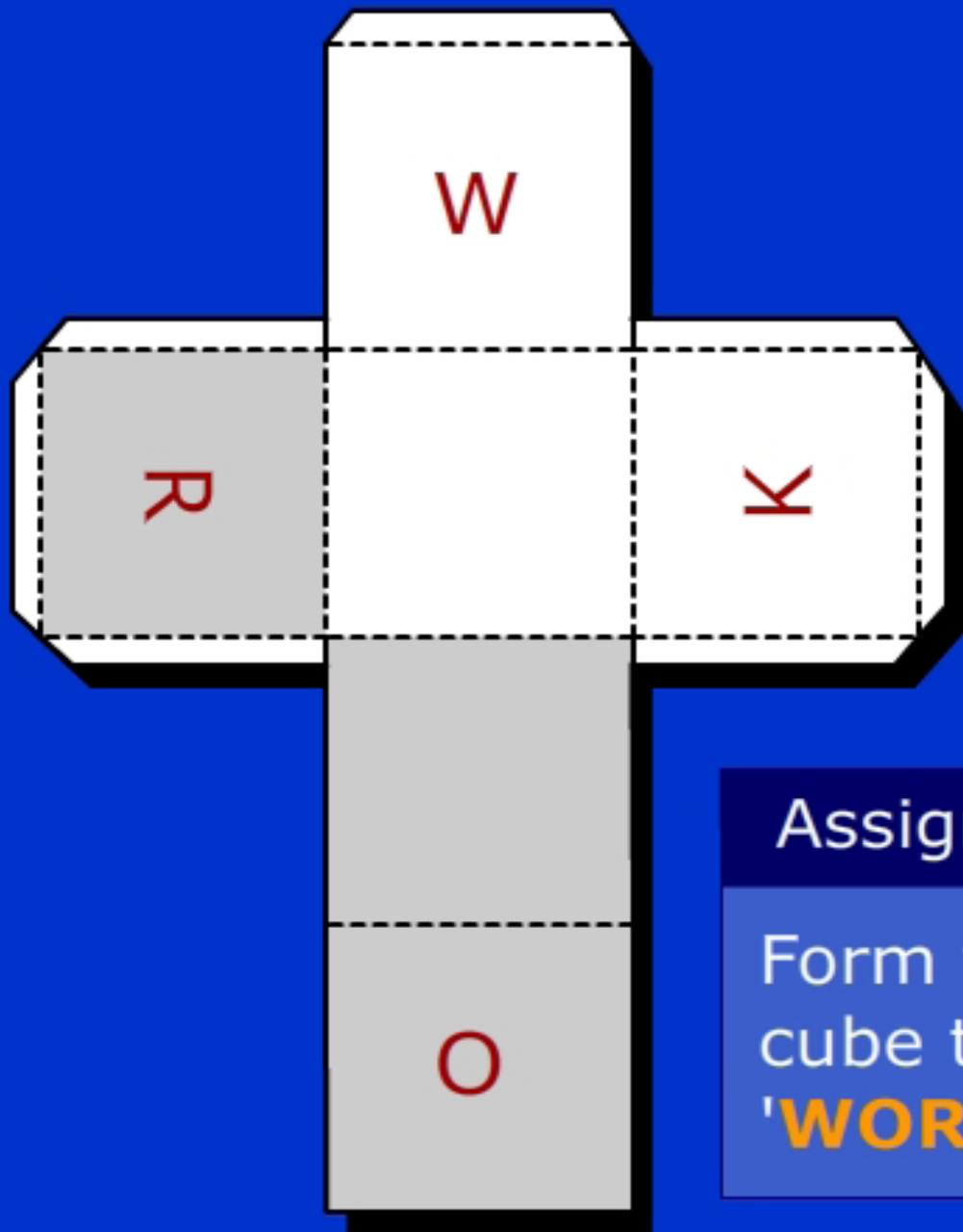
4. Which function returns the remainder of the first argument divided by the second?
 - a. LOG
 - b. EXP



Check Your Progress

Concepts

- c. POWER
 - d. MOD
5. Which function returns the non-negative square root of the value specified in the argument?
- a. SIN
 - b. CRC
 - c. PI
 - d. SQRT



Assignment

Form the
cube to read
'WORK'.

"Practice does not make perfect. Only perfect practice makes perfect."
- Vince Lombardi

For perfection, solve assignments @

www.onlinevarsity.com

Objectives

At the end of this session, the student will be able to:

- *Use the Aggregate functions.*
- *Use the Mathematical functions.*

The steps given in the session are detailed, comprehensive, and carefully thought through in order to meet the learning objectives and understand the tool completely. Please follow the steps carefully.

Part I - For the first 1.5 hours:

Using Aggregate Functions in MySQL

Aggregate functions operate on a group of values and return a single value as the final result. Aggregate functions can be used with the GROUP BY and HAVING clause. The GROUP BY clause is used to group rows having similar values for a specific column into a single row and the HAVING clause specifies the condition to be applied on the selected columns. You can use the GROUP BY and HAVING clause with the SELECT statement to retrieve data that satisfies conditions defined in the HAVING clause. The following steps illustrate the use of GROUP BY and HAVING clause:

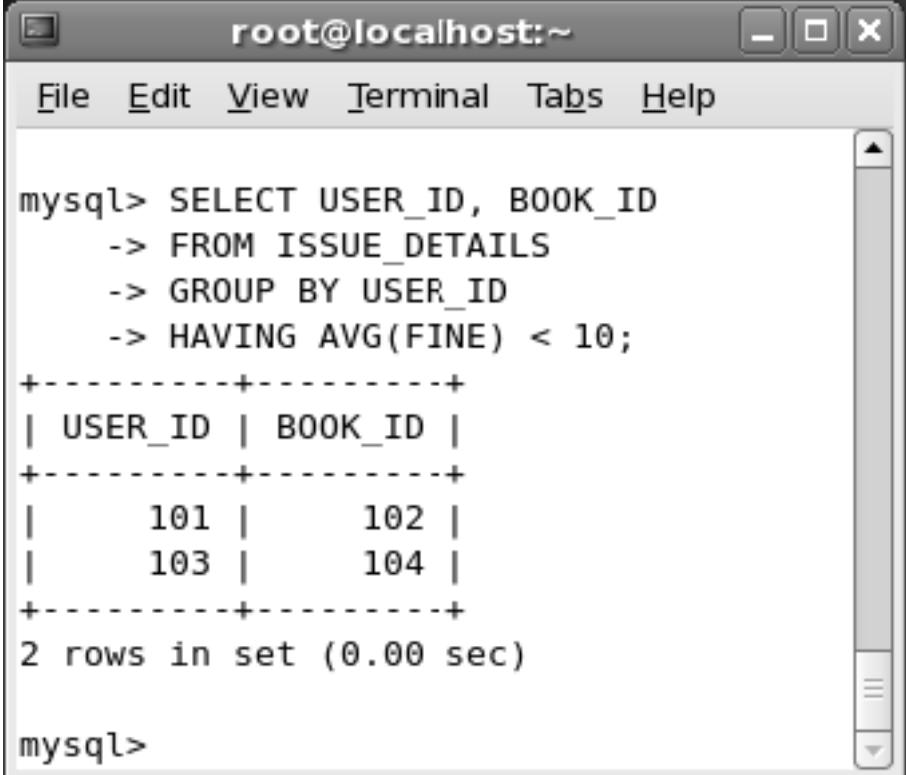
1. Activate the LIBRARY database.
2. The librarian wants to view the USER_ID and the BOOK_ID to get details about the fines imposed on the members. However, the average fine amount must not be greater than 10. To display the USER_ID and BOOK_ID columns of the ISSUE_DETAILS table having average fine less than 10, enter the following command at the command prompt:

```
SELECT USER_ID, BOOK_ID FROM ISSUE_DETAILS  
GROUP BY USER_ID  
HAVING AVG(FINE)<10;
```

Figure 13.1 displays the output of the command.

Session 13

Using Basic Functions in MySQL - I (Lab)



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area displays the following MySQL query and its results:

```
mysql> SELECT USER_ID, BOOK_ID
-> FROM ISSUE_DETAILS
-> GROUP BY USER_ID
-> HAVING AVG(FINE) < 10;
+-----+-----+
| USER_ID | BOOK_ID |
+-----+-----+
|      101 |      102 |
|      103 |      104 |
+-----+-----+
2 rows in set (0.00 sec)

mysql>
```

The output shows two rows of data: one row where USER_ID is 101 and BOOK_ID is 102, and another row where USER_ID is 103 and BOOK_ID is 104.

Figure 13.1: Displays Output of AVG Function

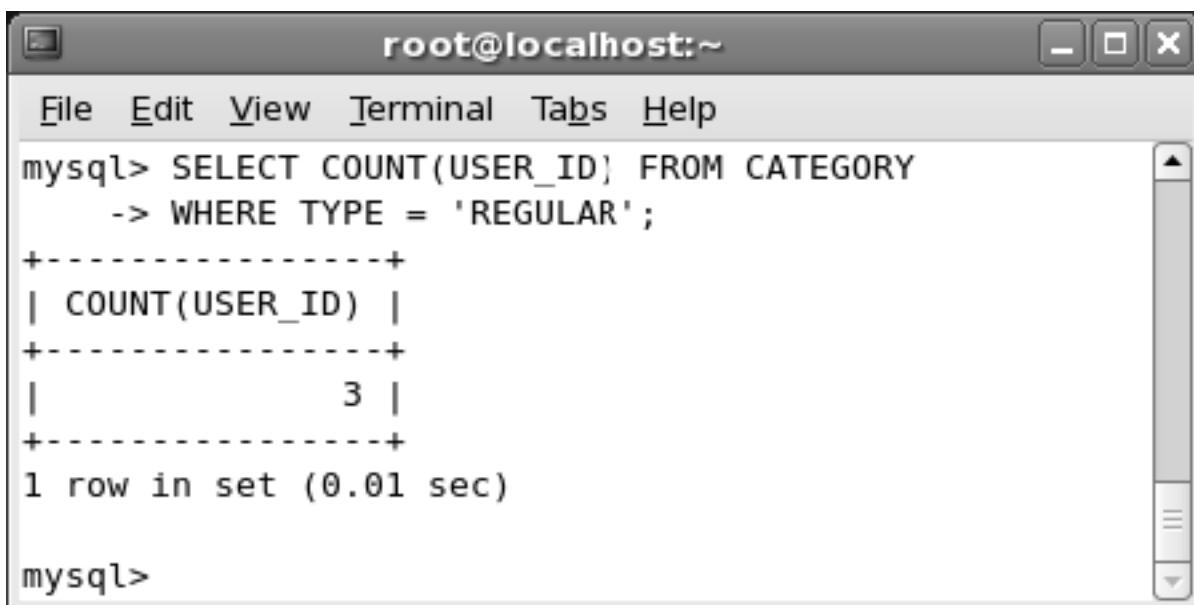
3. The librarian wants to view the total number of REGULAR members. To calculate the number of users whose membership type is REGULAR, enter the following command at the command prompt:

```
SELECT COUNT(USER_ID) FROM CATEGORY
WHERE TYPE = 'REGULAR';
```

Figure 13.2 displays the output of the command.

Session 13

Using Basic Functions in MySQL - I (Lab)



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its output:

```
mysql> SELECT COUNT(USER_ID) FROM CATEGORY
   -> WHERE TYPE = 'REGULAR';
+-----+
| COUNT(USER_ID) |
+-----+
|            3 |
+-----+
1 row in set (0.01 sec)

mysql>
```

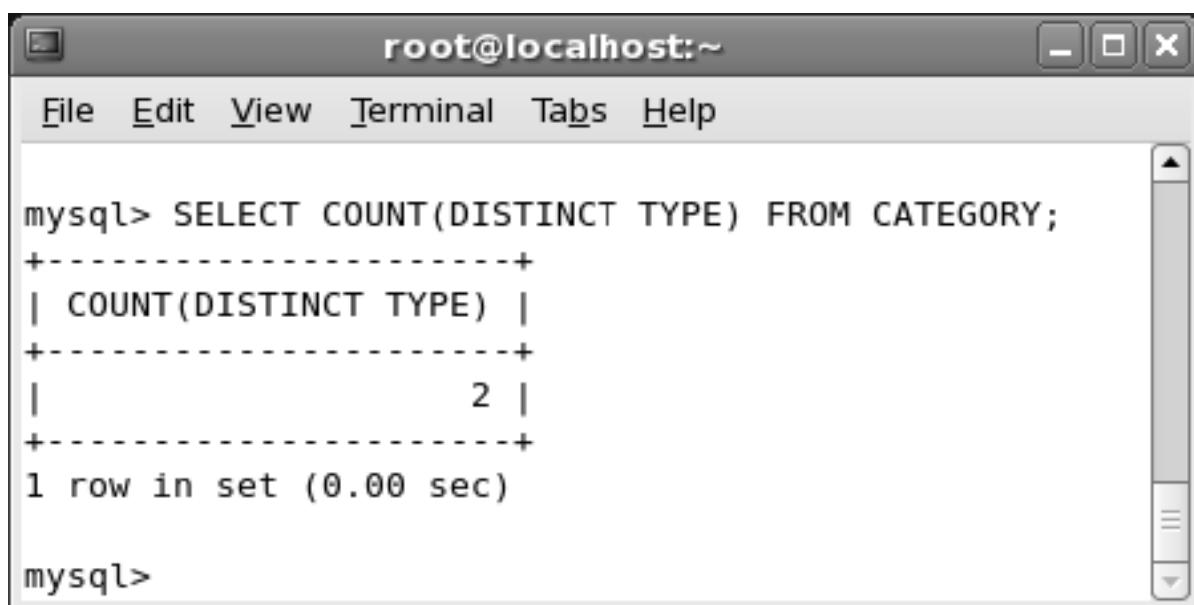
Lab Guide

Figure 13.2: Displays Output of COUNT Function

4. The librarian wants to view the total number of unique memberships available. To calculate the total number of unique membership type, enter the following command at the command prompt:

```
SELECT COUNT(DISTINCT TYPE) FROM CATEGORY;
```

Figure 13.3 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its output:

```
mysql> SELECT COUNT(DISTINCT TYPE) FROM CATEGORY;
+-----+
| COUNT(DISTINCT TYPE) |
+-----+
|            2 |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 13.3: Displays Output of COUNT(DISTINCT TYPE) Function

Session 13

Using Basic Functions in MySQL - I (Lab)

5. To display the concatenated values of name and user ID of the users belonging to the location, 'MEXICO', enter the following command at the command prompt:

```
SELECT GROUP_CONCAT(USER_ID), GROUP_CONCAT(USER_NAME), LOCATION  
FROM USER_DETAILS WHERE LOCATION = 'MEXICO';
```

Figure 13.4 displays the output of the command.

The screenshot shows a terminal window titled "root@localhost:~". The window has a standard Linux-style interface with a menu bar (File, Edit, View, Terminal, Tabs, Help) and a scroll bar on the right. The main area contains a MySQL command-line session. The command entered is:

```
mysql> SELECT GROUP_CONCAT(USER_ID),  
-> GROUP_CONCAT(NAME),  
-> LOCATION  
-> FROM USER_DETAILS  
-> WHERE LOCATION = 'MEXICO';
```

The output shows the results of the query:

GROUP_CONCAT(USER_ID)	GROUP_CONCAT(NAME)	LOCATION
100,101,102,103	TOBY,BINCY,DENVER,HELEN	MEXICO

Below the table, the message "1 row in set (0.01 sec)" is displayed. The MySQL prompt "mysql>" appears again at the bottom of the window.

Figure 13.4: GROUP_CONCAT Function

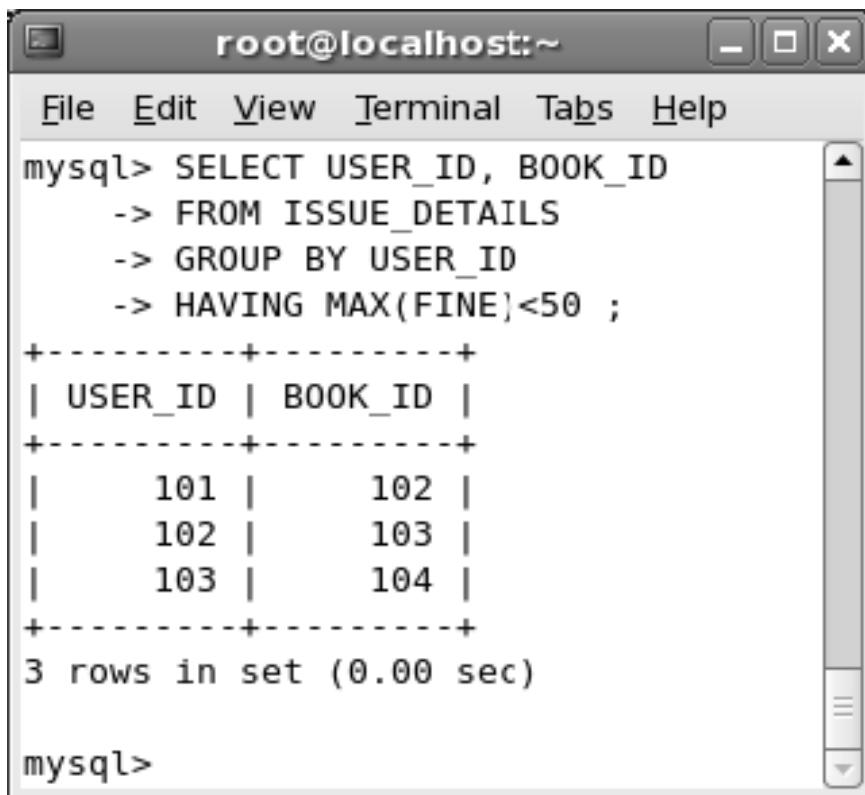
6. The librarian wants to view the users who have been imposed a fine less than 50 and also have been issued a book. To display the users, enter the following command at the command prompt:

```
SELECT USER_ID, BOOK_ID  
FROM ISSUE_DETAILS  
GROUP BY USER_ID  
HAVING MAX(FINE)<50;
```

Figure 13.5 displays the output of the command.

Session 13

Using Basic Functions in MySQL - I (Lab)



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL query and its results:

```
mysql> SELECT USER_ID, BOOK_ID
   -> FROM ISSUE_DETAILS
   -> GROUP BY USER_ID
   -> HAVING MAX(FINE)<50 ;
+-----+-----+
| USER_ID | BOOK_ID |
+-----+-----+
|      101 |      102 |
|      102 |      103 |
|      103 |      104 |
+-----+-----+
3 rows in set (0.00 sec)

mysql>
```

The results show three rows of data with columns labeled "USER_ID" and "BOOK_ID". The data is as follows:

USER_ID	BOOK_ID
101	102
102	103
103	104

Lab Guide

Figure 13.5: MAX Function

For steps 8 to 10, you will be working with the EMPLOYEE database that consists of the EMP_DETAILS, EMP_DEPARTMENT, and EMP_SALARY tables. You must activate the EMPLOYEE database before executing steps from 8 to 10.

7. To activate the EMPLOYEE database, enter the following command at the command prompt:

```
USE EMPLOYEE;
```

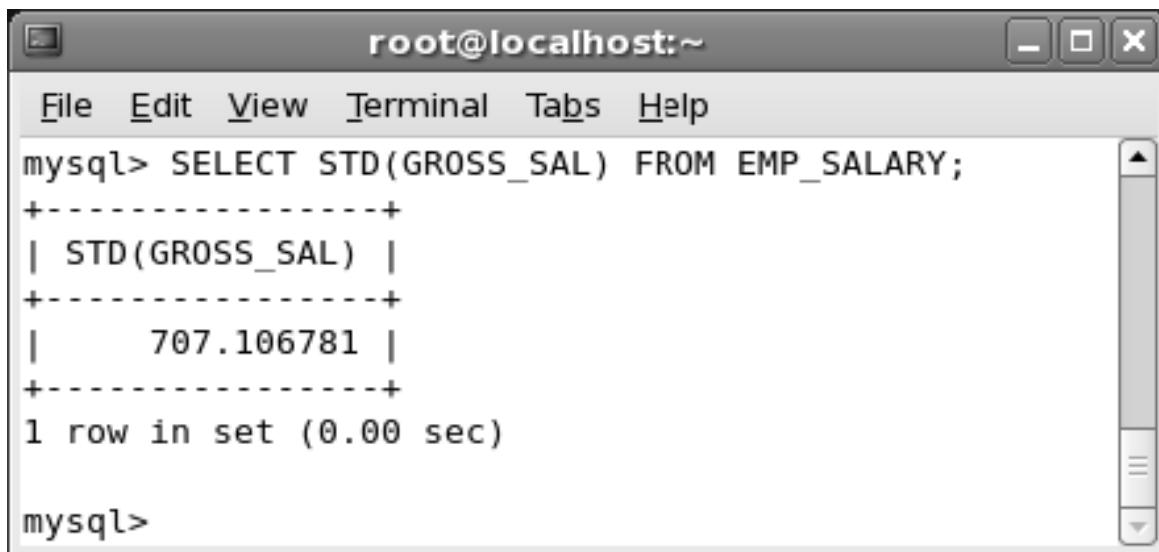
8. To calculate the standard deviation of the gross salary of all the employees, enter the following command at the command prompt:

```
SELECT STD(GROSS_SAL) FROM EMP_SALARY;
```

Figure 13.6 displays the output of the command.

Session 13

Using Basic Functions in MySQL - I (Lab)



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area displays a MySQL command and its result:

```
mysql> SELECT STD(GROSS_SAL) FROM EMP_SALARY;
+-----+
| STD(GROSS_SAL) |
+-----+
|      707.106781 |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 13.6: STD Function

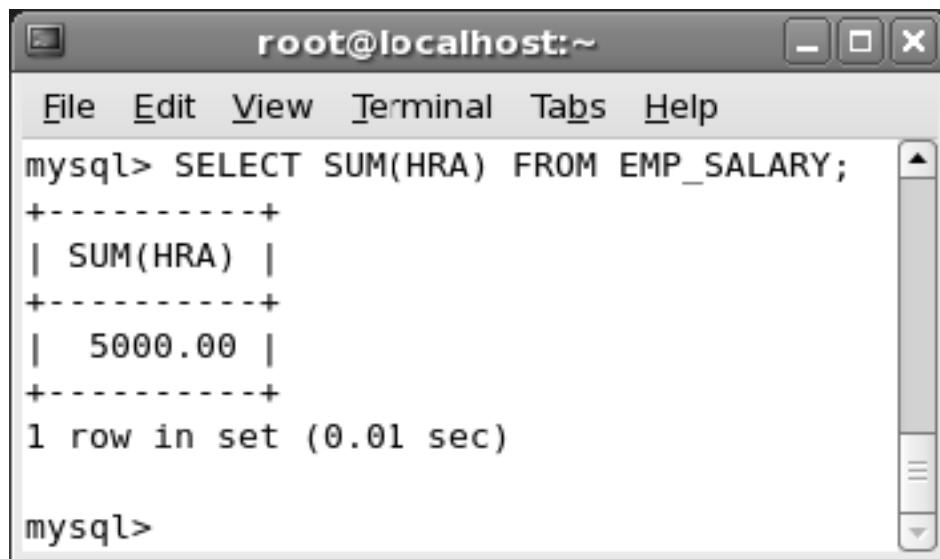
9. To calculate the sum of HRA of all the employees belonging to the same department, enter the following command at the command prompt:

```
SELECT SUM(HRA), D_NAME  
FROM EMP_SALARY  
GROUP BY D_NAME;
```

Figure 13.7 displays the output of the command.

Session 13

Using Basic Functions in MySQL - I (Lab)



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". Below the menu is a command-line interface. The user has run the following SQL command:

```
mysql> SELECT SUM(HRA) FROM EMP_SALARY;
```

The output is:

SUM(HRA)
5000.00

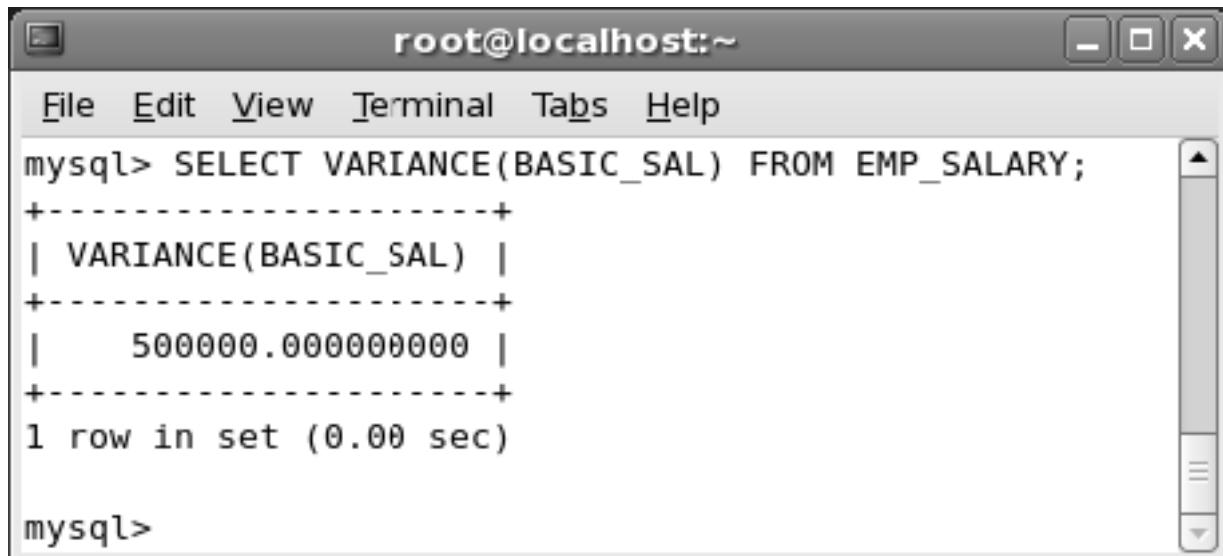
Below the table, it says "1 row in set (0.01 sec)". At the bottom of the window, the prompt "mysql>" is visible.

Figure 13.7: SUM Function

10. To calculate the variance of basic salary for all the employees, enter the following command at the command prompt:

```
SELECT VARIANCE(BASIC_SAL) FROM EMP_SALARY;
```

Figure 13.8 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". Below the menu is a command-line interface. The user has run the following SQL command:

```
mysql> SELECT VARIANCE(BASIC_SAL) FROM EMP_SALARY;
```

The output is:

VARIANCE(BASIC_SAL)
500000.0000000000

Below the table, it says "1 row in set (0.00 sec)". At the bottom of the window, the prompt "mysql>" is visible.

Figure 13.8: VARIANCE Function

Session 13

Using Basic Functions in MySQL - I (Lab)

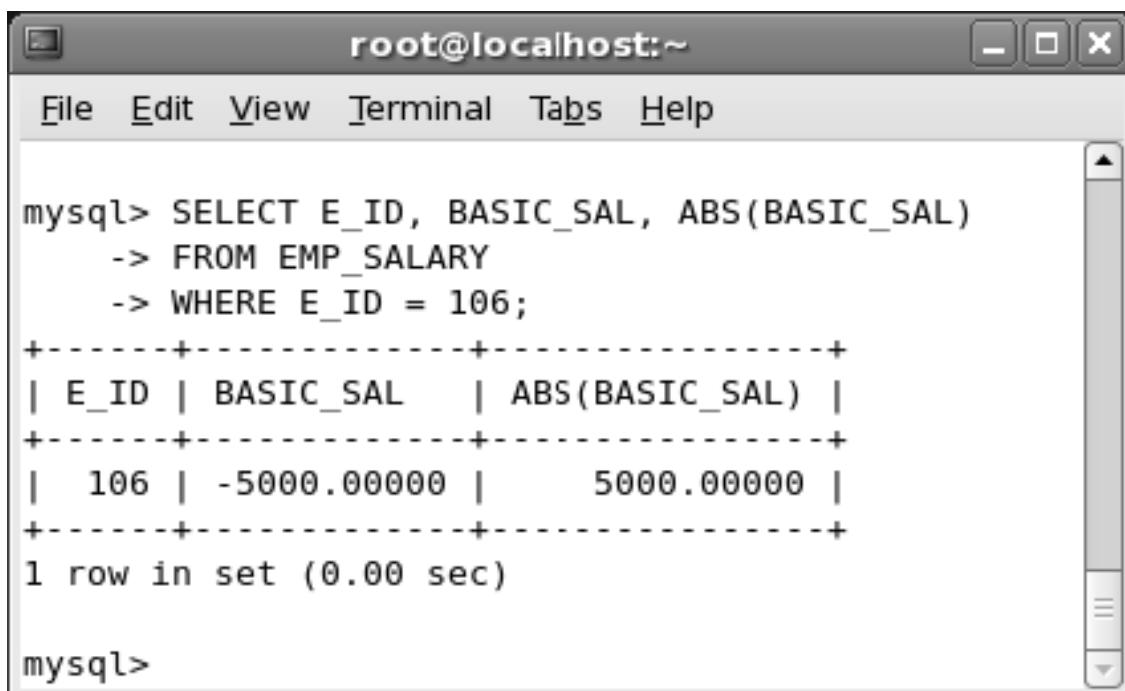
Using Mathematical Functions in MySQL

Mathematical functions operate on numerical data. The following steps illustrate the use of mathematical functions in MySQL.

1. The basic salary of an employee is in negative. This negative salary has to be converted to a positive value in order to compute results. To calculate the absolute value of the basic salary where the value is in negative, enter the following command at the command prompt:

```
SELECT E_ID, BASIC_SAL, ABS(BASIC_SAL)  
  
FROM EMP_SALARY  
  
WHERE E_ID = 106;
```

Figure 13.9 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window contains a MySQL command-line session. The user has entered the following SQL query:

```
mysql> SELECT E_ID, BASIC_SAL, ABS(BASIC_SAL)
-> FROM EMP_SALARY
-> WHERE E_ID = 106;
```

The query returns one row of data:

E_ID	BASIC_SAL	ABS(BASIC_SAL)
106	-5000.00000	5000.00000

Below the table, the message "1 row in set (0.00 sec)" is displayed. The MySQL prompt "mysql>" appears at the bottom of the window.

Figure 13.9: ABS Function

2. To calculate the arc cosine value of 0, enter the following command at the command prompt:

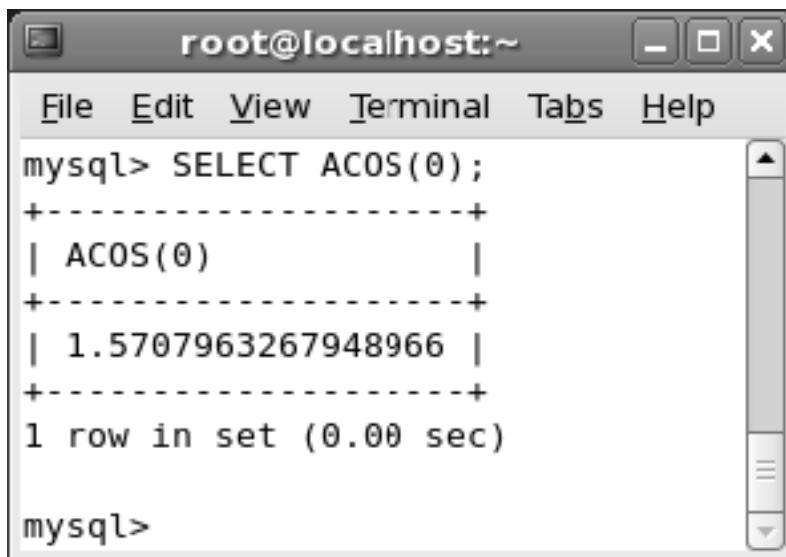
```
SELECT ACOS(0);
```

Figure 13.10 displays the output of the command.

Session 13

Using Basic Functions in MySQL - I (Lab)

Lab Guide



The screenshot shows a terminal window titled "root@localhost:~". The MySQL prompt "mysql>" is at the bottom. The command "SELECT ACOS(0);" is entered, followed by its execution result:

ACOS(0)
1.5707963267948966

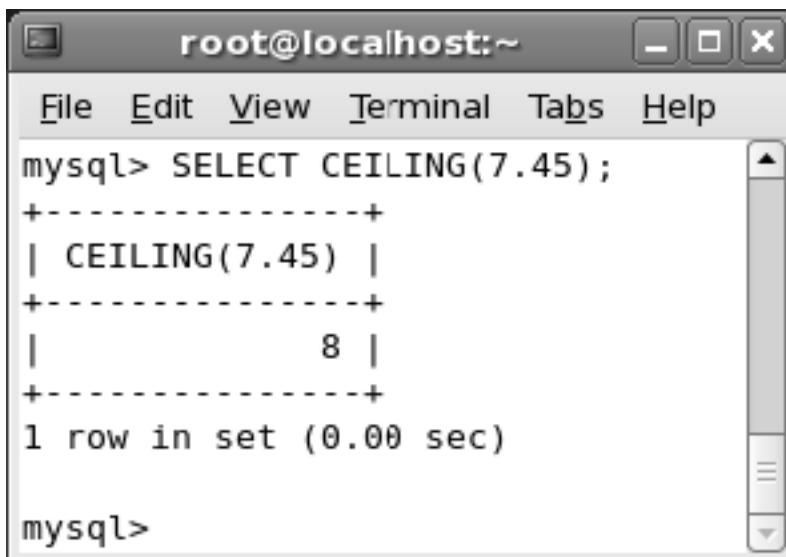
Below the table, the message "1 row in set (0.00 sec)" is displayed.

Figure 13.10: ACOS Function

3. To use the CEILING function on 7.45, enter the following command at the command prompt:

```
SELECT CEILING(7.45);
```

Figure 13.11 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The MySQL prompt "mysql>" is at the bottom. The command "SELECT CEILING(7.45);" is entered, followed by its execution result:

CEILING(7.45)
8

Below the table, the message "1 row in set (0.00 sec)" is displayed.

Figure 13.11: CEILING Function

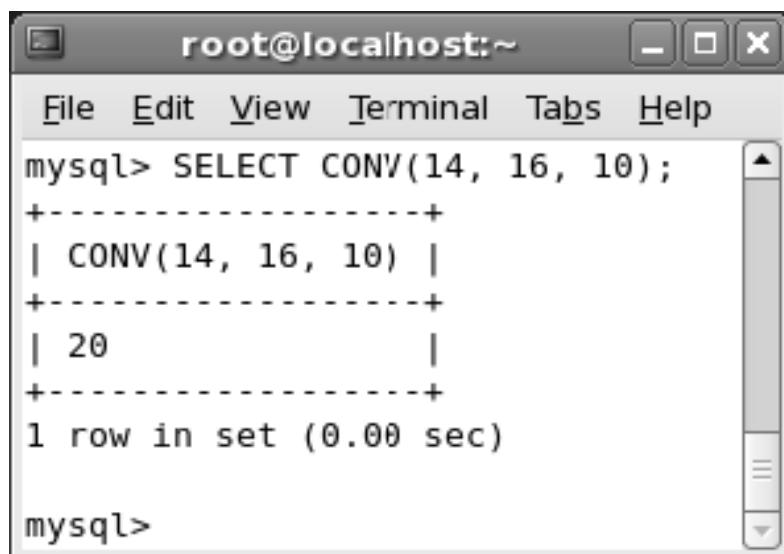
Session 13

Using Basic Functions in MySQL - I (Lab)

4. To convert 14 from base 16 to base 10, enter the following command at the command prompt:

```
SELECT CONV(14, 16, 10);
```

Figure 13.12 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following text:

```
File Edit View Terminal Tabs Help
mysql> SELECT CONV(14, 16, 10);
+-----+
| CONV(14, 16, 10) |
+-----+
| 20               |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 13.12: CONVERT Function

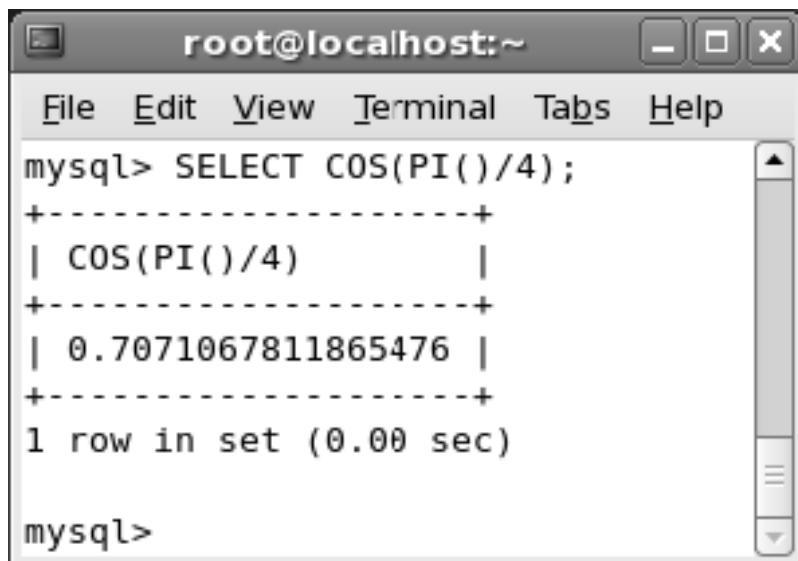
5. To calculate the cosine value of pi()/4, enter the following command at the command prompt:

```
SELECT COS(PI()/4);
```

Figure 13.13 displays the output of the command.

Session 13

Using Basic Functions in MySQL - I (Lab)



```
root@localhost:~ File Edit View Terminal Tabs Help mysql> SELECT COS(PI()/4); +-----+ | COS(PI()/4) | +-----+ | 0.7071067811865476 | +-----+ 1 row in set (0.00 sec)

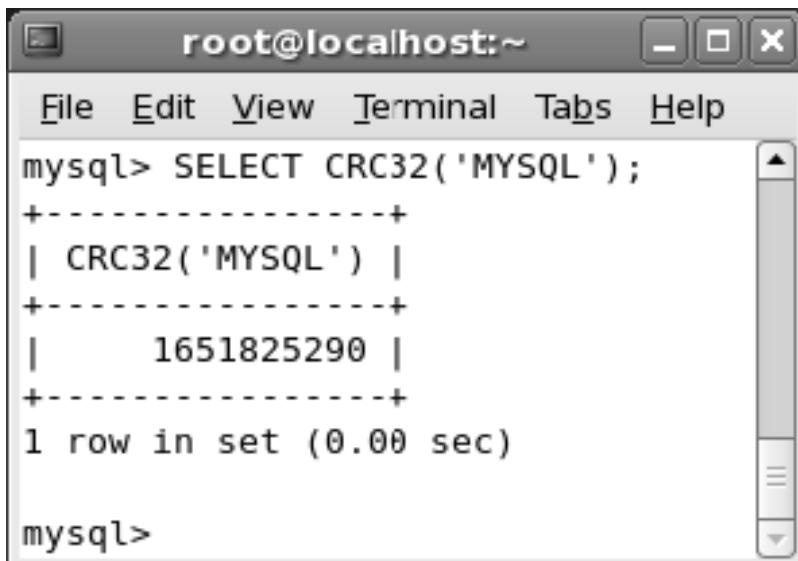
mysql>
```

Figure 13.13: COS Function

6. To calculate the cyclic redundancy check value of the string MYSQL, enter the following command at the command prompt:

```
SELECT CRC32('MYSQL');
```

Figure 13.14 displays the output of the command.



```
root@localhost:~ File Edit View Terminal Tabs Help mysql> SELECT CRC32('MYSQL'); +-----+ | CRC32('MYSQL') | +-----+ | 1651825290 | +-----+ 1 row in set (0.00 sec)

mysql>
```

Figure 13.14: CRC32 Function

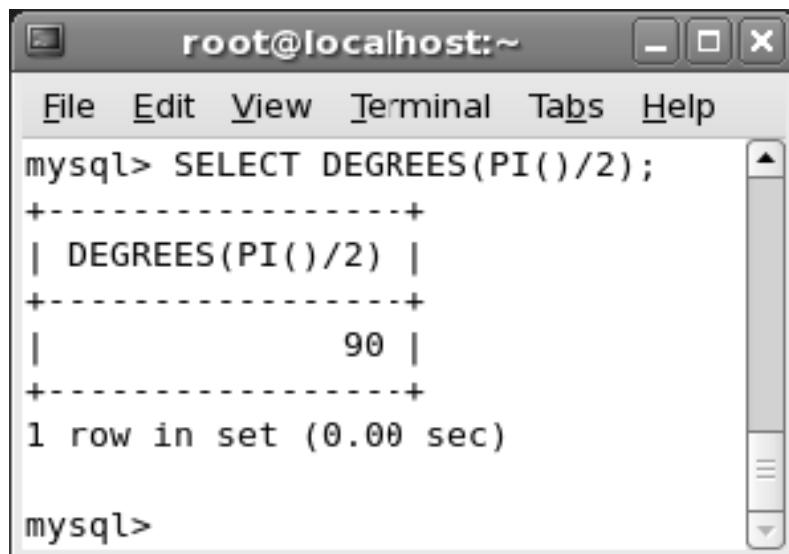
Session 13

Using Basic Functions in MySQL - I (Lab)

7. To calculate the value of $\pi()/2$ in degrees, enter the following command at the command prompt:

```
SELECT DEGREES(PI()/2);
```

Figure 13.15 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following text:

```
File Edit View Terminal Tabs Help
mysql> SELECT DEGREES(PI()/2);
+-----+
| DEGREES(PI()/2) |
+-----+
|          90 |
+-----+
1 row in set (0.00 sec)

mysql>
```

The terminal window has a standard window title bar with minimize, maximize, and close buttons. The main area shows the MySQL command-line interface with the prompt "mysql>". The command "SELECT DEGREES(PI()/2);" is entered, followed by its execution result, which is a single row containing the value 90. The output is formatted with column headers and row separators.

Figure 13.15: DEGREES Function

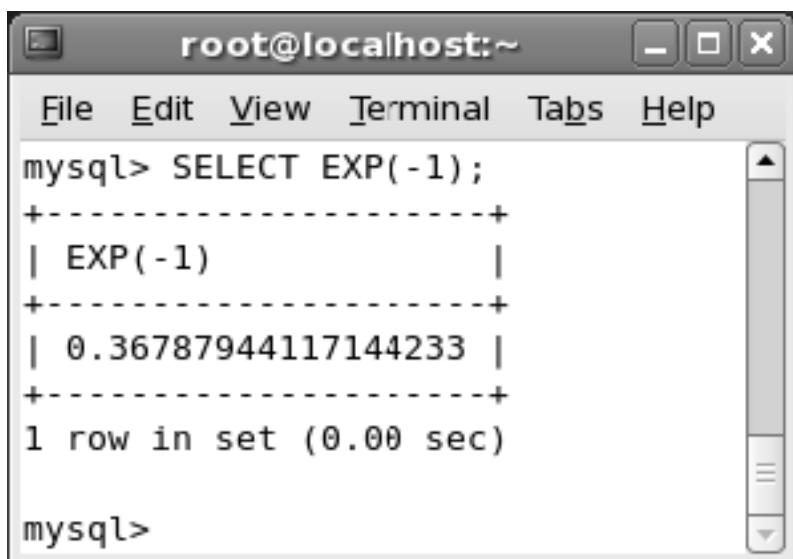
8. To calculate the exponential value of -1, enter the following command at the command prompt:

```
SELECT EXP(-1);
```

Figure 13.16 displays the output of the command.

Session 13

Using Basic Functions in MySQL - I (Lab)



A screenshot of a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area displays the following MySQL command and its output:

```
mysql> SELECT EXP(-1);
+-----+
| EXP(-1)           |
+-----+
| 0.36787944117144233 |
+-----+
1 row in set (0.00 sec)

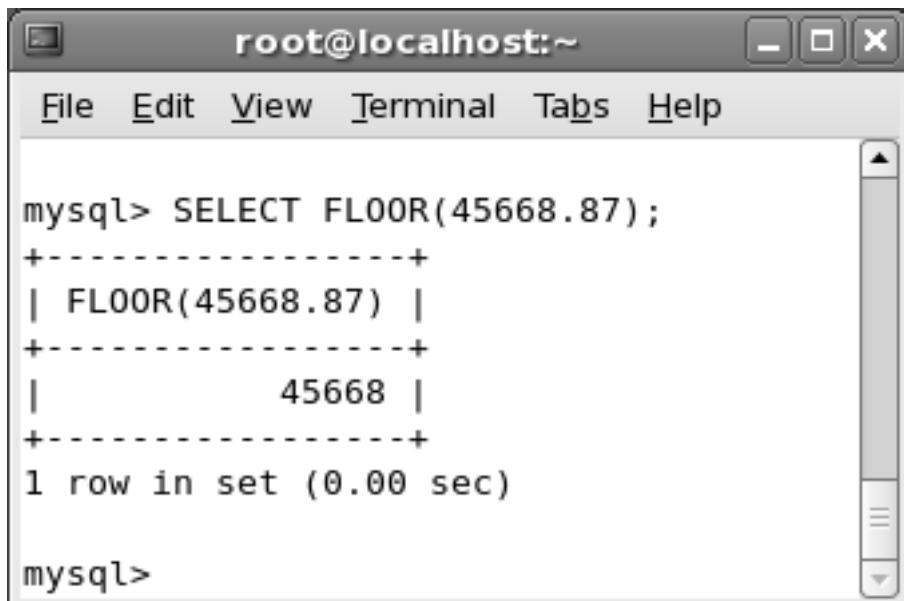
mysql>
```

Figure 13.16: EXP Function

9. John has a bank balance of 45,668.87. He wants to view the account balance without the decimals. To display the bank balance without the decimal, enter the following command at the command prompt:

```
SELECT FLOOR(45,668.87);
```

Figure 13.17 displays the output of the command.



A screenshot of a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area displays the following MySQL command and its output:

```
mysql> SELECT FLOOR(45668.87);
+-----+
| FLOOR(45668.87) |
+-----+
|        45668   |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 13.17: FLOOR Function

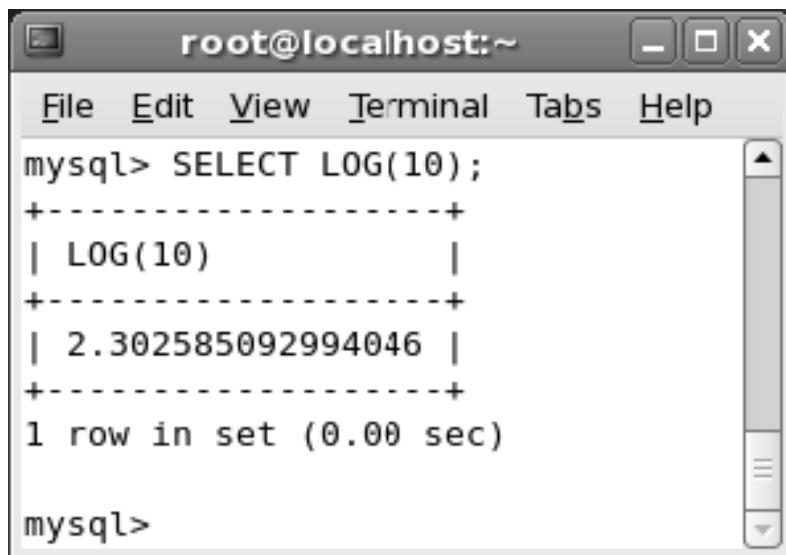
Session 13

Using Basic Functions in MySQL - I (Lab)

10. To calculate the natural logarithm of 10, enter the following command at the command prompt:

```
SELECT LOG(10);
```

Figure 13.18 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area displays the following MySQL command and its output:

```
mysql> SELECT LOG(10);
+-----+
| LOG(10) |
+-----+
| 2.302585092994046 |
+-----+
1 row in set (0.00 sec)

mysql>
```

The output shows the natural logarithm of 10, which is approximately 2.302585092994046, displayed in a tabular format with one column.

Figure: 13.18: LOG Function

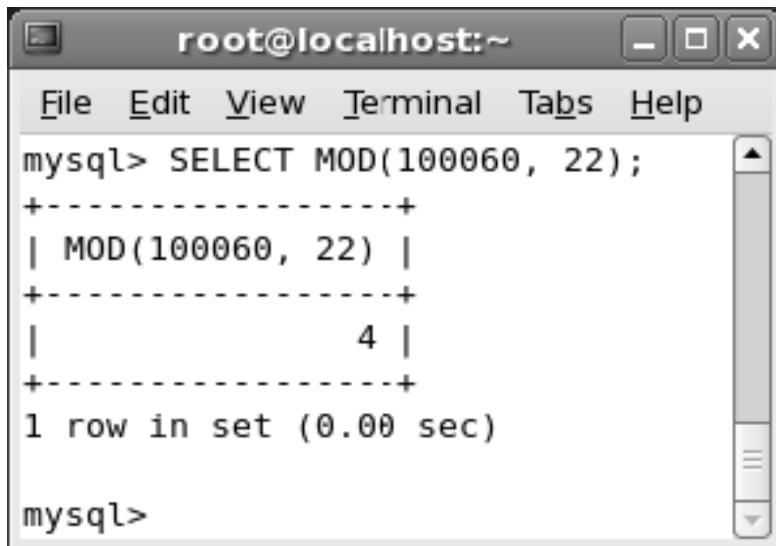
11. A restaurant bill of 10060 is to be split among 22 people. They will pay only round figure. Now, if 22 people pay round figure, how much from the total bill amount will be pending to be paid? You need to calculate the pending amount from 10060. To calculate the remainder of 10060 divided by 22, enter the following command at the command prompt:

```
SELECT MOD(100060,22);
```

Figure 13.19 displays the output of the command.

Session 13

Using Basic Functions in MySQL - I (Lab)



A screenshot of a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area displays the following MySQL command and its result:

```
mysql> SELECT MOD(100060, 22);
+-----+
| MOD(100060, 22) |
+-----+
|          4       |
+-----+
1 row in set (0.00 sec)

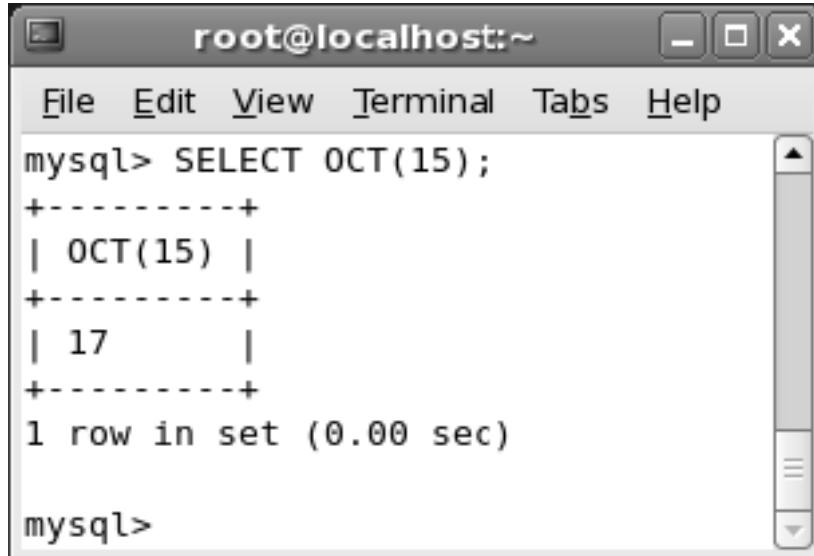
mysql>
```

Figure 13.19: MOD Function

12. To calculate the octal value of 15, enter the following command at the command prompt:

```
SELECT OCT(15);
```

Figure 13.20 displays the output of the command.



A screenshot of a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area displays the following MySQL command and its result:

```
mysql> SELECT OCT(15);
+-----+
| OCT(15) |
+-----+
|      17    |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 13.20: OCT Function

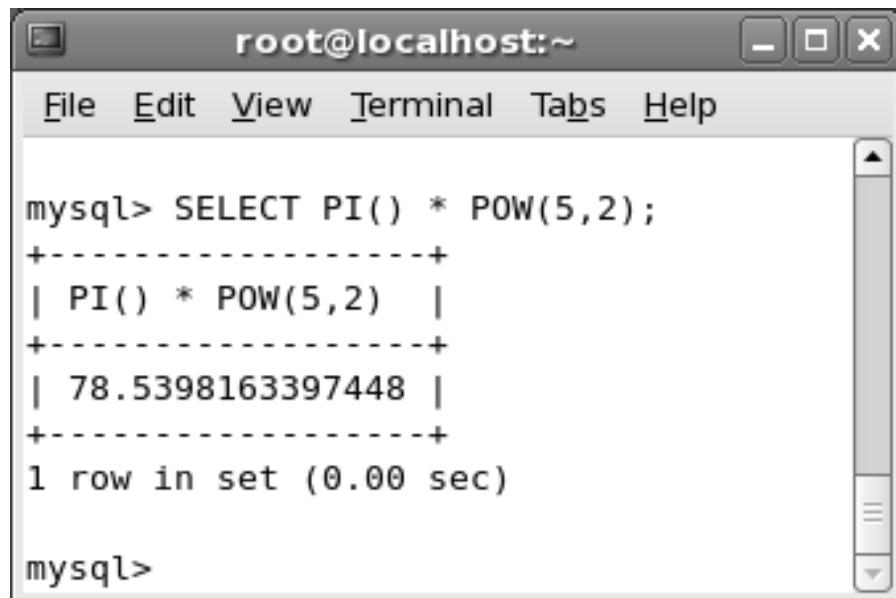
Session 13

Using Basic Functions in MySQL - I (Lab)

13. John wants to calculate the area of the circle whose radius is 5. The formula to calculate the area of a circle is $\pi * r^2$. To calculate the area of the circle, enter the following command at the command prompt:

```
SELECT PI() * POW(5,2);
```

Figure 13.21 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area contains a MySQL command-line interface. The user has entered the following SQL query:

```
mysql> SELECT PI() * POW(5,2);
```

The output shows the result of the calculation:

PI() * POW(5,2)
78.5398163397448

Below the result, it says "1 row in set (0.00 sec)". At the bottom of the window, the MySQL prompt "mysql>" is visible.

Figure 13.21: PI and POW Function

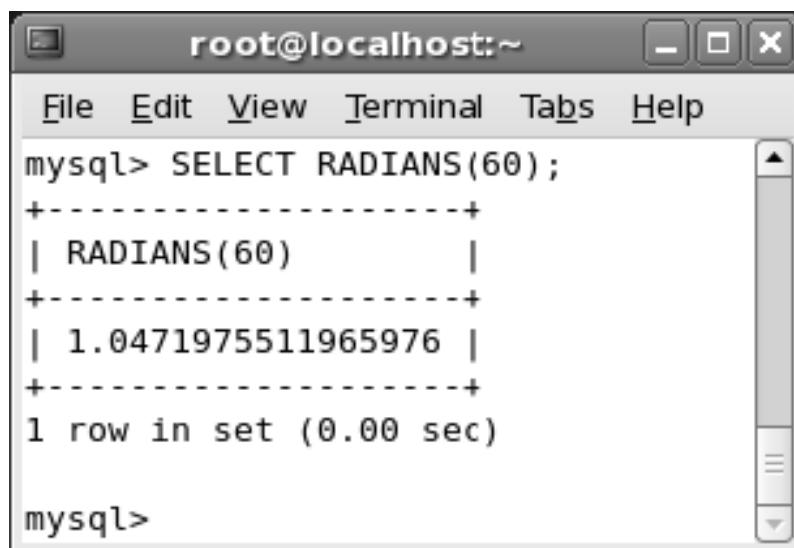
14. To calculate the value of 60 in radians, enter the following command at the command prompt:

```
SELECT RADIANS(60);
```

Figure 13.22 displays the output of the command.

Session 13

Using Basic Functions in MySQL - I (Lab)



A screenshot of a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area displays the following MySQL command and its output:

```
mysql> SELECT RADIANS(60);
+-----+
| RADIANS(60)           |
+-----+
| 1.0471975511965976 |
+-----+
1 row in set (0.00 sec)

mysql>
```

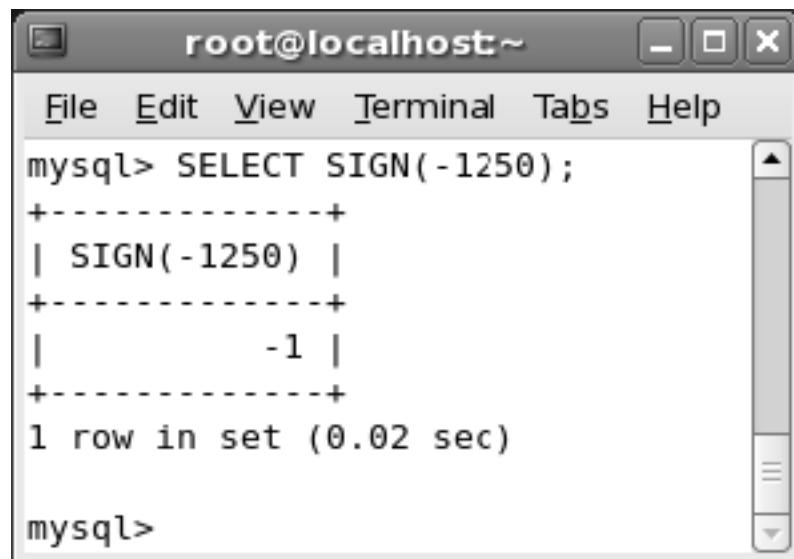
Lab Guide

Figure 13.22: RADIANS Function

15. To calculate the sign of -1250, enter the following command at the command prompt:

```
SELECT SIGN(-1250);
```

Figure 13.23 displays the output of the command.



A screenshot of a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area displays the following MySQL command and its output:

```
mysql> SELECT SIGN(-1250);
+-----+
| SIGN(-1250)   |
+-----+
|          -1  |
+-----+
1 row in set (0.02 sec)

mysql>
```

Figure 13.23: SIGN Function

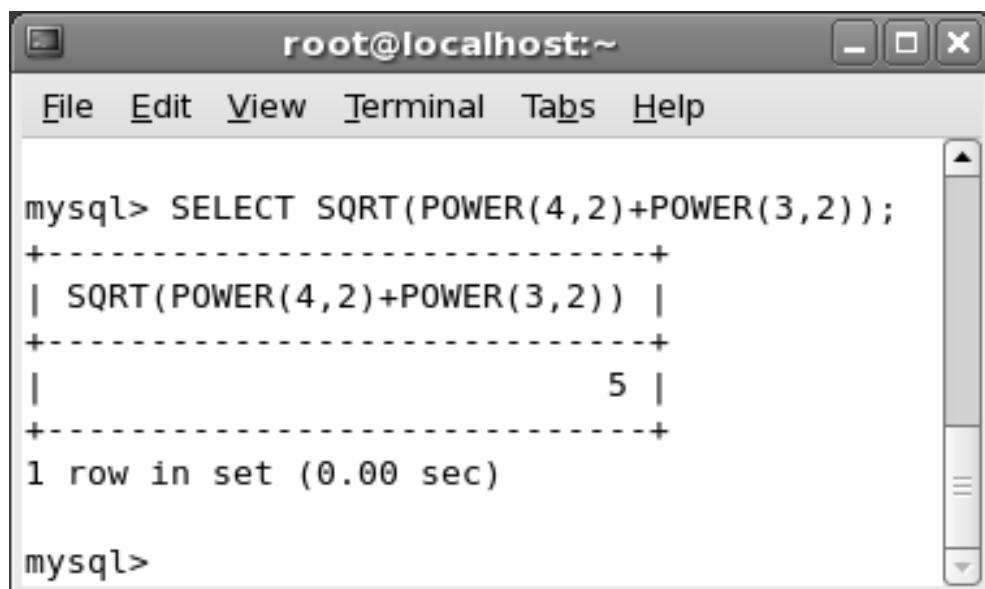
Session 13

Using Basic Functions in MySQL - I (Lab)

16. John wants to calculate the measurement of one side of a right angled triangle using the Pythagoras theorem. The equation for Pythagoras theorem is $c^2 = a^2 + b^2$ where c is the hypotenuse and a and b are the other two sides. Therefore the formula to calculate the length of the hypotenuse will be $c = \sqrt{a^2 + b^2}$. To calculate the length of the hypotenuse of a right angled triangle having length of sides as 3 cm and 4 cm, enter the following command at the command prompt:

```
SELECT SQRT(POWER(4,2)+POWER(3,2));
```

Figure 13.24 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area contains the following MySQL command and its output:

```
mysql> SELECT SQRT(POWER(4,2)+POWER(3,2));
+-----+
| SQRT(POWER(4,2)+POWER(3,2)) |
+-----+
|                      5 |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 13.24: SQRT Function

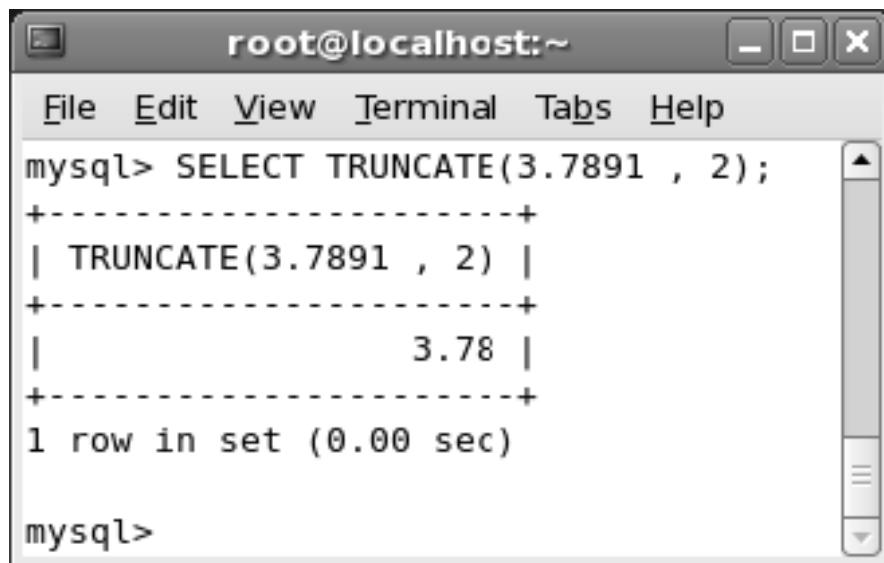
17. To truncate the BASIC_SAL from 5 decimal places to 2 decimal places, enter the following command at the command prompt:

```
SELECT BASIC_SAL, TRUNCATE(BASIC_SAL,2)
FROM EMP_SALARY;
```

Figure 13.25 displays the output of the command.

Session 13

Using Basic Functions in MySQL - I (Lab)



A screenshot of a terminal window titled "root@localhost:~". The window contains the following MySQL command and its output:

```
mysql> SELECT TRUNCATE(3.7891 , 2);
+-----+
| TRUNCATE(3.7891 , 2) |
+-----+
|          3.78 |
+-----+
1 row in set (0.00 sec)

mysql>
```

The terminal window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. There are also standard window control buttons (minimize, maximize, close) at the top right.

Figure 13.25: TRUNCATE Function

Session 13

Using Basic Functions in MySQL - I (Lab)



Do It Yourself

1. Calculate the BITWISE AND for 15 and 27.
2. Calculate the BITWISE OR for 21 and 25.
3. Calculate the BITWISE XOR for 16 and 20.
4. From the LIBRARY database, display all the columns of the ISSUE_DETAILS table having minimum fine greater than 10.
5. From the EMPLOYEE database, calculate the standard deviation of the gross salary of the employees.
6. Calculate the sample standard deviation of the gross salary of all the employees.
7. Calculate the sample variance of the basic salary of all the employees.
8. Calculate the arc sine value of 1.
9. Calculate the cotangent value of 30.
10. Calculate the logarithm of 1000 to the base 10.
11. Calculate the logarithm of 64 to the base 2.
12. Calculate the sin value of $\pi/2$.
13. Calculate the tangent value of 60.
14. Calculate the length of the hypotenuse of a right-angled triangle having two sides with length as 6 cm and 7 cm.

Objectives

At the end of this session, the student will be able to:

- *Describe the use of Date functions in MySQL.*
- *Describe the use of String functions in MySQL.*
- *Describe the use of System Information functions in MySQL.*

14.1 Introduction

Functions are independent programs that execute on the parameter passed and return a value depending on the parameter. Parameters are also known as arguments. You can use MySQL functions within a query to operate on single or multiple columns of a table.

In this session, you will learn about the basics of using functions. You will use different types of functions, such as date, string, and system information functions in MySQL.

14.2 Date Functions in MySQL

MySQL provides date functions that use the date and time data type. The date functions enable you to perform different functions on a date.

14.2.1 ADDDATE Function

The `ADDDATE` function adds two date expressions. The syntax for adding two date expressions is:

```
SELECT ADDDATE(expr1,expr2);
```

where,

`ADDDATE` – calculates the date

`expr1` – specifies a date

`expr2` – defines the date to be added

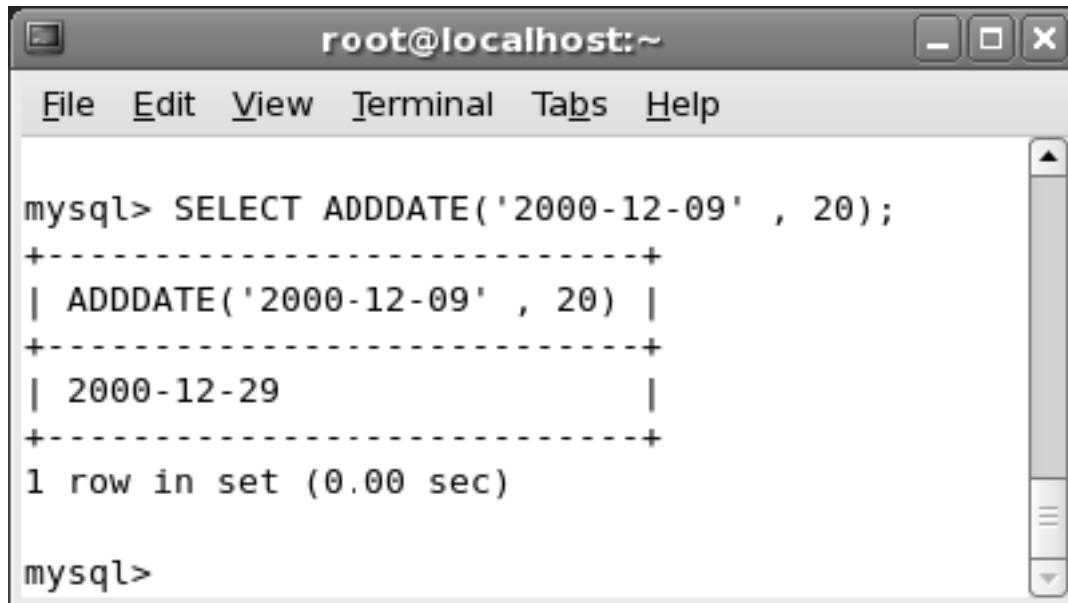
To add 20 days to the date 2000-12-09, enter the following command at the command prompt:

Session 14

Using Basic Functions in MySQL - II

```
SELECT ADDDATE('2000-12-09', 20);
```

Figure 14.1 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The main area contains the following MySQL command and its output:

```
mysql> SELECT ADDDATE('2000-12-09' , 20);
+-----+
| ADDDATE('2000-12-09' , 20) |
+-----+
| 2000-12-29 |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 14.1: ADDDATE Function

14.2.2 ADDTIME Function

The ADDTIME function is used to add two time expressions. The syntax for using this function is:

```
SELECT ADDTIME(expr1, expr2);
```

where,

ADDTIME – calculates the time

expr1 – defines a time or datetime expression

expr2 – specifies a time expression

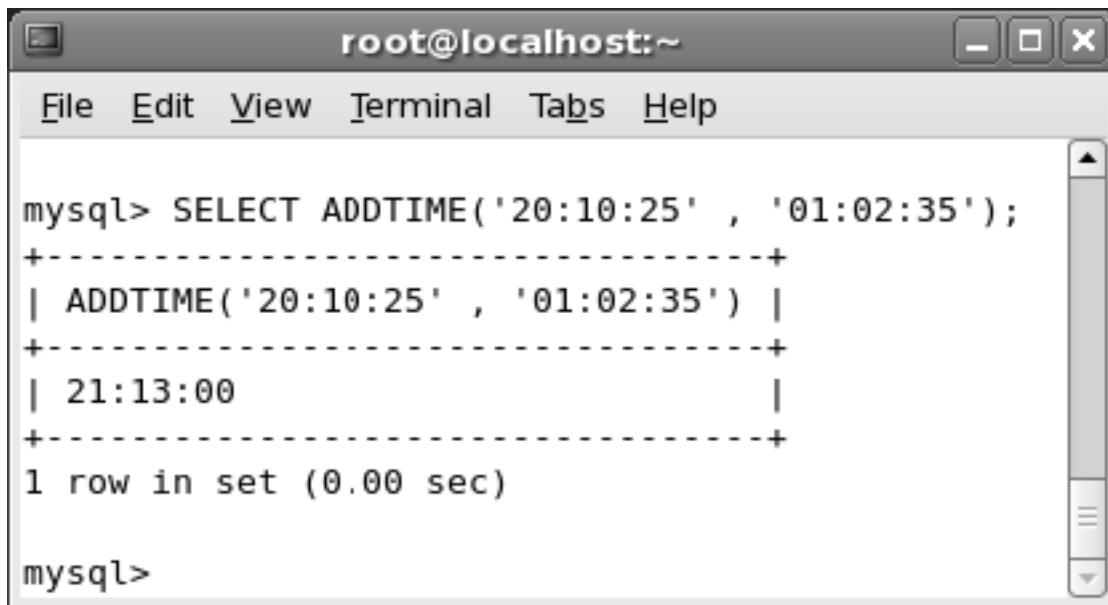
To calculate the addition of 20:10:25 and 01:02:35, enter the following command at the command prompt:

```
SELECT ADDTIME('20:10:25','01:02:35');
```

Session 14

Using Basic Functions in MySQL - II

Figure 14.2 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The main area contains the following MySQL command and its output:

```
mysql> SELECT ADDTIME('20:10:25' , '01:02:35');
+-----+
| ADDTIME('20:10:25' , '01:02:35') |
+-----+
| 21:13:00 |
+-----+
1 row in set (0.00 sec)

mysql>
```

Concepts

Figure 14.2: ADDTIME Function

14.2.3 CURDATE Function

The CURDATE function returns the current date in the YYYY-MM-DD or YYYYMMDD format. The syntax to retrieve the current date is:

```
SELECT CURDATE();
```

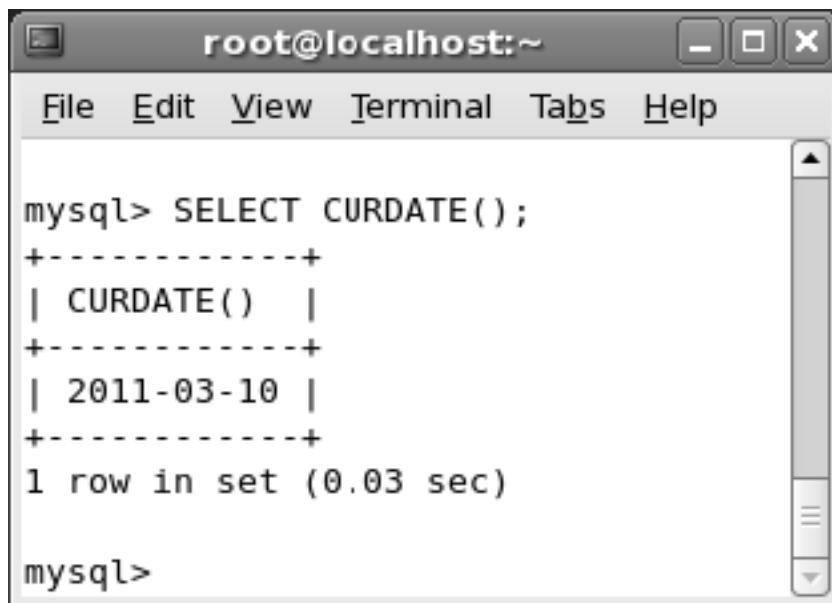
For example, to view the current date, enter the following command at the command prompt:

```
SELECT CURDATE();
```

Figure 14.3 displays the output of the command.

Session 14

Using Basic Functions in MySQL - II



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its output:

```
mysql> SELECT CURDATE();
+-----+
| CURDATE() |
+-----+
| 2011-03-10 |
+-----+
1 row in set (0.03 sec)

mysql>
```

Figure 14.3: CURDATE Function

The alternative syntax for obtaining the current date is:

```
SELECT CURRENT_DATE ;
```

14.2.4 CURTIME Function

The CURTIME function displays the current time. The syntax to view the current time is:

```
SELECT CURTIME();
```

For example, to obtain the current time, enter the following command at the command prompt:

```
SELECT CURTIME();
```

The alternative syntax for viewing the current time is:

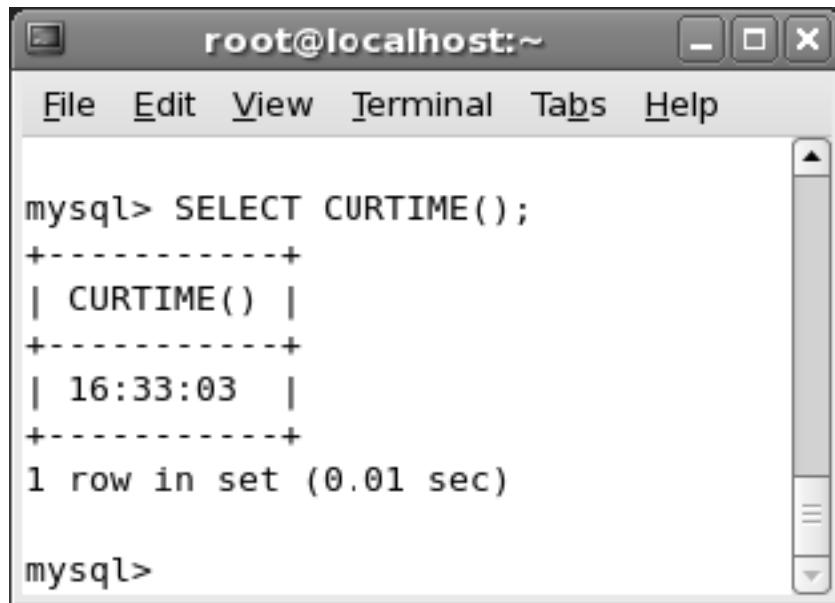
```
SELECT CURRENT_TIME ;
```

Figure 14.4 displays the output of the command.

Session 14

Using Basic Functions in MySQL - II

Concepts



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its output:

```
mysql> SELECT CURTIME();
+-----+
| CURTIME() |
+-----+
| 16:33:03 |
+-----+
1 row in set (0.01 sec)

mysql>
```

Figure 14.4: CURTIME Function

14.2.5 DATE_ADD Function

The `DATE_ADD` function appends a specified time interval to the given date. The syntax for this function is:

```
SELECT DATE_ADD(date, INTERVAL expr unit);
```

where,

`DATE_ADD` – alters the date with the specified interval

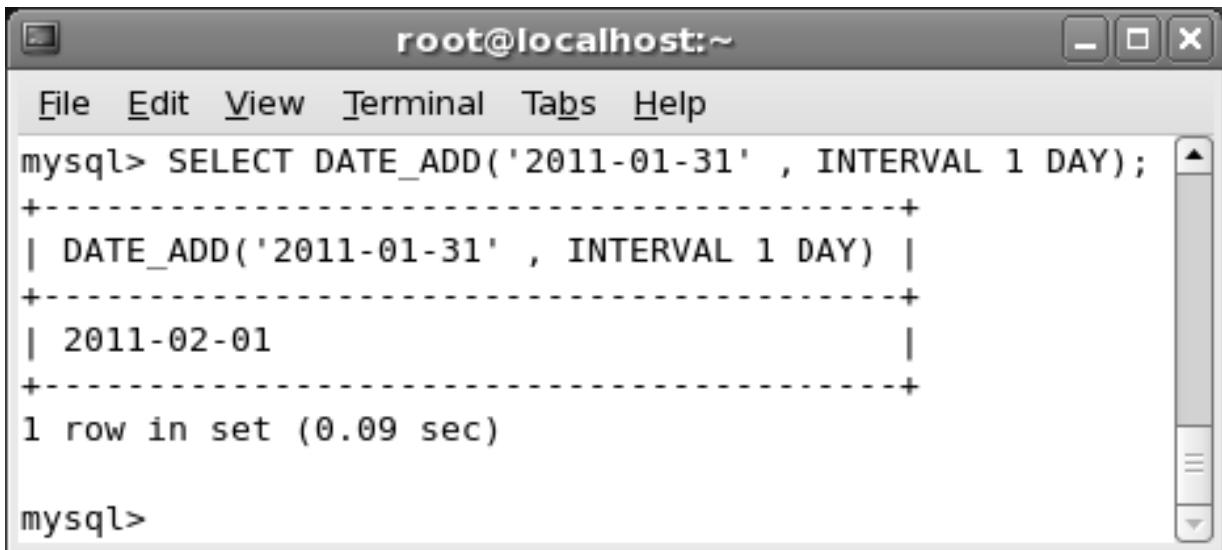
`date` – specifies the starting date or datetime value

`INTERVAL expr unit` – specifies the interval to be added to the given date

For example, to add 1 day to the date 2011-01-31, enter the following command at the command prompt:

```
SELECT DATE_ADD('2011-01-31', INTERVAL 1 DAY);
```

Figure 14.5 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its output:

```
mysql> SELECT DATE_ADD('2011-01-31' , INTERVAL 1 DAY);
+-----+
| DATE_ADD('2011-01-31' , INTERVAL 1 DAY) |
+-----+
| 2011-02-01                                |
+-----+
1 row in set (0.09 sec)

mysql>
```

Figure 14.5: DATE_ADD Function

Note: The value returned can be a datetime or a string depending upon the arguments.

14.2.6 DATE function

The DATE function displays the date part of the specified date or timestamp. The syntax for this function is:

```
SELECT DATE(expression);
```

For example, to view only the date part from 2007-11-12 21:20:00, enter the following command at the command prompt:

```
SELECT DATE('2007-11-12 21:20:00');
```

Figure 14.6 displays the output of the command.

Session 14

Using Basic Functions in MySQL - II

Concepts

The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The main area displays the following MySQL command and its output:

```
mysql> SELECT DATE('2007-11-12 21:20:00');
+-----+
| DATE('2007-11-12 21:20:00') |
+-----+
| 2007-11-12                      |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 14.6: DATE Function

14.2.7 DATEDIFF Function

The `DATEDIFF` function returns the number of days between a start date and an end date. These dates are entered as arguments. The syntax to calculate the date difference is:

```
SELECT DATEDIFF(expr1,expr2);
```

where,

`DATEDIFF` – calculates the number of days

`expr1` – defines the start date

`expr2` – defines the end date

For example, to calculate the difference between the dates 2003-10-12 and 2003-09-03, enter the following command at the command prompt:

```
SELECT DATEDIFF('2003-10-12','2003-09-03');
```

Figure 14.7 displays the output of the command.

Session 14

Using Basic Functions in MySQL - II

The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The main area displays the following MySQL command and its result:

```
mysql> SELECT DATEDIFF('2003-10-12' , '2003-09-03');
+-----+
| DATEDIFF('2003-10-12' , '2003-09-03') |
+-----+
| 39 |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 14.7: DATEDIFF Function

14.2.8 DATE_FORMAT Function

The `DATE_FORMAT` function displays the specified date in a particular format. The syntax to use this function is:

```
SELECT DATE_FORMAT(date, format);
```

where,

`date` – defines a valid date

`format` – defines the output format to display the date

To display a particular date in the `MM-DD-YYYY` format, enter the following command at the command prompt:

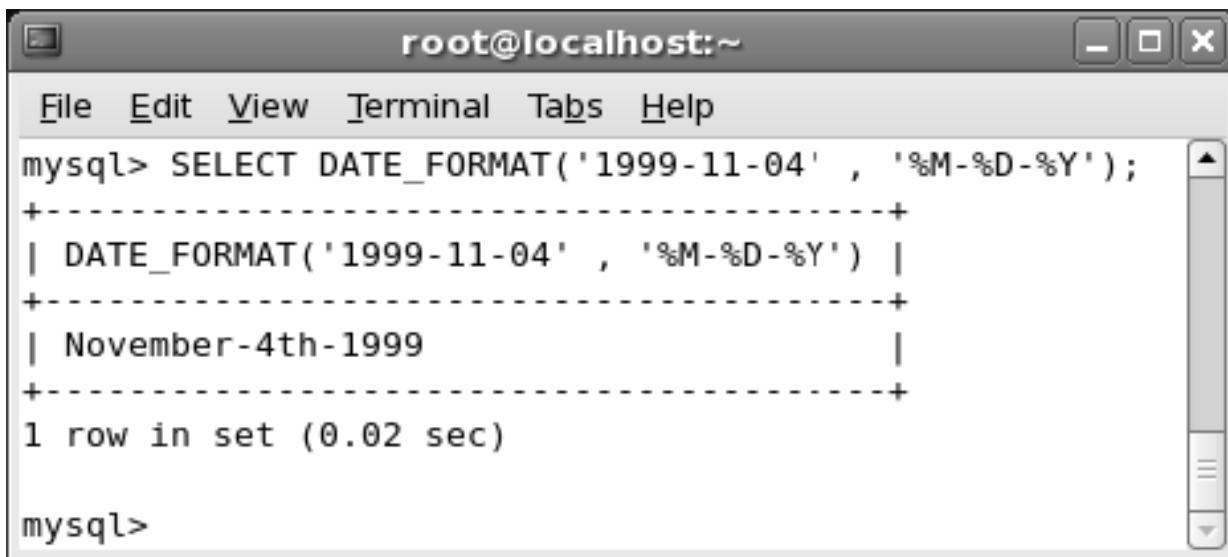
```
SELECT DATE_FORMAT('1999-11-04','%M-%D-%Y');
```

Figure 14.8 displays the output of the command.

Session 14

Using Basic Functions in MySQL - II

Concepts



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its output:

```
mysql> SELECT DATE_FORMAT('1999-11-04' , '%M-%D-%Y');
+-----+
| DATE_FORMAT('1999-11-04' , '%M-%D-%Y') |
+-----+
| November-4th-1999                         |
+-----+
1 row in set (0.02 sec)

mysql>
```

Figure 14.8: DATE_FORMAT Function

Table 14.1 lists some of the format specifiers.

Specifier	Description
%a	Abbreviated weekday name
%f	Microseconds
%H	Hour (00 – 23)
%h	Hour (00 – 12)
%M	Month name (January – December)
%m	Month, numeric (01 – 12)
%p	A.M. or P.M.
%s	Seconds
%D	Date
%Y	Year, numeric, four digits

Table 14.1: Format Specifiers

14.2.9 DAY Function

The DAY function returns the day of the month for the specified date. The range is from 1 to 31. The syntax to display the day of the month is:

```
SELECT DAY(date);
```

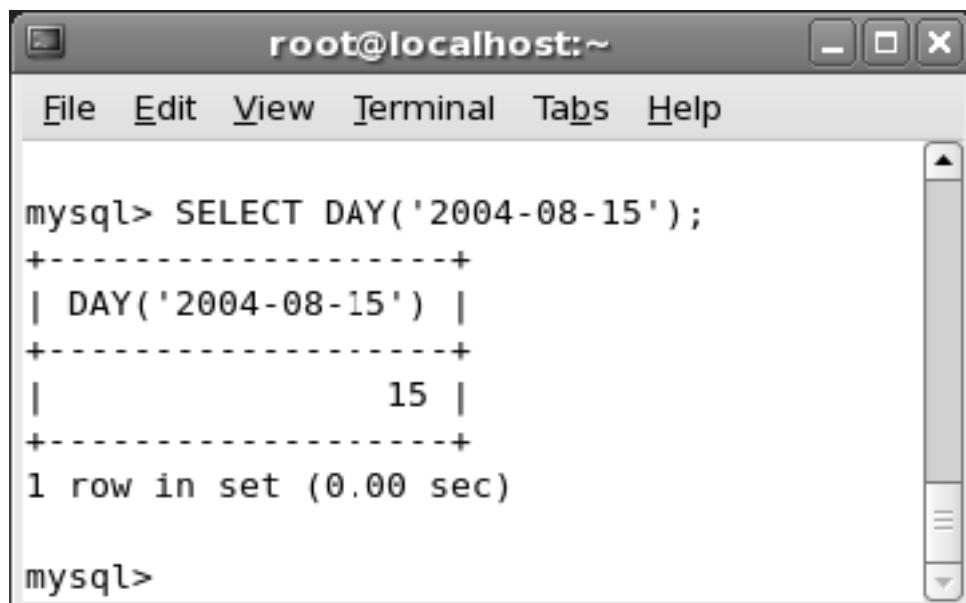
Session 14

Using Basic Functions in MySQL - II

For example, to obtain the day for the date 2004-08-15, enter the following command at the command prompt:

```
SELECT DAY('2004-08-15');
```

Figure 14.9 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area contains the following MySQL query and its result:

```
mysql> SELECT DAY('2004-08-15');
+-----+
| DAY('2004-08-15') |
+-----+
|          15 |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 14.9: DAY Function

14.2.10 DAYNAME Function

The DAYNAME function returns the name of the weekday for a date entered as an argument. The syntax to display the name of the weekday is:

```
SELECT DAYNAME(date);
```

For example, to obtain the name of the day for the date 2002-05-09, enter the following command at the command prompt:

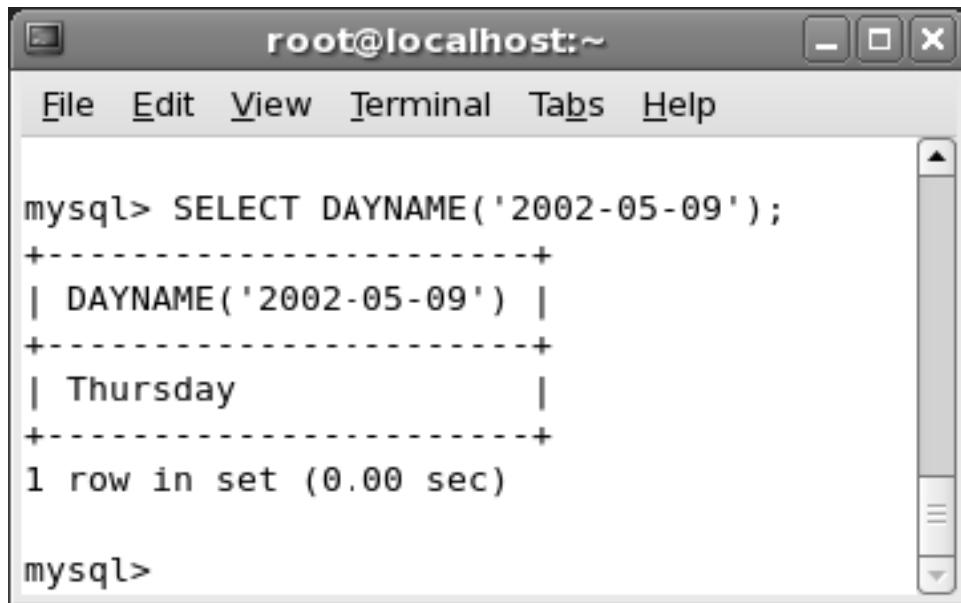
```
SELECT DAYNAME('2002-05-09');
```

Figure 14.10 displays the output of the command.

Session 14

Using Basic Functions in MySQL - II

Concepts



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The main area displays the following MySQL command and its output:

```
mysql> SELECT DAYNAME('2002-05-09');
+-----+
| DAYNAME('2002-05-09') |
+-----+
| Thursday                |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 14.10: DAYNAME Function

14.2.11 HOUR Function

The `HOUR` function returns the hour of the time specified as an argument. The syntax for this function is:

```
SELECT HOUR(expression);
```

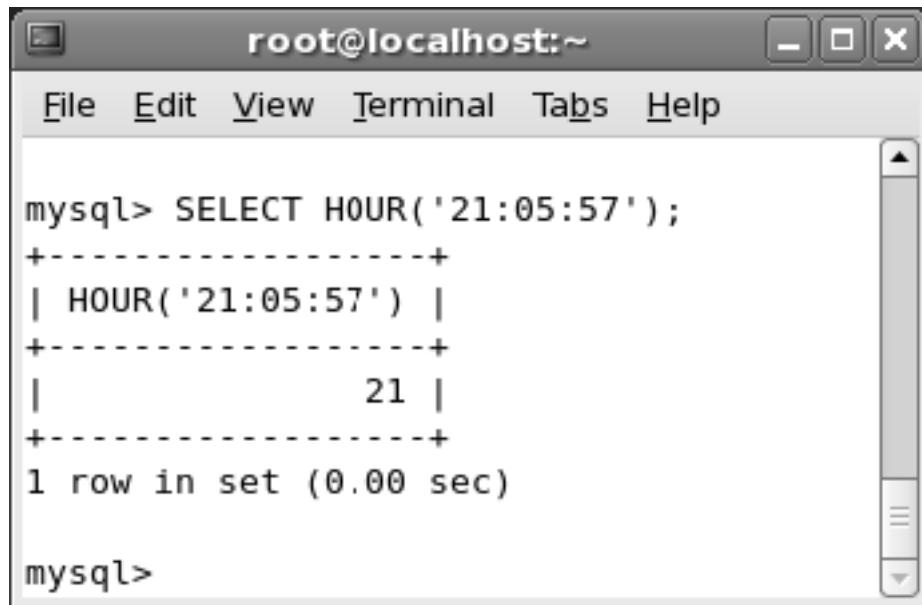
For example, to retrieve the hour from the expression `21:05:57`, enter the following command at the command prompt:

```
SELECT HOUR('21:05:57');
```

Figure 14.11 displays the output of the command.

Session 14

Using Basic Functions in MySQL - II



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The main area displays the following MySQL command and its output:

```
mysql> SELECT HOUR('21:05:57');
+-----+
| HOUR('21:05:57') |
+-----+
|          21      |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 14.11: HOUR Function

14.2.12 MINUTE Function

The MINUTE function extracts the minutes from the specified time argument. The output of this function is a numeric value ranging from 0 to 59. The syntax for this function is:

```
SELECT MINUTE(expression);
```

For example, to display the minutes from a datetime expression, enter the following command at the command prompt:

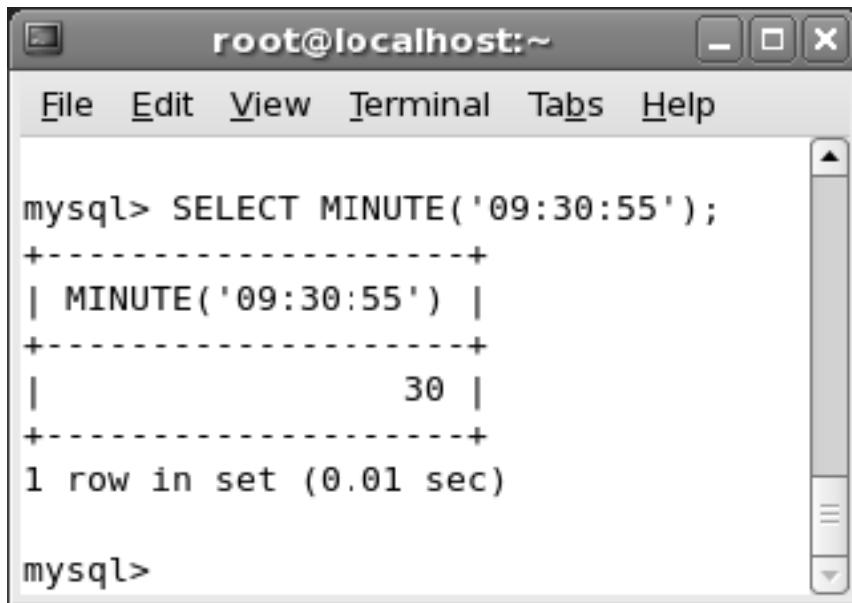
```
SELECT MINUTE('09:30:55');
```

Figure 14.12 displays the output of the command.

Session 14

Using Basic Functions in MySQL - II

Concepts



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area displays the following MySQL command and its output:

```
mysql> SELECT MINUTE('09:30:55');
+-----+
| MINUTE('09:30:55') |
+-----+
|          30 |
+-----+
1 row in set (0.01 sec)

mysql>
```

Figure 14.12: MINUTE Function

14.2.13 MONTH Function

The `MONTH` function extracts the month from the specified argument. The output of this function is a numeric value ranging from 1 to 12. The syntax for this function is:

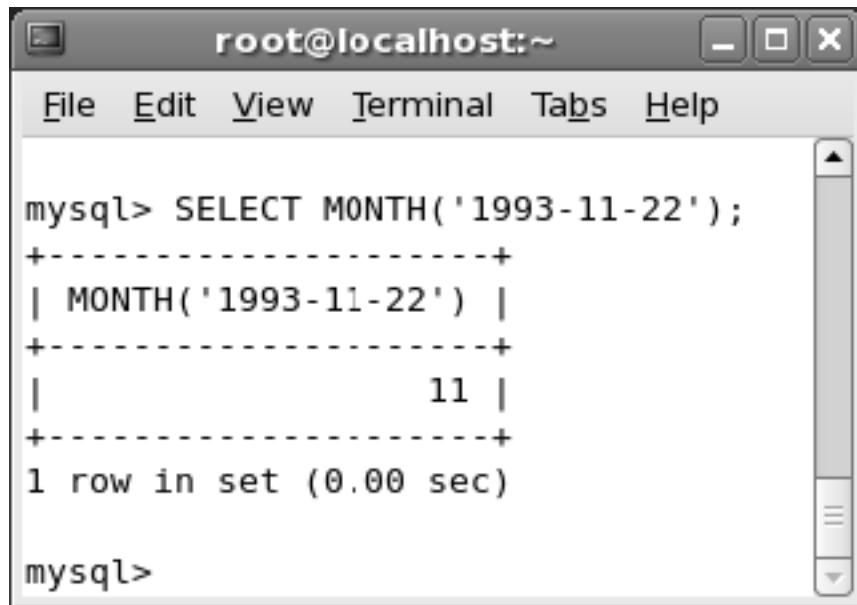
```
SELECT MONTH(expression);
```

For example, to display the month from the datetime expression `1993-11-22`, enter the following command at the command prompt:

```
SELECT MONTH('1993-11-22');
```

Figure 14.13 displays the output of the command.

Session 14



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The main area displays the following MySQL command and its result:

```
mysql> SELECT MONTH('1993-11-22');
+-----+
| MONTH('1993-11-22') |
+-----+
|           11 |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 14.13: MONTH Function

14.2.14 MONTHNAME Function

The `MONTHNAME` function returns the name of the month for the date passed as an argument. The syntax for displaying the name of the month is:

```
SELECT MONTHNAME(date);
```

For example, to display the name of the month for the date `2006-07-03`, enter the following command at the command prompt:

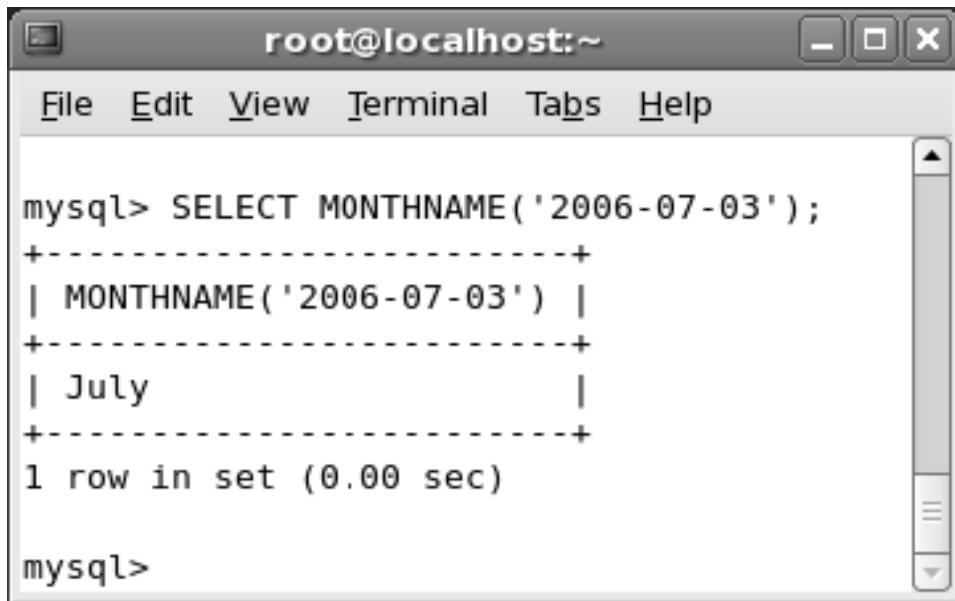
```
SELECT MONTHNAME('2006-07-03');
```

Figure 14.14 displays the output of the command.

Session 14

Using Basic Functions in MySQL - II

Concepts



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its output:

```
mysql> SELECT MONTHNAME('2006-07-03');
+-----+
| MONTHNAME('2006-07-03') |
+-----+
| July |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 14.14: MONTHNAME Function

14.2.15 NOW Function

The `NOW` function returns the current date and time. The output of this function is in the `YYYY-MM-DD HH:MM:SS` or `YYYYMMDDHHMMSS.uuuuuu` format, and the value is expressed in the current time zone. The syntax to view the current date and time value is:

```
SELECT NOW();
```

For example, to view the current timestamp, enter the following command at the command prompt:

```
SELECT NOW();
```

The alternative commands to display the current time and date are:

```
SELECT CURRENT_TIMESTAMP();
SELECT LOCALTIME();
SELECT LOCALTIMESTAMP();
```

Figure 14.15 displays the output of the `NOW` function.

Session 14

Using Basic Functions in MySQL - II



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The main area displays the following MySQL command and its output:

```
mysql> SELECT NOW();
+-----+
| NOW() |
+-----+
| 2011-03-10 16:54:48 |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 14.15: NOW Function

14.2.16 SECOND Function

The `SECOND` function returns the number of seconds specified in the argument. The output of this function is a numeric value ranging from 1 to 59. The syntax for this function is:

```
SELECT SECOND(expression);
```

For example, to calculate the number of seconds for 21:10:25, enter the following command at the command prompt:

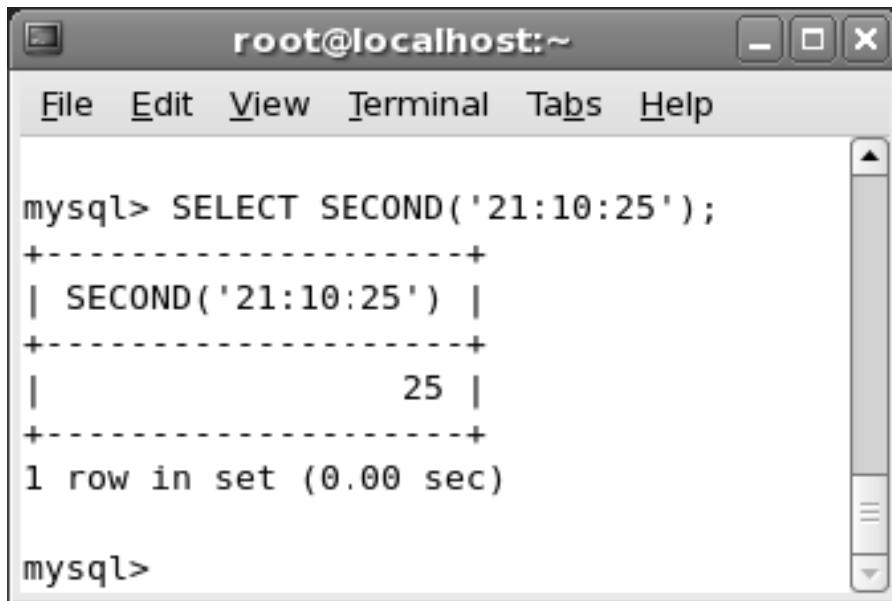
```
SELECT SECOND('21:10:25');
```

Figure 14.16 displays the output of the command.

Session 14

Using Basic Functions in MySQL - II

Concepts



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its output:

```
mysql> SELECT SECOND('21:10:25');
+-----+
| SECOND('21:10:25') |
+-----+
|          25 |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 14.16: SECOND Function

14.2.17 SEC_TO_TIME Function

The `SEC_TO_TIME` function converts the number of seconds specified as an argument to `HH:MM:SS` format. The syntax for this function is:

```
SELECT SEC_TO_TIME(expression);
```

where,

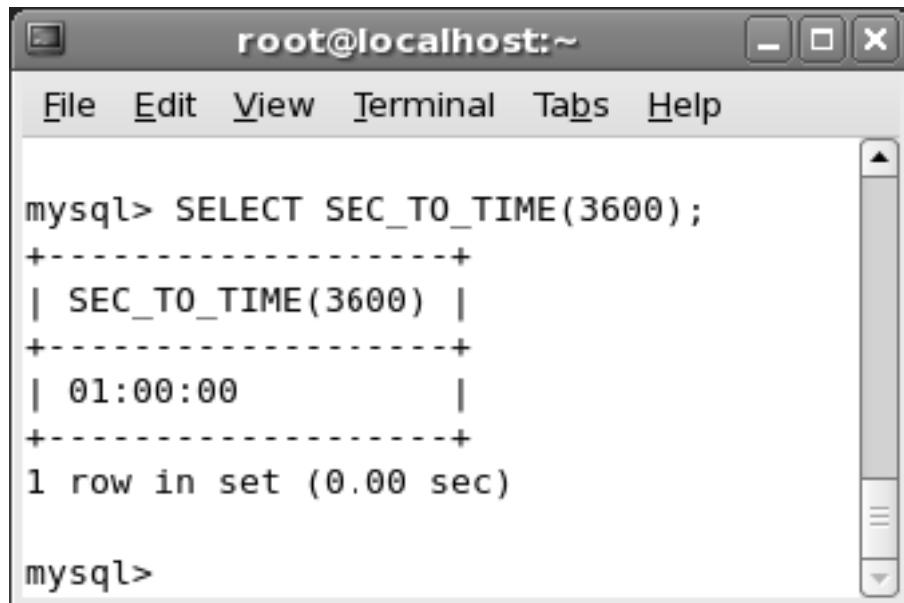
`SEC_TO_TIME` – converts seconds to `HH:MM:SS` format

`expression` – defines a numeric value specifying the number of seconds

To convert `3600` to the `HH:MM:SS` format, enter the following command at the command prompt:

```
SELECT SEC_TO_TIME(3600);
```

Figure 14.17 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area displays the following MySQL command and its output:

```
mysql> SELECT SEC_TO_TIME(3600);
+-----+
| SEC_TO_TIME(3600) |
+-----+
| 01:00:00          |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 14.17: SEC_TO_TIME Function

14.2.18 SYSDATE Function

The SYSDATE function returns the current date and time value after execution. The SYSDATE function is different from the NOW function. Consider an example, where you have to add 50,000 records to a table in the database. These rows will have a different timestamp because all the rows will not be added at the same time to the table in the database. The SYSDATE function returns this unique time required to execute a function. On the other hand, the NOW function will display the same execution time for adding all the 50,000 records to the table. The syntax for this function is:

```
SELECT SYSDATE(expression);
```

For example, to display the current date and time value, enter the following command at the command prompt:

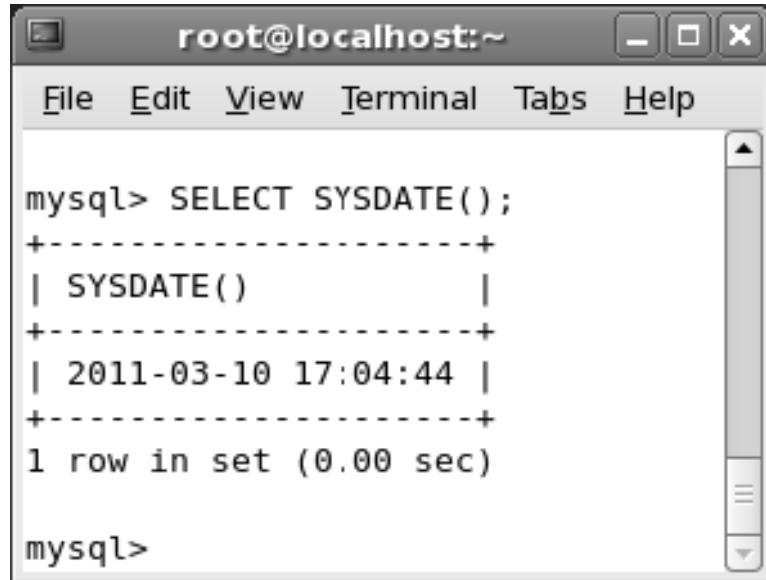
```
SELECT SYSDATE();
```

Figure 14.18 displays the output of the command.

Session 14

Using Basic Functions in MySQL - II

Concepts



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The main area displays the following MySQL query and its result:

```
mysql> SELECT SYSDATE();
+-----+
| SYSDATE()           |
+-----+
| 2011-03-10 17:04:44 |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 14.18: SYSDATE Function

14.2.19 TIME Function

The `TIME` function displays only the time part of the specified date or timestamp. The syntax for this function is:

```
SELECT TIME(expression);
```

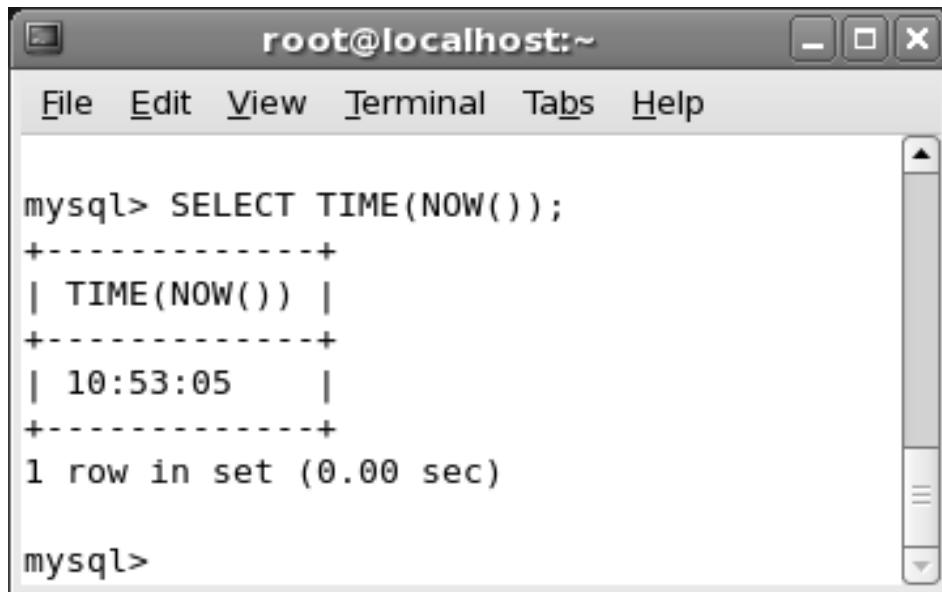
For example, to display only the time from `2007-09-12 20:20:00`, enter the following command at the command prompt:

```
SELECT TIME(NOW());
```

Figure 14.19 displays the output of the command.

Session 14

Using Basic Functions in MySQL - II



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The main area displays the following MySQL command and its output:

```
mysql> SELECT TIME(NOW());
+-----+
| TIME(NOW()) |
+-----+
| 10:53:05   |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 14.19: TIME Function

14.2.20 TIMEDIFF Function

The **TIMEDIFF** function returns time interval between two time arguments. The syntax to calculate the time difference is:

```
SELECT TIMEDIFF(time1, time2);
```

where,

TIMEDIFF – calculates the time difference

time1, time2 – specifies the time expressions

Note: In the syntax, the values for **time1** and **time2** must be of the Date data type.

For example, to calculate the difference between 10:15:20 and 12:45:20, enter the following command at the command prompt:

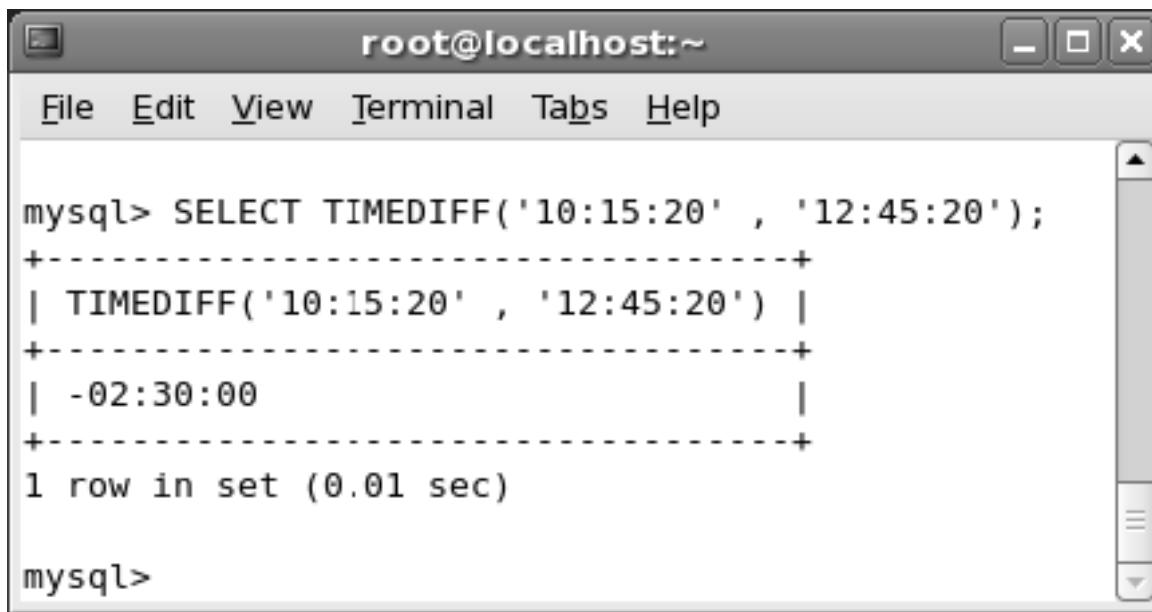
```
SELECT TIMEDIFF('10:15:20','12:45:20');
```

Figure 14.20 displays the output of the command.

Session 14

Using Basic Functions in MySQL - II

Concepts



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The main area displays the following MySQL command and its result:

```
mysql> SELECT TIMEDIFF('10:15:20' , '12:45:20');
+-----+
| TIMEDIFF('10:15:20' , '12:45:20') |
+-----+
| -02:30:00 |
+-----+
1 row in set (0.01 sec)

mysql>
```

Figure 14.20: TIMEDIFF Function

14.2.21 WEEK Function

The **WEEK** function returns the week number for the value specified in the argument. The syntax for this function is:

```
SELECT WEEK(date, mode);
```

where,

WEEK – displays the week number

date – specifies the date

mode – defines the return value. The return value can be within a range of 1 to 53 or 0 to 53 depending on whether the week starts on Monday or Sunday

Note: When you do not specify the mode, value for the default week format is used.

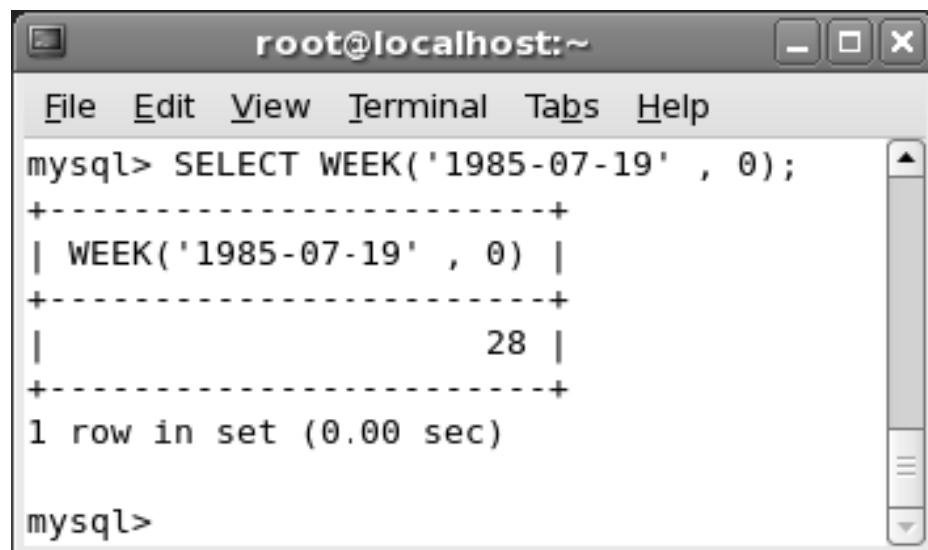
For example, to retrieve the week number for 1985-07-19, enter the following command at the command prompt:

```
SELECT WEEK('1985-07-19', 0);
```

Session 14

Using Basic Functions in MySQL - II

Figure 14.21 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its output:

```
mysql> SELECT WEEK('1985-07-19', 0);
+-----+
| WEEK('1985-07-19', 0) |
+-----+
| 28 |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 14.21: WEEK Function

14.2.22 WEEKOFYEAR Function

The WEEKOFYEAR function returns the week number for the specified date. The output of this function is a numeric value ranging between 1 to 53. The syntax for this function is:

```
SELECT WEEKOFYEAR(expression);
```

For example, to calculate the week number for the date 2005-04-25, enter the following command at the command prompt:

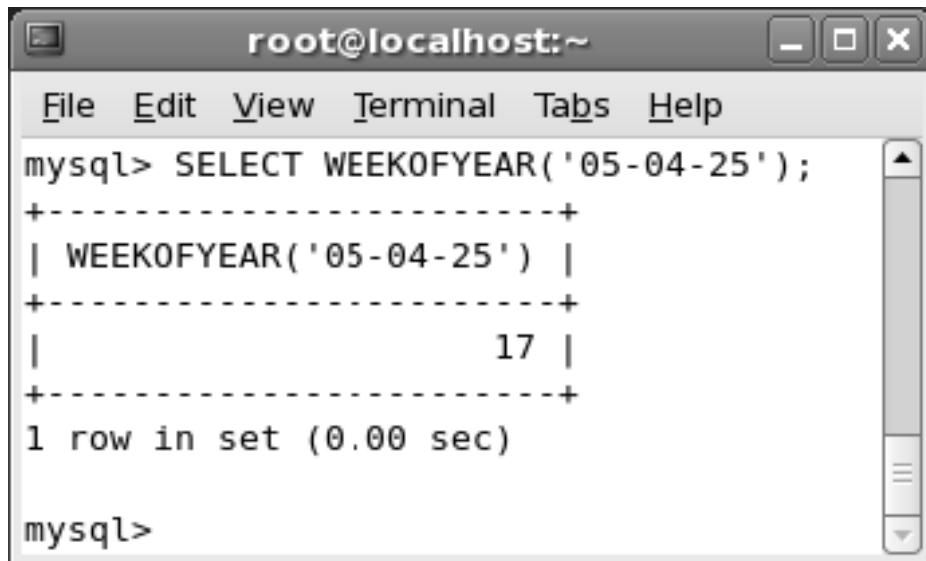
```
SELECT WEEKOFYEAR('2005-04-25');
```

Figure 14.22 displays the output of the command.

Session 14

Using Basic Functions in MySQL - II

Concepts



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area displays the following MySQL command and its result:

```
mysql> SELECT WEEKOFYEAR('05-04-25');
+-----+
| WEEKOFYEAR('05-04-25') |
+-----+
| 17 |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 14.22: WEEKOFYEAR Function

14.2.23 YEAR Function

The YEAR function returns the year for the date specified in the argument. The syntax for obtaining the year from a date argument is:

```
SELECT YEAR(date);
```

For example, to display the year from the date 2000-09-09, enter the following command at the command prompt:

```
SELECT YEAR('2000-09-09');
```

Figure 14.23 displays the output of the command.

Session 14

Using Basic Functions in MySQL - II

The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its output:

```
mysql> SELECT YEAR('2000-09-09');
+-----+
| YEAR('2000-09-09') |
+-----+
|          2000 |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 14.23: YEAR Function

Table 14.2 describes the additional Date functions in MySQL.

Name	Description	Example
CONVERT_TZ	<p>The CONVERT_TZ function changes the time zone of an argument. The syntax for using this function is:</p> <pre>SELECT CONVERT_TZ(dt, from_tz, to_tz);</pre> <p>where,</p> <ul style="list-style-type: none"> CONVERT – edits the time zone dt – specifies the date in the current time zone from_tz – specifies the current time zone of the argument to_tz – specifies the new time zone 	<p>To convert the time zone for the date 2004-01-01, enter the following command at the command prompt:</p> <pre>SELECT CONVERT_TZ('2004-01-01 12:00:00', '+00:00', '+10:00');</pre> <p>The output of this function is:</p> <p>2004-01-01 22:00:00</p>
CURRENT_TIMESTAMP	The CURRENT_TIMESTAMP function displays the current date and time as a value in YYYY-MM-DD HH:MM:SS or YYYYMMDDHHMMSS.uuuuuu format. The value is expressed in	For example, to view the current timestamp, enter the following command at the command prompt:

Session 14

Using Basic Functions in MySQL - II

Concepts

Name	Description	Example
CURRENT_TIMESTAMP	<p>the current time zone. The syntax to view the current timestamp is:</p> <pre>SELECT CURRENT_TIMESTAMP();</pre>	<pre>SELECT CURRENT_TIMESTAMP();</pre> <p>The output of this function will be the current date and time.</p>
DATE_SUB	<p>The DATE_SUB function subtracts a specified time interval from the given date. The syntax for this function is:</p> <pre>SELECT DATE_SUB(date, INTERVAL expr unit);</pre> <p>where,</p> <p>DATE_SUB – subtracts the time</p> <p>date – defines the starting date or datetime value</p> <p>INTERVAL expr – specifies the interval to be subtracted from the given date</p>	<p>To subtract 1 day from the current date, enter the following command at the command prompt:</p> <pre>SELECT DATE_SUB(CURDATE(), INTERVAL 1 DAY);</pre> <p>The output of this function will be yesterday's date.</p>
DAYOFMONTH	<p>The DAYOFMONTH function returns the day of the month for a date. The output of this function is a numeric value between 1 and 31. The syntax for obtaining the day of the month is:</p> <pre>SELECT DAYOFMONTH(date);</pre>	<p>For example, to extract the day of the month for the date 2005-03-18, enter the following command at the command prompt:</p> <pre>SELECT DAYOFMONTH('2005-03-18');</pre> <p>The output of this function is:</p> <p>18</p>
DAYOFWEEK	<p>The DAYOFWEEK function returns the day of the week for a date. The output for this function is a numeric value between 1 and 7, where 1 is Sunday and 7 is Saturday. The syntax to display the day of the week is:</p> <pre>SELECT DAYOFWEEK(date);</pre>	<p>For example, to display the day of the week for a particular date, enter the following command at the command prompt:</p> <pre>SELECT DAYOFWEEK('1995-06-07');</pre> <p>The output of this function is:</p> <p>4</p>

Session 14

Using Basic Functions in MySQL - II

Concepts

Name	Description	Example
DAYOFYEAR	<p>The DAYOFYEAR function returns the day of the year for a date. The output of this function is a numeric value between 1 and 366. The syntax for obtaining the day of the month is:</p> <pre>SELECT DAYOFYEAR(date);</pre>	<p>For example, to display the day of the year for a particular date, enter the following command at the command prompt:</p> <pre>SELECT DAYOFYEAR('2001-10-21');</pre> <p>The output of this function is:</p> <p>294</p>
EXTRACT	<p>The EXTRACT function extracts a particular part from a given date. The syntax for this function is:</p> <pre>SELECT EXTRACT(unit from date);</pre> <p>where,</p> <p>date – defines a valid date expression</p> <p>unit – specifies any valid unit, such as year, month, second, minute, or day</p>	<p>For example, to display only the year from a particular date, enter the following command at the command prompt:</p> <pre>SELECT EXTRACT(YEAR FROM '2008-04-06');</pre> <p>The output of this function is:</p> <p>2008</p>
FROM_DAYS	<p>The FROM_DAYS function returns the date of the number specified as an argument. This function works only for dates after the advent of the Gregorian calendar (1582). The syntax for this function is:</p> <pre>SELECT FROM_DAYS(argument);</pre>	<p>For example, to display the date for 5,00,000, enter the following command at the command prompt:</p> <pre>SELECT FROM_DAYS(500000);</pre> <p>The output of this function is:</p> <p>1368-12-14</p>
LAST_DAY	<p>The LAST_DAY function returns the last date of the month for the argument. The output is a numeric value ranging from 1 to 31, if the argument is valid. However, if the argument is invalid, the output is NULL. The syntax for using this function is:</p> <pre>SELECT LAST_DAY(expression);</pre>	<p>For example, enter the following command at the command prompt:</p> <pre>SELECT LAST_DAY('2008-02-05');</pre> <p>The output of this function is:</p> <p>2008-02-29</p>

Session 14

Using Basic Functions in MySQL - II

Concepts

Name	Description	Example
MICROSECOND	<p>The MICROSECOND function returns the number of microseconds in the argument specified as a datetime value. The output of this function is a numeric value ranging from 0 to 999999. The syntax for this function is:</p> <pre>SELECT MICROSECOND(expression);</pre>	<p>For example, to display the microseconds from the datetime expression 11:45:45.000123, enter the following command at the command prompt:</p> <pre>SELECT MICROSECOND (‘11:45:45.000123’);</pre> <p>The output of this function is:</p> <p>123</p>
PERIOD_ADD	<p>The PERIOD_ADD function adds a specified number of months to a given period expressed either as YYMM or YYYYMM. The final output is displayed in the form of YYYYMM. The syntax for this function is:</p> <pre>SELECT PERIOD_ADD(P, N);</pre> <p>where,</p> <p>PERIOD_ADD – adds the specified number of months</p> <p>P – defines the specified period</p> <p>N – defines the number of months to be added</p>	<p>For example, to add two months to the period 200705, enter the following command at the command prompt:</p> <pre>SELECT PERIOD_ADD(‘200705’, 02);</pre> <p>The output of this function is:</p> <p>200707</p>
PERIOD_DIFF	<p>The PERIOD_DIFF function calculates the difference in number of months between two periods. The final output is a numeric value. The syntax for this function is:</p> <pre>SELECT PERIOD_DIFF(P1, P2);</pre> <p>where,</p> <p>PERIOD_DIFF – calculates the number of months between two periods</p>	<p>For example, to calculate the number of months between 199608 and 199503, enter the following command at the command prompt:</p> <pre>SELECT PERIOD_DIFF(‘199608’, ‘199503’);</pre> <p>The output of this function is:</p> <p>17</p>

Session 14

Name	Description	Example
PERIOD_DIFF	P1 - stands for Period1 P2 - stands for Period2	
QUARTER	The QUARTER function returns the quarter number for the argument specified as a date value. The output of this function is a numeric value ranging from 1 - 4. The syntax for this function is: SELECT QUARTER(expression);	For example, to calculate the quarter for the date 1987-12-24, enter the following command at the command prompt: SELECT QUARTER('1987-12-24'); The output of this function is: 4
STR_TO_DATE	The STR_TO_DATE function converts a string into a specified format. The output of this function can be a date, time, or a datetime value depending upon the format. The syntax of this function is: SELECT STR_TO_DATE(String, format);	To convert the string 01,05,2013 into the YYYY-MM-DD format, enter the following command at the command prompt: SELECT STR_TO_DATE('01,05,2013', '%m, %d, %Y'); The output of this function is: 2013-01-05
SUBTIME	The SUBTIME function subtracts a specified time interval from the given datetime or time value. The syntax for this function is: SELECT SUBTIME(expr1, expr2); where, SUBTIME – subtracts a specified time interval expr1 – defines the starting datetime or time value expr2 – specifies the time value to be subtracted	To subtract 45 minutes from the current date, enter the following command at the command prompt: SELECT SUBTIME(NOW(), '00:45:00'); The output of this function will be a timestamp of 45 minutes prior to the current time.

Session 14

Using Basic Functions in MySQL - II

Concepts

Name	Description	Example
TIME_FORMAT	<p>The TIME_FORMAT function displays the time in a specific format.</p> <p>The syntax for this function is:</p> <pre>SELECT TIME_FORMAT(time, format);</pre> <p>where,</p> <p>time – defines a valid time</p> <p>format – defines the output to display the time</p>	<p>To display the current time in the HH-MM-SS format, enter the following command at the command prompt:</p> <pre>SELECT TIME_FORMAT(NOW(), '%H-%I-%S');</pre> <p>The output of this function will be the current time in HH-MM-SS format.</p>
TIME_TO_SEC	<p>The TIME_TO_SEC function converts a time argument into seconds. The output of this function is a numeric value and the syntax for this function is:</p> <pre>SELECT TIME_TO_SEC(expression);</pre>	<p>To convert the time expression 23:45:02 into seconds, enter the following command at the command prompt:</p> <pre>SELECT TIME_TO_SEC('23:45:02');</pre> <p>The output of this function is:</p> <p>85502</p>
TIMESTAMP	<p>The TIMESTAMP function returns the datetime expression of the specified argument. When you specify only a single argument, the output is a datetime value of only that argument. If you specify two arguments, the output would be the sum of the two expressions.</p> <p>The syntax for this function is:</p> <pre>SELECT TIMESTAMP(expression);</pre>	<p>For example, to display the current timestamp value, enter the following command at the command prompt:</p> <pre>SELECT TIMESTAMP(NOW());</pre> <p>The output of this function will be the current date and time.</p>
TO_DAYS	<p>The TO_DAYS function converts a datetime value into a numeric value, which is the day number. It starts counting the number of days after the advent of the Gregorian calendar (1582). The syntax for this function is:</p> <pre>SELECT TO_DAYS(expression);</pre>	<p>For example, to calculate the day number for the date 1982-06-24, enter the following command at the command prompt:</p> <pre>SELECT TO_DAYS('1982-06-24');</pre>

Session 14

Using Basic Functions in MySQL - II

Name	Description	Example
TO_DAYS		The output of this function will be: 724085
TO_SECONDS	The TO_SECONDS function converts the argument into seconds. The argument is a datetime value and the output is a numeric value. The syntax for this function is: SELECT TO_SECONDS(expression);	To convert 1975-02-25 into seconds, enter the following command at the command prompt: SELECT TO_SECONDS('1975-02-25'); The output of this function is: 62329737600
UTC_DATE	The UTC_DATE function returns the current UTC date as a value in YYYY-MM-DD or YYYYMMDD format, depending on whether the function is used in a string or numeric context. The syntax for this function is: SELECT UTC_DATE(expression);	For example, to retrieve the current UTC date, enter the following command at the command prompt: SELECT UTC_DATE(); The output of this function will be the current UTC date.
UTC_TIME	The UTC_TIME function returns the current UTC time as a value in HH:MM:SS or HHMMSS.uuuuuu format, depending on whether the function is used in a string or numeric context. The syntax for this function is: SELECT UTC_TIME(expression);	For example, to retrieve the current UTC time, enter the following command at the command prompt: SELECT UTC_TIME(); The output of this function will be the current UTC time.
UTC_TIMESTAMP	The UTC_TIMESTAMP function returns the current UTC date and time as a value in YYYY-MM-DD HH:MM:SS or YYYYMMDDHHMMSS.uuuuuu format, depending on whether the function is used in a string or numeric context. The syntax for this function is: SELECT UTC_TIMESTAMP(expression);	For example, to retrieve the current UTC date and time, enter the following command at the command prompt: SELECT UTC_TIMESTAMP(); The output of this function will be the current date and time.
WEEKDAY	The WEEKDAY function returns the weekday index for the argument. The output of this function is a numeric value between 0 to 6	The output of this function will be a numerical value corresponding to the weekday.

Session 14

Using Basic Functions in MySQL - II

Concepts

Name	Description	Example
WEEKDAY	where, 0 represents Monday and 6 represents Sunday. The syntax for this function is: SELECT WEEKDAY(expression);	
YEARWEEK	The YEARWEEK function returns the year and the week number of the date specified in the argument. The syntax for this function is: SELECT YEARWEEK(expression);	For example, to retrieve the year and week for the date 1982-12-29, enter the following command at the command prompt: SELECT YEARWEEK('1982-12-29'); The output of this function is: 198252

Table 14.2: Additional Date Functions in MySQL

14.3 String Functions in MySQL

MySQL provides string functions that work with the string data type. The string functions consider the specified arguments as strings even if you enter numeric data.

14.3.1 CHAR Function

The CHAR function interprets the values specified in the argument as integers. This function returns a string that contain characters returned by the code value of the integers. If the arguments contain NULL values, then they are skipped. The syntax to retrieve the string of characters is:

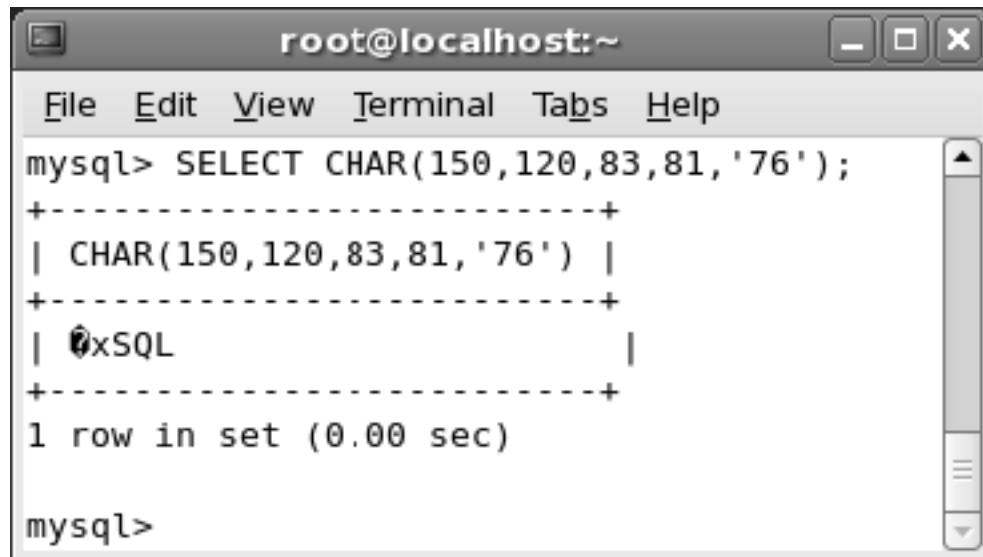
```
SELECT CHAR(N, . . .);
```

For example, to retrieve the code value for the integers 150, 120, 83, 81, '76', enter the following command at the command prompt:

```
SELECT CHAR(150, 120, 83, 81, '76');
```

Figure 14.24 displays the output of the command.

Session 14



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The main area displays the following MySQL command and its output:

```
mysql> SELECT CHAR(150,120,83,81,'76');
+-----+
| CHAR(150,120,83,81,'76') |
+-----+
| 0xSQL                      |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 14.24: CHAR Function

14.3.2 CHARACTER_LENGTH Function

The `CHARACTER_LENGTH` function also known as `CHAR_LENGTH` function returns the number of characters present in the string. This function counts a multibyte character as a single character. The syntax for this function is:

```
SELECT CHARACTER_LENGTH(expression);
```

For example, to calculate the number of characters present in the string `abc`, enter the following command at the command prompt:

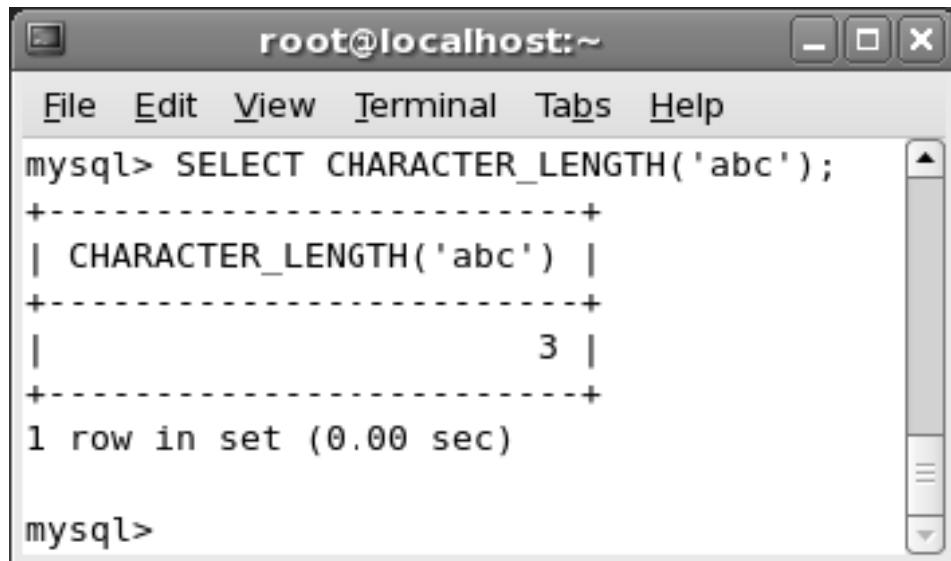
```
SELECT CHARACTER_LENGTH('abc');
```

Figure 14.25 displays the output of the command.

Session 14

Using Basic Functions in MySQL - II

Concepts



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its output:

```
mysql> SELECT CHARACTER_LENGTH('abc');
+-----+
| CHARACTER_LENGTH('abc') |
+-----+
|                      3 |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 14.25: CHARACTER_LENGTH Function

14.3.3 CONCAT Function

The CONCAT function returns a string after joining the specified arguments. This function returns a NULL value if NULL arguments are specified. The syntax for this function is:

```
SELECT CONCAT(STR1,STR2,...);
```

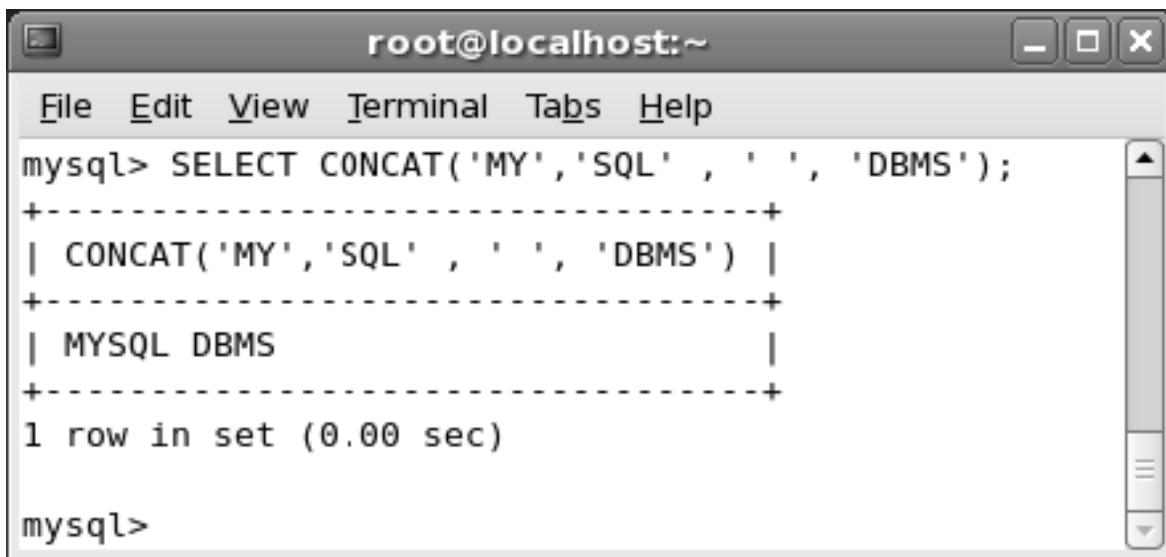
To concatenate the strings My, SQL, DBMS, enter the following command at the command prompt:

```
SELECT CONCAT('My','SQL','',' ','DBMS');
```

Figure 14.26 displays the output of the command.

Session 14

Using Basic Functions in MySQL - II



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its output:

```
mysql> SELECT CONCAT('MY', 'SQL' , ' ', 'DBMS');
+-----+
| CONCAT('MY', 'SQL' , ' ', 'DBMS') |
+-----+
| MYSQL DBMS                         |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 14.26: CONCAT Function

14.3.4 INSERT Function

The `INSERT` function adds a new string to the existing string at a specified position. The output of the function is also a string. The syntax for the `INSERT` function is:

```
INSERT (str, pos, len, newstr);
```

where,

`INSERT` – adds a new string

`str` – specifies the existing string

`pos` – defines the starting position of the new string

`len` – specifies the length of the new string

`newstr` – specifies the new string to be inserted

This function returns a string where the substring starting at position, `pos`, and of length, `len`, is replaced by the string `newstr`.

To insert `MATHEMATICAL` at position 6 of the string `MATHE`, enter the following command at the command prompt:

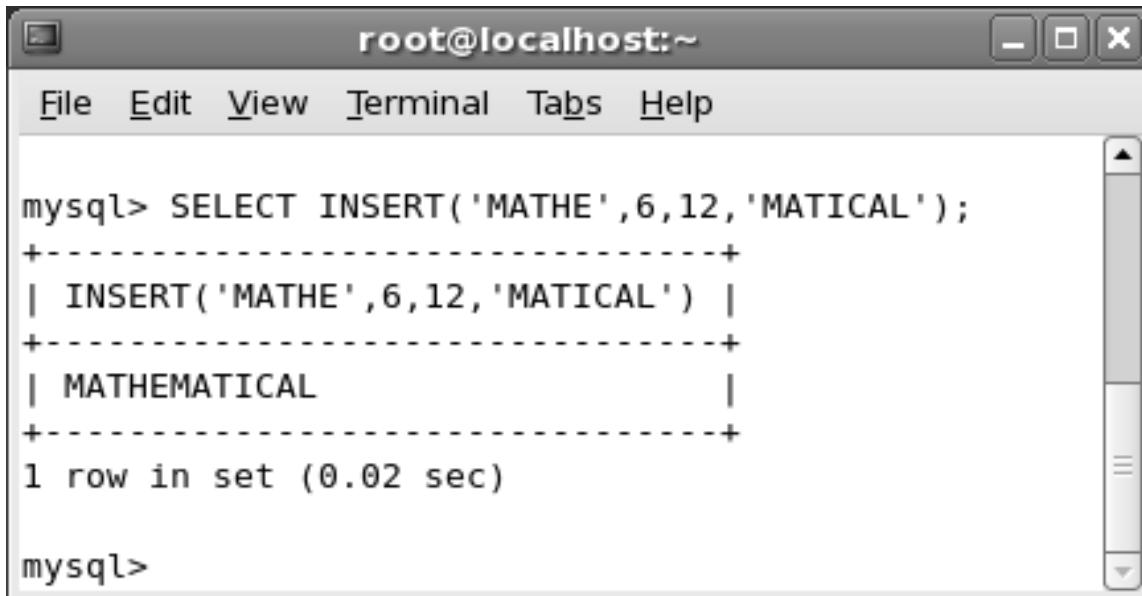
Session 14

Using Basic Functions in MySQL - II

Concepts

```
SELECT INSERT('MATHE', 6, 12, 'MATICAL');
```

Figure 14.27 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The main area displays the following MySQL query and its result:

```
mysql> SELECT INSERT('MATHE',6,12,'MATICAL');
+-----+
| INSERT('MATHE',6,12,'MATICAL') |
+-----+
| MATHEMATICAL                     |
+-----+
1 row in set (0.02 sec)

mysql>
```

Figure 14.27: INSERT Function

14.3.5 INSTR Function

The `INSTR` function works with two arguments, such as string and substring. It returns the position of the first instance of a substring in a string. The output of this function is a numeric value ranging from 0 to the length of the string. The syntax for this function is:

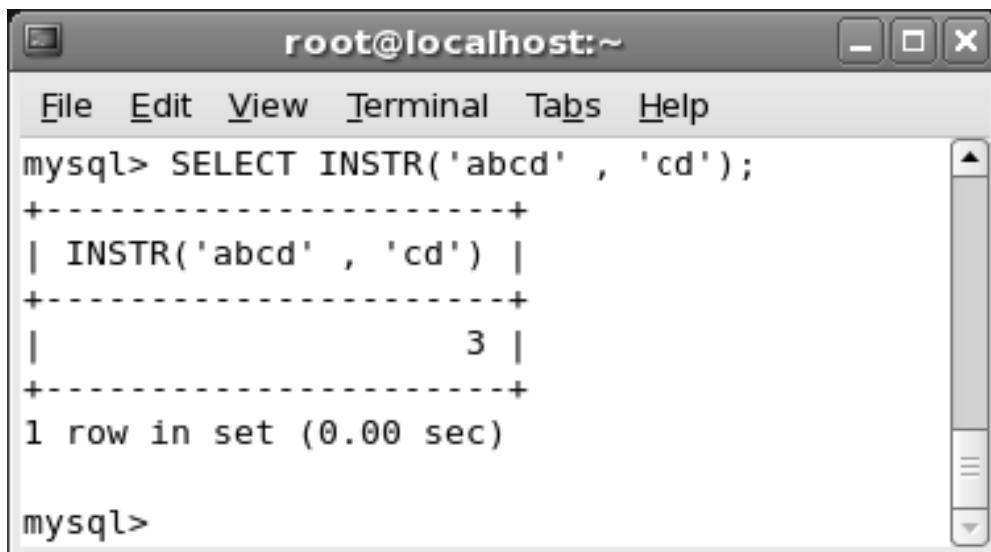
```
SELECT INSTR(string, substring);
```

For example, to retrieve the position of the string `cd` from the string `abcd`, enter the following command at the command prompt:

```
SELECT INSTR('abcd', 'cd');
```

Figure 14.28 displays the output of the command.

Session 14



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area displays a MySQL command and its output:

```
mysql> SELECT INSTR('abcd' , 'cd');
+-----+
| INSTR('abcd' , 'cd') |
+-----+
|            3          |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 14.28: INSTR Function

14.3.6 LCASE Function

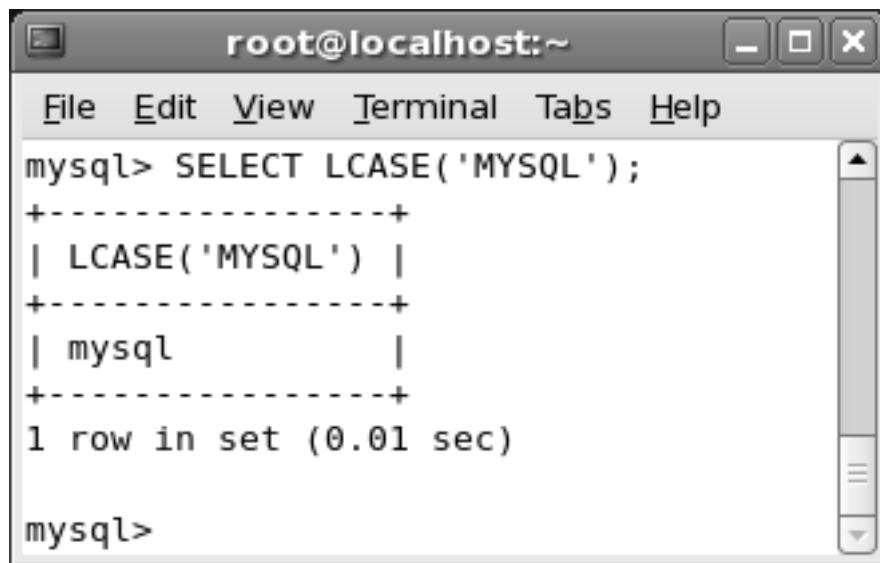
The LCASE function changes all the characters of the values specified in the argument to lowercase. This function does not work with binary strings. The syntax for this function is:

```
SELECT LCASE(expression);
```

To convert MYSQL to lowercase, enter the following command at the command prompt:

```
SELECT LCASE ('MYSQL');
```

Figure 14.29 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area displays the following MySQL command and its output:

```
mysql> SELECT LCASE('MYSQL');
+-----+
| LCASE('MYSQL') |
+-----+
| mysql          |
+-----+
1 row in set (0.01 sec)

mysql>
```

Figure 14.29: LCASE Function

14.3.7 LEFT Function

The `LEFT` function returns the leftmost characters of the values specified in the argument. The syntax for this function is:

```
SELECT LEFT('str', len);
```

where,

`LEFT` – returns the leftmost character

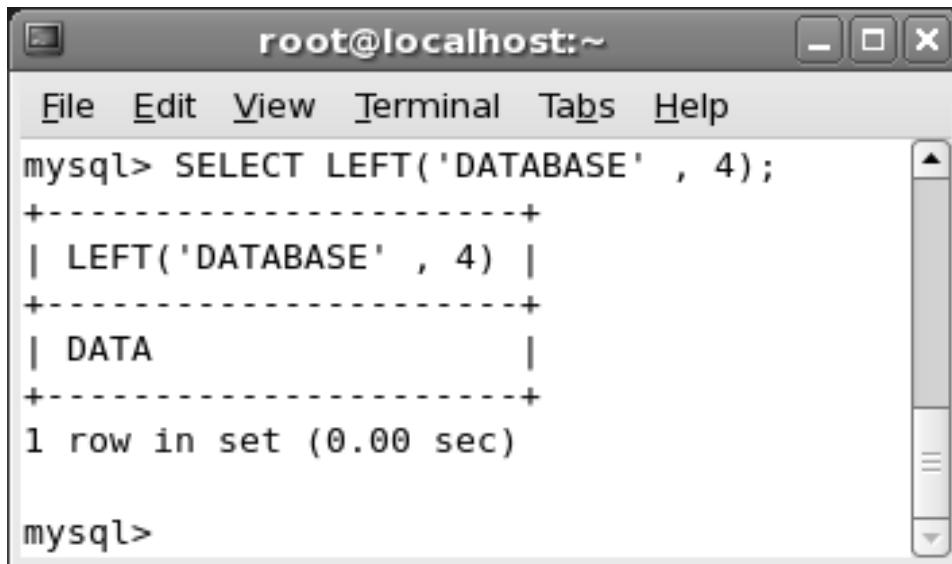
`str` – contains the original string

`len` – specifies the number of characters to be extracted from the leftmost end

For example, to obtain the first four character of the string `DATABASE`, enter the following command at the command prompt:

```
SELECT LEFT('DATABASE' , 4);
```

Figure 14.30 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area displays the following MySQL command and its output:

```
mysql> SELECT LEFT('DATABASE' , 4);
+-----+
| LEFT('DATABASE' , 4) |
+-----+
| DATA                |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 14.30: LEFT Function

14.3.8 LENGTH Function

The LENGTH function returns the length of a string. The syntax to calculate the length of a string is:

```
SELECT LENGTH(str);
```

For example, to calculate the length of the string MySQL, enter the following command at the command prompt:

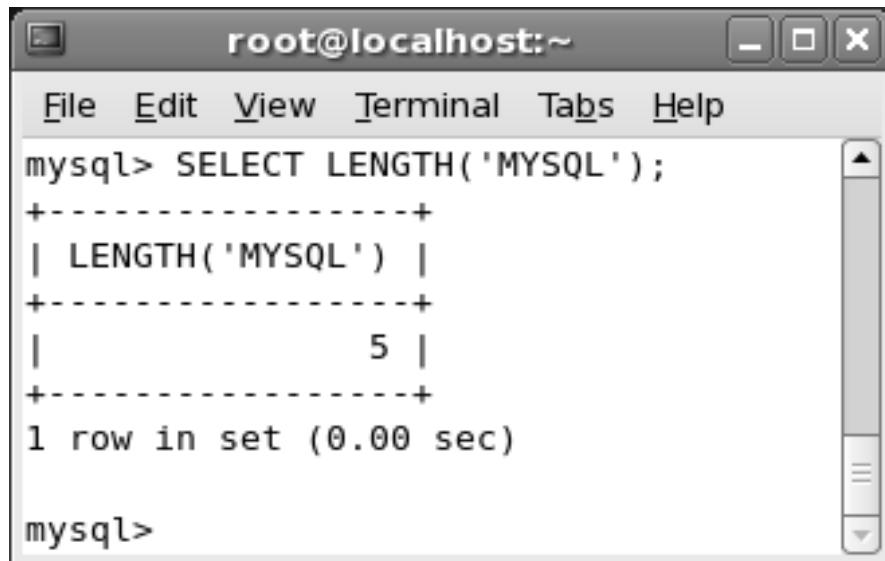
```
SELECT LENGTH('MySQL');
```

Figure 14.31 displays the output of the command.

Session 14

Using Basic Functions in MySQL - II

Concepts



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The main area displays the following MySQL command and its output:

```
mysql> SELECT LENGTH('MYSQL');
+-----+
| LENGTH('MYSQL') |
+-----+
|           5 |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 14.31: LENGTH Function

14.3.9 LOCATE Function

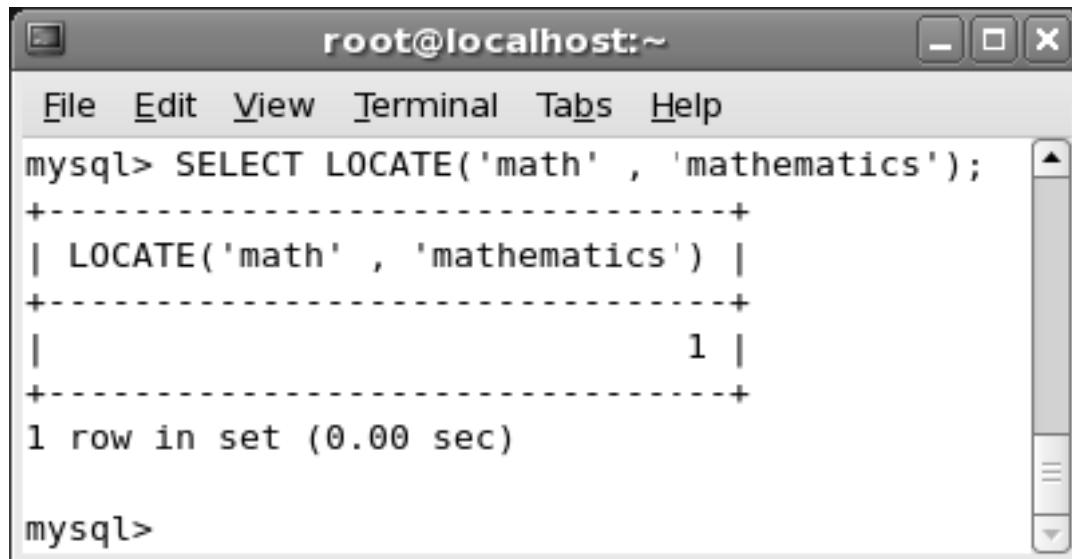
The LOCATE function returns the position of the first instance of a substring in a string. The syntax to retrieve the position of the first instance of a substring in a string is:

```
SELECT LOCATE(substr, str);
```

For example, to identify the position of the substring math in the string mathematics, enter the following command at the command prompt:

```
SELECT LOCATE('math', 'mathematics');
```

Figure 14.32 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its output:

```
mysql> SELECT LOCATE('math' , 'mathematics');
+-----+
| LOCATE('math' , 'mathematics') |
+-----+
|                               1 |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 14.32: LOCATE Function

14.3.10 REPEAT Function

The REPEAT function copies the given string N number of times. If N is less than 1, the output is an empty string. The syntax for this function is:

```
SELECT REPEAT(String, count);
```

where,

REPEAT – copies the string

String – specifies the string to copy

count – specifies the number of times to copy the string

To replicate the string DATA four times, enter the following command at the command prompt:

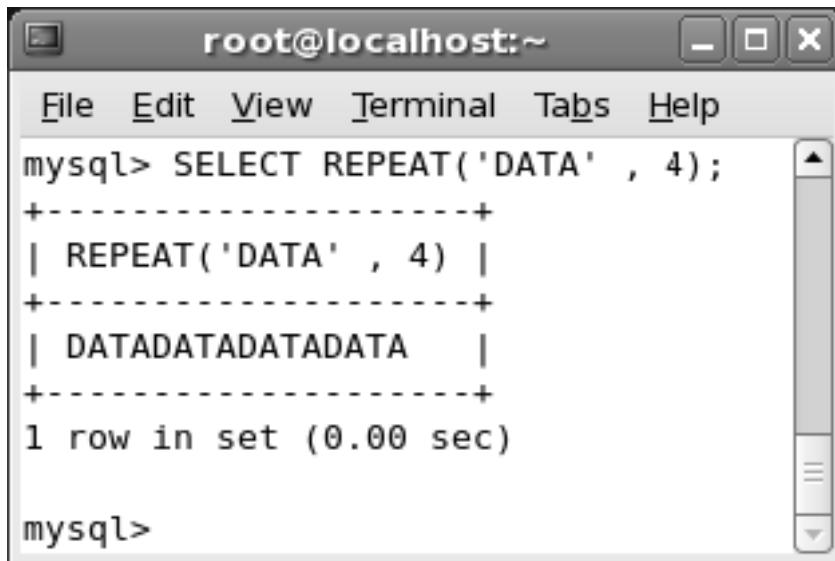
```
SELECT REPEAT('DATA', 4);
```

Figure 14.33 displays the output of the command.

Session 14

Using Basic Functions in MySQL - II

Concepts



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its output:

```
mysql> SELECT REPEAT('DATA' , 4);
+-----+
| REPEAT('DATA' , 4) |
+-----+
| DATADATADATADATA |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 14.33: REPEAT Function

14.3.11 REPLACE Function

The REPLACE function modifies a part or whole of the string with the new string. The syntax for this function is:

```
SELECT REPLACE(str, from_str, to_str);
```

where,

REPLACE – modifies the string

str – specifies the original string

from_str – defines the string to be replaced

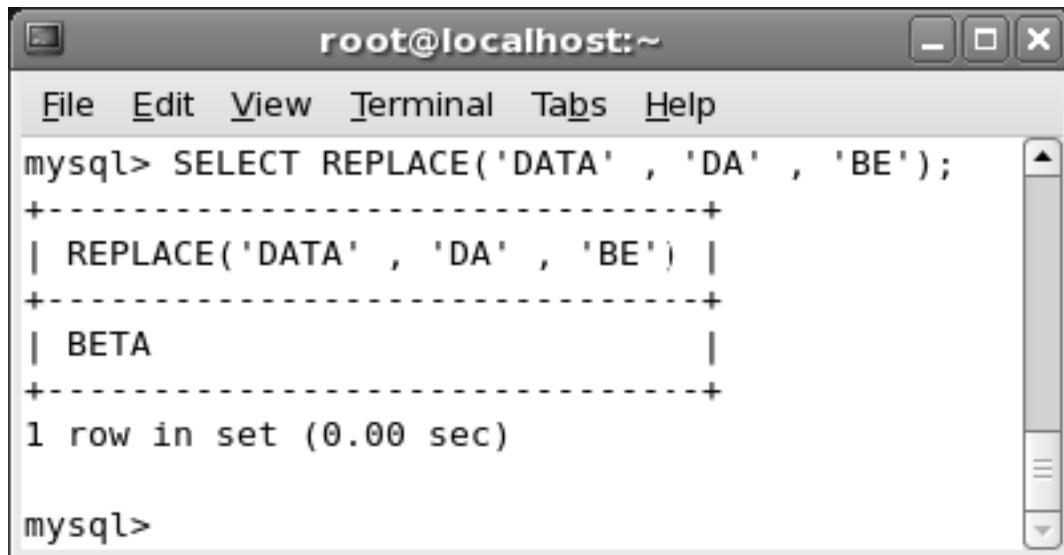
to_str – defines the new string to be inserted

Note: The REPLACE function is case-sensitive.

To change the string DATA to BETA, enter the following command at the command prompt:

```
SELECT REPLACE('DATA', 'DA', 'BE');
```

Figure 14.34 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its output:

```
mysql> SELECT REPLACE('DATA' , 'DA' , 'BE');
+-----+
| REPLACE('DATA' , 'DA' , 'BE') |
+-----+
| BETA
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 14.34: REPLACE Function

14.3.12 REVERSE Function

The REVERSE function inverts the sequence of characters in an argument. The syntax for this function is:

```
SELECT REVERSE(expression);
```

To change the order of characters of the string XYZ, enter the following command at the command prompt:

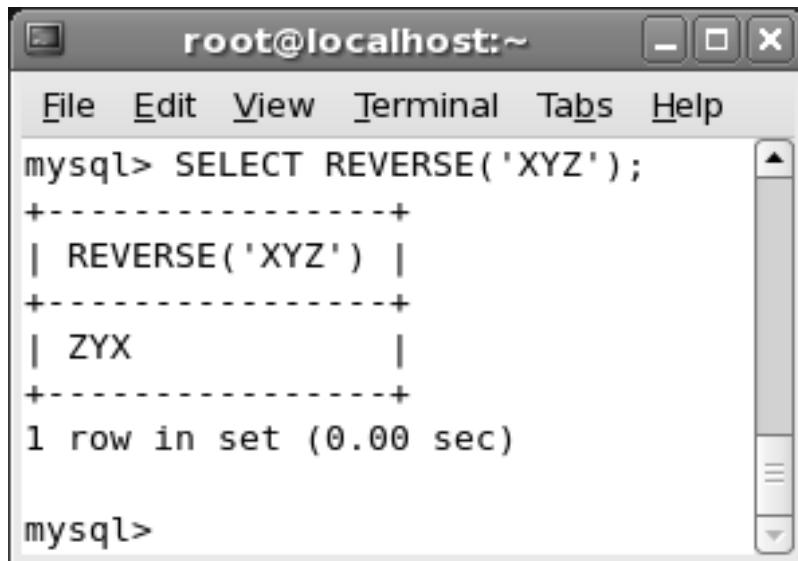
```
SELECT REVERSE('XYZ');
```

Figure 14.35 displays the output of the command.

Session 14

Using Basic Functions in MySQL - II

Concepts



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its output:

```
mysql> SELECT REVERSE('XYZ');
+-----+
| REVERSE('XYZ') |
+-----+
| ZYX           |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 14.35: REVERSE Function

14.3.13 RIGHT Function

The `RIGHT` function accepts two arguments, such as string and length. The function returns an output that contains string values starting from the right end of the input string. The syntax for this function is:

```
SELECT RIGHT(string, length);
```

where,

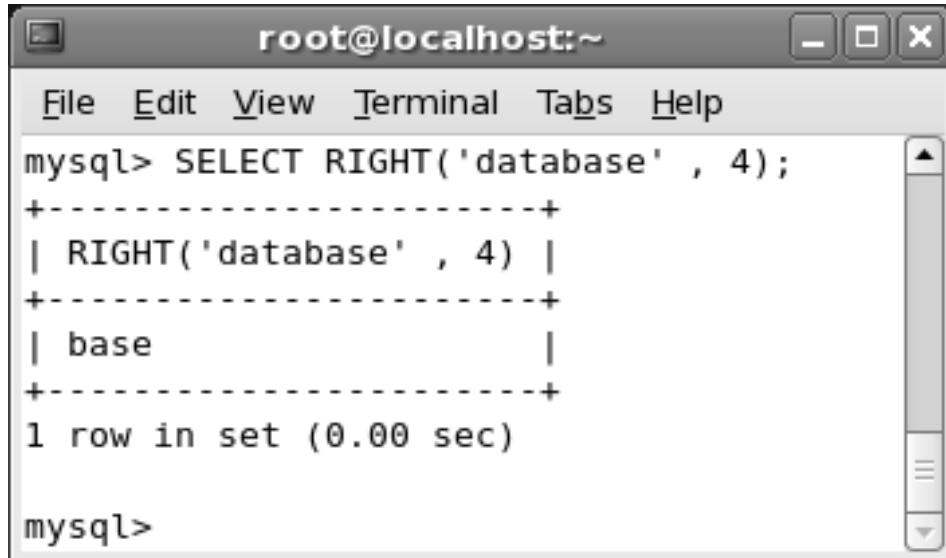
`string` – defines the given string

`length` – specifies the number of rightmost characters

For example, to obtain the last four characters of the string `database`, enter the following command at the command prompt:

```
SELECT RIGHT('database', 4);
```

Figure 14.36 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its output:

```
mysql> SELECT RIGHT('database' , 4);
+-----+
| RIGHT('database' , 4) |
+-----+
| base                |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 14.36: RIGHT Function

14.3.14 STRCMP Function

The STRCMP function accepts two strings as arguments. It compares the strings and returns a numeric value ranging from -1 to 1 in the output. The function returns 0 if the strings have same values. The output of the function is -1 when the first argument is smaller than the second based on the conditions specified in the sort order. This function returns 1 when the first argument is greater than the second based on the conditions specified in the sort order. The syntax for this function is:

```
SELECT STRCMP(str1, str2);
```

where,

str1, str2 – specify the strings to compare

If str1 = str2, the output is 0.

If str1 < str2, the output is -1.

If str1 > str2, the output is 1.

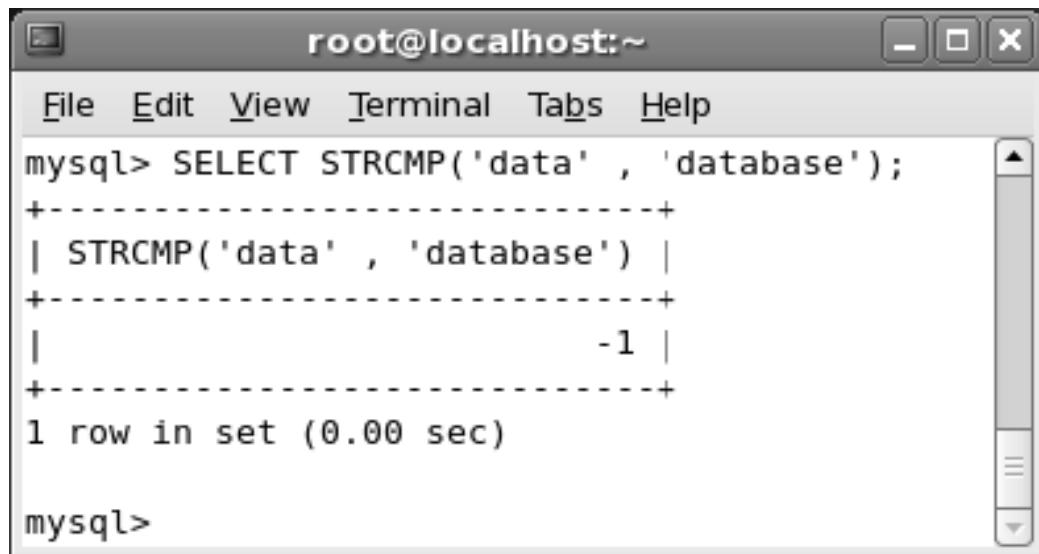
To compare two strings data and database, enter the following command at the command prompt:

```
SELECT STRCMP('data', 'database');
```

Session 14

Using Basic Functions in MySQL - II

Figure 14.37 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its output:

```
mysql> SELECT STRCMP('data' , 'database');
+-----+
| STRCMP('data' , 'database') |
+-----+
| -1 |
+-----+
1 row in set (0.00 sec)

mysql>
```

The output shows that the STRCMP function returns -1, indicating that 'data' is less than 'database' lexicographically.

Concepts

Figure 14.37: STRCMP Function

14.3.15 TRIM Function

The TRIM function removes a specified prefix or suffix from a given string. The syntax for this function is:

```
SELECT TRIM(location 'remstring' from 'string');
```

where,

location – specifies the starting position for deletion

remstring – specifies the string to be removed

string – specifies the original string

To remove the string `quant` from the string `quantitative`, enter the following command at the command prompt:

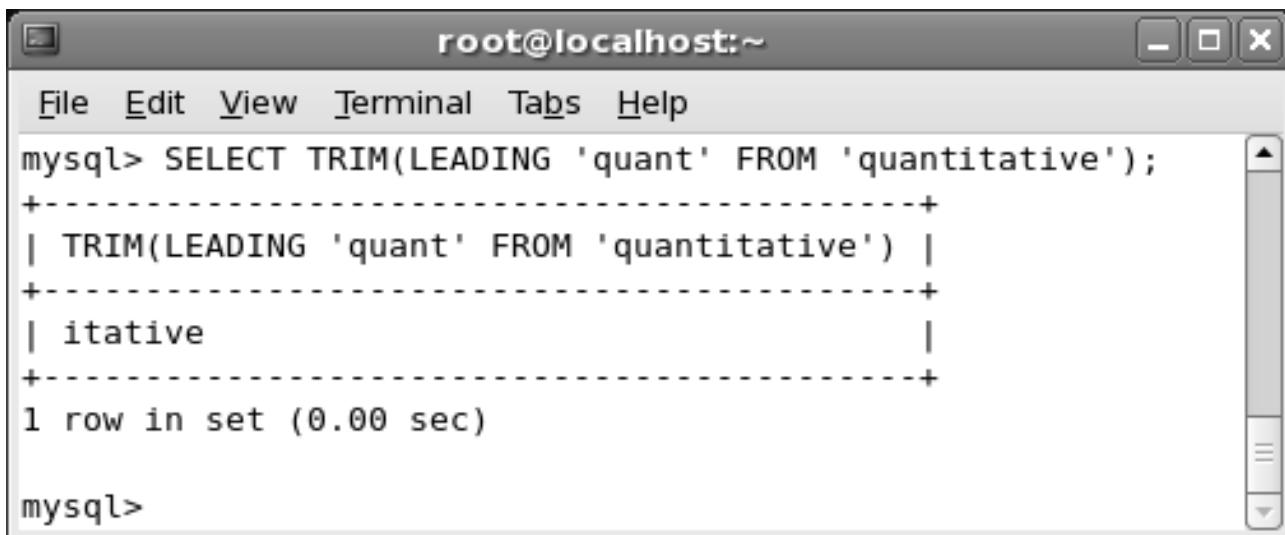
```
SELECT TRIM(LEADING 'quant' from 'quantitative');
```

Figure 14.38 displays the output of the command.

Session 14

Using Basic Functions in MySQL - II

Concepts



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its output:

```
File Edit View Terminal Tabs Help
mysql> SELECT TRIM(LEADING 'quant' FROM 'quantitative');
+-----+
| TRIM(LEADING 'quant' FROM 'quantitative') |
+-----+
| itative |
+-----+
1 row in set (0.00 sec)

mysql>
```

The command `SELECT TRIM(LEADING 'quant' FROM 'quantitative');` is executed, resulting in the output `itative`, which is the string 'quantitative' with the leading 'quant' removed.

Figure 14.38: TRIM Function

Note: In the syntax, if the location is not specified, then the string to be removed is deleted from both ends. If no remstr is specified, then whitespaces are removed.

14.3.16 UCASE Function

The UCASE function changes all the characters of the argument into uppercase. This function does not work with binary strings. The syntax for this function is:

```
SELECT UCASE(expression);
```

To convert database to uppercase, enter the following command at the command prompt:

```
SELECT UCASE('database');
```

Figure 14.39 displays the output of the command.

Session 14

Using Basic Functions in MySQL - II

Concepts

The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its output:

```
mysql> SELECT UCASE('database');
+-----+
| UCASE('database') |
+-----+
| DATABASE          |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 14.39: UCASE Function

Table 14.3 describes the additional String functions in MySQL.

Name	Description	Example
ASCII	The ASCII function returns the numeric value of the leftmost character of the string, specified as the argument. This function returns a value of 0 if you enter an empty string. This function is applicable for characters whose numeric value ranges from 0 to 255. The syntax to retrieve the ASCII value is: SELECT ASCII(string);	For example, to retrieve the ASCII value of the string JOHN, enter the following command at the command prompt: SELECT ASCII('JOHN'); The output of this function is: 74
BIN	The BIN function returns a string representation of the binary value of N, where N contains values of BIGINT data type. The syntax to retrieve the string representation is: SELECT BIN(expression);	For example, to retrieve the binary value of 55, enter the following command at the command prompt: SELECT BIN(55); The output of this function is: 110111
BIT_LENGTH	The BIT_LENGTH function returns the length of the string specified in the argument. The syntax to calculate the length of the string in bits is:	For example, to calculate the length of the string MYSQL in bits, enter the following command at the command prompt:

Session 14

Using Basic Functions in MySQL - II

Name	Description	Example
BIT_LENGTH	<pre>SELECT BIT_LENGTH(string);</pre>	<pre>SELECT BIT_LENGTH('MYSQL');</pre> The output of this function is: 40
COMPRESS	<p>The COMPRESS function compresses a string. The syntax to compress a string is:</p> <pre>SELECT COMPRESS(string);</pre> <p>The compressed string outputs are stored as follows:</p> <ul style="list-style-type: none">➤ Empty strings are stored as it is.➤ Non-empty strings are stored as a four-byte length of the uncompressed string (low byte first), followed by the compressed string.➤ If the string contains a space at the end, the space is replaced by the '.' character to resolve trim errors while storing data in a CHAR or VARCHAR column.	<p>To compress the string XYZ and obtain the length of the compressed string, enter the following command at the command prompt:</p> <pre>SELECT LENGTH(COMPRESS('XYZ'));</pre> <p>The output of this function is:</p> <p>15</p>
CONCAT_WS	<p>The CONCAT_WS function is similar to the CONCAT() function. However, it allows you to add a separator between the arguments. The syntax for this function is:</p> <pre>CONCAT_WS(separator, str1, str2, ...);</pre>	<p>To join the strings My and SQL with a comma as the separator, enter the following command at the command prompt:</p> <pre>SELECT CONCAT_WS(',', 'My', 'SQL');</pre> <p>The output of this function is:</p> <p>MY,SQL</p>
ELT	<p>The ELT function extracts the value of the string whose position is specified in the argument. The position is a numeric value. The syntax for using this function is:</p>	<p>Enter the following command at the command prompt:</p> <pre>SELECT ELT(1, 'a', 'b', 'c');</pre>

Session 14

Using Basic Functions in MySQL - II

Concepts

Name	Description	Example
ELT	<pre>SELECT ELT(N, str1, str2, ...);</pre> <p>where,</p> <p>N - specifies the numeric position of the value to be searched</p> <p>str1, str2 - defines the strings passed as arguments</p>	The output of this function is: a Note: For the ELT function, the numeric value specified must be between 1 and the number of strings present in the given argument, else the output is NULL.
FIELD	<p>The FIELD function returns the position of a particular string in a given set of strings specified in the argument. If the string is not found, this function returns a value of zero. The syntax for this function is:</p> <pre>SELECT FIELD(string1,string2, string3....);</pre> <p>where,</p> <p>string1 – represents the string to search for</p> <p>string2, string3 – list of strings to be searched from</p>	To find the index of string C in the strings A, B, C, D, and E, enter the following command at the command prompt:
		<pre>SELECT FIELD('C', 'A', 'B', 'C', 'D', 'E');</pre> <p>The output of this function is:</p> 3
FIND_IN_SET	<p>The FIND_IN_SET function returns the position of the character present in the string list passed as an argument. A string list contains strings that are separated by a comma. The output of this function is a numeric value between 0 and N, where N is the number of substrings in the string list. If either of the argument is NULL, the final result is also NULL. The syntax for this function is:</p> <pre>SELECT FIND_IN_SET(expression);</pre>	For example, to retrieve the index value of B in the string B,Y,E, enter the following command at the command prompt:
		<pre>SELECT FIND_IN_SET('B', 'B,Y,E');</pre> <p>The output of this function is:</p> 1
HEX	The HEX function displays the values specified in the argument in its hexadecimal format. The syntax for this function is:	For example, to retrieve the hexadecimal value of MYSQL, enter the following command at the command prompt:

Session 14

Using Basic Functions in MySQL - II

Name	Description	Example
HEX	<code>SELECT HEX(expression);</code>	<code>SELECT HEX('MYSQL');</code> The output of this function is: 4D5953514C
LOWER	The <code>LOWER</code> function changes the given string into lowercase. The syntax to change the case for a string is: <code>SELECT LOWER(str);</code>	For example, to convert the string <code>PETER</code> to lowercase, enter the following command at the command prompt: <code>SELECT LOWER('PETER');</code> The output of this function is: peter
LTRIM	The <code>LTRIM</code> function removes the whitespaces before the values specified in the argument. The syntax for this function is: <code>SELECT LTRIM(expression);</code>	To delete the whitespaces before ' <code>xyz</code> ', enter the following command at the command prompt: <code>SELECT LTRIM(' xyz');</code> The output of this function is: xyz
ORD	The <code>ORD</code> function returns the ASCII value of the first character for the string specified in the argument. The arguments should be a string. The syntax for this function is: <code>SELECT ORD(expression);</code>	To display the ASCII value of the first character from the string <code>BAR</code> , enter the following command at the command prompt: <code>SELECT ORD('BAR');</code> The output of this function is: 66
RTRIM	The <code>RTRIM</code> function deletes the spaces after the values specified in the argument. The syntax for this function is: <code>SELECT RTRIM(expression);</code>	To remove the trailing space from the string ' <code>SPACE </code> ', enter the following command at the command prompt: <code>SELECT RTRIM('SPACE ');</code> The output of this function is: SPACE

Session 14

Using Basic Functions in MySQL - II

Concepts

Name	Description	Example
SPACE	<p>The SPACE function returns a string that contains only space characters. You can specify the number of space characters as a numeric value in the argument. The syntax for this function is:</p> <pre>SELECT SPACE(N);</pre> <p>where,</p> <p>N – defines the number of spaces</p>	<p>For example, to obtain a string with seven spaces, enter the following command at the command prompt:</p> <pre>SELECT SPACE(7);</pre> <p>The output of this function will be a string of 7 spaces.</p>
UNHEX	<p>The UNHEX function interprets each character of the string in the hexadecimal format and returns the character as the output. The syntax for this function is:</p> <pre>SELECT UNHEX(expression);</pre>	<p>For example, to obtain the characters for the hexadecimal string 'AB', enter the following command at the command prompt:</p> <pre>SELECT UNHEX('AB');</pre> <p>The output of this function is:</p> <p>?</p> <p>Note: For the UNHEX function, if the argument is not in the hexadecimal format (0 to 9, A to F, a to f), then the final output is NULL.</p>

Table 14.3: Additional String Functions in MySQL

14.4 System Information Functions in MySQL

System functions provide information about the system. MySQL provides certain system functions that return system-related information such as, connection ID, current users and so on.

14.4.1 CHARSET Function

The CHARSET function returns the character set of the argument. The argument entered must be a string argument. The syntax for obtaining the character set of a string is:

```
SELECT CHARSET(str);
```

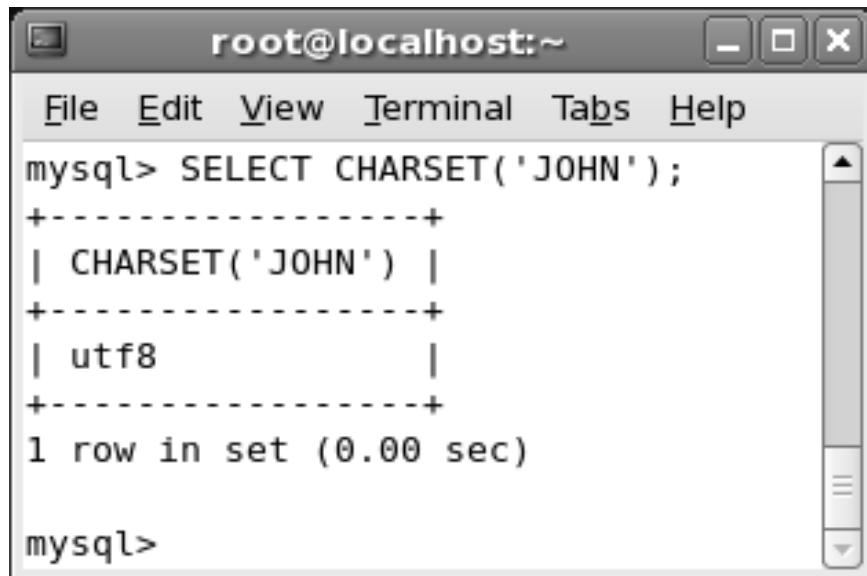
Session 14

Using Basic Functions in MySQL - II

For example, to obtain the character set of ‘John’, enter the following command at the command prompt:

```
SELECT CHARSET('JOHN');
```

Figure 14.40 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following text:

```
mysql> SELECT CHARSET('JOHN');
+-----+
| CHARSET('JOHN') |
+-----+
| utf8           |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 14.40: CHARSET Function

14.4.2 CURRENT_USER Function

The CURRENT_USER function returns the username and the hostname of the current session. The syntax to view the hostname and the username is:

```
SELECT CURRENT_USER;
```

For example, to obtain the username of the current account, enter the following command at the command prompt:

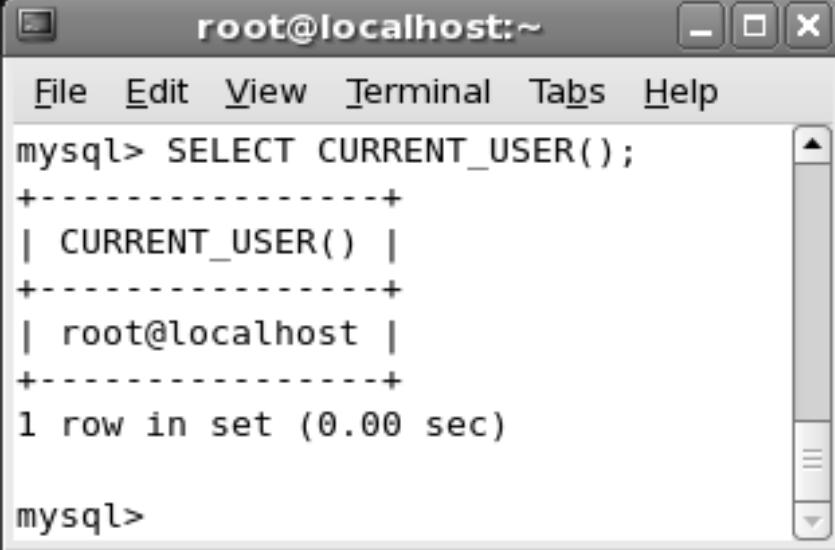
```
SELECT CURRENT_USER;
```

Figure 14.41 displays the output of the command.

Session 14

Using Basic Functions in MySQL - II

Concepts



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area displays the following MySQL command and its output:

```
mysql> SELECT CURRENT_USER();
+-----+
| CURRENT_USER() |
+-----+
| root@localhost |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 14.41: CURRENT_USER Function

14.4.3 DATABASE Function

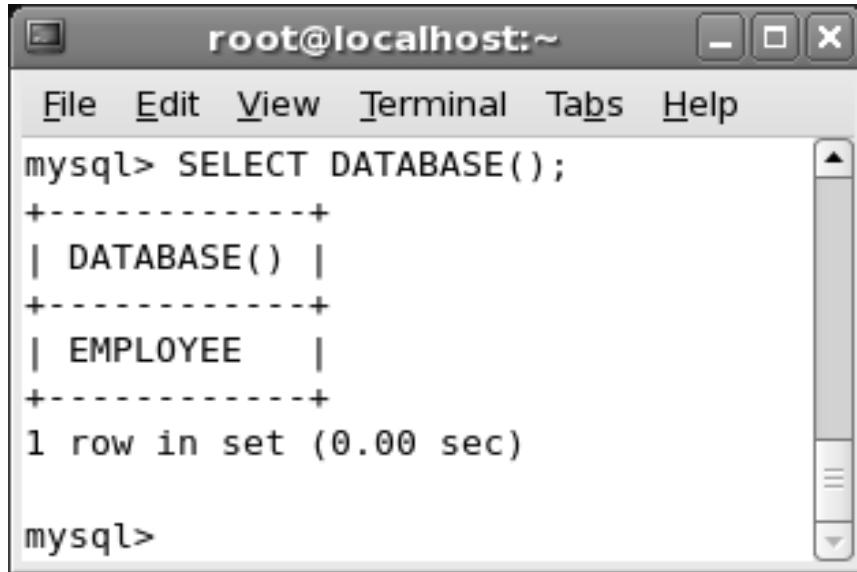
The DATABASE function returns the name of the current database. If no database is activated, then this function returns a NULL value. The syntax to view the name of the current database is:

```
SELECT DATABASE();
```

For example, to retrieve the name of the current database, enter the following command at the command prompt:

```
SELECT DATABASE();
```

Figure 14.42 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its output:

```
mysql> SELECT DATABASE();
+-----+
| DATABASE() |
+-----+
| EMPLOYEE   |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 14.42: DATABASE Function

14.4.4 ROW_COUNT Function

The `ROW_COUNT` function returns the number of rows inserted, updated, or deleted after the `INSERT`, `UPDATE`, or `DELETE` statements are executed. The number of affected-rows returned in an `UPDATE` statement is the number of rows actually changed. In `REPLACE` statement, the number of affected rows is two if the new row replaces an old row, because, the `REPLACE` statement inserts one row after deleting the duplicate row. In case of an `INSERT` statement, the number of rows affected is one if a new row is inserted and the number of rows affected is two if an existing row is updated. By default, MySQL displays the row count stored in the `mysql_affected_rows()` function when the `ROW_COUNT()` function is executed.

The syntax for this function is:

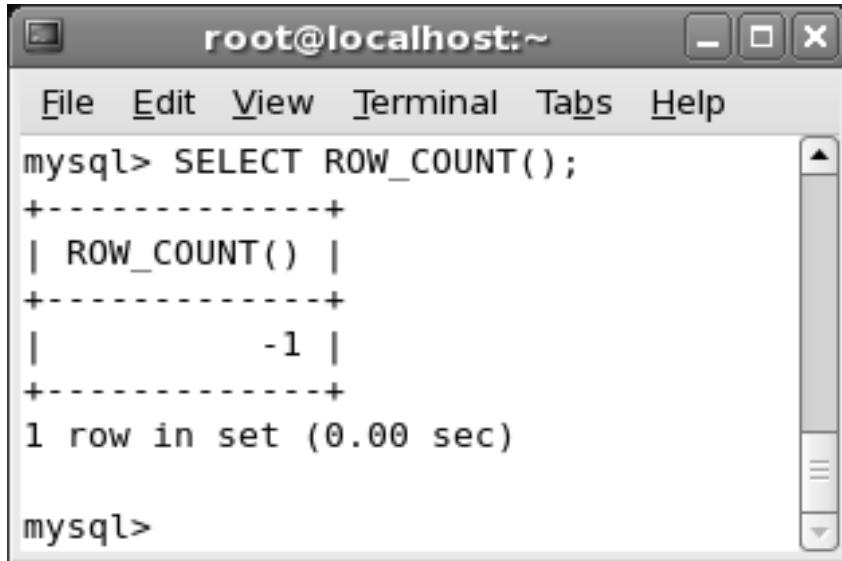
```
SELECT ROW_COUNT();
```

Figure 14.43 displays the output of the command.

Session 14

Using Basic Functions in MySQL - II

Concepts



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its output:

```
mysql> SELECT ROW_COUNT();
+-----+
| ROW_COUNT() |
+-----+
|          -1 |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 14.43: ROW_COUNT Function

14.4.5 VERSION Function

The VERSION function returns the version of MySQL. The syntax to view the version of MySQL is:

```
SELECT VERSION();
```

For example, to view the version of MySQL, enter the following command at the command prompt:

```
SELECT VERSION();
```

Figure 14.44 displays the output of the command.

Session 14

Using Basic Functions in MySQL - II

```
root@localhost:~
```

File Edit View Terminal Tabs Help

```
mysql> SELECT VERSION();
+-----+
| VERSION()      |
+-----+
| 5.1.56-community |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 14.44: VERSION Function

Table 14.4 displays the description and use of other system information functions.

Name	Description	Example
CONNECTION_ID	The CONNECTION_ID function returns the thread ID for the connection. MySQL assigns a unique ID for every thread or connection. The syntax to retrieve the thread ID is: SELECT CONNECTION_ID();	For example, to obtain the thread ID for the current connection, enter the following command at the command prompt: SELECT CONNECTION_ID(); The output of this function is: 2
ROW_COUNT	The ROW_COUNT function returns the number of rows inserted, updated, or deleted after the INSERT, UPDATE, or DELETE statements are executed. The syntax for this function is: SELECT ROW_COUNT();	Execute the following command and view the output. SELECT ROW_COUNT(); The output of this function is: -1

Session 14

Using Basic Functions in MySQL - II

Concepts

Name	Description	Example
USER	The USER function returns the current username and hostname. The syntax to view the username is: SELECT USER();	For example, to obtain the current username and hostname, enter the following command at the command prompt: SELECT USER();
USER		The output of this function is: root@localhost.com

Table 14.4: Additional System Information Functions in MySQL



Summary

- In SQL queries, functions can be used in place of the column name and in the WHERE clause.
- Date functions can execute computations on date, time, and datetime data types. These functions can add and subtract the date, extract a part of the date, and convert a date from one format to another. Some examples of date functions are ADDDATE, YEAR, DATEDIFF, and CONVERT _ TZ.
- String functions operate on string data type. They can be used to retrieve the ASCII value of a string, calculate the number of bits in a string, join two or more strings with or without the separator, convert the given string to lowercase or uppercase, and calculate the hexadecimal value of string. Some examples of STRING functions are ASCII, CHARACTER _ LENGTH, BIT _ LENGTH, LCASE, and UCASE.
- System information functions display system-related information. The version number of the database, name of the current user and host, and number of rows affected due to the preceding INSERT, UPDATE, or DELETE statement can be retrieved using system information functions. Some examples of system information functions are BENCHMARK, CHARSET, USER, and VERSION.



Check Your Progress

1. Which of the following function returns the numeric value of the leftmost character of the string specified in the argument?
 - a. CHAR
 - b. ASCII
 - c. MOD
 - d. BIN

2. Which of the following function returns the index of the string in the other strings specified in the argument?
 - a. FIELD
 - b. CHAR
 - c. BIN
 - d. BIT_LENGTH

3. Which of the following function returns the length of an argument in bits?
 - a. LENGTH
 - b. COMPRESS
 - c. BIT_LENGTH
 - d. COUNT

4. Diana wants to extract the time from a given expression. Which of the following functions will help Diana to achieve this?
 - a. TIME_FORMAT
 - b. TIME



Check Your Progress

- c. TIMEDIFF
 - d. TIMESTAMP
5. John wants to delete leading and trailing spaces before displaying the records from a table. Which of the following functions will help John to achieve this?
- a. LTRIM
 - b. TRIM
 - c. RTRIM
 - d. TRUNCATE
6. Which of the following function returns the value of the string whose position in the argument is passed as a numeric value?
- a. ELT
 - b. FIELD
 - c. ORD
 - d. FIND_IN_SET
7. John wants to view the username and the hostname. Which of the following functions will help John to display the current username and hostname?
- a. SCHEMA
 - b. DATABASE
 - c. USER
 - d. VERSION

Session 14

Using Basic Functions in MySQL - II



Check Your Progress

Concepts

8. Which of the following function returns the name of the month for the date entered as the argument?
 - a. MONTH
 - b. MONTHNAME
 - c. DAYOFMONTH
 - d. LAST_DAY

9. Which of the following function changes all the characters of the argument into uppercase?
 - a. LOWER
 - b. LCASE
 - c. UPPER
 - d. REPLACE

10. Which of the following function returns the day number for a datetime value?
 - a. TO_DAYS
 - b. DAY
 - c. FROM_DAYS
 - d. LAST_DAY



To enhance your knowledge,

visit **REFERENCES**



www.onlinevarsity.com

Objectives

At the end of this session, the student will be able to:

- *Use Date functions in MySQL.*
- *Use String functions in MySQL.*
- *Use System Information functions in MySQL.*

The steps given in the session are detailed, comprehensive, and carefully thought through in order to meet the learning objectives and understand the tool completely. Please follow the steps carefully.

Part I - For the first 1.5 hours:

Date Functions in MySQL

Date functions operate on date and time type of data. The date data type allows you to manipulate date values, allowing a user to add or subtract dates. In addition, several functions extract the required part from the date argument.

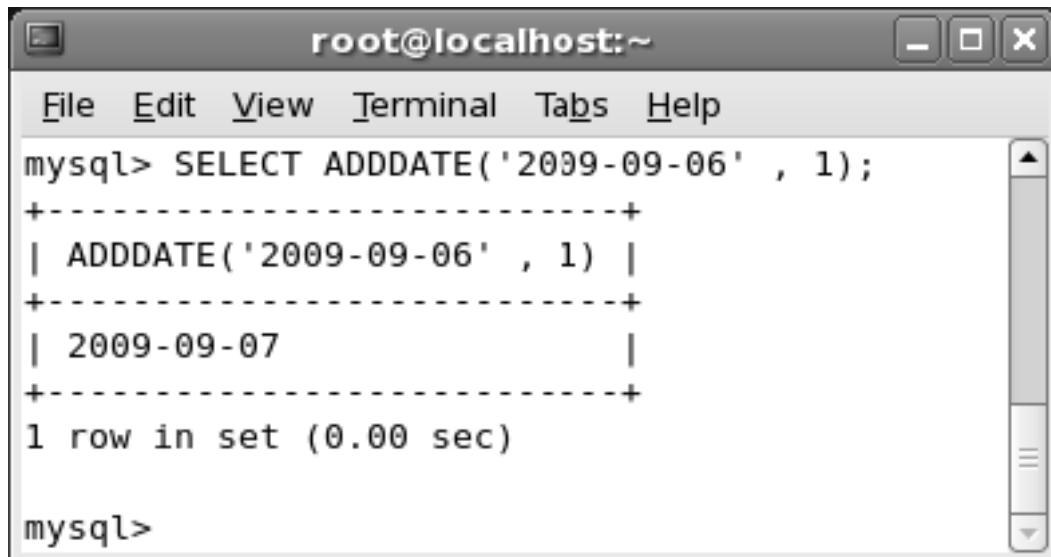
1. John's hire date is '2009-09-06'. He starts actual work one day after the hire date. You have to display the date when he actually started working. To achieve this, you will add 1 day to the hire date. To display the working date, enter the following command at the command prompt:

```
SELECT ADDDATE('2009-09-06', 1);
```

Figure 15.1 displays the output of the command.

Session 15

Using Basic Functions in MySQL - II (Lab)



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its output:

```
mysql> SELECT ADDDATE('2009-09-06' , 1);
+-----+
| ADDDATE('2009-09-06' , 1) |
+-----+
| 2009-09-07 |
+-----+
1 row in set (0.00 sec)

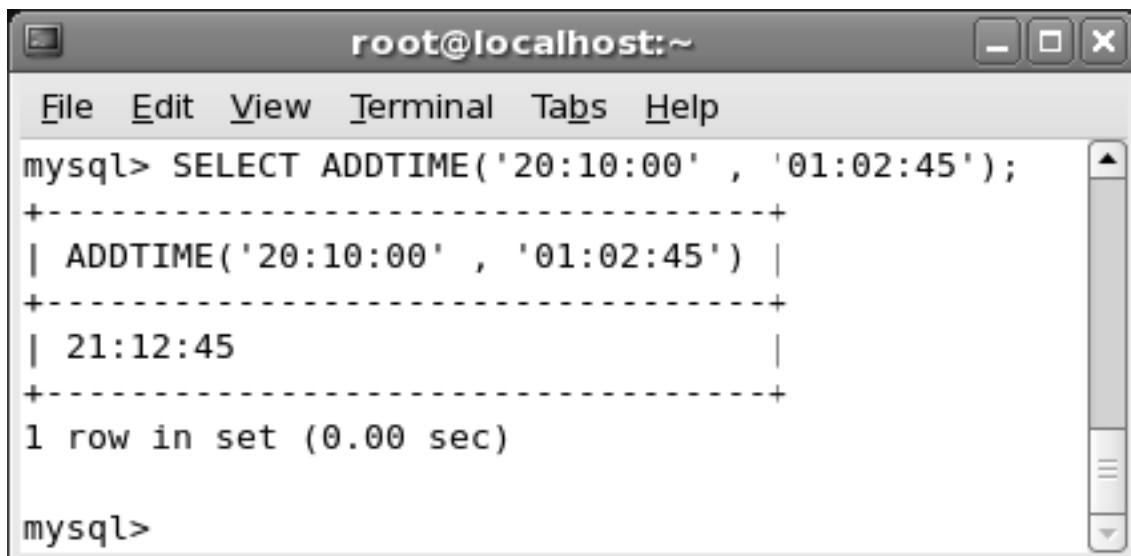
mysql>
```

Figure 15.1: ADDDATE Function

2. John convened a meeting with his team at '20:10:00', which continued for an interval of '01:02:45'. To calculate the time when the meeting ended, you will add the two time expressions. To calculate the meeting end time, enter the following command at the command prompt:

```
SELECT ADDTIME('20:10:00' , '01:02:45');
```

Figure 15.2 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its output:

```
mysql> SELECT ADDTIME('20:10:00' , '01:02:45');
+-----+
| ADDTIME('20:10:00' , '01:02:45') |
+-----+
| 21:12:45 |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 15.2: ADDTIME Function

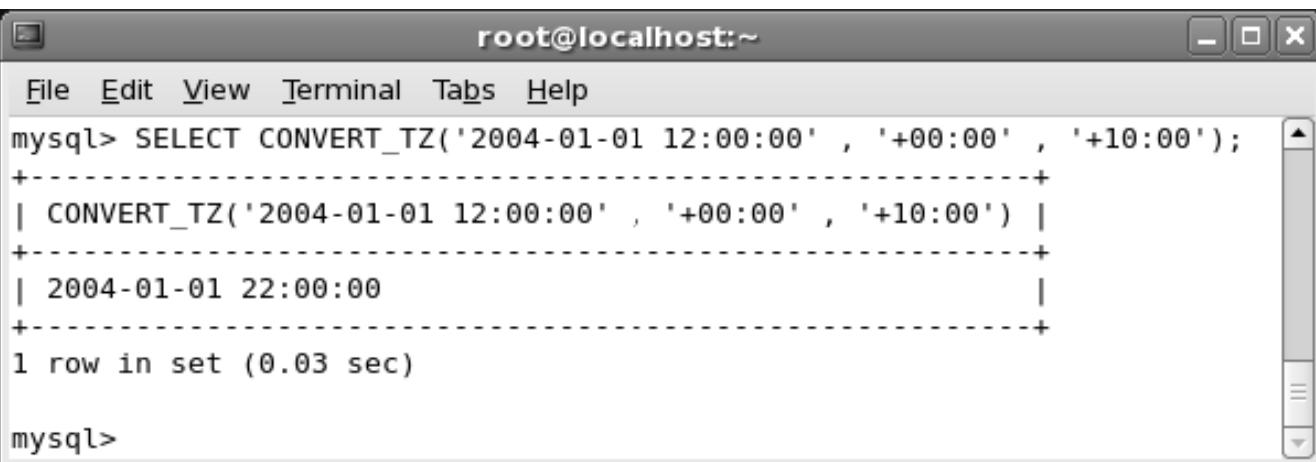
Session 15

Using Basic Functions in MySQL - II (Lab)

3. John wants to have a Web conference with his client at '2004-01-01 12:00:00' GMT. However, the client resides in a city which is 10 hours ahead of the specified time zone. To adjust his schedule, he needs to adapt to his client's time zone accordingly. To calculate the new time zone, enter the following command at the command prompt:

```
SELECT CONVERT_TZ('2004-01-01 12:00:00', '+00:00', '+10:00');
```

Figure 15.3 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following text:

```
File Edit View Terminal Tabs Help
mysql> SELECT CONVERT_TZ('2004-01-01 12:00:00', '+00:00', '+10:00');
+-----+
| CONVERT_TZ('2004-01-01 12:00:00', '+00:00', '+10:00') |
+-----+
| 2004-01-01 22:00:00 |
+-----+
1 row in set (0.03 sec)

mysql>
```

Figure 15.3: CONVERT_TZ Function

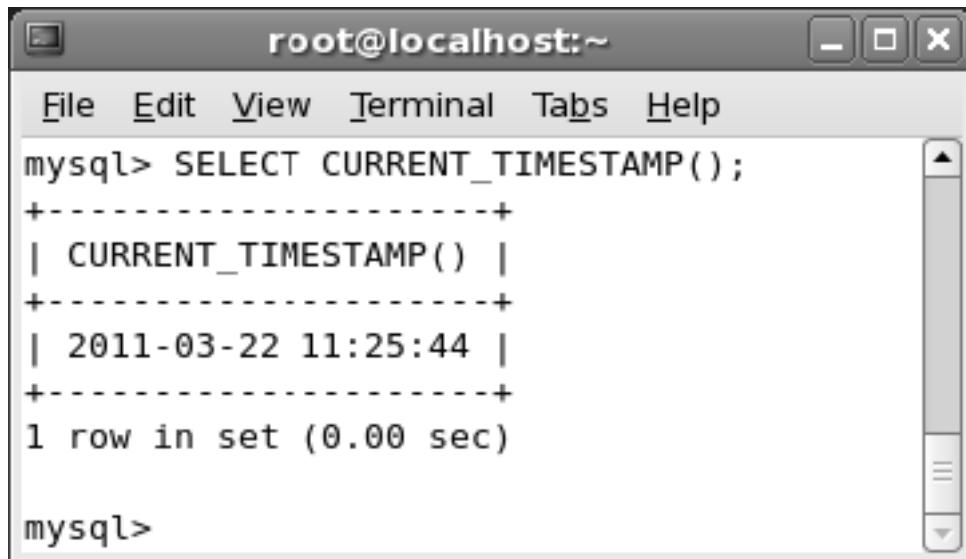
4. John wants to know the current date and time because he lost his watch and has to proceed for a meeting. To view the current timestamp, enter the following command at the command prompt:

```
SELECT CURRENT_TIMESTAMP();
```

Figure 15.4 displays the output of the command.

Session 15

Using Basic Functions in MySQL - II (Lab)



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its output:

```
mysql> SELECT CURRENT_TIMESTAMP();
+-----+
| CURRENT_TIMESTAMP() |
+-----+
| 2011-03-22 11:25:44 |
+-----+
1 row in set (0.00 sec)

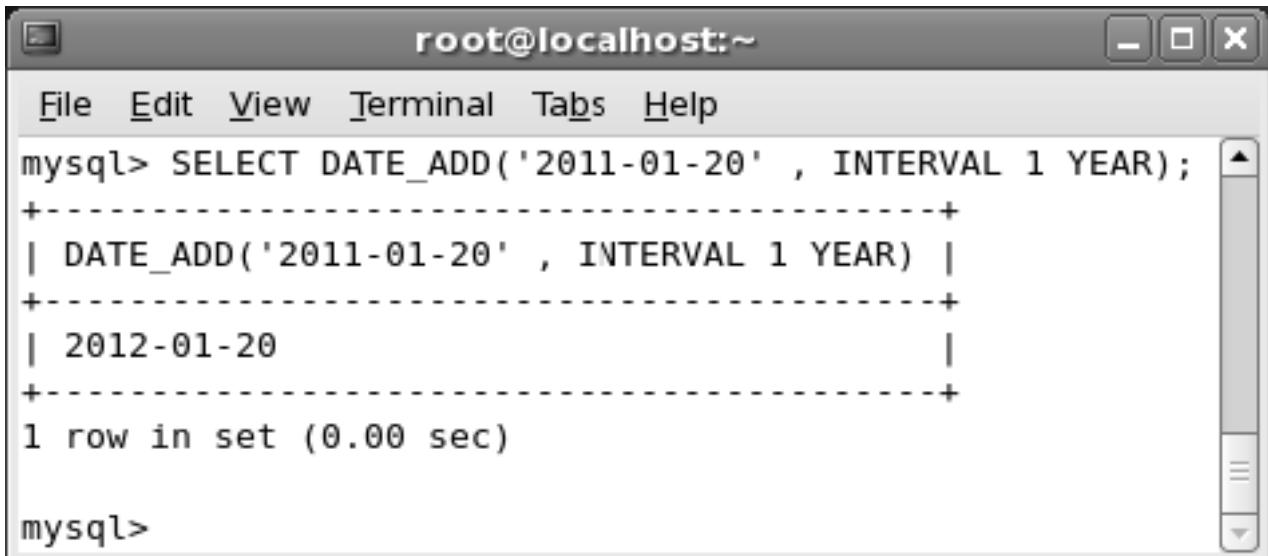
mysql>
```

Figure 15.4: CURRENT_TIMESTAMP Function

5. Peter joined the library on '2011-01-20' and he will have to renew the subscription exactly one year from the joining date. To calculate the subscription renewal date, enter the following command at the command prompt:

```
SELECT DATE_ADD('2011-01-20' , INTERVAL 1 YEAR);
```

Figure 15.5 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its output:

```
mysql> SELECT DATE_ADD('2011-01-20' , INTERVAL 1 YEAR);
+-----+
| DATE_ADD('2011-01-20' , INTERVAL 1 YEAR) |
+-----+
| 2012-01-20 |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 15.5: DATE_ADD Function

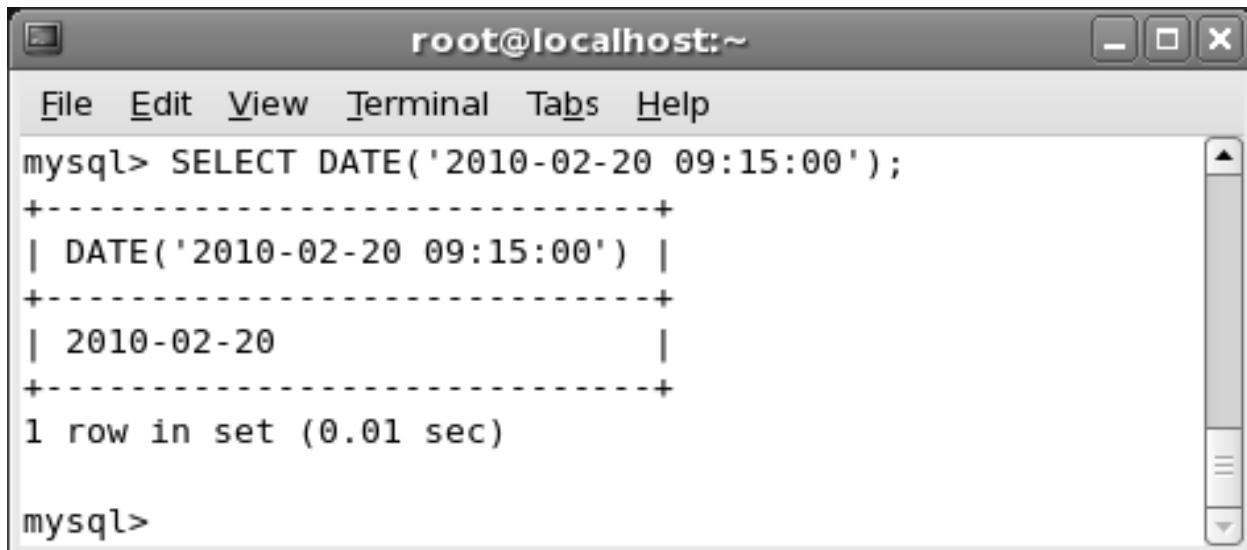
Session 15

Using Basic Functions in MySQL - II (Lab)

6. Tim wants to know the date from the expression ‘2010-02-20 09:15:00’. To display the date from the expression, enter the following command at the command prompt:

```
SELECT DATE('2010-02-20 09:15:00');
```

Figure 15.6 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area contains a MySQL command-line interface. The user has entered the command "SELECT DATE('2010-02-20 09:15:00');". The output shows the result of the query in a tabular format:

+	- - - - +
	DATE('2010-02-20 09:15:00')
+	- - - - +
	2010-02-20
+	- - - - +
1	row in set (0.01 sec)

At the bottom of the terminal window, the prompt "mysql>" is visible.

Figure 15.6: DATE Function

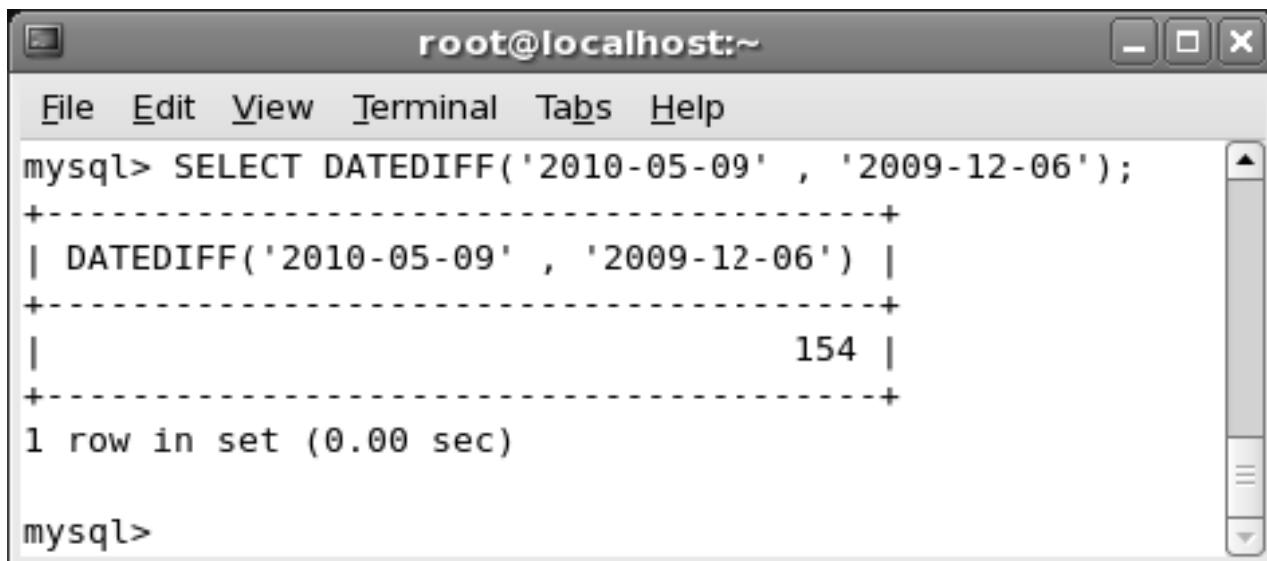
7. Diana joined the firm on ‘2009-12-06’ and sent her resignation letter on ‘2010-05-09’. To calculate her tenure in number of days in the firm you must calculate the difference between the join date and the end date. To calculate the tenure, enter the following command at the command prompt:

```
SELECT DATEDIFF('2010-05-09' , '2009-12-06');
```

Figure 15.7 displays the output of the command.

Session 15

Using Basic Functions in MySQL - II (Lab)



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its output:

```
mysql> SELECT DATEDIFF('2010-05-09' , '2009-12-06');
+-----+
| DATEDIFF('2010-05-09' , '2009-12-06') |
+-----+
| 154 |
+-----+
1 row in set (0.00 sec)

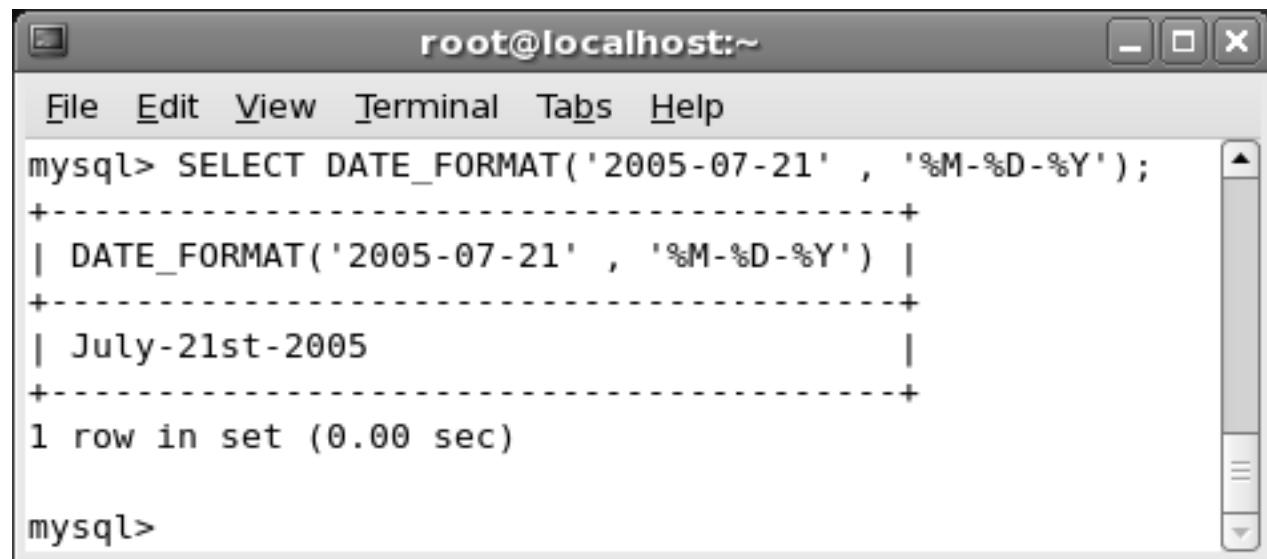
mysql>
```

Figure 15.7: DATEDIFF Function

8. Allen does not know the format in which the date is stored in the system. He wants to view the date '2005-07-21' in the 'MM-DD-YYYY' format. To view the date in the specified format, enter the following command at the command prompt:

```
SELECT DATE_FORMAT('2005-07-21' , '%M-%D-%Y');
```

Figure 15.8 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its output:

```
mysql> SELECT DATE_FORMAT('2005-07-21' , '%M-%D-%Y');
+-----+
| DATE_FORMAT('2005-07-21' , '%M-%D-%Y') |
+-----+
| July-21st-2005 |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 15.8: DATE_FORMAT Function

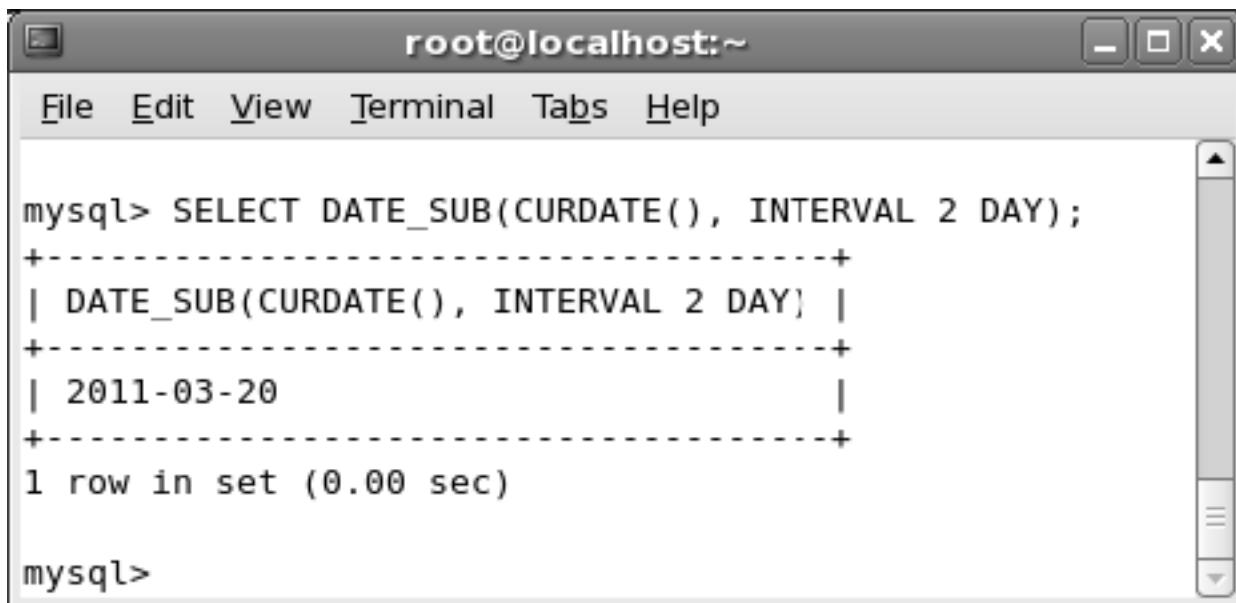
Session 15

Using Basic Functions in MySQL - II (Lab)

9. Tim started a project two days earlier and will be finishing today. The manager wants to know the date on which the project work started. To calculate the start date, you will subtract two days from the current date. To subtract two days from the current date, enter the following command at the command prompt:

```
SELECT DATE_SUB(CURDATE(), INTERVAL 2 DAY);
```

Figure 15.9 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The main area contains the following text:

```
mysql> SELECT DATE_SUB(CURDATE(), INTERVAL 2 DAY);
+-----+
| DATE_SUB(CURDATE(), INTERVAL 2 DAY) |
+-----+
| 2011-03-20                                |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 15.9: DATE_SUB Function

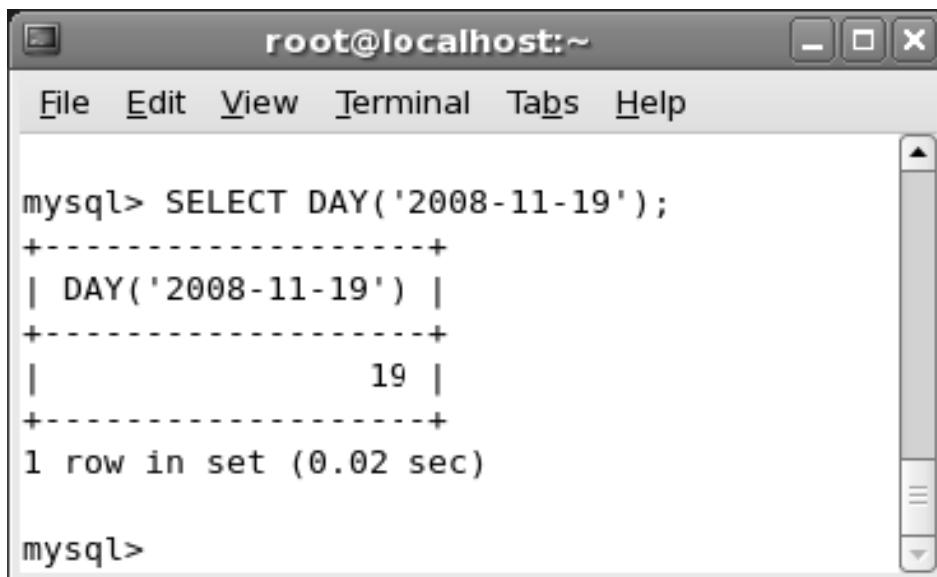
10. Adam wants to know the day for his date of joining '2008-11-19'. To view the day for a given date, enter the following command at the command prompt:

```
SELECT DAY('2008-11-19');
```

Figure 15.10 displays the output of the command.

Session 15

Using Basic Functions in MySQL - II (Lab)



A screenshot of a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area displays the following MySQL command and its output:

```
mysql> SELECT DAY('2008-11-19');
+-----+
| DAY('2008-11-19') |
+-----+
|          19 |
+-----+
1 row in set (0.02 sec)

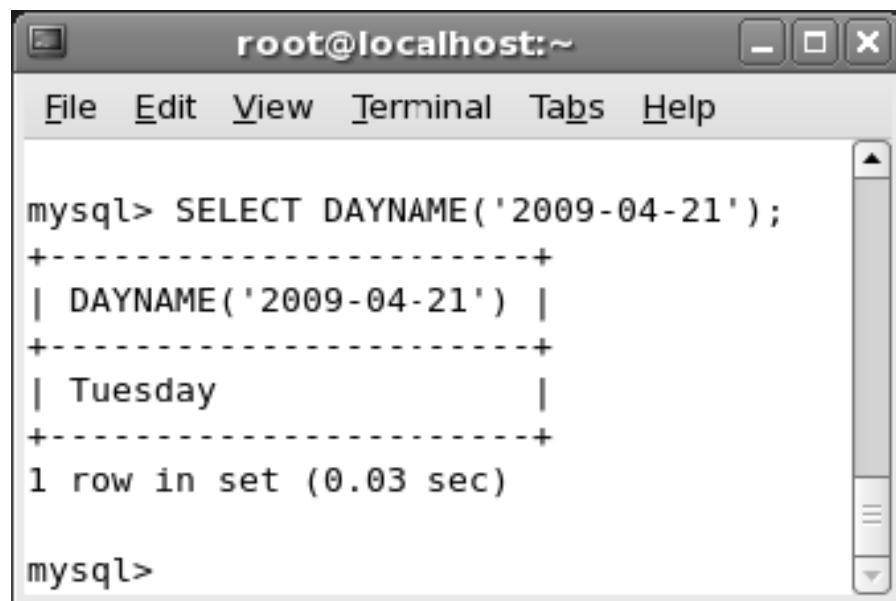
mysql>
```

Figure 15.10: DAY Function

11. Peter wanted to know the day of the week when he started working on '2009-04-21'. To display the name of day for the date '2009-04-21', enter the following command at the command prompt:

```
SELECT DAYNAME('2009-04-21');
```

Figure 15.11 displays the output of the command.



A screenshot of a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area displays the following MySQL command and its output:

```
mysql> SELECT DAYNAME('2009-04-21');
+-----+
| DAYNAME('2009-04-21') |
+-----+
| Tuesday                |
+-----+
1 row in set (0.03 sec)

mysql>
```

Figure 15.11: DAYNAME Function

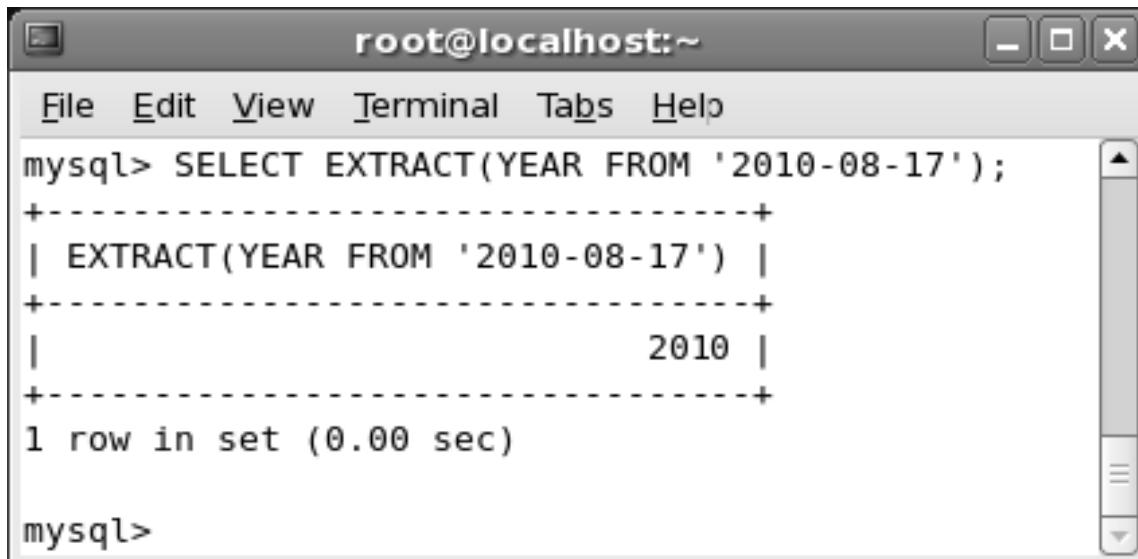
Session 15

Using Basic Functions in MySQL - II (Lab)

12. John wants to know the year in the date '2010-08-17'. To view the year from a date, enter the following command at the command prompt:

```
SELECT EXTRACT(YEAR FROM '2010-08-17');
```

Figure 15.12 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area contains the following text:

```
mysql> SELECT EXTRACT(YEAR FROM '2010-08-17');
+-----+
| EXTRACT(YEAR FROM '2010-08-17') |
+-----+
|          2010 |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 15.12: EXTRACT Function

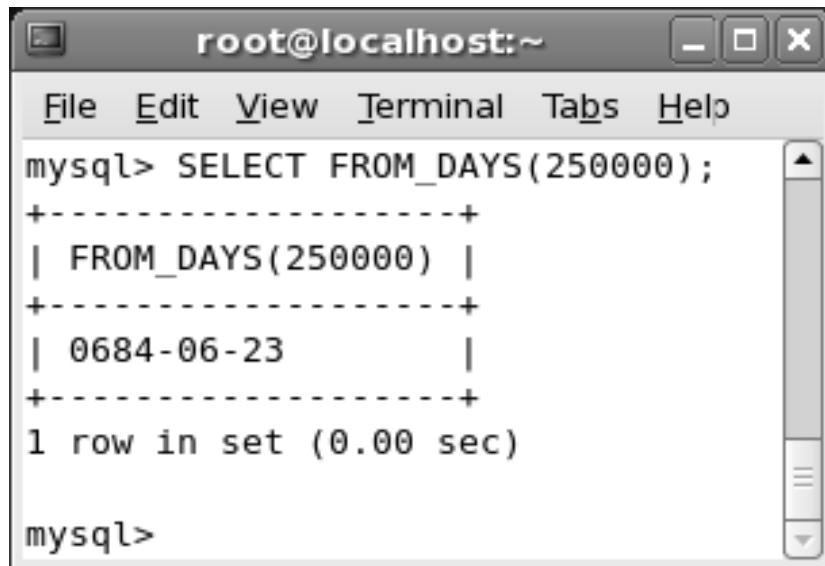
13. As a part of an assignment, John requires to calculate the date for the number '250000'. To calculate the date, enter the following command at the command prompt:

```
SELECT FROM_DAYS(250000);
```

Figure 15.13 displays the output of the command.

Session 15

Using Basic Functions in MySQL - II (Lab)



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its output:

```
mysql> SELECT FROM_DAYS(250000);
+-----+
| FROM_DAYS(250000) |
+-----+
| 0684-06-23       |
+-----+
1 row in set (0.00 sec)

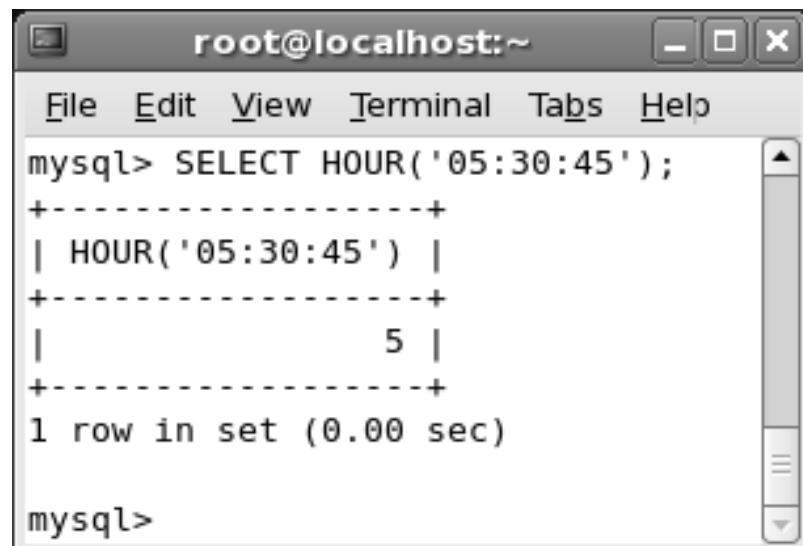
mysql>
```

Figure 15.13: FROM_DAYS Function

14. Jeanne wants to view the hours for the time '05:30:45'. To display the hours from the time expression, enter the following command at the command prompt:

```
SELECT HOUR('05:30:45');
```

Figure 15.14 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its output:

```
mysql> SELECT HOUR('05:30:45');
+-----+
| HOUR('05:30:45') |
+-----+
|      5           |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 15.14: HOUR Function

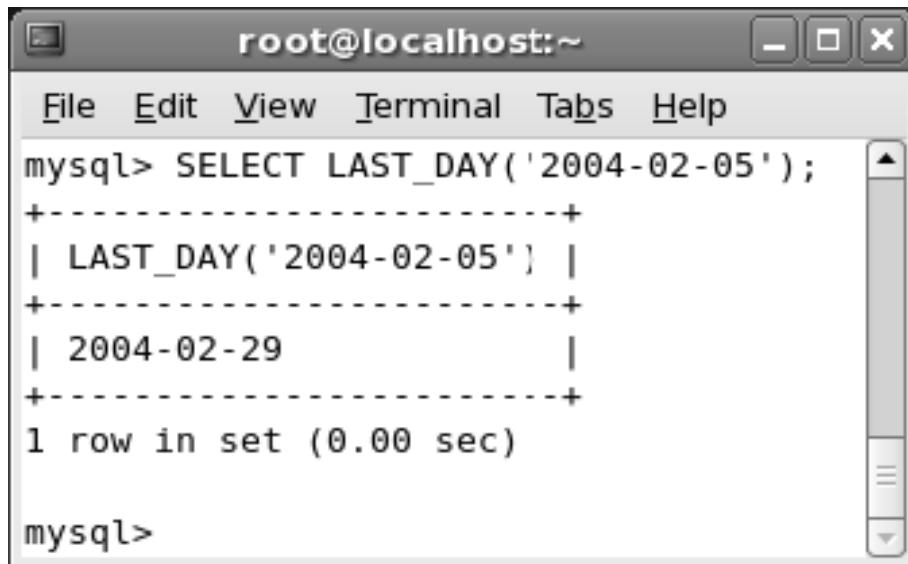
Session 15

Using Basic Functions in MySQL - II (Lab)

15. Sarah wants to view the last day of February 2004 to know whether it was a leap year. To view the last day, enter the following command at the command prompt:

```
SELECT LAST_DAY('2004-02-05');
```

Figure 15.15 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area contains the following text:

```
mysql> SELECT LAST_DAY('2004-02-05');
+-----+
| LAST_DAY('2004-02-05') |
+-----+
| 2004-02-29           |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 15.15: LAST_DAY Function

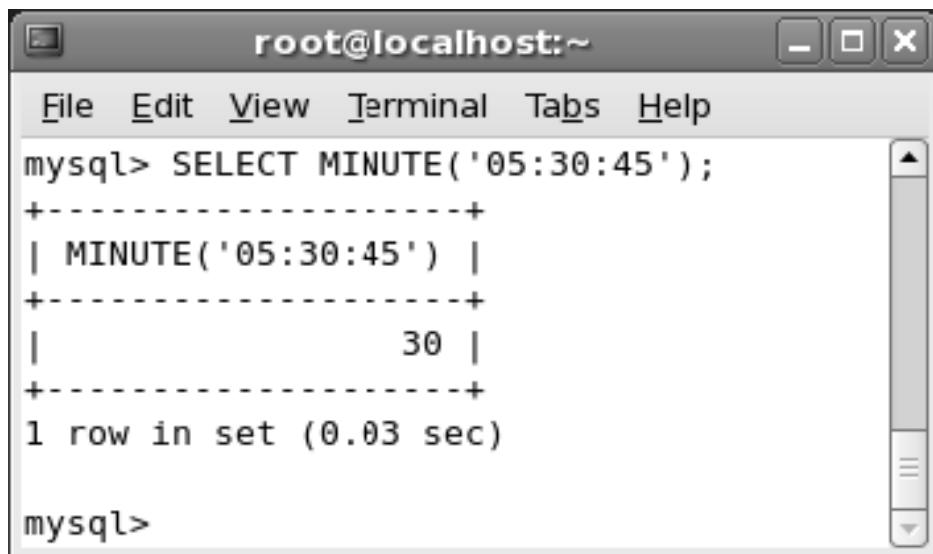
16. Lavina wants to view the exact minutes when the space shuttle was launched from the time expression '05:30:45'. To display the minutes from a time expression, enter the following command at the command prompt:

```
SELECT MINUTE('05:30:45');
```

Figure 15.16 displays the output of the command.

Session 15

Using Basic Functions in MySQL - II (Lab)



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area displays the following MySQL command and its output:

```
mysql> SELECT MINUTE('05:30:45');
+-----+
| MINUTE('05:30:45') |
+-----+
|          30         |
+-----+
1 row in set (0.03 sec)

mysql>
```

Figure 15.16: MINUTE Function

17. Barbara wants to view the month and the month name for the books that have been issued. To display the month and the month name for books that have been issued, enter the following command at the command prompt:

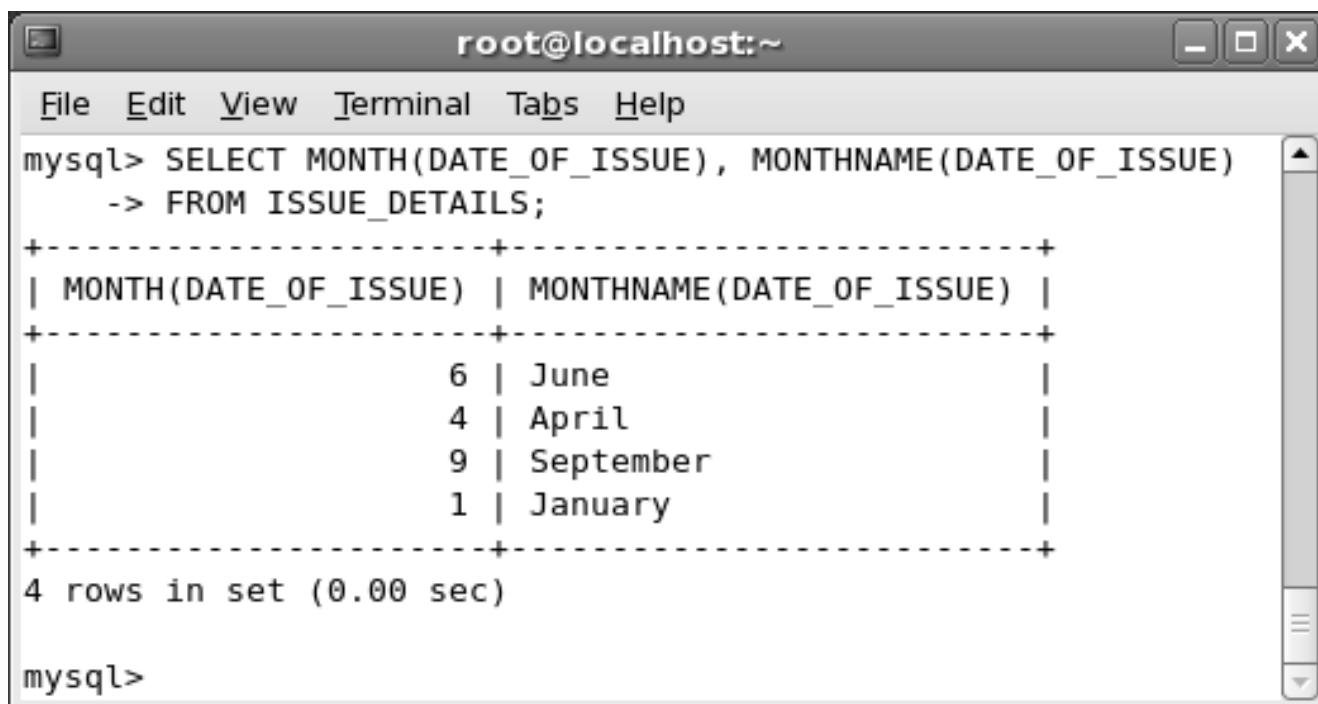
```
SELECT MONTH(DATE_OF_ISSUE), MONTHNAME(DATE_OF_ISSUE)
FROM ISSUE_DETAILS;
```

Figure 15.17 displays the output of the command.

Session 15

Using Basic Functions in MySQL - II (Lab)

Lab Guide



A screenshot of a terminal window titled "root@localhost:~". The window contains the following MySQL query and its output:

```
mysql> SELECT MONTH(DATE_OF_ISSUE), MONTHNAME(DATE_OF_ISSUE)
   -> FROM ISSUE_DETAILS;
+-----+-----+
| MONTH(DATE_OF_ISSUE) | MONTHNAME(DATE_OF_ISSUE) |
+-----+-----+
| 6 | June
| 4 | April
| 9 | September
| 1 | January
+-----+-----+
4 rows in set (0.00 sec)

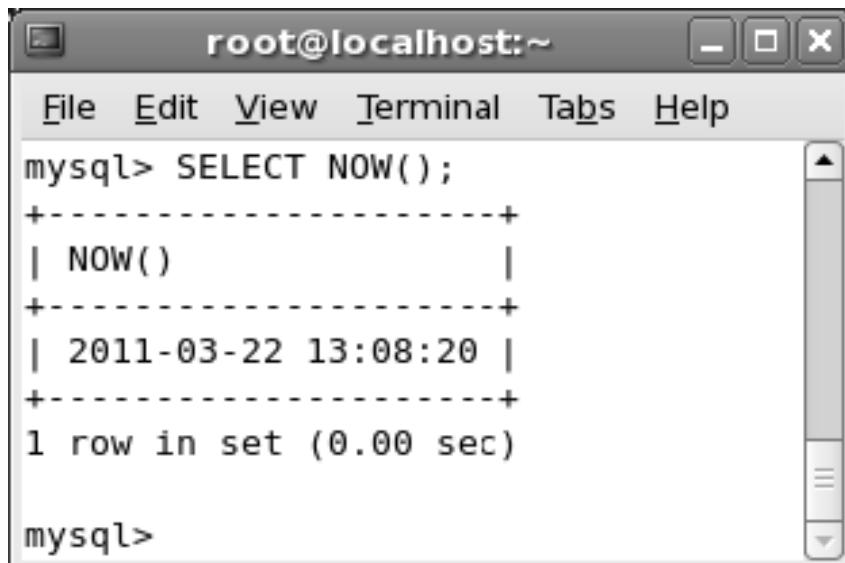
mysql>
```

Figure 15.17: MONTH and MONTHNAME Function

18. To display the current timestamp, enter the following command at the command prompt:

```
SELECT NOW();
```

Figure 15.18 displays the output of the command.



A screenshot of a terminal window titled "root@localhost:~". The window contains the following MySQL query and its output:

```
mysql> SELECT NOW();
+-----+
| NOW()           |
+-----+
| 2011-03-22 13:08:20 |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 15.18: NOW Function

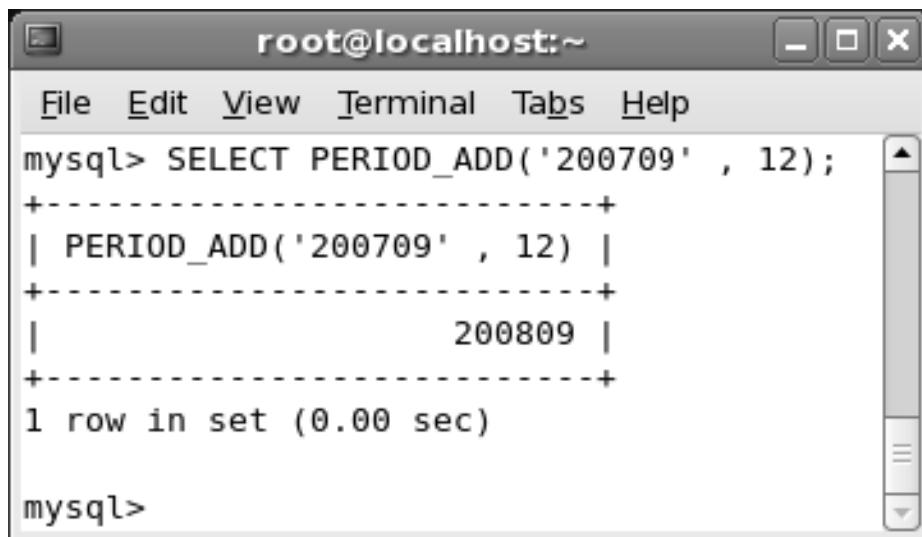
Session 15

Using Basic Functions in MySQL - II (Lab)

19. Celina joined the firm in '200709' and will be on probation for a period of 12 months. To display the probation end date, enter the following command at the command prompt:

```
SELECT PERIOD_ADD('200709' , 12);
```

Figure 15.19 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following text:

```
File Edit View Terminal Tabs Help
mysql> SELECT PERIOD_ADD('200709' , 12);
+-----+
| PERIOD_ADD('200709' , 12) |
+-----+
|          200809 |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 15.19: PERIOD_ADD Function

20. To convert 4200 into the HH:MM:SS format, enter the following command at the command prompt:

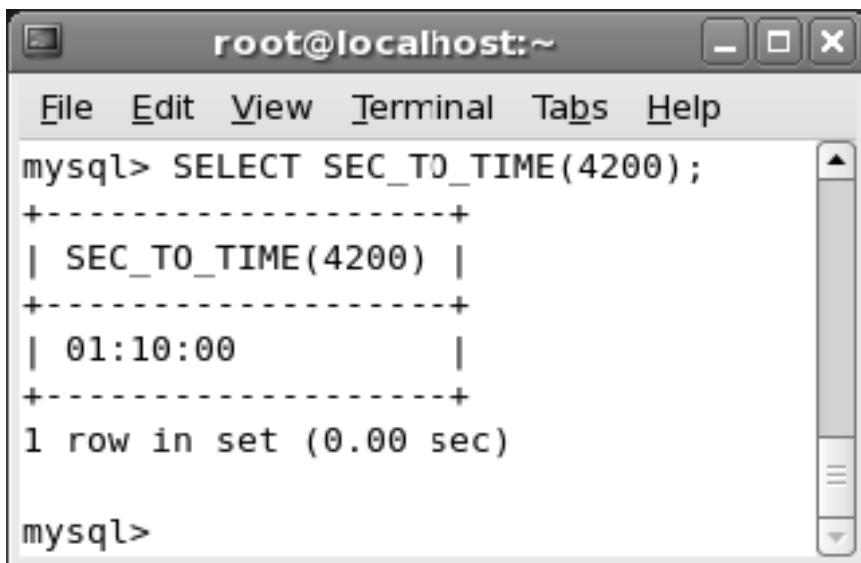
```
SELECT SEC_TO_TIME(4200);
```

Figure 15.20 displays the output of the command.

Session 15

Using Basic Functions in MySQL - II (Lab)

Lab Guide



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The command entered is "SELECT SEC_TO_TIME(4200);". The output is:

```
mysql> SELECT SEC_TO_TIME(4200);
+-----+
| SEC_TO_TIME(4200) |
+-----+
| 01:10:00           |
+-----+
1 row in set (0.00 sec)

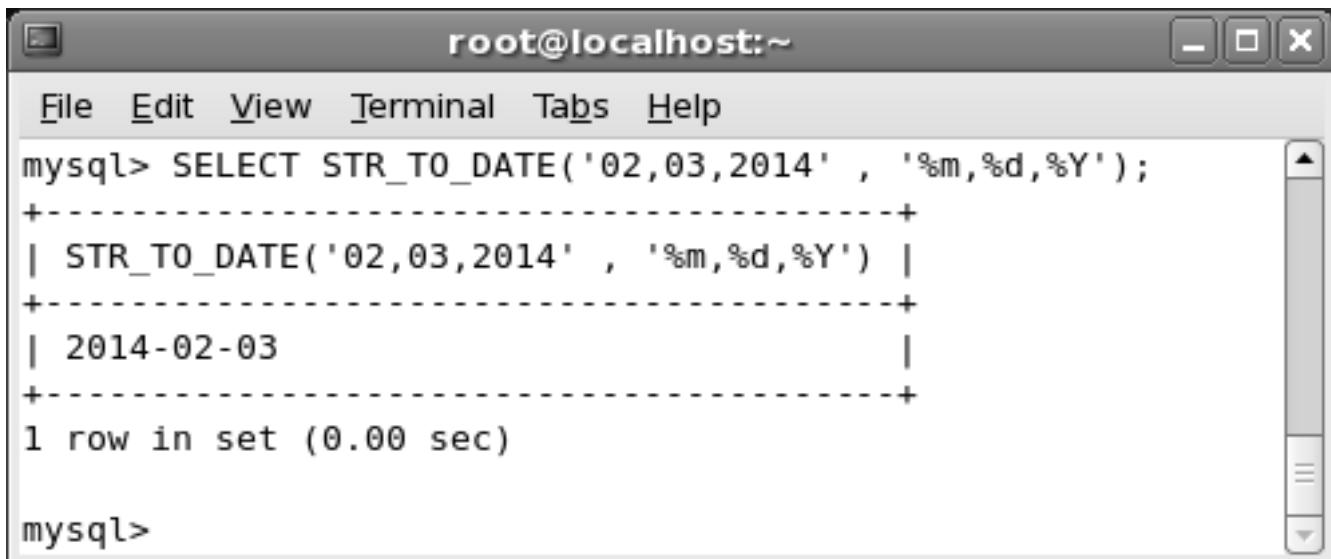
mysql>
```

Figure 15.20: SEC_TO_TIME Function

21. Michael wants to convert the sting '02, 03, 2014' into the 'YYYY-MM-DD' format. To convert the string to 'YYYY-MM-DD' format, enter the following command at the command prompt:

```
SELECT STR_TO_DATE('02,03,2014' , '%m,%d,%Y');
```

Figure 15.21 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The command entered is "SELECT STR_TO_DATE('02,03,2014' , '%m,%d,%Y');". The output is:

```
mysql> SELECT STR_TO_DATE('02,03,2014' , '%m,%d,%Y');
+-----+
| STR_TO_DATE('02,03,2014' , '%m,%d,%Y') |
+-----+
| 2014-02-03                                |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 15.21: STR_TO_DATE Function

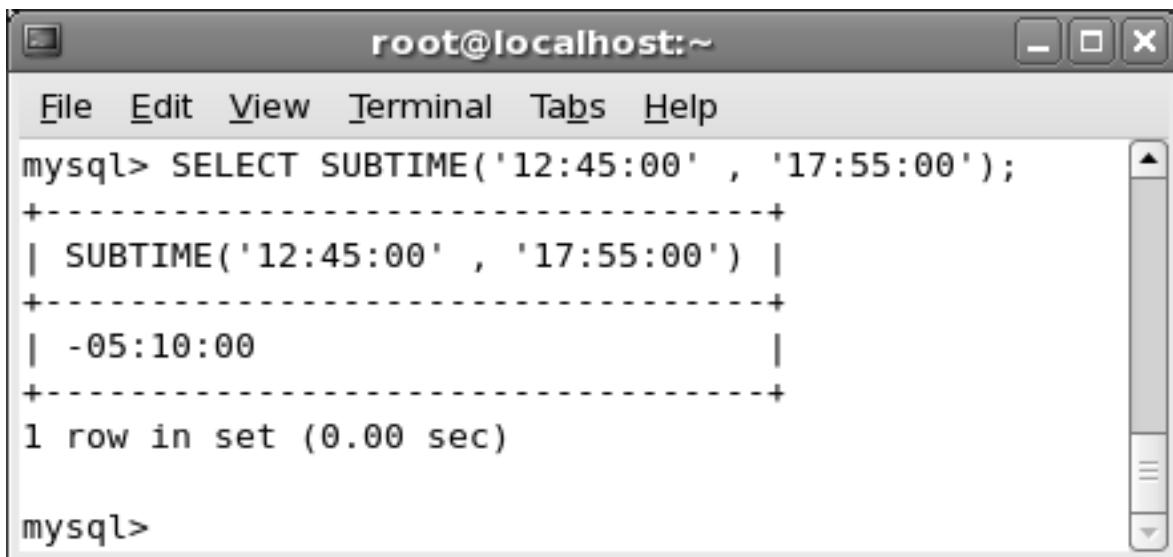
Session 15

Using Basic Functions in MySQL - II (Lab)

22. Peter reached office at '12:45:00' and left at '17:55:00'. To calculate the total working hours, enter the following command at the command prompt:

```
SELECT SUBTIME('12:45:00' , '17:55:00');
```

Figure 15.22 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area contains the following text:

```
mysql> SELECT SUBTIME('12:45:00' , '17:55:00');
+-----+
| SUBTIME('12:45:00' , '17:55:00') |
+-----+
| -05:10:00 |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 15.22: SUBTIME Function

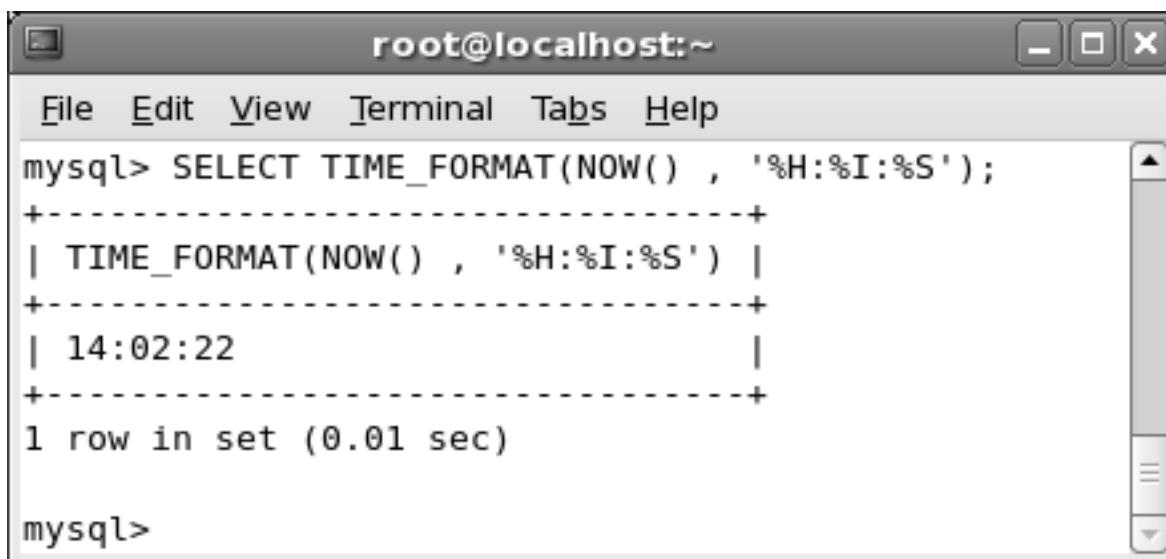
23. Annie wants to view the current time in HH:MM:SS format. To display the current time in the 'HH:MM:SS' format, enter the following command at the command prompt:

```
SELECT TIME_FORMAT(NOW() , '%H:%I:%S');
```

Figure 15.23 displays the output of the command.

Session 15

Using Basic Functions in MySQL - II (Lab)



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". Below the menu is a command-line interface. The user has run the command: "SELECT TIME_FORMAT(NOW(), '%H:%I:%S');". The output is a single row: "+-----+ | TIME_FORMAT(NOW(), '%H:%I:%S') | +-----+ | 14:02:22 | +-----+ 1 row in set (0.01 sec)" followed by a blank line "mysql>".

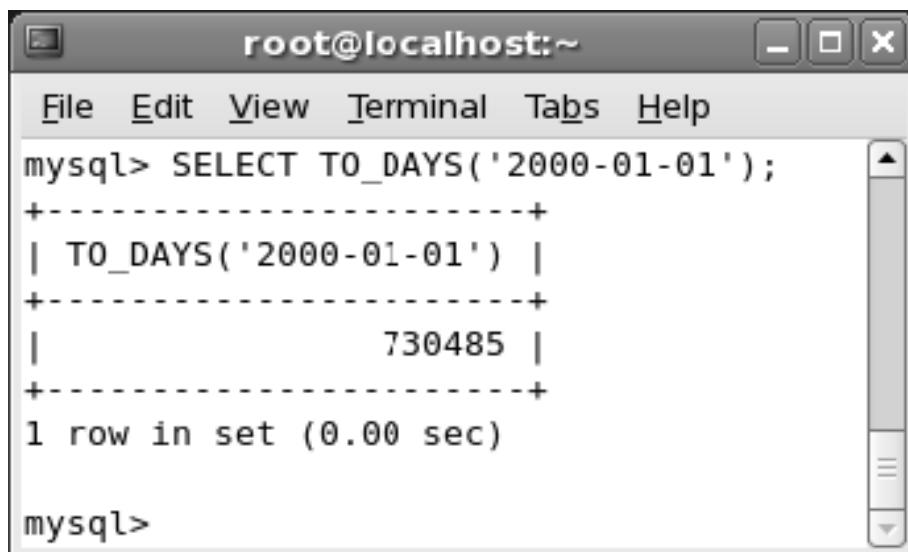
Lab Guide

Figure 15.23: TIME_FORMAT Function

24. Dave wants to calculate the day number for the date '2000-01-01'. To calculate the number of days that have elapsed, enter the following command at the command prompt:

```
SELECT TO_DAYS('2000-01-01');
```

Figure 15.24 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". Below the menu is a command-line interface. The user has run the command: "SELECT TO_DAYS('2000-01-01');". The output is a single row: "+-----+ | TO_DAYS('2000-01-01') | +-----+ | 730485 | +-----+ 1 row in set (0.00 sec)" followed by a blank line "mysql>".

Figure 15.24: TO_DAYS Function

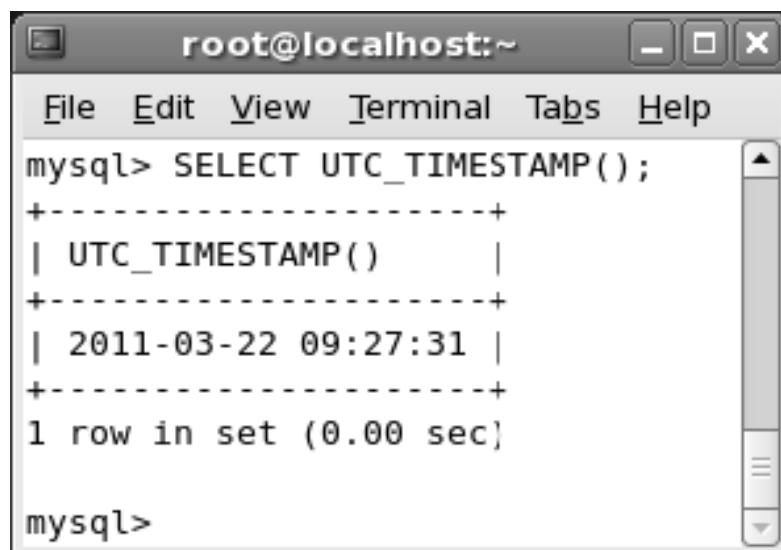
Session 15

Using Basic Functions in MySQL - II (Lab)

25. To display the current UTC date and time, enter the following command at the command prompt:

```
SELECT UTC_TIMESTAMP;
```

Figure 15.25 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area contains the following text:

```
mysql> SELECT UTC_TIMESTAMP();
+-----+
| UTC_TIMESTAMP() |
+-----+
| 2011-03-22 09:27:31 |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 15.25: UTC_TIMESTAMP Function

26. Gia is recording all important events and dates. She completed one year in the organization on '2005-04-20'. To display the week number for the date '2005-04-20', enter the following command at the command prompt:

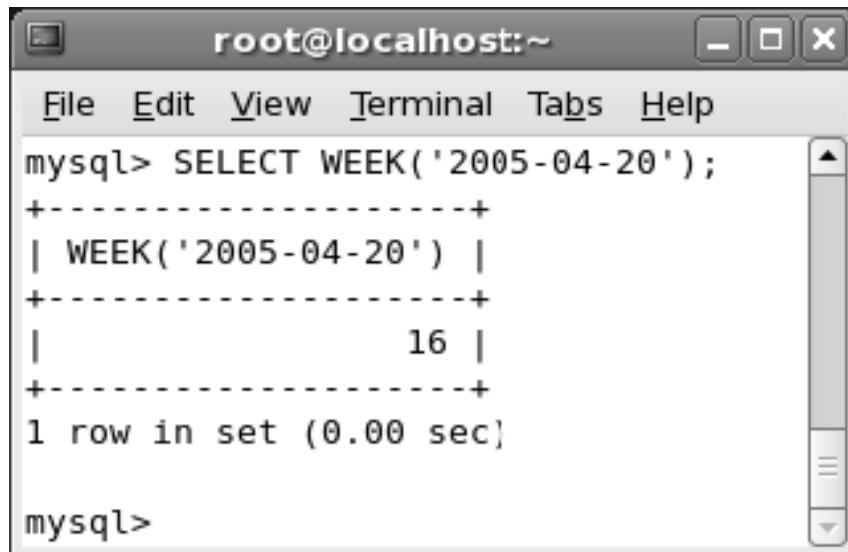
```
SELECT WEEK('2005-04-20');
```

Figure 15.26 displays the output of the command.

Session 15

Using Basic Functions in MySQL - II (Lab)

Lab Guide



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area displays the following MySQL command and its result:

```
mysql> SELECT WEEK('2005-04-20');
+-----+
| WEEK('2005-04-20') |
+-----+
|          16         |
+-----+
1 row in set (0.00 sec)

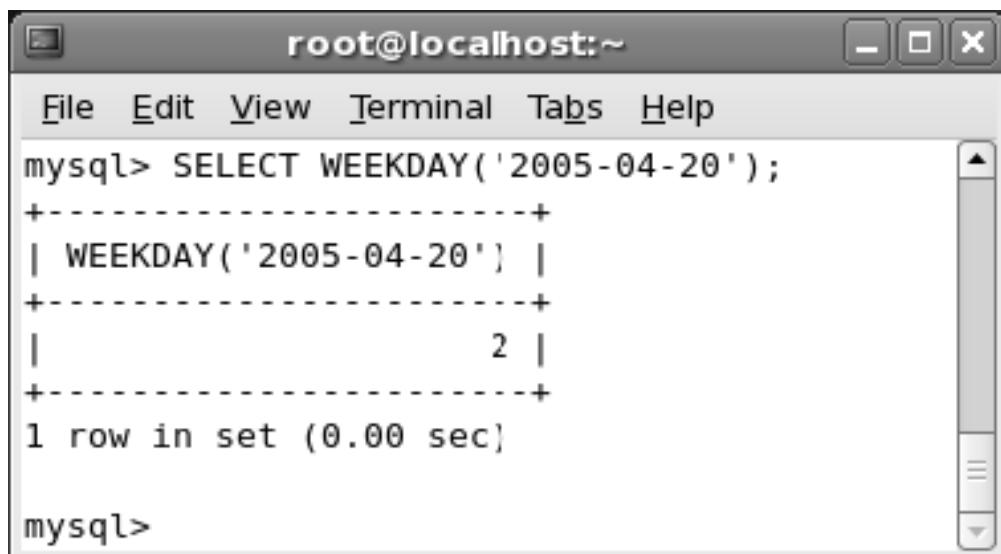
mysql>
```

Figure 15.26: WEEK Function

27. Gia also wants to view the weekday index for the date when she completed one year in the organization. To display the weekday index for '2005-04-20', enter the following command at the command prompt:

```
SELECT WEEKDAY('2005-04-20');
```

Figure 15.27 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area displays the following MySQL command and its result:

```
mysql> SELECT WEEKDAY('2005-04-20');
+-----+
| WEEKDAY('2005-04-20') |
+-----+
|          2           |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 15.27: WEEKDAY Function

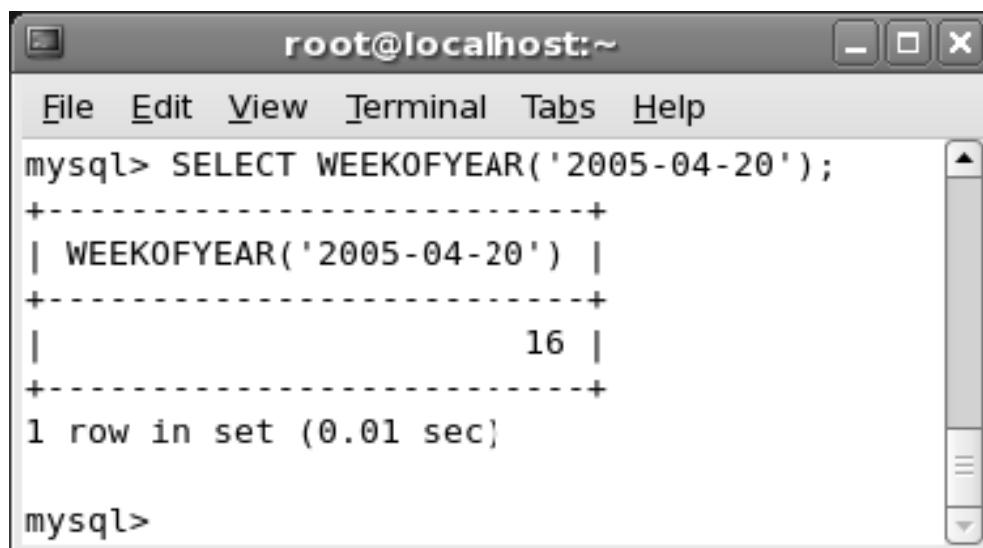
Session 15

Using Basic Functions in MySQL - II (Lab)

28. George wants to view the week number for the date '2005-04-20'. To display the week number for '2005-04-20', enter the following command at the command prompt:

```
SELECT WEEKOFYEAR('2005-04-20');
```

Figure 15.28 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area contains the following MySQL command and its output:

```
mysql> SELECT WEEKOFYEAR('2005-04-20');
+-----+
| WEEKOFYEAR('2005-04-20') |
+-----+
| 16 |
+-----+
1 row in set (0.01 sec)

mysql>
```

The output shows that the week number for April 20, 2005, is 16.

Figure 15.28: WEEKOFYEAR Function

29. To display the year from '2010-12-05', enter the following command at the command prompt:

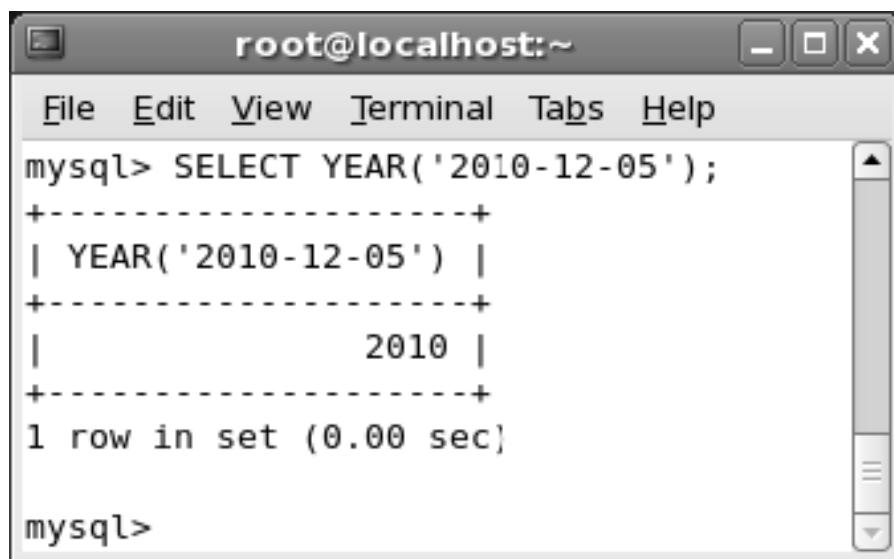
```
SELECT YEAR('2010-12-05');
```

Figure 15.29 displays the output of the command.

Session 15

Using Basic Functions in MySQL - II (Lab)

Lab Guide



A screenshot of a terminal window titled "root@localhost:~". The window shows the MySQL command-line interface. A single command is entered: "SELECT YEAR('2010-12-05');". The output is a table with one row, displaying the year '2010'. The terminal prompt "mysql>" appears at the bottom.

```
mysql> SELECT YEAR('2010-12-05');
+-----+
| YEAR('2010-12-05') |
+-----+
|           2010 |
+-----+
1 row in set (0.00 sec)

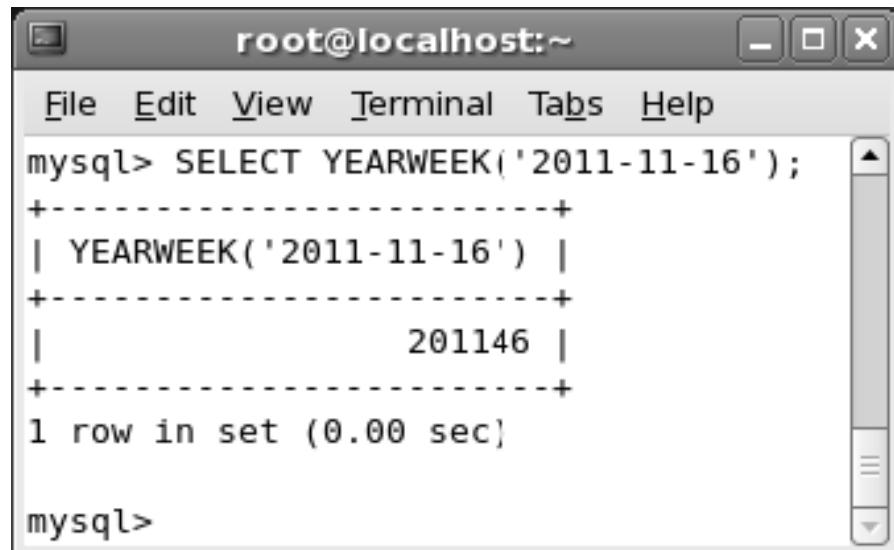
mysql>
```

Figure 15.29: YEAR Function

30. To display the year and week for '2011-11-16', enter the following command at the command prompt:

```
SELECT YEARWEEK('2011-11-16');
```

Figure 15.30 displays the output of the command.



A screenshot of a terminal window titled "root@localhost:~". The window shows the MySQL command-line interface. A single command is entered: "SELECT YEARWEEK('2011-11-16');". The output is a table with one row, displaying the year-week value '201146'. The terminal prompt "mysql>" appears at the bottom.

```
mysql> SELECT YEARWEEK('2011-11-16');
+-----+
| YEARWEEK('2011-11-16') |
+-----+
|           201146 |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 15.30: YEARWEEK Function

Session 15

Using Basic Functions in MySQL - II (Lab)

String functions in MySQL

MySQL provides string functions that operate on string data type. The string functions accept arguments as strings even if you input numeric data. You will work with the **LIBRARY** database to use string functions in MySQL.

1. Activate the **LIBRARY** database.
2. To display the name of the users and the ASCII value of the names, enter the following command at the command prompt:

```
SELECT NAME, ASCII(NAME) FROM USER_DETAILS;
```

Figure 15.31 displays the output of the command.

The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area displays the following MySQL query and its result:

```
mysql> SELECT ASCII('DATA');
+-----+
| ASCII('DATA') |
+-----+
|          68   |
+-----+
1 row in set (0.03 sec)

mysql>
```

Figure 15.31: ASCII Function

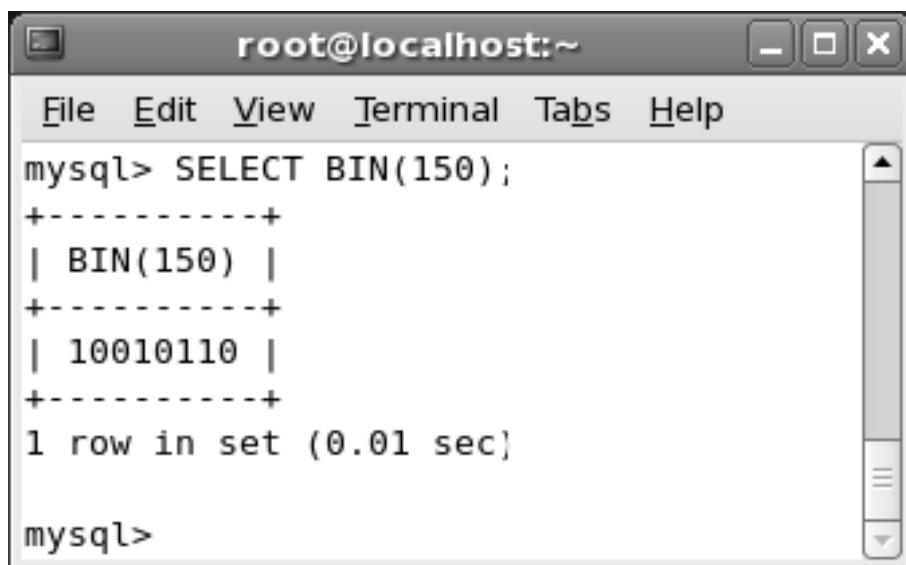
3. John wants to view the phone numbers of users as encrypted values. To display the phone numbers as encrypted values, enter the following command at the command prompt:

```
SELECT PHONE_NO, BIN(PHONE_NO) FROM USER_DETAILS;
```

Figure 15.32 displays the output of the command.

Session 15

Using Basic Functions in MySQL - II (Lab)



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area displays the following MySQL command and its output:

```
mysql> SELECT BIN(150);
+-----+
| BIN(150) |
+-----+
| 10010110 |
+-----+
1 row in set (0.01 sec)

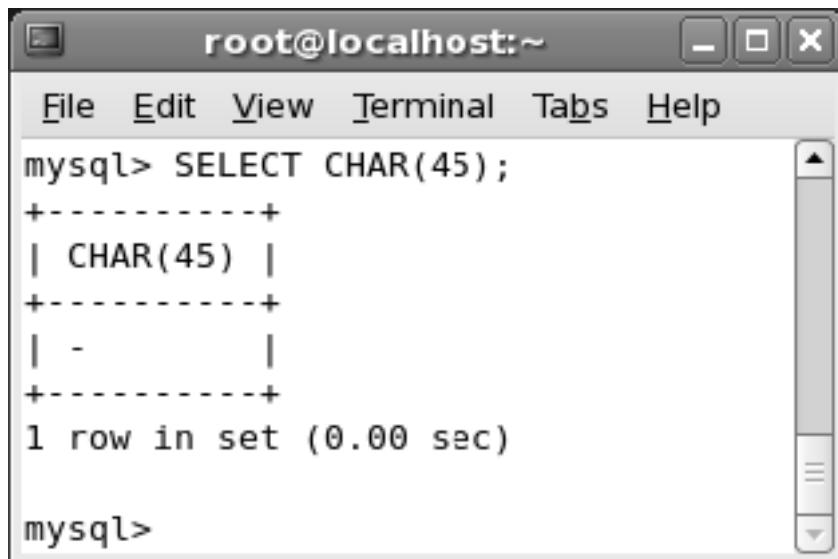
mysql>
```

Figure 7.32: BIN Function

4. To display the BOOK_ID, char value of BOOK_ID, enter the following command at the command prompt:

```
SELECT BOOK_ID, CHAR(BOOK_ID) FROM BOOK_TABLE;
```

Figure 15.33 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area displays the following MySQL command and its output:

```
mysql> SELECT CHAR(45);
+-----+
| CHAR(45) |
+-----+
| -        |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 15.33: CHAR Function

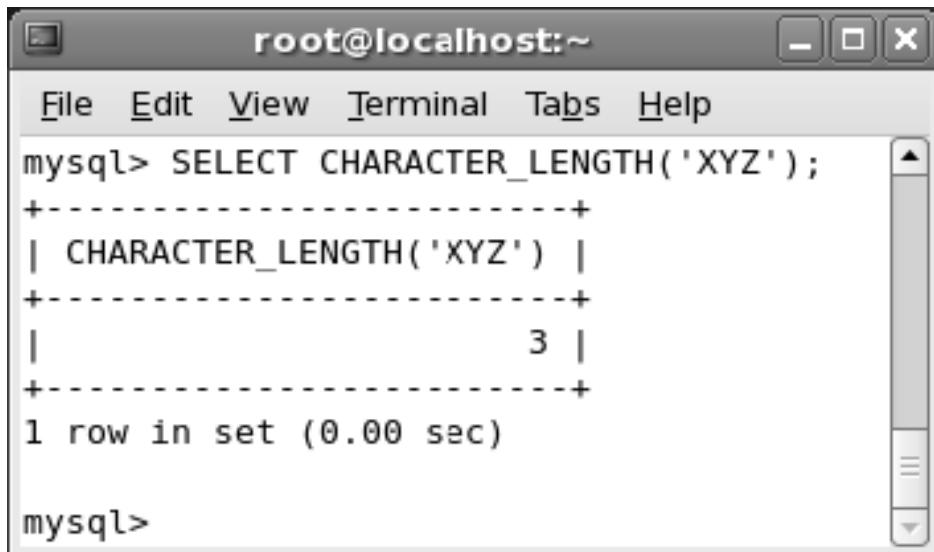
Session 15

Using Basic Functions in MySQL - II (Lab)

5. To display the BOOK_NAME and the maximum number of characters in the BOOK_NAME column, enter the following command at the command prompt:

```
SELECT BOOK_NAME, CHARACTER_LENGTH(BOOK_NAME)  
FROM BOOK_TABLE;
```

Figure 15.34 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area contains the following MySQL command and its output:

```
mysql> SELECT CHARACTER_LENGTH('XYZ');  
+-----+  
| CHARACTER_LENGTH('XYZ') |  
+-----+  
| 3 |  
+-----+  
1 row in set (0.00 sec)  
  
mysql>
```

The output shows that the length of the string 'XYZ' is 3.

Figure 15.34: CHARACTER_LENGTH Function

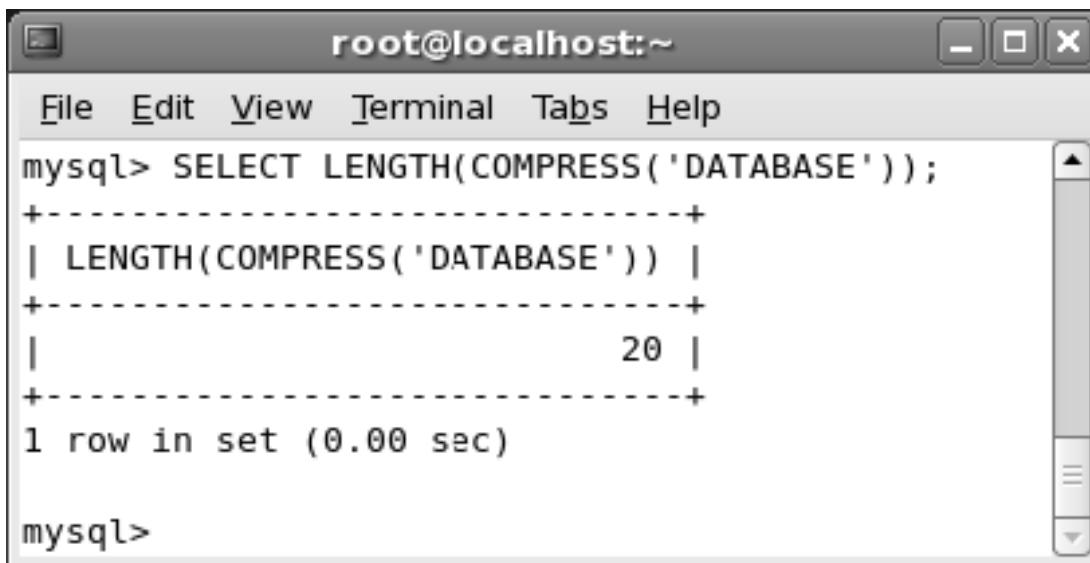
6. To display the names of the author and the length of the compressed string of the names of the author, enter the following command at the command prompt:

```
SELECT AUTHOR, LENGTH(COMPRESS(AUTHOR))  
FROM BOOK_TABLE;
```

Figure 15.35 displays the output of the command.

Session 15

Using Basic Functions in MySQL - II (Lab)



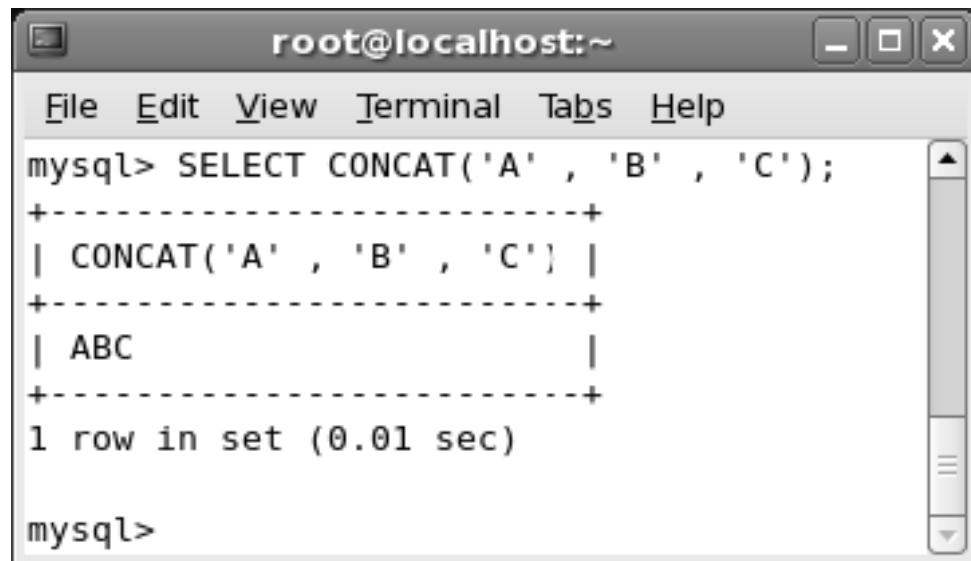
```
root@localhost:~ File Edit View Terminal Tabs Help mysql> SELECT LENGTH(COMPRESS('DATABASE')); +-----+ | LENGTH(COMPRESS('DATABASE')) | +-----+ | 20 | +-----+ 1 row in set (0.00 sec) mysql>
```

Figure 15.35: COMPRESS Function

7. To display the USER_ID and the NAME as a single string, enter the following command at the command prompt:

```
SELECT CONCAT(USER_ID, NAME) FROM USER_DETAILS;
```

Figure 15.36 displays the output of the command.



```
root@localhost:~ File Edit View Terminal Tabs Help mysql> SELECT CONCAT('A' , 'B' , 'C'); +-----+ | CONCAT('A' , 'B' , 'C') | +-----+ | ABC | +-----+ 1 row in set (0.01 sec) mysql>
```

Figure 15.36: CONCAT Function

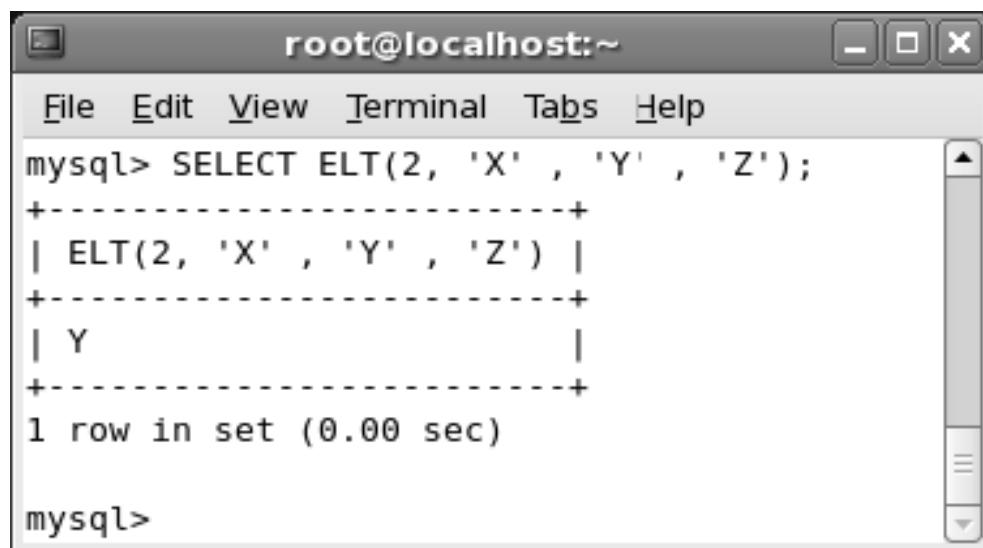
Session 15

Using Basic Functions in MySQL - II (Lab)

8. To display the character at position two of the string 'X', 'Y', 'Z', enter the following command at the command prompt:

```
SELECT ELT(2, 'X', 'Y', 'Z');
```

Figure 15.37 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area contains the following MySQL command and its output:

```
mysql> SELECT ELT(2, 'X' , 'Y' , 'Z');
+-----+
| ELT(2, 'X' , 'Y' , 'Z') |
+-----+
| Y
+-----+
1 row in set (0.00 sec)

mysql>
```

The output shows a single row with the value 'Y'.

Figure 15.37: ELT Function

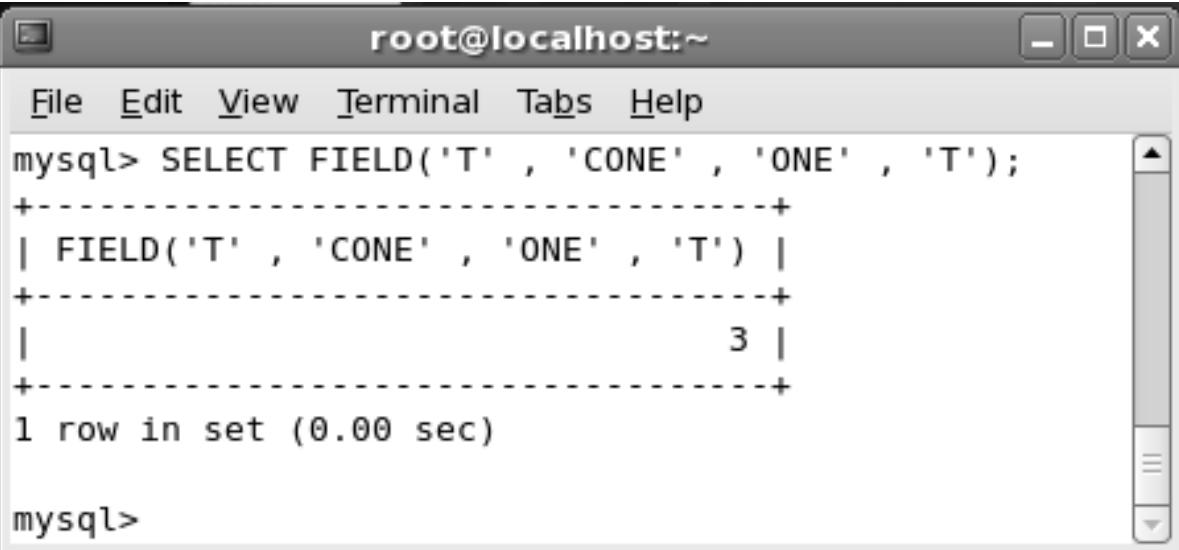
9. To display the index of the field 'T' in the strings 'CONE', 'ONE', and 'T', enter the following command at the command prompt:

```
SELECT FIELD('T', 'CONE', 'ONE', 'T');
```

Figure 15.38 displays the output of the command.

Session 15

Using Basic Functions in MySQL - II (Lab)



The screenshot shows a terminal window titled "root@localhost:~". The command entered is "SELECT FIELD('T', 'CONE', 'ONE', 'T');". The output is a single row with a value of 3, indicating the position of 'T' in the string 'CONE'. The terminal prompt "mysql>" is visible at the bottom.

```
root@localhost:~\nFile Edit View Terminal Tabs Help\nmysql> SELECT FIELD('T', 'CONE', 'ONE', 'T');\n+-----+\n| FIELD('T', 'CONE', 'ONE', 'T') |\n+-----+\n| 3 |\n+-----+\n1 row in set (0.00 sec)\n\nmysql>
```

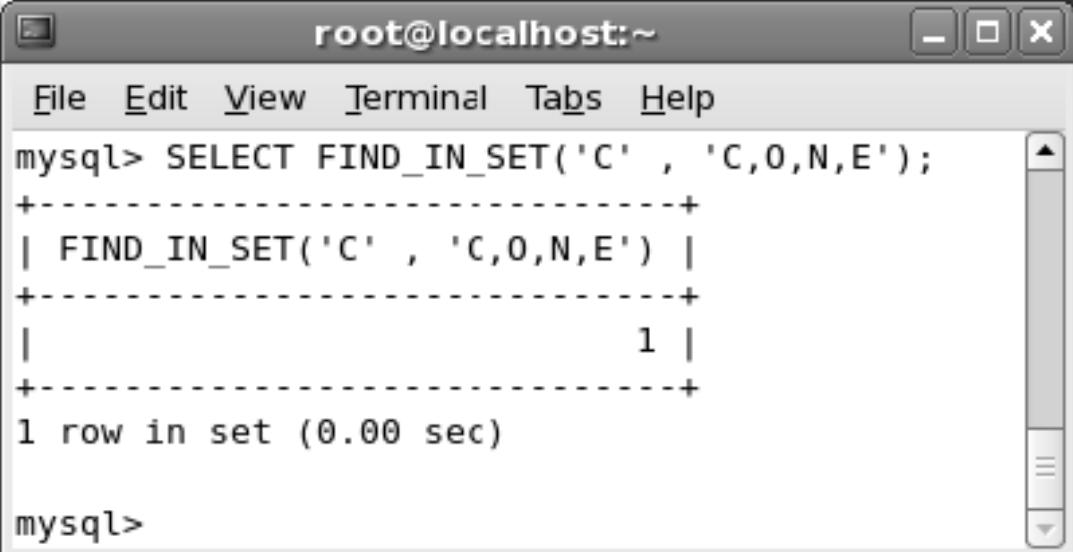
Lab Guide

Figure 15.38: FIELD Function

10. To display the position of the field 'C' in the string 'C, O, N, E', enter the following command at the command prompt:

```
SELECT FIND_IN_SET('C', 'C,O,N,E');
```

Figure 15.39 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The command entered is "SELECT FIND_IN_SET('C', 'C,O,N,E');". The output is a single row with a value of 1, indicating the position of 'C' in the string 'C,O,N,E'. The terminal prompt "mysql>" is visible at the bottom.

```
root@localhost:~\nFile Edit View Terminal Tabs Help\nmysql> SELECT FIND_IN_SET('C', 'C,O,N,E');\n+-----+\n| FIND_IN_SET('C', 'C,O,N,E') |\n+-----+\n| 1 |\n+-----+\n1 row in set (0.00 sec)\n\nmysql>
```

Figure 15.39: FIND_IN_SET Function

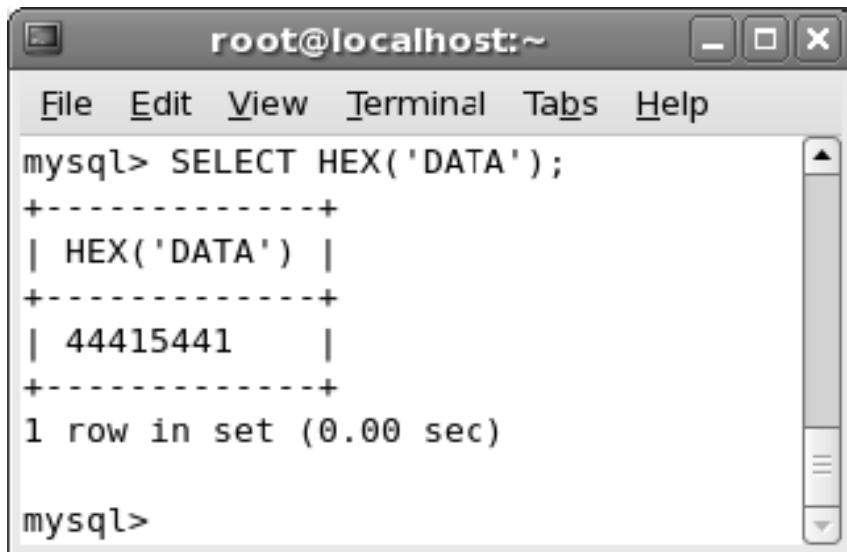
Session 15

Using Basic Functions in MySQL - II (Lab)

11. To display the hexadecimal value of book names, enter the following command at the command prompt:

```
SELECT BOOK_NAME , HEX(BOOK_NAME)  
FROM BOOK_TABLE;
```

Figure 15.40 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area contains the following MySQL query and its result:

```
mysql> SELECT HEX('DATA');  
+-----+  
| HEX('DATA') |  
+-----+  
| 44415441 |  
+-----+  
1 row in set (0.00 sec)  
  
mysql>
```

Figure 15.40: HEX Function

12. The librarian wants to concatenate word 'MEMBER' where the type of membership has the value as REGULAR. To insert the string MEMBER where the TYPE is REGULAR, enter the following command at the command prompt:

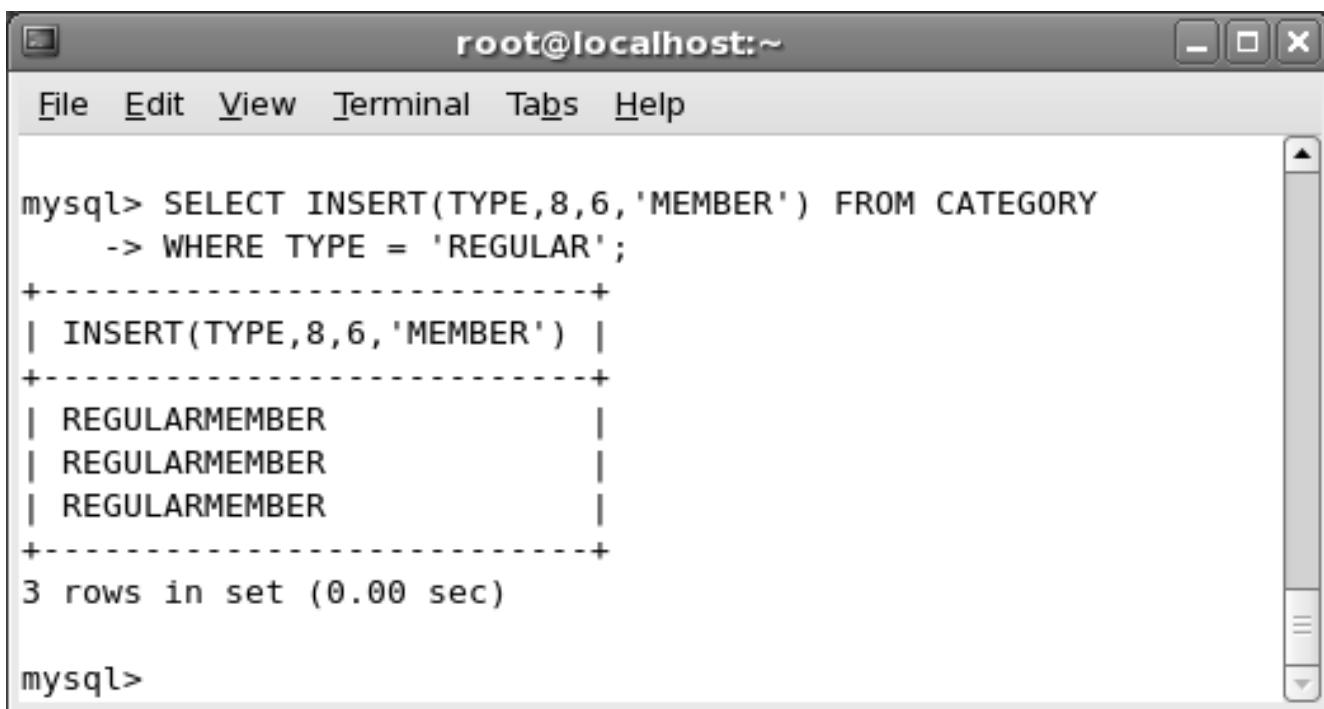
```
SELECT INSERT(TYPE, 8, 6, 'MEMBER') FROM CATEGORY WHERE TYPE =  
'REGULAR';
```

Figure 15.41 displays the output of the command.

Session 15

Using Basic Functions in MySQL - II (Lab)

Lab Guide



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The main area displays the following MySQL query and its results:

```
mysql> SELECT INSERT(TYPE,8,6,'MEMBER') FROM CATEGORY  
      -> WHERE TYPE = 'REGULAR';  
+-----+  
| INSERT(TYPE,8,6,'MEMBER') |  
+-----+  
| REGULARMEMBER           |  
| REGULARMEMBER           |  
| REGULARMEMBER           |  
+-----+  
3 rows in set (0.00 sec)  
  
mysql>
```

Figure 15.41: INSERT Function

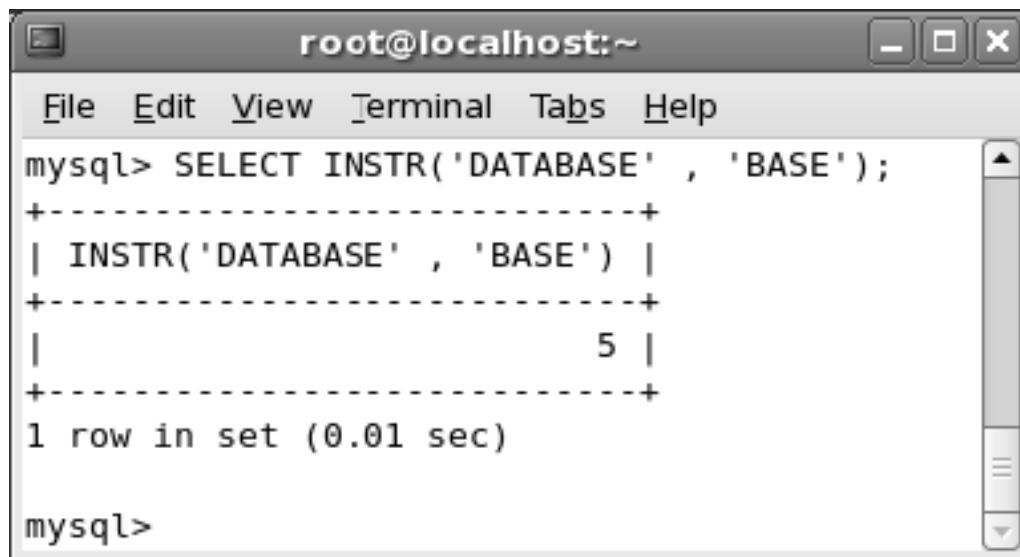
13. To display the position of the substring 'BASE' in the string 'DATABASE', enter the following command at the command prompt:

```
SELECT INSTR('DATABASE', 'BASE');
```

Figure 15.42 displays the output of the command.

Session 15

Using Basic Functions in MySQL - II (Lab)



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The command entered is "SELECT INSTR('DATABASE' , 'BASE');". The output is a single row with the value 5, indicating the position of the character 'B' in the string 'DATABASE'. The terminal prompt "mysql>" appears at the bottom.

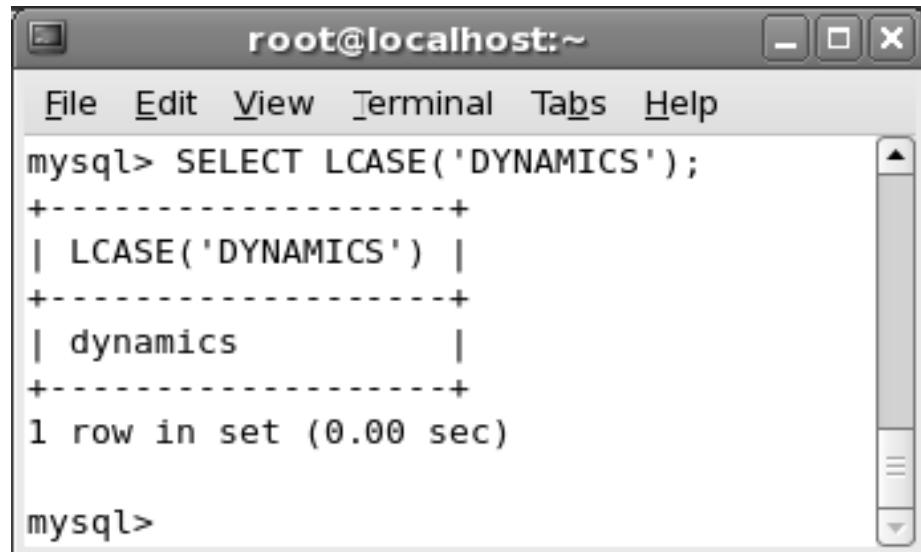
```
root@localhost:~  
File Edit View Terminal Tabs Help  
mysql> SELECT INSTR('DATABASE' , 'BASE');  
+-----+  
| INSTR('DATABASE' , 'BASE') |  
+-----+  
| 5 |  
+-----+  
1 row in set (0.01 sec)  
  
mysql>
```

Figure 15.42: INSTR Function

14. To convert the book names to lowercase, enter the following command at the command prompt:

```
SELECT BOOK_NAME, LCASE(BOOK_NAME) FROM BOOK_TABLE;
```

Figure 15.43 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The command entered is "SELECT LCASE('DYNAMICS');". The output is a single row with the value "dynamics", showing the lowercase conversion of the input string. The terminal prompt "mysql>" appears at the bottom.

```
root@localhost:~  
File Edit View Terminal Tabs Help  
mysql> SELECT LCASE('DYNAMICS');  
+-----+  
| LCASE('DYNAMICS') |  
+-----+  
| dynamics |  
+-----+  
1 row in set (0.00 sec)  
  
mysql>
```

Figure 15.43: LCASE Function

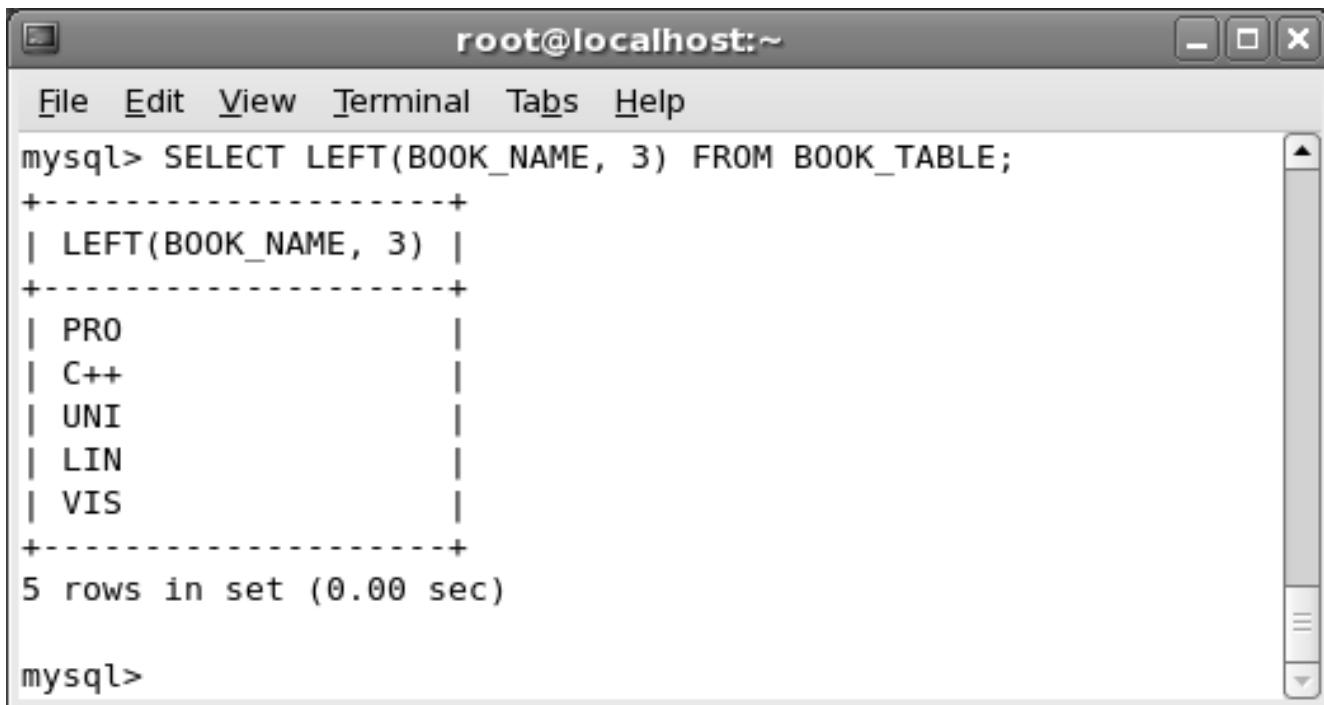
Session 15

Using Basic Functions in MySQL - II (Lab)

15. The librarian wants to view the first three characters of the names of the books in the library. To display the first three characters of the book name, enter the following command at the command prompt:

```
SELECT LEFT(BOOK_NAME, 3) FROM BOOK_TABLE;
```

Figure 15.44 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following text:

```
File Edit View Terminal Tabs Help
mysql> SELECT LEFT(BOOK_NAME, 3) FROM BOOK_TABLE;
+-----+
| LEFT(BOOK_NAME, 3) |
+-----+
| PRO          |
| C++          |
| UNI          |
| LIN          |
| VIS          |
+-----+
5 rows in set (0.00 sec)

mysql>
```

The terminal window has a standard Linux-style interface with a title bar, menu bar, and scroll bars.

Figure 15.44: LEFT Function

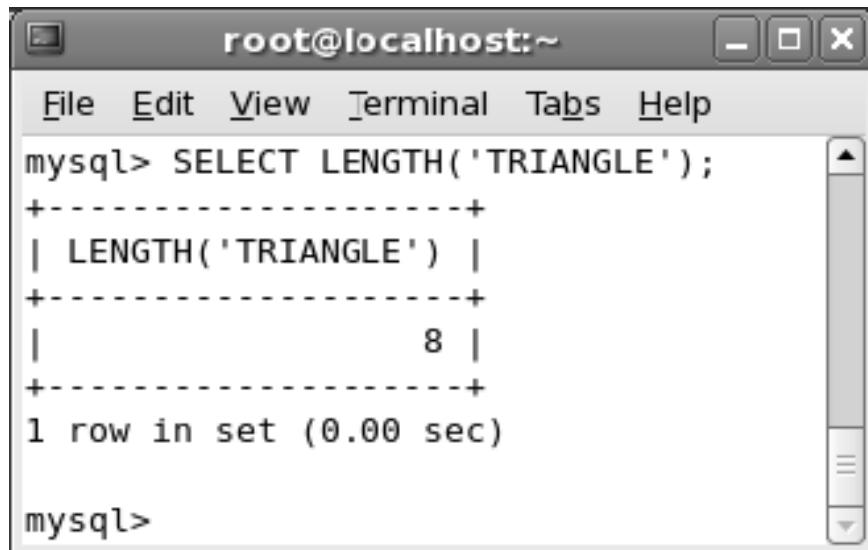
16. To display the name of the author and the number of characters present in the author field, enter the following command at the command prompt:

```
SELECT AUTHOR, LENGTH(AUTHOR) FROM BOOK_TABLE;
```

Figure 15.45 displays the output of the command.

Session 15

Using Basic Functions in MySQL - II (Lab)



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its output:

```
mysql> SELECT LENGTH('TRIANGLE');
+-----+
| LENGTH('TRIANGLE') |
+-----+
|          8         |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 15.45: LENGTH Function

17. The librarian wants to find the index position of the substring 'C' in the book name column. To find the index position of C in the BOOK_NAME column, enter the following command at the command prompt:

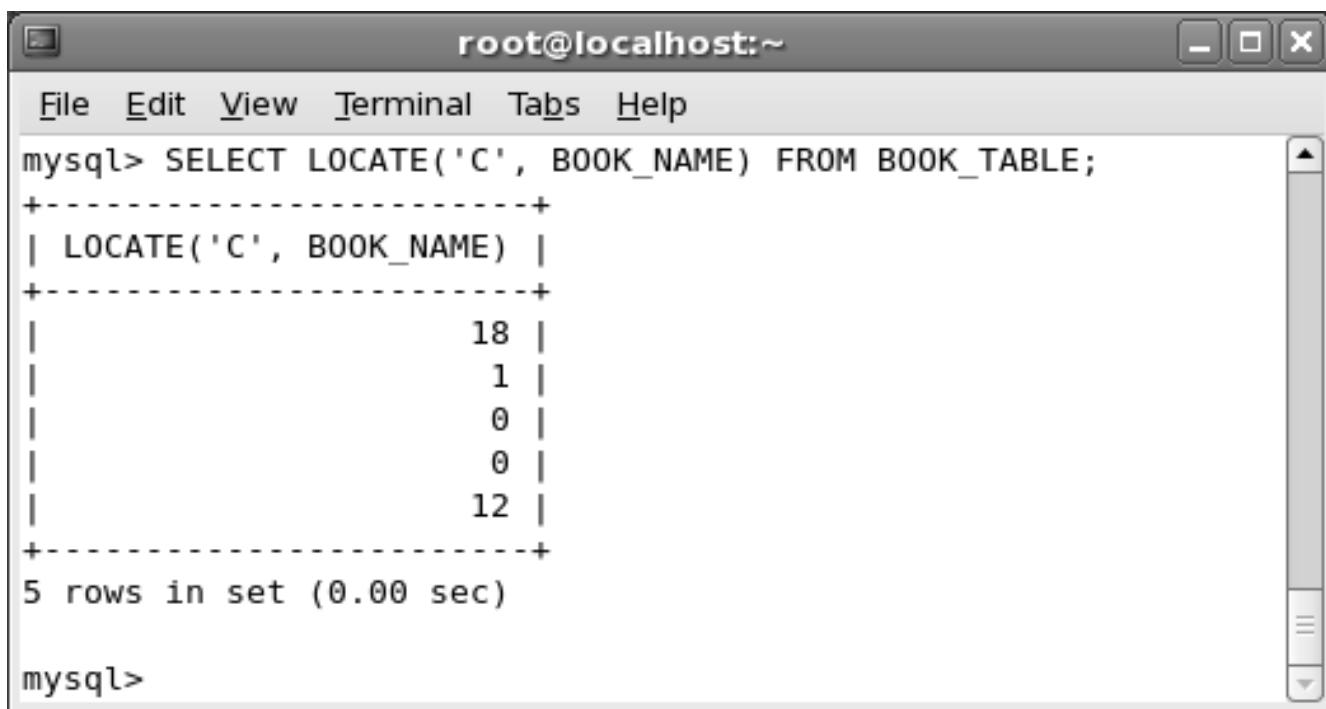
```
SELECT LOCATE('C', BOOK_NAME) FROM BOOK_TABLE;
```

Figure 15.46 displays the output of the command.

Session 15

Using Basic Functions in MySQL - II (Lab)

Lab Guide



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL command and its output:

```
mysql> SELECT LOCATE('C', BOOK_NAME) FROM BOOK_TABLE;
+-----+
| LOCATE('C', BOOK_NAME) |
+-----+
|      18   |
|        1   |
|        0   |
|        0   |
|      12   |
+-----+
5 rows in set (0.00 sec)

mysql>
```

The output shows five rows of results from the query, each containing a value from the LOCATE function applied to the BOOK_NAME column of the BOOK_TABLE.

Figure 15.46: LOCATE Function

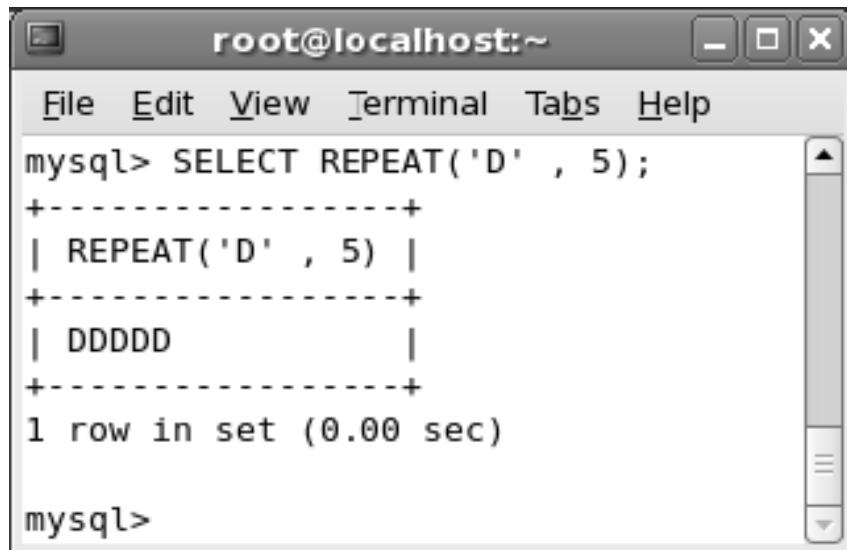
18. To replicate the USER_ID thrice, enter the following command at the command prompt:

```
SELECT USER_ID, REPEAT(USER_ID, 3)
FROM USER_DETAILS;
```

Figure 15.47 displays the output of the command.

Session 15

Using Basic Functions in MySQL - II (Lab)



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The main area displays the following MySQL command and its output:

```
mysql> SELECT REPEAT('D' , 5);
+-----+
| REPEAT('D' , 5) |
+-----+
| DDDDD |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 15.47: REPEAT Function

19. The librarian wants to replace the word REGULAR with REG for the membership type. To replace the letter 'REGULAR' with 'REG' in the field TYPE, enter the following command at the command prompt:

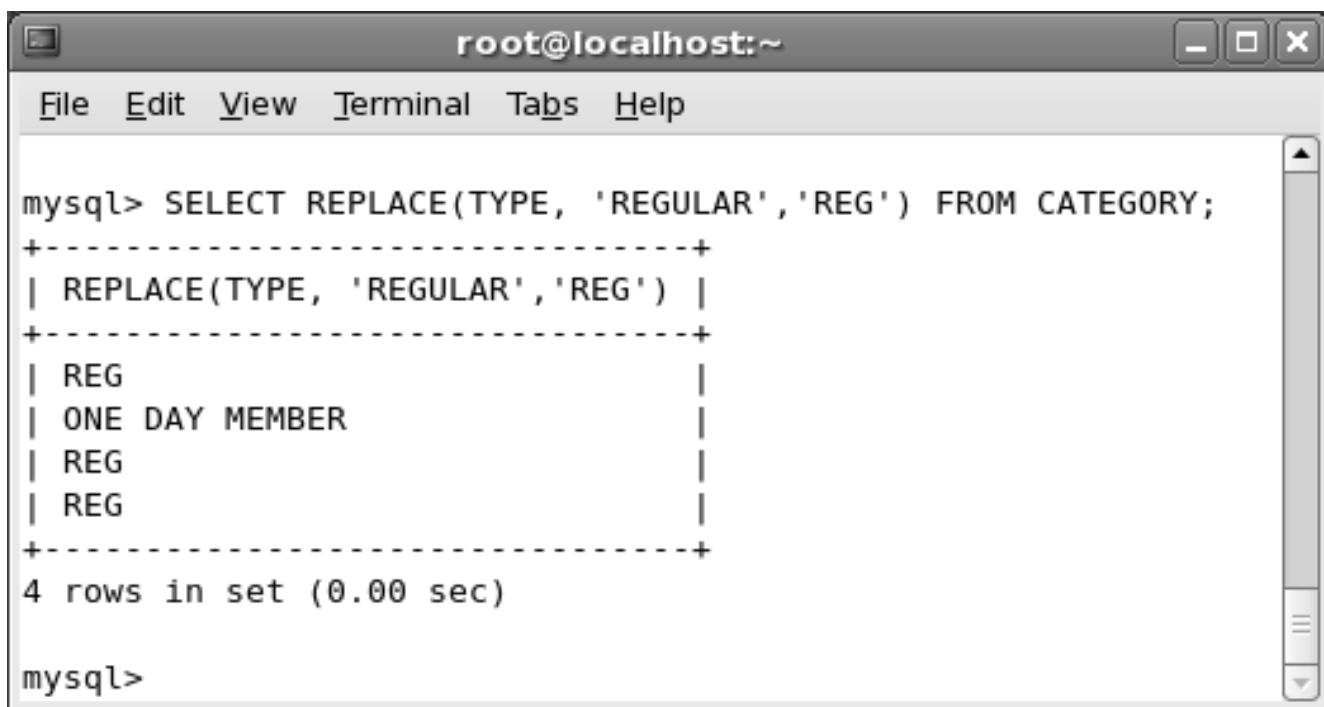
```
SELECT REPLACE(TYPE, 'REGULAR', 'REG') FROM CATEGORY;
```

Figure 15.48 displays the output of the command.

Session 15

Using Basic Functions in MySQL - II (Lab)

Lab Guide



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The main area displays the following MySQL command and its output:

```
mysql> SELECT REPLACE(TYPE, 'REGULAR', 'REG') FROM CATEGORY;
+-----+
| REPLACE(TYPE, 'REGULAR', 'REG') |
+-----+
| REG
| ONE DAY MEMBER
| REG
| REG
+-----+
4 rows in set (0.00 sec)

mysql>
```

The output shows four rows where the string 'REGULAR' has been replaced by 'REG'.

Figure 15.48: REPLACE Function

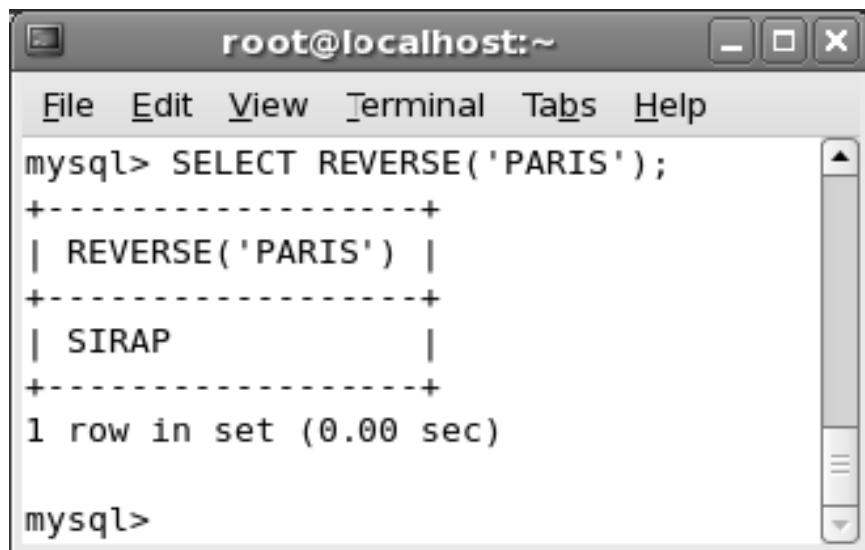
20. To invert the sequence of the characters of EDITION, enter the following command at the command prompt:

```
SELECT EDITION, REVERSE(EDITION)  
FROM BOOK_TABLE;
```

Figure 15.49 displays the output of the command.

Session 15

Using Basic Functions in MySQL - II (Lab)



A screenshot of a terminal window titled "root@localhost:~". The window shows the MySQL command-line interface. The user has run the command "SELECT REVERSE('PARIS');". The output is a single row: "+-----+", "| REVERSE('PARIS') |", "+-----+", "| SIRAP |", "+-----+", "1 row in set (0.00 sec)". The MySQL prompt "mysql>" is visible at the bottom.

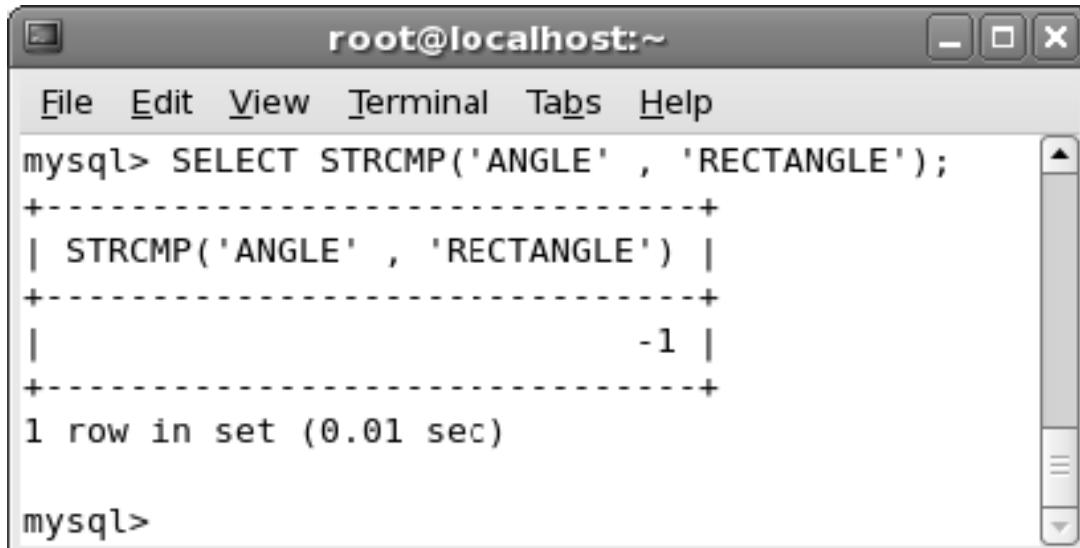
```
root@localhost:~  
File Edit View Terminal Tabs Help  
mysql> SELECT REVERSE('PARIS');  
+-----+  
| REVERSE('PARIS') |  
+-----+  
| SIRAP |  
+-----+  
1 row in set (0.00 sec)  
  
mysql>
```

Figure 15.49: REVERSE Function

21. To compare the strings 'ANGLE' and 'RECTANGLE', enter the following command at the command prompt:

```
SELECT STRCMP('ANGLE', 'RECTANGLE');
```

Figure 15.50 displays the output of the command.



A screenshot of a terminal window titled "root@localhost:~". The window shows the MySQL command-line interface. The user has run the command "SELECT STRCMP('ANGLE', 'RECTANGLE');". The output is a single row: "+-----+", "| STRCMP('ANGLE', 'RECTANGLE') |", "+-----+", "| -1 |", "+-----+", "1 row in set (0.01 sec)". The MySQL prompt "mysql>" is visible at the bottom.

```
root@localhost:~  
File Edit View Terminal Tabs Help  
mysql> SELECT STRCMP('ANGLE', 'RECTANGLE');  
+-----+  
| STRCMP('ANGLE', 'RECTANGLE') |  
+-----+  
| -1 |  
+-----+  
1 row in set (0.01 sec)  
  
mysql>
```

Figure 15.50: STRCMP Function

Session 15

Using Basic Functions in MySQL - II (Lab)

22. The librarian wants to remove all the extra spaces existing before the book names from the BOOK_TABLE table. To remove the leading spaces in BOOK_NAME from the BOOK_TABLE table, enter the following command at the command prompt:

```
SELECT TRIM(LEADING FROM BOOK_NAME) FROM BOOK_TABLE;
```

Figure 15.51 displays the output of the command.

The screenshot shows a terminal window titled "root@localhost:~". The window contains the following text:

```
File Edit View Terminal Tabs Help
mysql> SELECT TRIM(LEADING FROM BOOK_NAME) FROM BOOK_TABLE;
+-----+
| TRIM(LEADING FROM BOOK_NAME) |
+-----+
| PROGRAMMING WITH C           |
| C++ PROGRAMMING              |
| UNIX                          |
| LINUX                         |
| VISUAL BASIC                  |
+-----+
5 rows in set (0.00 sec)

mysql>
```

Figure 15.51: TRIM Function

System Information Functions in MySQL

System information functions are used to retrieve system-related information. The following steps illustrate the use of system information functions in MySQL:

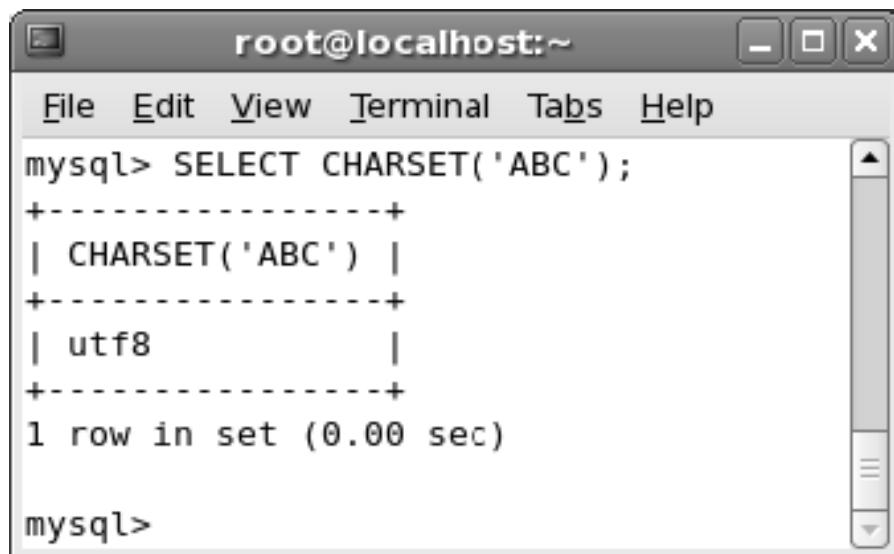
1. Will wants to know the charset being used in the active database, To display the character set being used for the string 'ABC', enter the following command at the command prompt:

```
SELECT CHARSET('ABC');
```

Figure 15.52 displays the output of the command.

Session 15

Using Basic Functions in MySQL - II (Lab)



A screenshot of a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area displays the following MySQL command and its output:

```
mysql> SELECT CHARSET('ABC');
+-----+
| CHARSET('ABC') |
+-----+
| utf8           |
+-----+
1 row in set (0.00 sec)

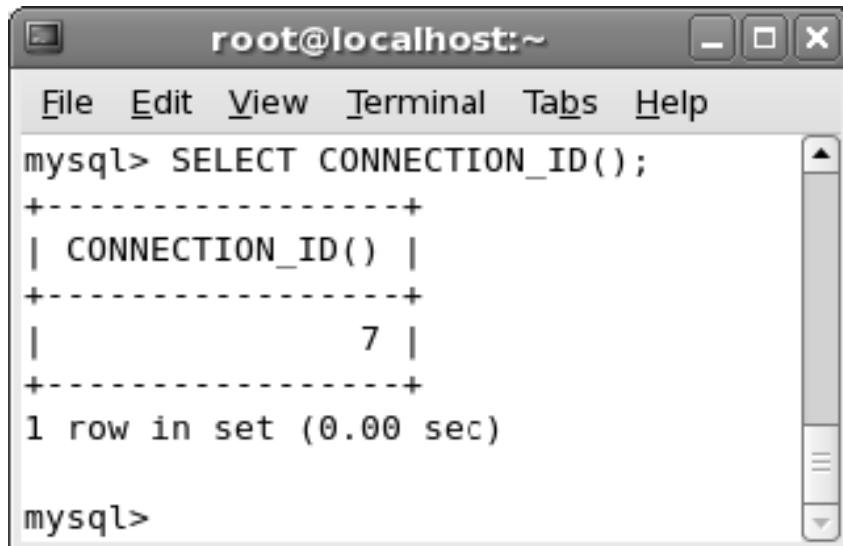
mysql>
```

Figure 15.52: CHARSET Function

2. Will wants to know the thread ID of the current connection. To display the thread id, enter the following command at the command prompt:

```
SELECT CONNECTION_ID();
```

Figure 15.53 displays the output of the command.



A screenshot of a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area displays the following MySQL command and its output:

```
mysql> SELECT CONNECTION_ID();
+-----+
| CONNECTION_ID() |
+-----+
|          7      |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 15.53: CONNECTION_ID Function

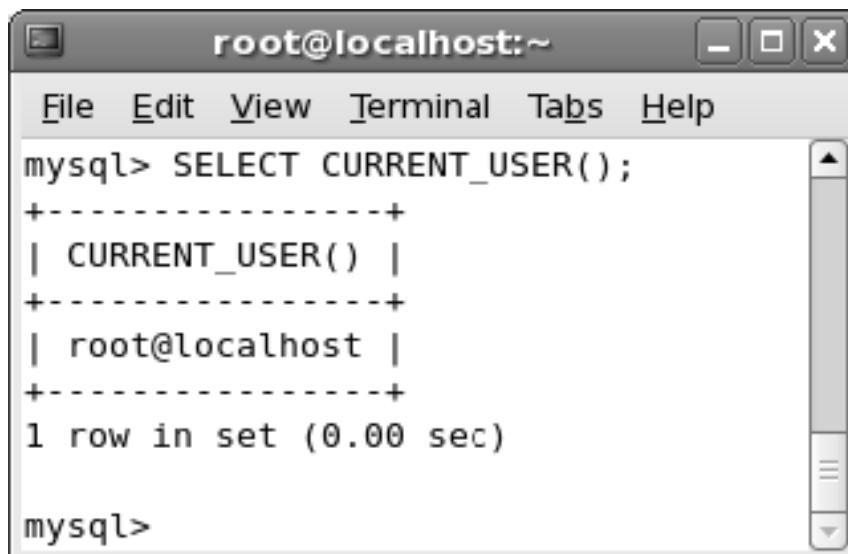
Session 15

Using Basic Functions in MySQL - II (Lab)

3. Will wants to know the current user name. To display the current user name, enter the following command at the command prompt:

```
SELECT CURRENT_USER();
```

Figure 15.54 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area displays the following MySQL command and its output:

```
mysql> SELECT CURRENT_USER();
+-----+
| CURRENT_USER() |
+-----+
| root@localhost |
+-----+
1 row in set (0.00 sec)

mysql>
```

Lab Guide

Figure 15.54: CURRENT_USER Function

4. To view the database currently in use, enter the following command at the command prompt:

```
SELECT DATABASE();
```

Figure 15.55 displays the output of the command.

Session 15

Using Basic Functions in MySQL - II (Lab)



A screenshot of a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area displays the following MySQL command and its output:

```
mysql> SELECT DATABASE();
+-----+
| DATABASE() |
+-----+
| LIBRARY   |
+-----+
1 row in set (0.01 sec)

mysql>
```

Figure 15.55: DATABASE Function

5. To display the version of MySQL in use, enter the following command at the command prompt:

```
SELECT VERSION();
```

Figure 15.56 displays the output of the command.



A screenshot of a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area displays the following MySQL command and its output:

```
mysql> SELECT VERSION();
+-----+
| VERSION()      |
+-----+
| 5.1.56-community |
+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 15.56: VERSION Function

Session 15

Using Basic Functions in MySQL - II (Lab)



Do It Yourself

Lab Guide

1. Display the current date.
2. Display the current time.
3. Add an interval of 1 year to the date 2011-01-20.
4. Subtract 2 days from the current date using the `DATE_SUB` function.
5. Display the day of month for the date 2009-06-25.
6. Display the day of week for the date 2007-08-11.
7. Display the day of year for 2009-10-20.
8. Display the microseconds from 09:15:45.123456.
9. Display the minute part from 05:30:45.
10. Display the month from the date 2010-12-05.
11. Calculate the difference in the number of months between the periods 199905 and 199604.
12. Calculate the quarter for the date 2010-04-05.
13. Display the seconds from 05:30:45.
14. Display the current date and time value.
15. Convert the time expression 10:30:00 into seconds.
16. Display the time part from the expression 2010-04-15 20:20:30.
17. Calculate the time difference between 21:30:00 and 23:30:00.
18. Display the current timestamp value.
19. Convert the date 2010-04-05 into seconds.

Session 15

Using Basic Functions in MySQL - II (Lab)



Do It Yourself

20. Display the current UTC DATE.
21. Display the current UTC TIME.
22. Calculate the week number for the year 2005-04-20.
23. Display the year from the date 2010-12-05.
24. Calculate the length of string ABC in bits.
25. Calculate the character length of the string XYZ.
26. Join the strings DATA and BASE with a comma as the separator.
27. Display the index of the field C in the string A, B, A, C, U, S.
28. Calculate the hexadecimal value of the string DATA.
29. Convert the string DYNAMICS to lowercase.
30. Delete the leading whitespace from the string ' DATA'.
31. Calculate and display the ASCII value of the first character of JOHN.
32. Replicate the string D five times.
33. Display only the last five characters of the string TRIANGLE as output.
34. Delete the trailing whitespace of the string 'XYZ '.
35. Display a string with only five spaces.
36. Convert the string abc to uppercase.
37. Display the character for the hexadecimal string EF.
38. Use the BENCHMARK function to execute 1,000 nine times.

Session 15

Using Basic Functions in MySQL - II (Lab)



Do It Yourself

39. Display the thread ID of the current connection.
40. Display the row count.
41. Display the current username and hostname.

WRITE-UPS BY

EXPERTS AND LEARNERS

TO PROMOTE NEW AVENUES AND
ENHANCE THE LEARNING EXPERIENCE



FOR FURTHER READING, LOGIN TO

www.onlinevarsity.com

Objectives

At the end of this session, the student will be able to:

- *Describe the creation of user accounts in MySQL.*
- *Identify the privileges in MySQL.*
- *Explain the privileges present in MySQL.*
- *Explain the commands for setting up of restrictions in MySQL.*

16.1 Introduction

A user account enables you to access a database. You require a user account to manage a database. User accounts enable you to add new users, create, modify, and delete tables and databases, and set privileges or access rights to other user accounts.

In this session, you will learn to create a user account, assign, and alter privileges to users. You will also learn to use privileges and control user access to databases. In addition, you will learn to rename a user account and delete a user account.

16.2 User Accounts

In MySQL, a user is a record present in the user table of the MySQL server. The table stores the username and password. After the installation of MySQL server is complete, the only existing user account is `root` user account. MySQL uses the `root` user account to execute administrative commands. A `root` user has unrestricted access across the database. All users are required to have a user account to access the database. MySQL server does not require a password for the `root` user, if you have not set a password at configuration. MySQL allows the `root` user to set a password using the `mysqladmin` command. MySQL stores user accounts and privileges as tables in a default database named `mysql`. In MySQL, you can create a new user account by using the `CREATE USER` or `GRANT` command. In addition, you can also create user accounts by manipulating the MySQL grant tables directly.

After an account is created, you can set the password, modify and revoke the privileges, rename the account, and delete the account. MySQL provides different commands and options to execute these tasks. Table 16.1 lists the commands for user accounts in MySQL.

MySQL Command	Description
<code>CREATE USER</code>	Allows the creation of a new account

Session 16

Controlling and Managing MySQL database

MySQL Command	Description
SET PASSWORD	Allows to assign a password
GRANT	Allows to assign privileges
RENAME USER	Allows to change the name of the account
REVOKE	Allows to cancel or remove privileges
DROP USER	Allows to delete an account

Table 16.1: User Account Tasks and MySQL Commands

You must login as the `root` user to create user accounts in MySQL.

Note: To connect as the `root` user, enter the following command at the command prompt:

```
mysql - u root -p
```

The MySQL Grant tables contain information about the privileges to user accounts. These tables are stored in the `mysql` database and their names are as follows:

- `columns_priv` – contains access details for columns in tables
- `db` – contains information of user accounts that can access the database
- `host` – contains list of user accounts that have access to the database
- `tables_priv` – contains access details for tables in a database
- `user` – contains access details for a user account

16.2.1 Adding User Accounts Using the CREATE USER Command

MySQL server provides the `CREATE USER` command to add new accounts. You must have the global `CREATE USER` privilege for the `mysql` database to execute the command. MySQL server creates a new row in the `user` table of `mysql` database and assigns privileges when you add a new user account.

You must login as the `root` user and connect to the `mysql` database to create user accounts.

To view the list of available databases, enter the following command at the command prompt:

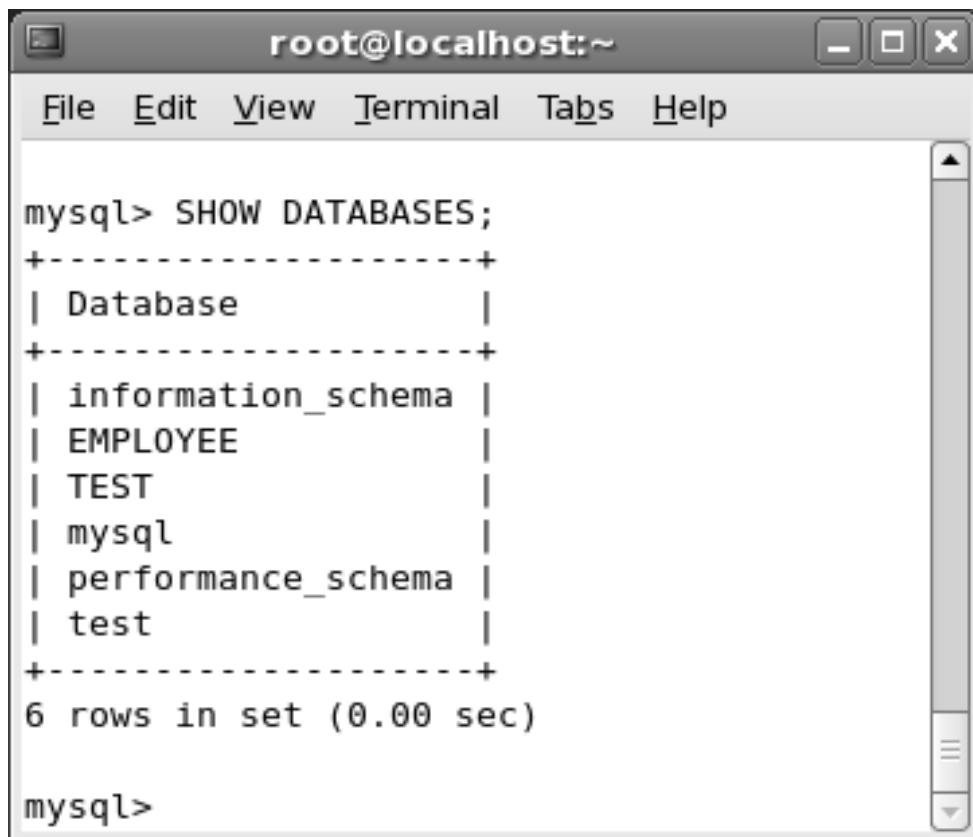
```
SHOW DATABASES;
```

Figure 16.1 displays a list of available databases, including `mysql`, the default database.

Session 16

Controlling and Managing MySQL database

Concepts



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area displays the following SQL command and its results:

```
mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| EMPLOYEE |
| TEST |
| mysql |
| performance_schema |
| test |
+-----+
6 rows in set (0.00 sec)

mysql>
```

Figure 16.1: Default mysql Database

Note: Ensure that the root user has the INSERT and RELOAD administrative privileges.

To load the default `mysql` database, enter the following command at the command prompt:

```
USE mysql;
```

This command will change the current database to `mysql`.

Figure 16.2 displays the output of the command.

Session 16

Controlling and Managing MySQL database

Concepts

```
root@localhost:~
```

```
File Edit View Terminal Tabs Help
```

```
mysql> USE mysql;
Reading table information for completion o
f table and column names
You can turn off this feature to get a qui
cker startup with -A

Database changed
mysql>
```

Figure 16.2: Activating mysql Database

To list the tables present in the default database `mysql`, enter the following command at the command prompt:

```
SHOW TABLES;
```

This command, lists all the available tables in the default database.

Figure 16.3 displays the output of the command.

Session 16

Controlling and Managing MySQL database

Concepts

The screenshot shows a terminal window titled "root@localhost:~". The window contains the following text:

```
mysql> SHOW TABLES;
+-----+
| Tables_in_mysql |
+-----+
| columns_priv    |
| db              |
| event           |
| func            |
| general_log     |
| help_category   |
| help_keyword    |
| help_relation   |
| help_topic      |
| host             |
| ndb_binlog_index|
| plugin          |
| proc             |
| procs_priv      |
| proxies_priv    |
| servers          |
| slow_log         |
| tables_priv     |
| time_zone        |
| time_zone_leap_second |
| time_zone_name  |
| time_zone_transition |

```

Figure 16.3: Tables in mysql Database

In figure 16.3, the `mysql` database lists the tables that describe the user access privileges. Table 16.2 lists the functions of some of the tables in the `mysql` database.

Table Name	Description
columns_priv	Contains access details for columns in tables

Session 16

Controlling and Managing MySQL database

Table Name	Description
db	Contains information of user accounts that can access the database
func	Contains list of functions and is updated at MySQL server restart
host	Contains list of user accounts that have access to the database
proc	Contains information about database and table modifications
tables_priv	Contains access details for tables in a database
user	Contains access details for a user account

Table 16.2: Default Tables in mysql Database

There are different commands for creating a user account. However, all these commands require you to connect as the `root` user and change the database to `mysql`. The `CREATE USER` command allows you to add a user account.

To add a user account using the `CREATE USER` command, enter the following command at the command prompt:

```
CREATE USER 'temp_user'@'localhost';
```

This command will add a new user named `temp_user` for the instance `localhost` of MySQL server.

16.2.2 Creating User Account Using the GRANT Command

You can also create a new user account using the `GRANT` command. In addition, you can also use the `GRANT` command to assign rights and privileges.

You can grant and revoke rights to a MySQL user account at four levels that are as follows:

- **Global Level** - grants and revokes privileges that apply to all the databases present on the server. These privileges are stored in the `user` table of the `mysql` database.
- **Database Level** - grants and revokes privileges that apply to all the tables of a given database. These privileges are stored in the `db` and `host` tables of the `mysql` database.
- **Table Level** - grants and revokes privileges that apply to all the columns of a given table. These privileges are stored in the `tables_priv` table of the `mysql` database.
- **Column Level** - grants and revokes privileges that apply to a single column in a given table. These privileges are stored in the `columns_priv` table of the `mysql` database.

Table 16.3 lists the privilege types that can be specified in the `GRANT` and `REVOKE` statements.

Session 16

Controlling and Managing MySQL database

Concepts

Privilege	Description
ALL [PRIVILEGES]	Assigns all rights except GRANT OPTION. The user cannot set access controls.
ALTER	Allows the use of the ALTER TABLE command
CREATE	Allows the use of the CREATE TABLE command
CREATE TEMPORARY TABLES	Allows the use of the CREATE TEMPORARY TABLE command
DELETE	Allows the use of the DELETE command
DROP	Allows the use of the DROP TABLE command
EXECUTE	Allows the user to run stored procedures
INDEX	Allows the use of the CREATE INDEX and DROP INDEX commands
INSERT	Allows the use of the INSERT command
SELECT	Allows the use of the SELECT command
SHOW DATABASES	Allows the use of the SHOW DATABASES command
SHUTDOWN	Allows the use of the mysqladmin shutdown privilege
UPDATE	Allows the use of the UPDATE command
USAGE	Allows the use of the 'no privileges' user account
GRANT OPTION	Allows privileges to be granted or revoked from other accounts

Table 16.3: Privileges in GRANT and REVOKE commands

Note: You will have to execute the `mysql_upgrade` script before using some of these commands.

The syntax for creating a new user with all privileges for all the tables in the database and who can connect to the localhost or any other host is as follows:

```
GRANT ALL [PRIVILEGES] ON *.* TO user_name [IDENTIFIED BY [PASSWORD] 'password'  
[WITH GRANT OPTION]]
```

where,

`user_name` – specifies the name for the new account

`[IDENTIFIED BY [PASSWORD]]` – sets the password for the new account

`password` – specifies the password for the account

`[WITH GRANT OPTION]` – allows the user to set privileges

In the syntax, the `PASSWORD` and the `GRANT OPTION` clauses are optional.

Session 16

Controlling and Managing MySQL database

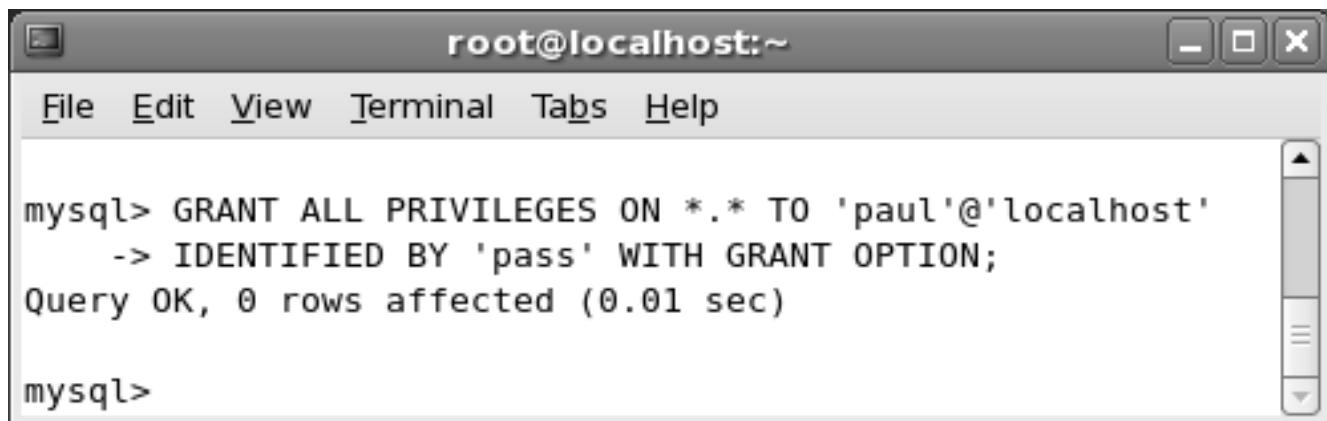
Note: The GRANT command supports table, database, column, and routine names upto 64 characters in length. Length of user name cannot be more than 16 characters long, while host names cannot be more than 60 characters long.

To create a new user with all the privileges and access to MySQL from localhost, enter the following command at the command prompt:

```
GRANT ALL PRIVILEGES ON *.* TO 'paul'@'localhost' IDENTIFIED BY 'pass' WITH  
GRANT OPTION;
```

The command will assign all privileges to the user `paul` present in the instance, `localhost`. The clause `WITH GRANT OPTION` will allow the user to set privileges.

Figure 16.4 displays the output of the GRANT command.



A screenshot of a terminal window titled "root@localhost:~". The window has a standard Linux-style interface with a menu bar (File, Edit, View, Terminal, Tabs, Help) and a scroll bar on the right. The terminal session shows the following command being entered and its output:

```
mysql> GRANT ALL PRIVILEGES ON *.* TO 'paul'@'localhost'  
-> IDENTIFIED BY 'pass' WITH GRANT OPTION;  
Query OK, 0 rows affected (0.01 sec)  
  
mysql>
```

Figure 16.4: Create User Using the GRANT command

In figure 16.4, a new user named `paul` with the password `pass` is created. MySQL server has assigned all the rights on all the databases present on this instance of MySQL server named `localhost`, to the user. Such a user is referred to as a super user.

Similarly, to create a new user with all the privileges without specifying the `localhost` or any other host, enter the following command at the command prompt:

```
GRANT ALL PRIVILEGES ON *.* TO 'paul'@'%' IDENTIFIED BY 'pass' WITH GRANT  
OPTION;
```

Figure 16.5 displays the output of the GRANT command.

Session 16

Controlling and Managing MySQL database

Concepts

```
root@localhost:~
```

File Edit View Terminal Tabs Help

```
mysql> GRANT ALL PRIVILEGES ON *.* TO 'paul'@'%'
-> IDENTIFIED BY 'pass' WITH GRANT OPTION;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

Figure 16.5: Creating a User Without Specifying the Localhost

In figure 16.5, MySQL creates a new user named `paul` with the password set as `pass`. This user is also a super user who has all the rights on all hosts.

16.2.3 Create User Account Using the INSERT command

MySQL also provides the `INSERT` command to create user accounts. The `INSERT` command adds a new entity to the `user` table of the `mysql` database. This table contains information of all user accounts in the database with their hostnames, passwords, and privileges. The newly created user can be assigned the rights using the `GRANT` command.

Note: To grant rights to the new user, set the privilege columns to `Y`; where `Y` stands for Yes. Thus, the user is granted only those privileges, which are set to `Y`.

To create a user by the name `martin` using the `INSERT` command, enter the following command at the command prompt:

```
INSERT INTO user VALUES ('localhost','martin', PASSWORD ('pass'),'Y','Y','Y',
 'Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y');
```

The command will add a record into the `user` table with the values specified in the `VALUES` clause.

Figure 16.6 displays the output of the command.

Session 16

Controlling and Managing MySQL database

Figure 16.6: Creating User Using INSERT Command

In figure 16.6, `user` is a table in the `mysql` default database that stores details of the user accounts and the privileges granted to the user. The `localhost` is the instance for the newly created user called `martin`. The password for the user has been set to `pass`. The values in the privilege columns have been set as `Y`.

Note: The command used in figure 16.6 will not execute in MySQL installed on the Microsoft Windows operating system.

Table 16.4 lists some of the column names in the user table of mysql database.

Column Name
Select_priv
Insert_priv
Update_priv
Delete_priv
Index_priv
Alter_priv
Create_priv
Drop_priv
Grant_priv
Create_view_priv
Show_view_priv
Create_routine_priv
Alter_routine_priv
Execute_priv
Create_tmp_table_priv
Lock_tables_priv
References_priv
Reload_priv
Shutdown_priv

Session 16

Controlling and Managing MySQL database

Concepts

Column Name
Process_priv
File_priv
Show_db_priv
Super_priv
Repl_slave_priv
Repl_client_priv
Create_user_priv

Table 16.4: Privilege Columns in user Table

After creating a user with the `INSERT` command, you must run the `FLUSH PRIVILEGES` command. This command reloads all the grant tables and assigns the new privileges.

The syntax for the `FLUSH PRIVILEGES` command is:

```
FLUSH PRIVILEGES;
```

where,

`FLUSH` – specifies the reloading of the object

`PRIVILEGES` – specifies to refresh and reload the privileges

Note: MySQL will not assign the new privileges until you execute the `FLUSH PRIVILEGES` command.

Figure 16.7 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following text:

```
File Edit View Terminal Tabs Help
, 'Y', 'Y',
', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y',
Query OK, 1 row affected, 5 warnings (0.00 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

Figure 16.7: FLUSH Privileges

Session 16

Controlling and Managing MySQL database

To create a user having only RELOAD and PROCESS privileges, enter the following command at the command prompt:

```
INSERT INTO user SET Host='localhost', User='john', Reload_priv='Y', Process_priv='Y';
```

The command will add an entry into the user table and will set the value in the Host, User, Reload_priv, and Process_priv columns only. The Host column contains the instance name of MySQL. The Reload_priv and Process_priv assigns the RELOAD and PROCESS privileges.

Figure 16.8 displays the output of the command.

The screenshot shows a terminal window titled "root@localhost:~". The window has a standard Linux-style interface with a menu bar (File, Edit, View, Terminal, Tabs, Help) and a scroll bar on the right. The main area contains the following text:

```
mysql> INSERT INTO user
    -> SET Host = 'localhost',
    -> User = 'john',
    -> Reload_priv = 'Y',
    -> Process_priv = 'Y';
Query OK, 1 row affected, 4 warnings (0.01 sec)
)

mysql>
```

Figure 16.8: Specific Privileges For User

In figure 16.8, the hostname is set as localhost for the user named john with the help of SET command. The user is created without any password. Only the Reload_priv and the Process_priv privilege columns are set to Y.

Note: The command used in figure 16.8 will not work in MySQL installed on Microsoft Windows operating system.

To create an account without any access and password privileges, enter the following command at the command prompt:

```
INSERT INTO user (Host, User, Password) VALUES ('localhost','martin', '');
```

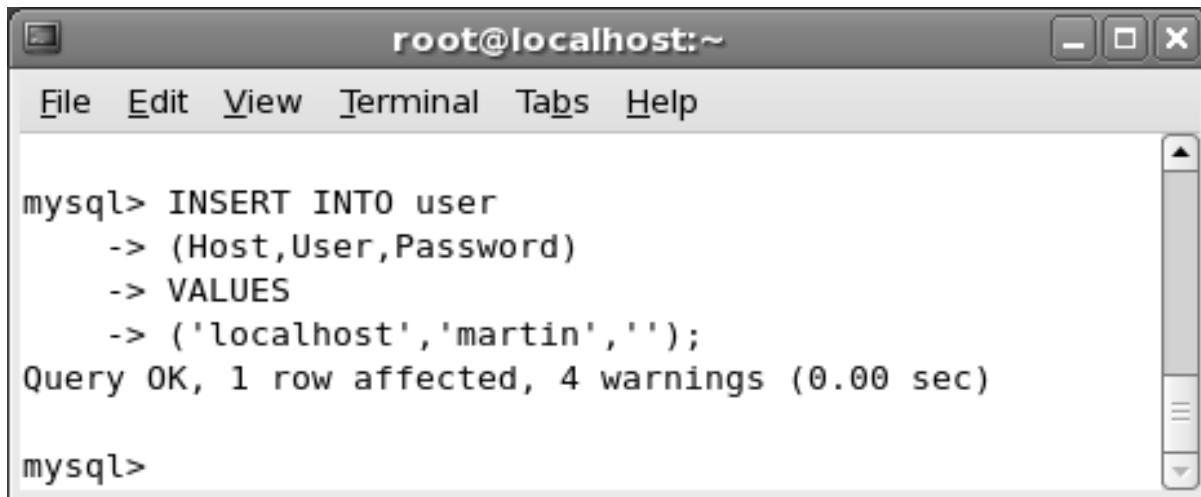
Session 16

Controlling and Managing MySQL database

The command adds a new entry into the user table and specifies the hostname, user name, and password.

Figure 16.9 displays the output of the command.

Concepts



```
root@localhost:~  
File Edit View Terminal Tabs Help  
  
mysql> INSERT INTO user  
-> (Host,User,Password)  
-> VALUES  
-> ('localhost','martin','');  
Query OK, 1 row affected, 4 warnings (0.00 sec)  
  
mysql>
```

Figure 16.9: Create User Without Setting Any Rights

In figure 16.9, the Host, User, and Password columns in the user table are updated. By default, MySQL assigns 'N' to all privilege columns if you do not specify the values for the privilege columns.

Note: Ensure to execute the `FLUSH PRIVILEGES` command after making changes to the grant tables. Also, the command used in figure 16.9 will not work in MySQL installed on Microsoft Windows operating system.

Ensuring Valid Connections

MySQL uses security for all connections and queries. The server controls access by ensuring that the following checks are performed:

- Checks whether the user has privileges to connect to the server. If the user has valid credentials, the connection is established.
- Verifies the privileges assigned to the user after the connection is established. MySQL executes the requested action if and only if the user has the necessary privileges.

Note: To control the users accessing the database, use administrative commands such as `mysql` or `mysqladmin`. To use the administrative commands, remember to login as the `root` user.

Session 16

Controlling and Managing MySQL database

16.2.4 Rename User Account

MySQL enables you to change the name of an existing user account. You can edit the name of the account using the `RENAME USER` command. MySQL enables you to change the name of the account without modifying the privileges. However, you must have the `UPDATE` or the `CREATE USER` privilege for the `mysql` database to execute the `RENAME USER` command.

For example, to rename a user account, enter the following command at the command prompt:

```
RENAME USER 'temp_user'@'localhost' TO 'temp-user1'@'localhost';
```

The command changes the name of the object by specifying the type of object to rename.

16.3 Altering Privileges

Managing privileges of users involves assigning and revoking user rights and privileges. You can use database-independent SQL commands, such as `GRANT` and `REVOKE` to manage user privileges. You can assign rights and privileges to an account using the `GRANT` command. The `REVOKE` command enables you to remove the rights granted to the user.

To alter privileges for the user, use `GRANT` command with selective privileges. For example, the command to grant permission to a user to read data from a specified table is as follows:

```
GRANT SELECT ON table_name TO user_name;
```

where,

`GRANT` – assigns privileges or rights

`SELECT` – specifies the type of privilege or right to assign

`table_name` – specifies the name of the table

`user_name` – specifies the name of the user or account

The command assigns the `SELECT` permission to the user specified in the `user_name` clause of the statement. The user can read data from the table specified in the `table_name` clause of the statement.

For example, to create a user who has the permission to read data from the `EMP_DETAILS` table, enter the following command at the command prompt:

```
GRANT SELECT ON EMP_DETAILS TO 'carol'@'localhost';
```

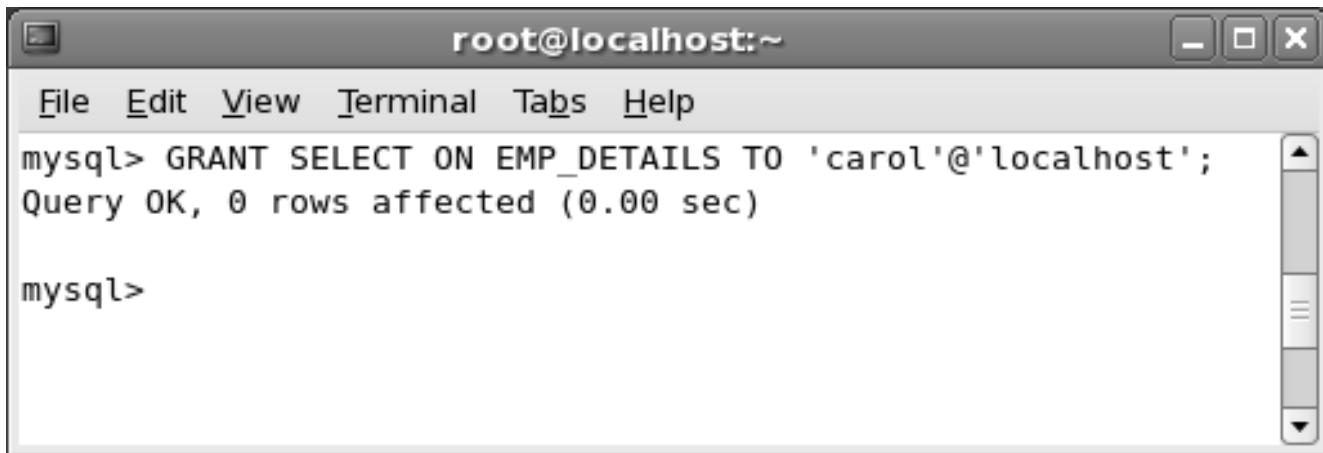
Session 16

Controlling and Managing MySQL database

The command will assign the SELECT privilege to the user, carol.

Figure 16.10 displays the output of the command.

Concepts



A screenshot of a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area shows the MySQL command-line interface. The user has entered the command "GRANT SELECT ON EMP_DETAILS TO 'carol'@'localhost';" followed by a semicolon. The response "Query OK, 0 rows affected (0.00 sec)" is displayed below the command. The prompt "mysql>" appears at the bottom of the window. A vertical scroll bar is visible on the right side of the terminal window.

```
root@localhost:~\nFile Edit View Terminal Tabs Help\nmysql> GRANT SELECT ON EMP_DETAILS TO 'carol'@'localhost';\nQuery OK, 0 rows affected (0.00 sec)\n\nmysql>
```

Figure 16.10: Create User Using GRANT command

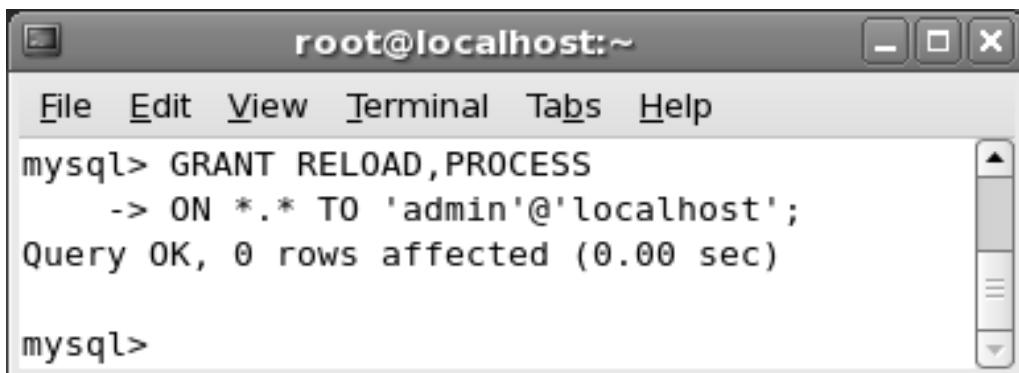
In figure 16.10, the user named carol is created and granted the privilege of reading data from the EMP_DETAILS table.

Note: The user has not been assigned any password and has access from all hosts.

Similarly, to create a user who is granted only with the RELOAD and the PROCESS rights for the entire database, with no password and has access only from the host named, localhost, enter the following command at the command prompt:

```
GRANT RELOAD,PROCESS ON *.* TO 'admin'@'localhost';
```

Figure 16.11 displays the output of the command.



A screenshot of a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area shows the MySQL command-line interface. The user has entered the command "GRANT RELOAD,PROCESS" followed by a semicolon. Below it, there is another line starting with "-> ON *.* TO 'admin'@'localhost';". The response "Query OK, 0 rows affected (0.00 sec)" is displayed below the command. The prompt "mysql>" appears at the bottom of the window. A vertical scroll bar is visible on the right side of the terminal window.

```
root@localhost:~\nFile Edit View Terminal Tabs Help\nmysql> GRANT RELOAD,PROCESS\n-> ON *.* TO 'admin'@'localhost';\nQuery OK, 0 rows affected (0.00 sec)\n\nmysql>
```

Figure 16.11: Create User with RELOAD and PROCESS Privileges

Session 16

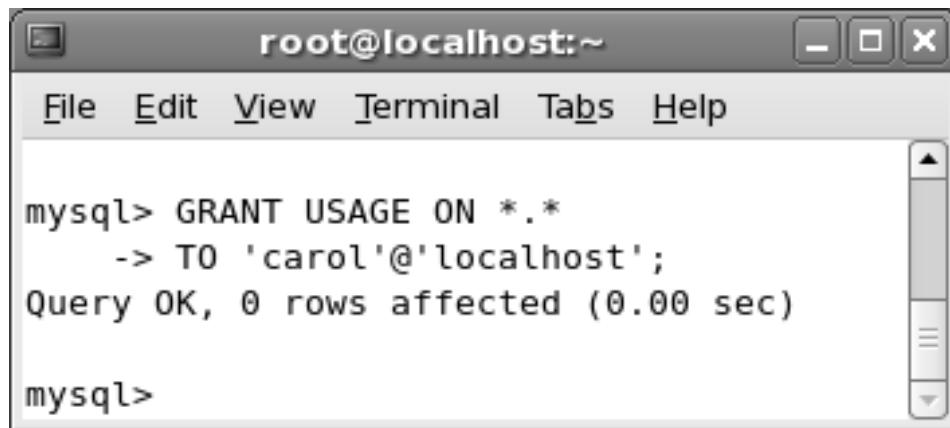
Controlling and Managing MySQL database

In figure 16.11, the RELOAD and PROCESS privileges are granted to the admin user for executing the mysqladmin reload, mysqladmin refresh, mysqladmin flush-* , and mysqladmin processlist commands.

To grant the USAGE privilege to the user, enter the following command at the command prompt:

```
GRANT USAGE ON *.* TO 'carol'@'localhost';
```

Figure 16.12 displays the output of the command.



```
root@localhost:~  
File Edit View Terminal Tabs Help  
  
mysql> GRANT USAGE ON *.*  
-> TO 'carol'@'localhost';  
Query OK, 0 rows affected (0.00 sec)  
  
mysql>
```

Figure 16.12: Creating User with the USAGE Privilege

In figure 16.12, MySQL assigns only the USAGE privilege to the new user named carol.

The USAGE privilege allows you to create an account without assigning any privileges. You can use such accounts only to establish a connection. The USAGE option sets all the global privileges to N. This privilege assumes that the user will grant specific privileges to the account later.

To create a user who has access to the EMPLOYEE database from localhost only and with selective rights, enter the following command at the command prompt:

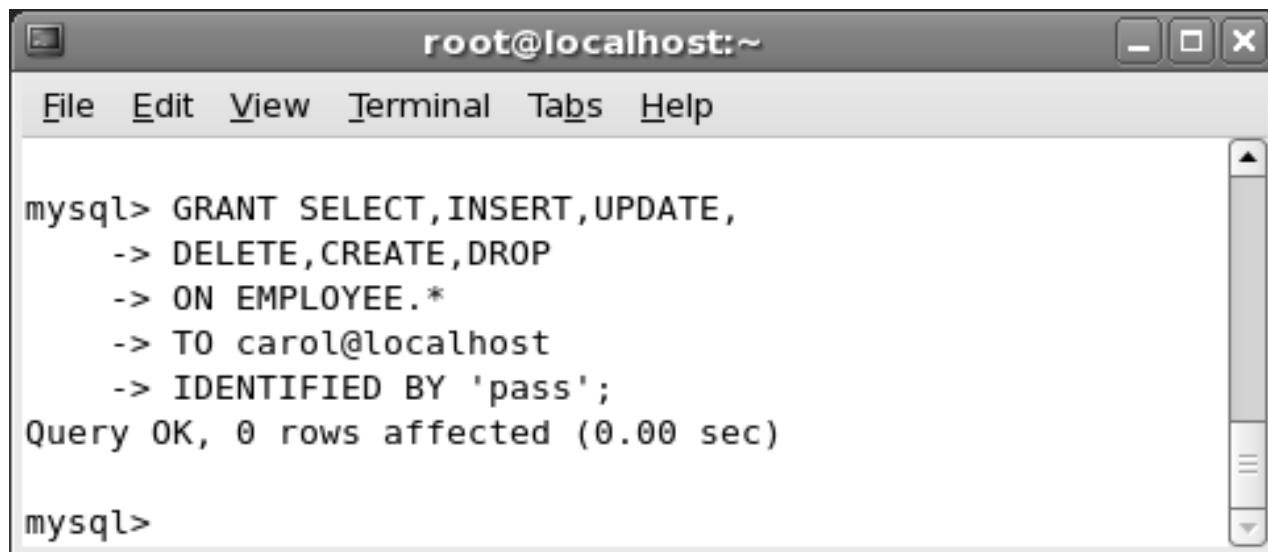
```
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP ON EMPLOYEE.* TO carol@  
localhost IDENTIFIED BY 'pass';
```

Figure 16.13 displays the output of the command.

Session 16

Controlling and Managing MySQL database

Concepts



The screenshot shows a terminal window titled "root@localhost:~". The menu bar includes "File", "Edit", "View", "Terminal", "Tabs", and "Help". The main area displays the following MySQL command and its execution:

```
mysql> GRANT SELECT,INSERT,UPDATE,
-> DELETE,CREATE,DROP
-> ON EMPLOYEE.*
-> TO carol@localhost
-> IDENTIFIED BY 'pass';
Query OK, 0 rows affected (0.00 sec)

mysql>
```

Figure 16.13: Create User with Selective Rights

In figure 16.13, MySQL assigns limited rights, such as select, insert, update, delete, create, and drop tables to the user name `carol` in the `EMPLOYEE` database.

The syntax to grant access to a specific database is:

```
GRANT privilege ON {db_name.*} TO user_name [IDENTIFIED BY [PASSWORD]
'password' [WITH GRANT OPTION]]
```

where,

`GRANT` – specifies to assign privileges

`privilege` – specifies the type of permission to assign

`db_name` – specifies the name of the database

`user_name` – specifies the name of the account

`[IDENTIFIED BY [PASSWORD]]` – sets a password for the user account

`[WITH GRANT OPTION]` – allows the account to set privileges

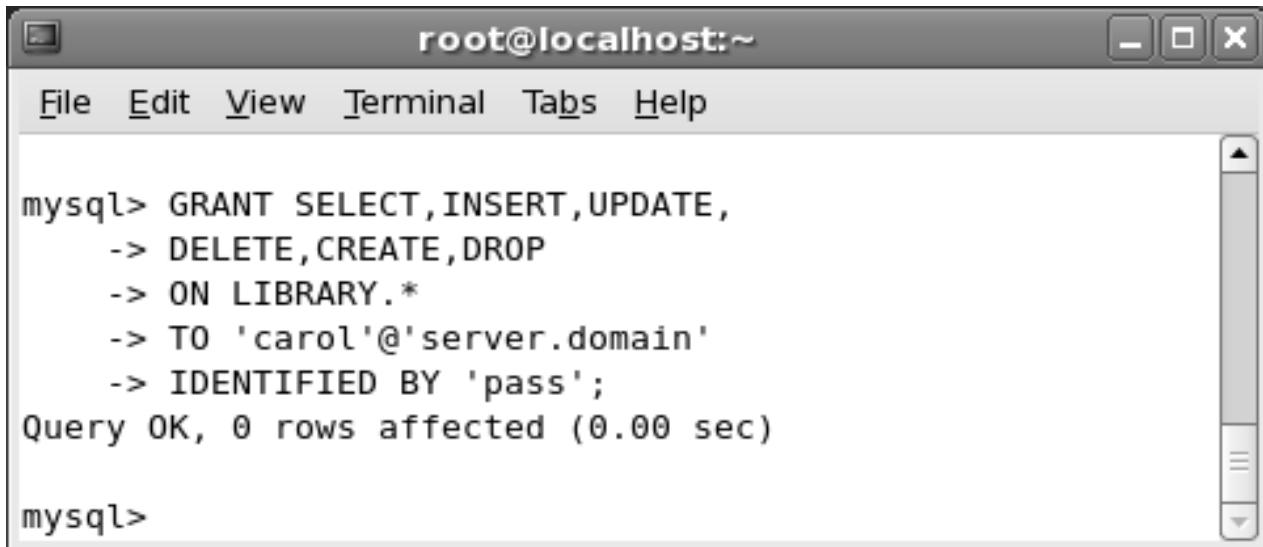
For example, to create a user who has access to the `LIBRARY` database from the `server.domain host`, enter the following command at the command prompt:

Session 16

Controlling and Managing MySQL database

```
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP ON LIBRARY.*  
TO 'carol'@'server.domain' IDENTIFIED BY 'pass';
```

Figure 16.14 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The menu bar includes "File", "Edit", "View", "Terminal", "Tabs", and "Help". The main area displays the following MySQL command and its execution:

```
mysql> GRANT SELECT,INSERT,UPDATE,  
-> DELETE,CREATE,DROP  
-> ON LIBRARY.*  
-> TO 'carol'@'server.domain'  
-> IDENTIFIED BY 'pass';  
Query OK, 0 rows affected (0.00 sec)  
  
mysql>
```

Figure 16.14: Create User with Access to Specific Database from a Specific Host

In figure 16.14, MySQL assigns limited rights and allows the user `carol` to access the `LIBRARY` database from the `server.domain` host.

To give access to a specific user from any machine in a given domain, enter the following command at the command prompt:

```
GRANT ALL ON *.* TO 'carol'@'%mydomain.com' IDENTIFIED BY 'pass';
```

Figure 16.15 displays the output of the command.

Session 16

Controlling and Managing MySQL database

Concepts

The screenshot shows a terminal window with the title 'root@localhost:~'. The menu bar includes 'File', 'Edit', 'View', 'Terminal', 'Tabs', and 'Help'. The main area displays the following MySQL command and its execution:

```
mysql> GRANT ALL ON *.*  
-> TO 'carol'@'%mydomain.com'  
-> IDENTIFIED BY 'pass';  
Query OK, 0 rows affected (0.00 sec)  
  
mysql>
```

Figure 16.15: Create User with Access to any Machine in a Given Domain

In figure 16.15, the user `carol` is assigned access from any machine included in the `mydomain.com` host. The user is provided with all rights and privileges.

16.3.1 Revoking Privileges

MySQL provides the REVOKE command to remove rights or privileges from user accounts.

Note: The REVOKE command only removes privileges. It does not delete `mysql.user` table entries.

For example, to cancel or revoke the `INSERT` privilege for a user, enter the following command at the command prompt:

```
REVOKE INSERT ON *.* FROM 'temp_user' @ 'localhost';
```

The command cancels the `INSERT` privilege assigned to `temp_user` on the `localhost` instance.

Note: To use the REVOKE command, you must have the `GRANT OPTION` privilege. Also, you must have the privileges that you are revoking.

16.3.2 Deleting User Account

MySQL provides the `DROP USER` command to remove a user account. You can also use this command to remove rows from the grant tables that contain privileges for user accounts.

To delete a user account, enter the following command at the command prompt:

Session 16

Controlling and Managing MySQL database

```
DROP USER user_name;
```

where,

DROP – specifies to remove the object from the database

USER – specifies the type of object to be removed from the database

user_name – specifies the name of the account to be removed from the database

Note: You must be careful when executing the `DROP USER` command.

You can also specify the domain to which the user account belongs while using the `DROP USER` command.

For example,

```
DROP USER 'carol'@'localhost';
```

This command will delete the user account `carol` on the `localhost` instance of MySQL.

MySQL also enables you to remove a user account with the `DELETE` command. This command removes the entry for the account that has been created in the `user` table of `mysql` database.

To delete the user using the `DELETE` command, enter the following command at the command prompt:

```
DELETE FROM user WHERE Host='localhost' AND User='carol';
```

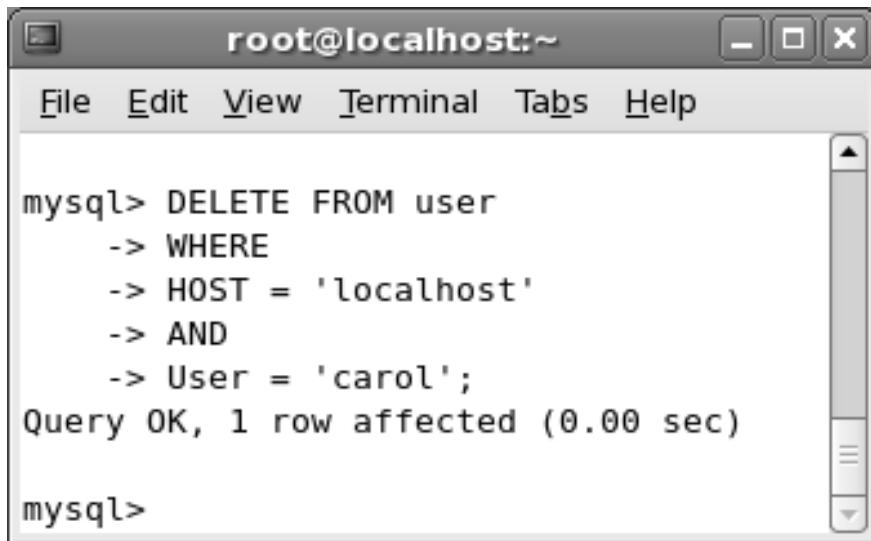
The command will remove the specified account from the `user` table. The `Host` and the `User` contains the instance name and the account name to delete.

Figure 16.16 displays the output of the command.

Session 16

Controlling and Managing MySQL database

Concepts



The screenshot shows a terminal window with the title 'root@localhost:~'. The window contains the following MySQL command and its execution:

```
mysql> DELETE FROM user
      -> WHERE
      -> HOST = 'localhost'
      -> AND
      -> User = 'carol';
Query OK, 1 row affected (0.00 sec)

mysql>
```

Figure 16.16: Deleting a User

In figure 16.16, MySQL removes the entry from the `user` table where the user is `carol` and the instance is `localhost`.

16.4 Setting Restrictions

You can ensure that single user do not take over the resources by setting a limit on the number of:

- Queries per account per hour
- Updates per account per hour
- Connections per account per hour
- Number of simultaneous connections per account

You can set these restrictions using the `GRANT` command as shown:

```
GRANT ... WITH MAX_QUERIES_PER_HOUR R1
              MAX_UPDATES_PER_HOUR R2
              MAX_CONNECTIONS_PER_HOUR R3
              MAX_USER_CONNECTIONS R4;
```

`R1`, `R2`, `R3` and `R4` are integer values that specify the limits. For example, once the user reaches the limit specified in `R3`, no further connections will be accepted till the hour elapses. Similarly, once the user reaches the limit of the queries specified in `R1`, no queries are accepted within the hour. Appropriate error

Session 16

Controlling and Managing MySQL database

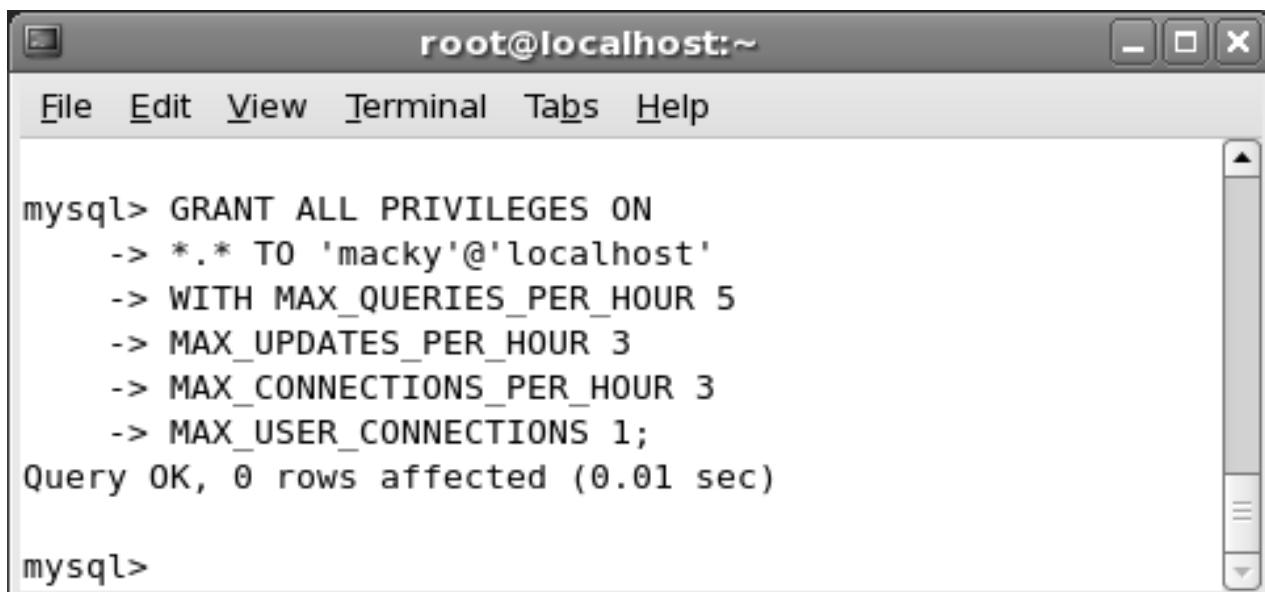
messages are displayed to indicate that the limit has been reached.

Note: You can specify any of the defined limits.

For example, to grant all privileges to the user for all the databases but with resource restrictions, enter the following command at the command prompt:

```
GRANT ALL PRIVILEGES ON *.* TO 'macky'@'localhost'  
WITH MAX_QUERIES_PER_HOUR 5  
MAX_UPDATES_PER_HOUR 3  
MAX_CONNECTIONS_PER_HOUR 3  
MAX_USER_CONNECTIONS 1;
```

Figure 16.17 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window contains a MySQL command-line interface. The user has run the following command:

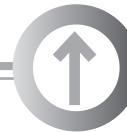
```
mysql> GRANT ALL PRIVILEGES ON
      -> *.* TO 'macky'@'localhost'
      -> WITH MAX_QUERIES_PER_HOUR 5
      -> MAX_UPDATES_PER_HOUR 3
      -> MAX_CONNECTIONS_PER_HOUR 3
      -> MAX_USER_CONNECTIONS 1;
Query OK, 0 rows affected (0.01 sec)

mysql>
```

The command grants all privileges on all databases to the user 'macky' from the localhost, with specific resource restrictions: 5 queries per hour, 3 updates per hour, 3 connections per hour, and 1 user connection.

Figure 16.17: Setting Restrictions

Figure 16.17 displays the privileges assigned to the user 'macky'. The `MAX_QUERIES_PER_HOUR` defines the number of queries that the user can execute in an hour. The `MAX_UPDATES_PER_HOUR` defines the number of updates the user can perform. In, addition, the `MAX_CONNECTIONS_PER_HOUR` defines the number of connections the user account can use to connect simultaneously to the server.



Summary

- MySQL stores details of user accounts and privileges in the `columns_priv`, `db`, `host`, `tables_priv`, and `user` tables under the default database named `mysql`.
- MySQL enables you to grant privileges to user accounts at four levels. They are global, database, table, and column level.
- A new user account can be created either by using `GRANT` command or by using `INSERT` command.
- The root user must have `INSERT` and `RELOAD` administrative privileges for creating user.
- The `USAGE` option creates a user without any privileges. An account with just `USAGE` privilege can only be used to establish a connection.
- The `GRANT OPTION` enables you to grant or revoke privileges to another user.
- A super user has all the administrative rights and can access the server from anywhere.
- MySQL provides the `GRANT` and `REVOKE` commands to assign and remove privileges to user accounts. To alter privileges for the user, `GRANT` command is used with selective privilege command.
- You can execute administrative commands, such as `mysql` or `mysqladmin` to control a user's access to the database. For executing administrative commands, the user has to login as the root user.
- You can ensure that no single user takes over the resources by setting a limit on the number of queries per account per hour, updates per account per hour, and so forth.

Session 16

Controlling and Managing MySQL database



Check Your Progress

1. For using the default database, the administrator has to log in as _____ user.
 - a. super
 - b. root
 - c. global
 - d. virtual

2. The _____ command grants and revokes privileges to all the columns of a given table.
 - a. Column level
 - b. Global level
 - c. Database level
 - d. Table level

3. A new user is added in the _____ table of _____ database.
 - a. func, test
 - b. db, default
 - c. user, mysql
 - d. host, user

4. When a user connects to the MySQL server, the location of the user gets detected from the _____ and by the _____ name.
 - a. Server, path
 - b. Hostname, user

Session 16

Controlling and Managing MySQL database



Check Your Progress

Concepts

- c. Client machine, user
 - d. Table level, target
5. Which of the following options help to set the maximum number of simultaneous connections per account?
- a. MAX_QUERIES_PER_HOUR
 - b. MAX_UPDATES_PER_HOUR
 - c. MAX_CONNECTIONS_PER_HOUR
 - d. MAX_USER_CONNECTIONS

ASK to LEARN

Questions
in your
mind?



are here to HELP

Post your queries in **ASK to LEARN** @

www.onlinevarsity.com

Objectives

At the end of this session, the student will be able to:

- *Create user accounts in MySQL.*
- *Use the GRANT command to assign privileges.*
- *Use the REVOKE command to alter privileges.*

The steps given in the session are detailed, comprehensive and carefully thought through. This has been done so that the learning objectives are met and the understanding of the tool is complete. Please follow the steps carefully.

Part I - For the first 1.5 hours:

Creating user accounts in MySQL

You can create new user accounts by using the GRANT command or by manipulating the MySQL grants tables directly. You will use the user table of the mysql database.

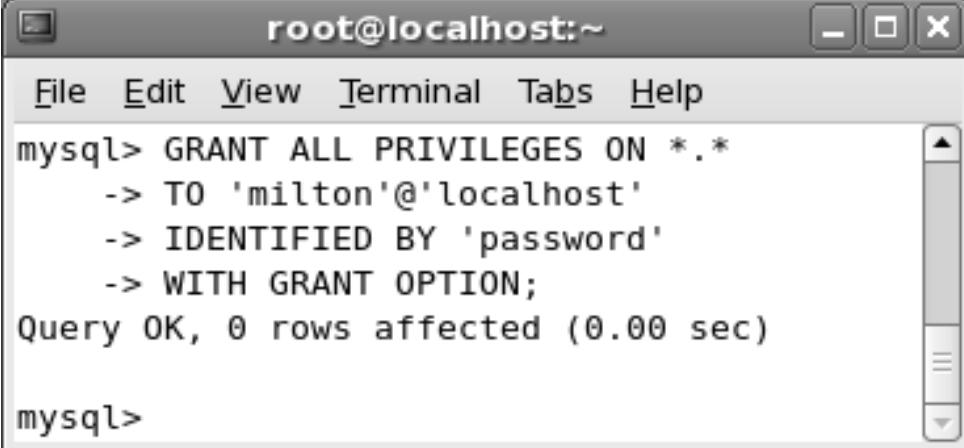
1. **Activate the mysql database.**
2. **The administrator wants to create a user having all the privileges and should also be able to connect through the localhost only. To do this using the GRANT command, enter the following command at the command prompt:**

```
GRANT ALL PRIVILEGES ON *.* TO 'milton'@'localhost' IDENTIFIED  
BY 'password' WITH GRANT OPTION;
```

Figure 17.1 displays the output of the command.

Session 17

Controlling and Managing MySQL Database (Lab)



```
root@localhost:~ 
File Edit View Terminal Tabs Help
mysql> GRANT ALL PRIVILEGES ON *.* 
    -> TO 'milton'@'localhost'
    -> IDENTIFIED BY 'password'
    -> WITH GRANT OPTION;
Query OK, 0 rows affected (0.00 sec)

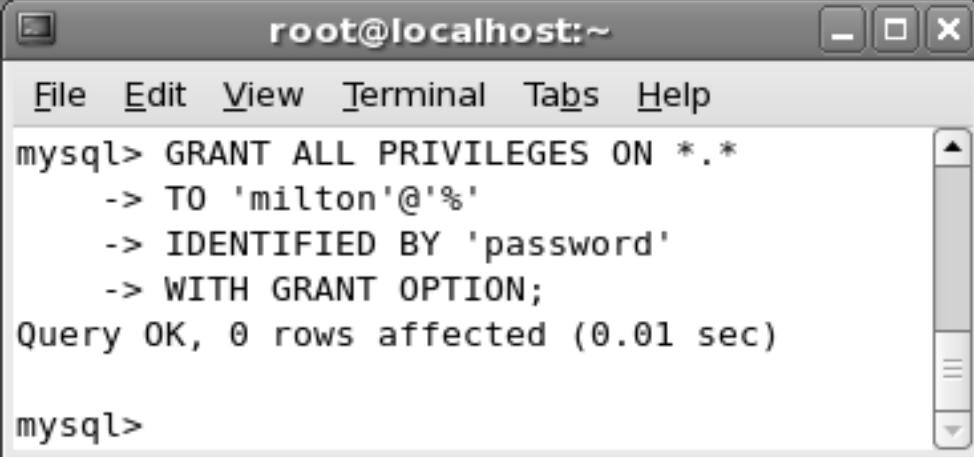
mysql>
```

Figure 17.1: Creating a User and Connecting through Localhost

3. The administrator wants to create a user having all the privileges and should be able to connect through any host. To perform this using the GRANT command, enter the following command at the command prompt:

```
GRANT ALL PRIVILEGES ON *.* TO 'milton'@'%' IDENTIFIED  
BY 'password' WITH GRANT OPTION;
```

Figure 17.2 displays the output of the command.



```
root@localhost:~ 
File Edit View Terminal Tabs Help
mysql> GRANT ALL PRIVILEGES ON *.* 
    -> TO 'milton'@'%'
    -> IDENTIFIED BY 'password'
    -> WITH GRANT OPTION;
Query OK, 0 rows affected (0.01 sec)

mysql>
```

Figure 17.2: Creating a User Connecting Through any Host

4. The administrator wants to create an account for Julie with all privileges that can connect to MySQL from the localhost instance. To create a user using the INSERT command with

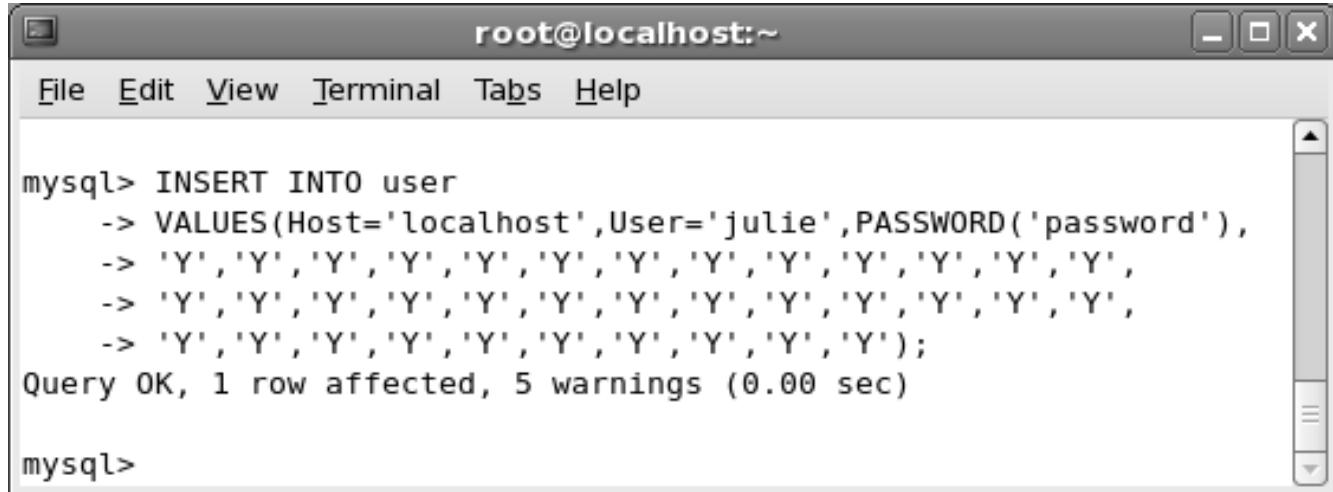
Session 17

Controlling and Managing MySQL Database (Lab)

all the privileges and to connect through the local host, enter the following command at the command prompt:

```
INSERT INTO user VALUES (Host='localhost',User='julie',PASSWORD ('password'),'Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y');
```

Figure 17.3 displays the output of the command.



```
root@localhost:~  
File Edit View Terminal Tabs Help  
  
mysql> INSERT INTO user  
-> VALUES(Host='localhost',User='julie',PASSWORD('password'),  
-> 'Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y');  
Query OK, 1 row affected, 5 warnings (0.00 sec)  
  
mysql>
```

Figure 17.3: Creating a User Connecting to the Localhost Using **INSERT** command

5. The administrator wants to create a user account for the employee bill who will have unrestricted access to MySQL and can connect from any client machine on the network. The user will have the password set as 'pass'. To create a user with all privileges and to connect through any host using the **INSERT** command, enter the following command at the command prompt:

```
INSERT INTO user VALUES (Host='%',User='bill',PASSWORD ('pass'),'Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y');
```

Figure 17.4 displays the output of the command.

Session 17

Controlling and Managing MySQL Database (Lab)

Figure 17.4: Creating a User Connecting to any Host Using `INSERT` command

6. The administrator wants to create a user with the File_priv and the Show_db_priv privileges only. The user should be able to connect through the localhost using the INSERT command. To do this, enter the following command at the command prompt:

```
INSERT INTO user SET Host='localhost', User='jonathan',
File priv='Y', Show db priv='Y';
```

Figure 17.5 displays the output of the command.

```
root@localhost:~ File Edit View Terminal Tabs Help mysql> INSERT INTO user
    -> SET Host='localhost',
    -> User='jonathan',
    -> File_priv='Y',
    -> Show_db_priv='Y';
Query OK, 1 row affected, 3 warnings (0.00 sec)

mysql>
```

Figure 17.5: Creating a User with the `File_priv` and the `Show_db_priv` Privileges

Session 17

Controlling and Managing MySQL Database (Lab)

7. The administrator wants to create a user account for the employee, katie, using the INSERT command. To create a user account using the INSERT command, and allowing connection through localhost only, enter the following command at the command prompt.

```
INSERT INTO user (Host, User, Password) VALUES  
('localhost','katie','');
```

Figure 17.6 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area contains the following MySQL command and its output:

```
mysql> INSERT INTO user  
-> (Host,User,Password)  
-> VALUES  
-> ('localhost','katie','');  
Query OK, 1 row affected, 3 warnings (0.00 sec)  
  
mysql>
```

Figure 17.6: Creating a User Using INSERT command

The command will create a user in the `localhost` with no password and with no privileges granted to it. After using all the commands, use the `FLUSH PRIVILEGE` command to make the changes effective to the `user` table of `mysql` database.

Using GRANT command to assign privileges to users

A GRANT command is used for adding rights and privileges to the user. Perform an addition of privileges to the user, 'katie' using the GRANT command.

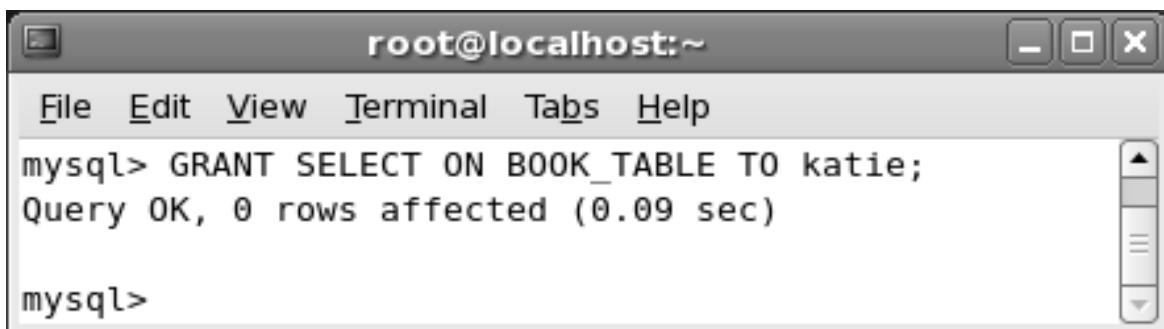
1. Activate the LIBRARY database.
2. The administrator wants to assign the SELECT privilege using the GRANT command for `BOOK_TABLE` of LIBRARY database to the user `katie`. To assign the SELECT privilege using the GRANT command for `BOOK_TABLE` of LIBRARY database, enter the following at the command prompt:

```
GRANT SELECT ON BOOK_TABLE TO katie;
```

Session 17

Controlling and Managing MySQL Database (Lab)

Figure 17.7 displays the output of the command.



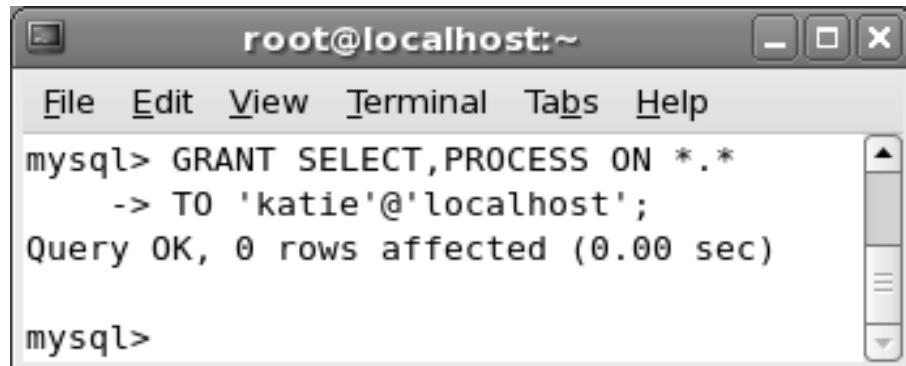
A screenshot of a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area shows the MySQL prompt "mysql>". A command is entered: "GRANT SELECT ON BOOK_TABLE TO katie;". The response is "Query OK, 0 rows affected (0.09 sec)". The prompt "mysql>" appears again at the bottom.

Figure 17.7: Assigning the SELECT Privilege

3. The administrator wants to assign the SELECT and the PROCESS privileges to the user katie using the GRANT command, for all the tables of LIBRARY database. To assign the SELECT and the PROCESS privileges using GRANT command, enter the following command at the command prompt:

```
GRANT SELECT,PROCESS ON *.* TO 'katie'@'localhost';
```

Figure 17.8 displays the output of the command.



A screenshot of a terminal window titled "root@localhost:~". The window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The main area shows the MySQL prompt "mysql>". A command is entered: "GRANT SELECT,PROCESS ON *.* TO 'katie'@'localhost';". The response is "Query OK, 0 rows affected (0.00 sec)". The prompt "mysql>" appears again at the bottom.

Figure 17.8: Assigning the SELECT and the PROCESS Privileges to the User

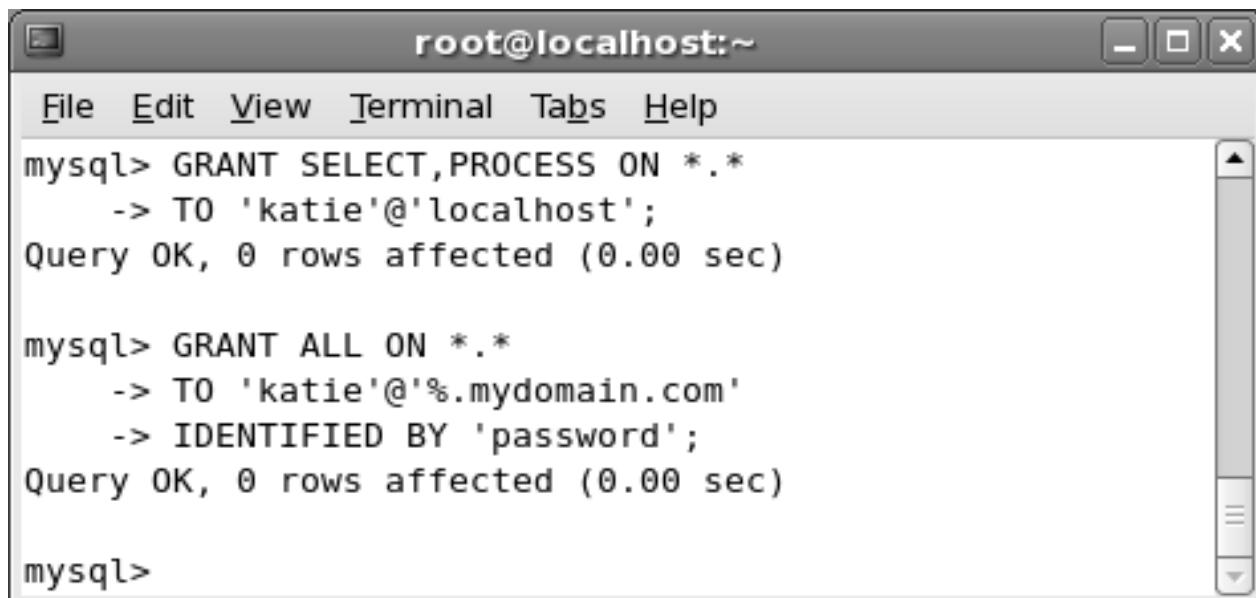
Session 17

Controlling and Managing MySQL Database (Lab)

4. The administrator wants to allow the user katie to access the MySQL database from any machine on the domain using the GRANT command. To provide access to the user from any machine in a given domain by using the GRANT command, enter the following command at the command prompt:

```
GRANT ALL ON *.* TO 'katie'@'%mydomain.com' IDENTIFIED BY  
'password';
```

Figure 17.9 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following MySQL session:

```
root@localhost:~  
File Edit View Terminal Tabs Help  
mysql> GRANT SELECT,PROCESS ON *.*  
      -> TO 'katie'@'localhost';  
Query OK, 0 rows affected (0.00 sec)  
  
mysql> GRANT ALL ON *.*  
      -> TO 'katie'@'%mydomain.com'  
      -> IDENTIFIED BY 'password';  
Query OK, 0 rows affected (0.00 sec)  
  
mysql>
```

Figure 17.9: Granting Access to the User from any Machine in a Given Domain

In Figure 17.9, the user `katie` is given access to any machine included in the `mydomain.com` host with all rights and privileges.

Using REVOKE command to alter privileges granted to users

The REVOKE command is used to remove the privileges from the user. This command drops all databases, tables, and columns level privileges from the user.

Session 17

Controlling and Managing MySQL Database (Lab)

1. Katie has submitted her resignation to the manager. As per the company policy, employees on notice period will not have any privileges on the MySQL database. Therefore the administrator wants to view the privileges assigned to katie. To check the privileges of the user, enter the following command at the command prompt:

```
SHOW PRIVILEGES;
```

Figure 17.10 displays the output of the command.



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following text:

```
mysql> SHOW PRIVILEGES;
+-----+-----+-----+
| Privilege          | Context           | Comment
+-----+-----+-----+
| Alter              | Tables            | To alter the table
| Alter routine      | Functions,Procedures | To alter or drop st
ions/procedures
| Create tables       | Databases,Tables,Indexes | To create new datab
ables
| Create routine     | Databases         | To use CREATE FUNCT
URE
| Create temporary   | Databases         | To use CREATE TEMPO
r
| Create view         | Tables            | To create new views
| Create user         | Server Admin      | To create new users
```

Figure 17.10: Displaying Privileges of a User

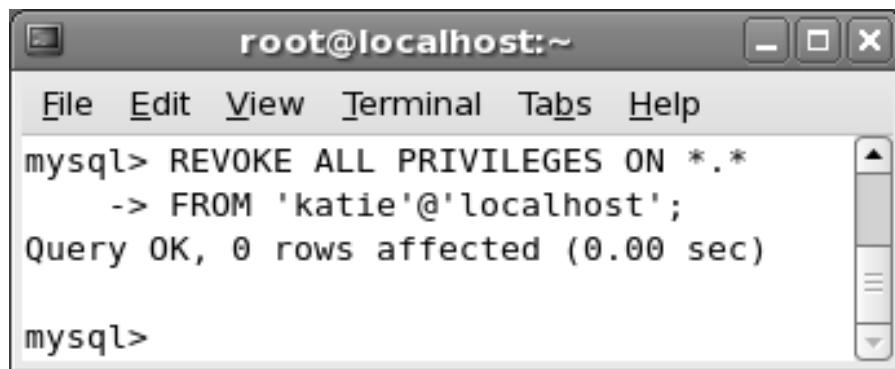
2. As per the company policy, employees on notice period will not have any privileges on the MySQL database. The administrator wants to remove all privileges assigned to katie. To remove all the privileges for the user katie, enter the following command at the command prompt:

```
REVOKE ALL PRIVILEGES ON *.* FROM 'katie'@'localhost';
```

Figure 17.11 displays the output of the command.

Session 17

Controlling and Managing MySQL Database (Lab)



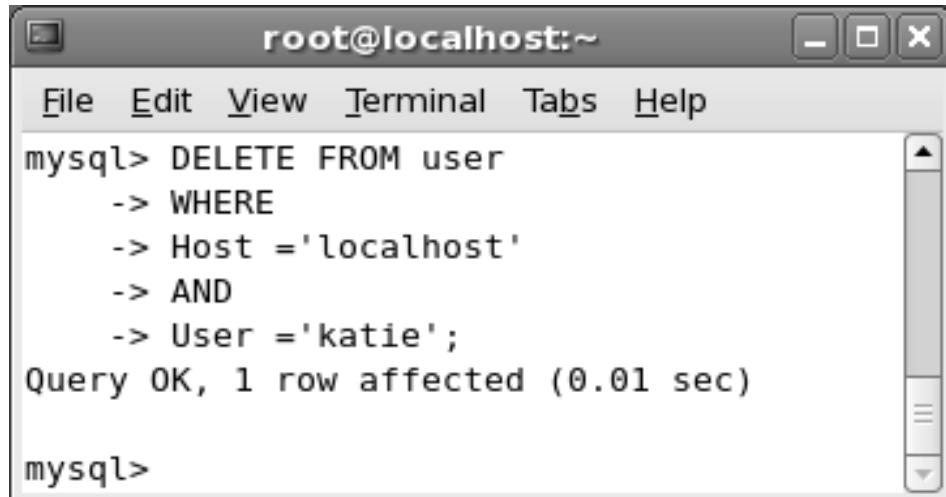
A screenshot of a terminal window titled "root@localhost:~". The window shows the MySQL command-line interface. The user has run the command "REVOKE ALL PRIVILEGES ON *.* FROM 'katie'@'localhost';" which results in "Query OK, 0 rows affected (0.00 sec)". The prompt "mysql>" is visible at the bottom.

Figure 17.11: Removing privileges from the user

3. Activate the mysql database.
4. Katie is no longer working in the organization. Therefore, the administrator wants to delete the user account, katie. To delete the user by using the DELETE command, enter the following command at the command prompt:

```
DELETE FROM user WHERE Host='localhost' AND User='katie';
```

Figure 17.12 displays the output of the command.



A screenshot of a terminal window titled "root@localhost:~". The window shows the MySQL command-line interface. The user has run the command "DELETE FROM user WHERE Host='localhost' AND User='katie';" which results in "Query OK, 1 row affected (0.01 sec)". The prompt "mysql>" is visible at the bottom.

Figure 17.12: Deleting User From Database



Do It Yourself

1. Create a user account having the name as `Betsy` with the help of `GRANT` command and assign it into `localhost` instance.
2. Assign all privileges to the user.
3. Create the same user with the help of `GRANT` command and assign it any host.
4. Add a new user `Helen` in the `user` table of `mysql` database. Assign the user in `server.domain.host`.
5. Assign `SELECT`, `PROCESS`, and `DROP` privileges to the user `Helen` using `GRANT` command.
6. Create a new user `Kathy` in `localhost` host and assign only the `Update_priv` and the `Create_priv` privileges to the user.
7. Assign the `INSERT` and `DROP` privileges to '`Kathy`' user for `Stock_Details` table using the `GRANT` command.
8. Check the privileges of user '`Betsy`'.
9. Remove the privileges of user '`Betsy`'.
10. Delete the user `Betsy` from `user` table of `mysql` database.