Prasad V. Potluri Siddhartha Institute of Technology

Autonomous, Kanuru,Vijayawada-07

Department of Computer Science and Engineering

DATABASE MANAGEMENT SYSTEMS

LABORATORY MANUAL

AICTE Approved, NBA and NACC A+ accredited, An ISO 9001: 2015 certified Institution, Permanent affiliation to JNTUK, Kakinada.

Institute Vision

To provide rich ambience for Academic and Professional Excellence, Research, Employability skills, Entrepreneurship and Social responsibility.

Institute Mission

To empower the students with Technical knowledge, Awareness of up-to-date technical trends, Inclination for research in the areas of human needs, Capacity building for Employment / Entrepreneurship, Application of technology for societal needs.

Department Vision

To be a center of excellence in academics and research in Computer Science and Engineering and take up challenges for the benefit of society.

Department Mission

Impart professional education through best curriculum in harmony with the industry needs.

Inculcate ethics, research capabilities and team work in the young minds so as to put efforts to the advancement of the nation.

Strive for student achievement and success with leadership qualities and preparing them for continuous learning in the global environment.

Program Educational Objective

PEO-I:

The graduates of the program will excel in the concepts of basic engineering and advanced concepts of computer science engineering.

PEO-II:

The graduates of the program will be professional in computing industry or pursuing higher studies.

PEO-III:

The graduates of the program will excel in team work, ethics, communication skills and contribute to the benefit to the society.

PO - 1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO - 2: Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO - 3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO - 4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO - 5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO - 6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO - 7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO - 8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO - 9: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO - 10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO - 11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO - 12: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Specific Outcomes

Engineering Graduates will be able to:

PSO - I: Apply the Knowledge of Computing Skills in building the Software Systems that meet the requirements of Industry and Society.

PSO - II: Apply the Knowledge of Data Engineering and Communication Technologies for Developing Applications in the Domain of Smart and Intelligent Computing.

Course Code	19CS3552	Year	III	Semester	I
Course Category	Program Core	Branch	CSE	Course Type	Practical
Credits	1	L-T-P	0-0-2	Prerequisites	Programming and Problem Solving
Continuous Internal Evaluation :	25	Semester End Evaluation:	50	Total Marks:	75

Course Outcomes		
Upon successful completion of the course, the student will be able to		
CO1	Apply database management techniques to solve problems	**L3**
CO2	Implement experiments by using modern tools like MYSQL, Oracle	**L3**
CO3	Develop an effective report based on various constructs implemented.	**L3**
CO4	Apply technical knowledge for a given problem and express with an effectiveoral communication.	**L3**
CO5	Analyze outputs of queries for a given problem	**L4**

Syllabus Copy

Contribution of Course Outcomes towards achievement of Program Outcomes & Strength of correlations(3:Substantial, 2: Moderate, 1:Slight)

	PO 1	PO 2	PO3	PO4	PO5	PO6	PO7	PO8	PO 9	PO 10	PO1 1	PO1 2	PSO 1	PSO 2
CO1	3											3		
CO2					3				3					
CO3										3				
CO4	3									3				
CO5		3												

Syllabus		
Expt No	**Contents**	**Mapped**
1	Introduction to MySQL Workbench. How to use MySql Workbench to run SQL Statements.	**CO1,CO2,CO3, CO4,CO5**
2	Examples on i)DDL Commands: CREATE , ALTER, DROP and TRUNCATE a Table. ii)Implementation of Constraints PRIMARY KEY, FOREIGNKEY, CHECK,NOT NULL, UNIQUE.	**CO1,CO2,CO3, CO4,CO5**
3	Examples on i)DML Commands. INSERT, UPDATE and DELETE ii)DCL Commands: COMMIT , ROLLBACK and SAVEPOINT.	**CO1,CO2,CO3, CO4,CO5**
4	Examples on retrieving data from a single table using i)SELECT statement ii) SELECT statement with where clause(Comparison Operators,AND, OR, NOT, IN, BETWEEN,LIKE) iii) ORDER BY clause(sort by column name) iv)LIMIT clause	**CO1,CO2,CO3, CO4,CO5**
5	Examples on Functions in MySQL: String, Numeric,Date, Time and Other Functions.	**CO1,CO2,CO3, CO4,CO5**
6	Examples on Summary Queries: Queries using Aggregate functions ,GROUP By and Having Clauses, ROLLUP Operator.	**CO1,CO2,CO3, CO4,CO5**
7	Examples on Inner join, outer join using USING, NATURAL Keywords	**CO1,CO2,CO3, CO4,CO5**
8	Examples on SUB/SUMMARY Queries Using IN, ANY, SOME,ALL , EXISTS and NOT EXISTS functions	**CO1,CO2,CO3, CO4,CO5**
9	Examples on i)Creating INDEXES and VIEWS ii)INSERT,DELETE and DROP on VIEWS	**CO1,CO2,CO3, CO4,CO5**
10	Examples on i) Create and Call STORED PROCEDURE (IN,OUT,INOUT Parameters) ,Drop a STORED PROCEDURE. ii) Create,call and Drop a FUNCTION. iii) Create and Drop a TRIGGER	**CO1,CO2,CO3, CO4,CO5**
11	Case Study using real world database applications	**CO1,CO2,CO3, CO4,CO5**

Learning Resources
Text Books
Murach's MySQL, JOEL MURACH, 2012, Shroff Publishers & Distributors Pvt.Ltd.
Reference Books
1. The Complete Reference MYSQL, VikramVaswani, 2017, McGrawHill Education. 2. DATABASE SYSTEMS Models, Languages, Design and Application Programming,RamezElmasri, ShamkantB. Navathe, Sixth Edition, Pearson. 3. Data base System Concepts, Abraham Silberschatz, Henry F Korth, S. Sudarshan, Fifth Edition,McGraw Hill.
e- Resources & other digital material
NPTEL Course: Data base Design: https://nptel.ac.in/courses/106/106/106106093/ COURSERA: i) SQL :https://www.coursera.org/lecture/gcp-exploring-preparing-data-bigquery/lab-walkthrough-enable-standard-sql-AGRJt ii):Built in Functions SQL: https://www.coursera.org/lecture/sql-data-science/built-in-database-functions-0bRtQ

Regulations –

For Laboratory courses, there shall be continuous evaluation during the semester for 25 marks and semester end evaluation for 50 marks.

Distribution of Marks (CIE)

S.No.	Criterion	Marks
1	Day to Day Evaluation	10
2	Record	05
3	Internal Examination	10

Lab Rubrics -

Day to Day Evaluation (10M) + Record (5M)			
Procedure	**Execution**	**Viva-Voce**	**Record**

Apply (2M)	Self-learning (1M)	Modern tools (3M)	Individual Performance (2M)	Apply (1M)	Communication (1M)	Communication (5M)

Internal Examination (10M)					
Procedure		**Execution**		**Viva-Voce**	
Apply (1M)	Self-learning (1M)	Modern tools (3M)	Individual Performance (2M)	Apply (2M)	Communication (1M)

External Examination (50M)					
Procedure	**Execution**		**Viva**		**Output**
Apply (10M)	Modern tools (8M)	Individual Performance (12M)	Apply (7M)	Communication (3M)	Analysis (10M)

EXPERIMENT-1

1. Introduction to MySQL Workbench.
2. How to use MySql Workbench to work with a database.
3. How to use MySql Workbench to run SQL Statements.

1. Introduction to MySQL Workbench.

MySQL Workbench is a graphical tool for working with MySQL servers and databases. MySQL

Workbench fully supports MySQL server versions 5.5 and higher. It is also compatible with older

MySQL server 5.x versions, except in certain situations (like displaying the process list) due to changed system tables. It does not support MySQL server versions 4.x.

- MySQL Workbench is a graphical tool for working with MySQL servers and databases. MySQL Workbench fully supports MySQL server versions 5.5 and higher. It is also compatible with older
- MySQL server 5.x versions, except in certain situations (like displaying the process list) due to changed system tables. It does not support MySQL server versions 4.x.
- MySQL Workbench functionality covers five main topics:
- **SQL Development**: Enables you to create and manage connections to database servers. Along with enabling you to configure connection parameters, MySQL Workbench provides the capability to execute SQL queries on the database connections using the built-in SQL Editor.
- **Data Modeling (Design):** Enables you to create models of your database schema graphically, reverse and forward engineer between a schema and a live database, and edit all aspects of your database using the comprehensive Table Editor. The Table Editor provides easy-to-use facilities for editing Tables, Columns, Indexes, Triggers, Partitioning, Options, Inserts and Privileges, Routines and Views.
- **Server Administration:** Enables you to administer MySQL server instances by administering users, performing backup and recovery, inspecting audit data, viewing database health, and monitoring the MySQL server performance.
- **Data Migration**: Allows you to migrate from Microsoft SQL Server, Microsoft Access, Sybase ASE, SQLite, SQL Anywhere, PostreSQL, and other RDBMS tables, objects and data to MySQL. Migration also supports migrating from earlier versions of MySQL to the latest releases.
- **MySQL Enterprise Support**: Support for Enterprise products such as MySQL Enterprise Backup, MySQL Firewall, and MySQL Audit.
- **MySQL Workbench is available in two editions**: the Community Edition and the Commercial Edition.
- The Community Edition is available free of charge. The Commercial Edition provides additional
- Enterprise features, such as access to MySQL Enterprise Backup, MySQL Firewall, and MySQL Audit.
- MySQLWorkbench is a Visual database designing and modeling access tool for MySQL server relational database. It facilitates creation of new physical data models and modification of existing MySQL databases with reverse/forward engineering and change management functions.

MySQL Workbench functionality covers five main topics:

SQL Development: Enables you to create and manage connections to database servers. Along with enabling you to configure connection parameters, MySQL Workbench provides the capability to execute SQL queries on the database connections using the built-in SQL Editor.

Data Modeling (Design): Enables you to create models of your database schema graphically, reverse and forward engineer between a schema and a live database, and edit all aspects of your database using the comprehensive Table Editor. The Table Editor provides easy-to-use facilities for editing Tables, Columns, Indexes, Triggers, Partitioning, Options, Inserts and Privileges, Routines and Views.

Server Administration: Enables you to administer MySQL server instances by administering users, performing backup and recovery, inspecting audit data, viewing database health, and monitoring the MySQL server performance.

Data Migration: Allows you to migrate from Microsoft SQL Server, Microsoft Access, Sybase ASE, SQLite, SQL Anywhere, PostreSQL, and other RDBMS tables, objects and data to MySQL. Migration

also supports migrating from earlier versions of MySQL to the latest releases.

MySQL Enterprise Support: Support for Enterprise products such as MySQL Enterprise Backup, MySQL Firewall, and MySQL Audit.

MySQL Workbench is available in two editions: the Community Edition and the Commercial Edition. The Community Edition is available free of charge. The Commercial Edition provides additional Enterprise features, such as access to MySQL Enterprise Backup, MySQL Firewall, and MySQL Audit.

MySQLWorkbench is a Visual database designing and modeling access tool for MySQL server relational database. It facilitates creation of new physical data models and modification of existing MySQL databases with reverse/forward engineering and change management functions.

Getting Started MySQL workbench- Modeling and Design tool Models are at the core of most valid and high performance databases. MySQL workbench has tools that allow developers and database administrators visually create physical database design models that can be easily translated into MySQL databases using forward engineering.

MySQL workbench supports creation of multiple models in the same environment.

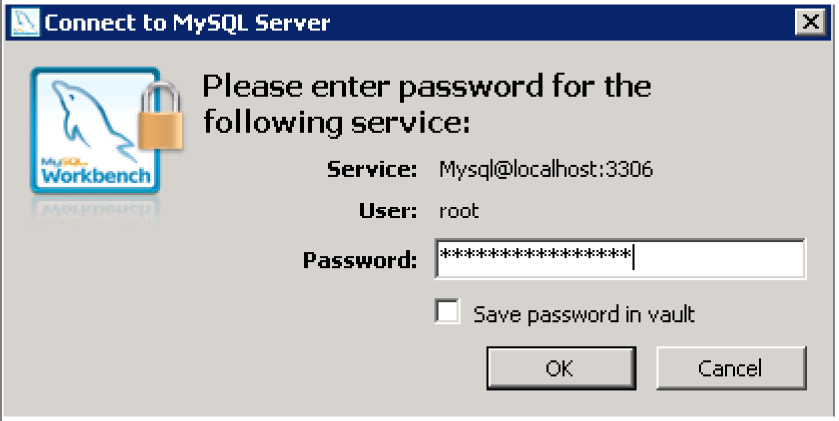
It supports all objects such as tables, views, stored procedures, triggers, etc. that make up a database.

MySQL workbench has a built in model validating utility that reports any issues that might be found to the data modeler.

It also allows for different modeling notations and can be extended by using LUA a scripting language.

2. How to use MySql Workbench to work with a database.

MySQL workbench may require a login to your MySQL server. Enter your root or user and password that has been assigned dba server privileges.



Please enter password for the following service:

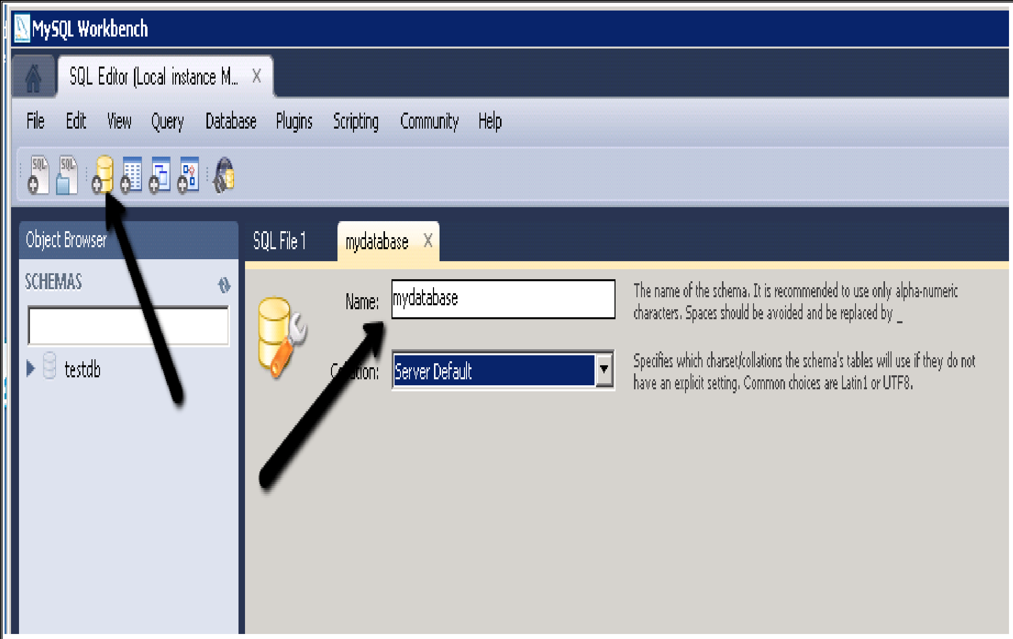
Service: Mysql@localhost:3306

User: root

Password: ****************

Save password in vault

Click on the **New Schema** icon in the menu, and then enter a **name** for your new database in the field as shown. Click the **Apply** button to generate the SQL script.

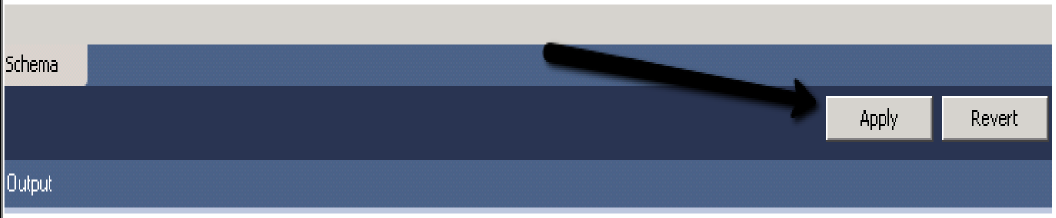


Name: mydatabase

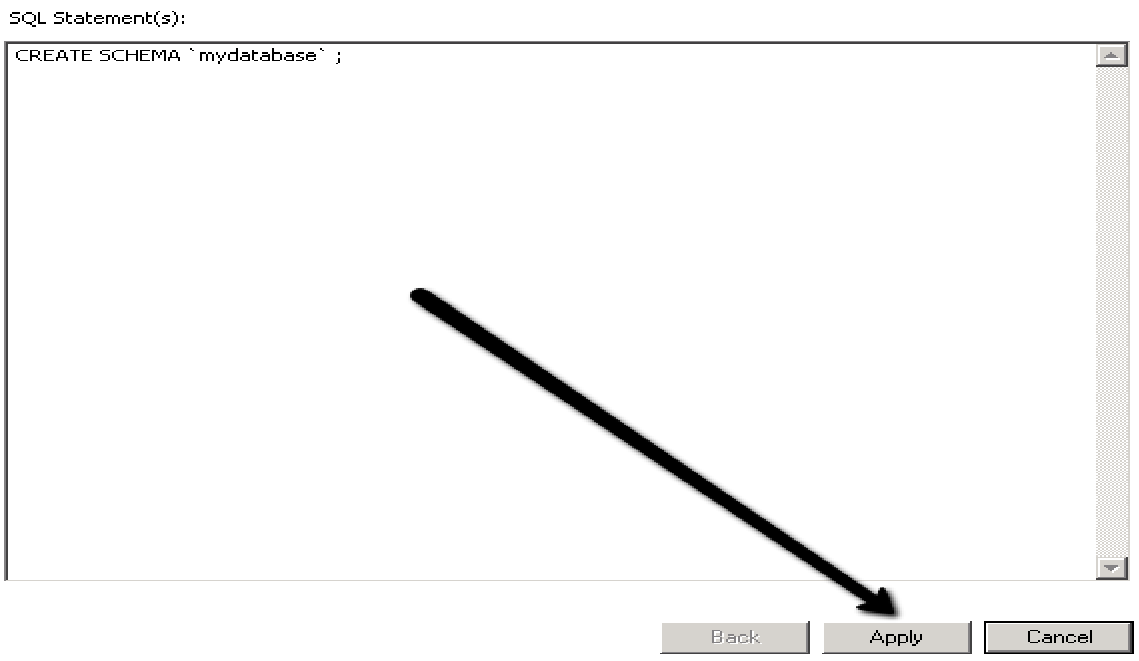
Collation: Server Default

The name of the schema. It is recommended to use only alpha-numeric characters. Spaces should be avoided and be replaced by _

Specifies which charset/collations the schema's tables will use if they do not have an explicit setting. Common choices are Latin1 or UTF8.

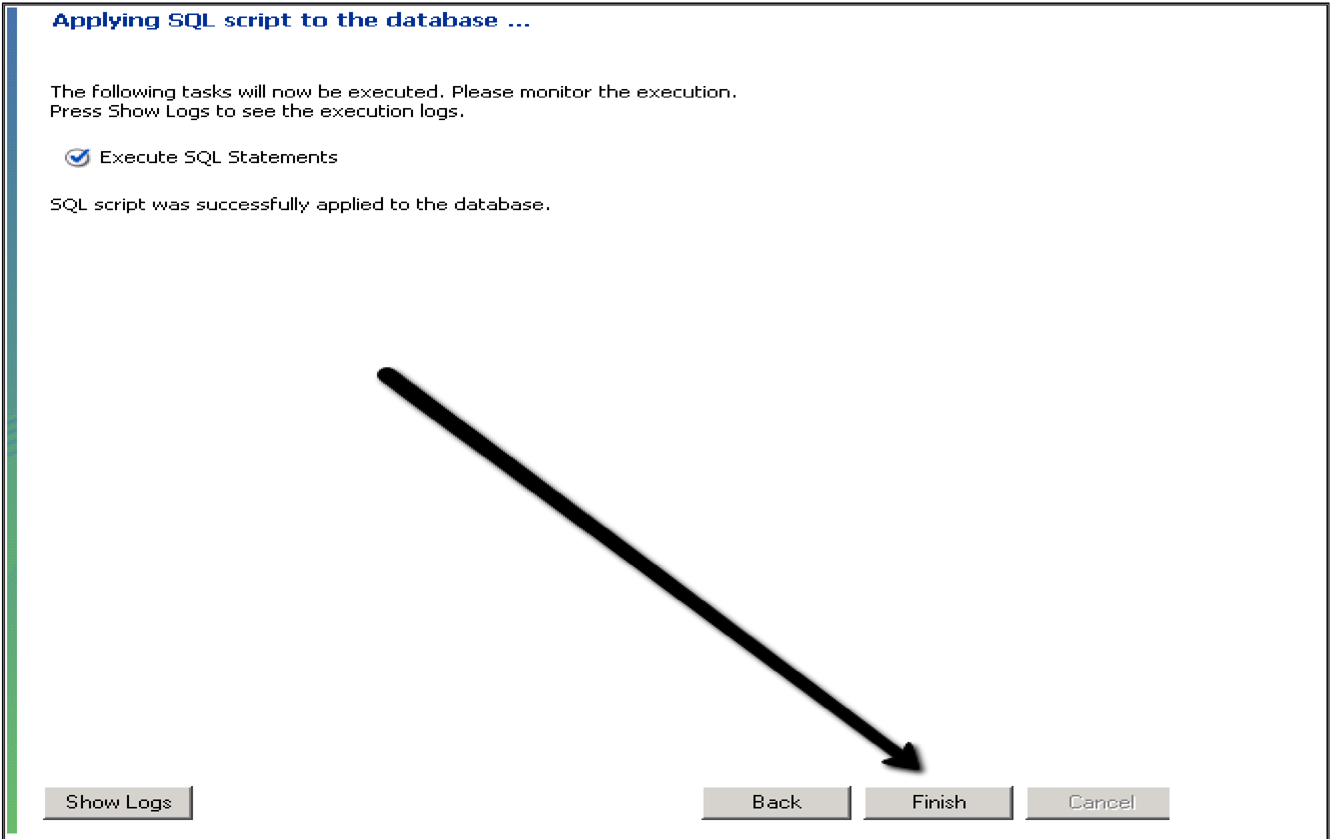


Click the **Apply** button again to execute the create database statement, and create your new database.



```
CREATE SCHEMA `mydatabase` ;
```

Click **Finish**.



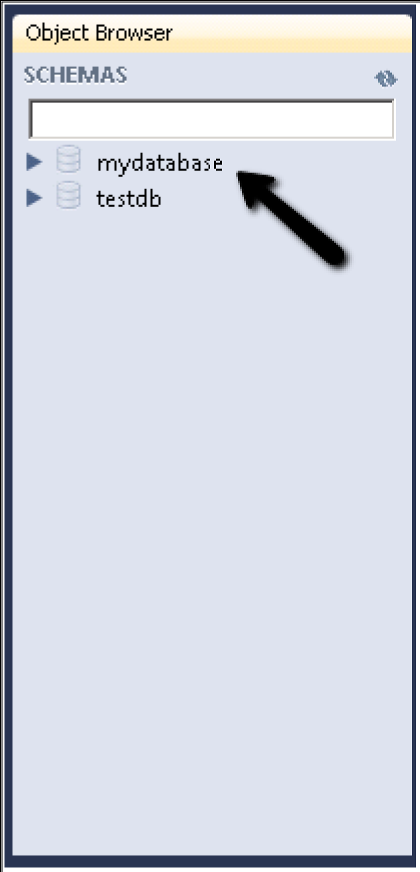
Applying SQL script to the database ...

The following tasks will now be executed. Please monitor the execution.
Press Show Logs to see the execution logs.

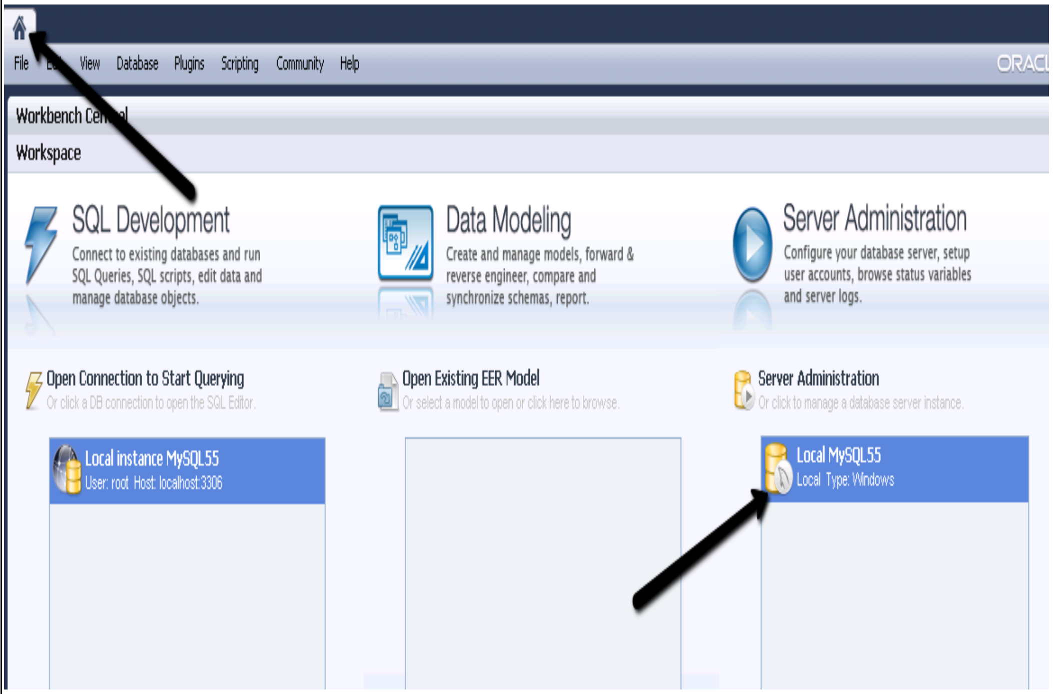
✓ Execute SQL Statements

SQL script was successfully applied to the database.

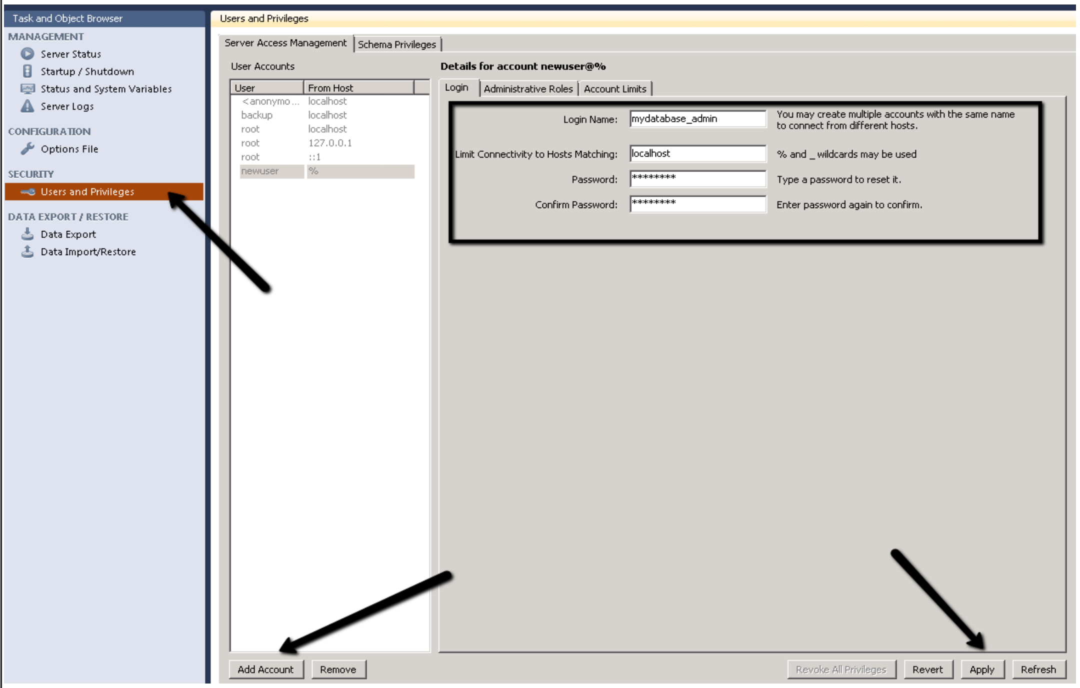
Your database should now be listed on the left with your other database schemas.



Click the **Home** icon in the top left corner to return to the Workbench Central screen. Click on your MySQL server instance under the **Server Administrator** section of MySQL workbench to create a new database user and assign privileges to your new database.


File
View
Database
Plugins
Scripting
Community
Help
Workbench Central
Workspace
SQL Development
Connect to existing databases and run SQL Queries, SQL scripts, edit data and manage database objects.
Data Modeling
Create and manage models, forward & reverse engineer, compare and synchronize schemas, report.
Server Administration
Configure your database server, setup user accounts, browse status variables and server logs.
Open Connection to Start Querying
Or click a DB connection to open the SQL Editor.
Open Existing EER Model
Or select a model to open or click here to browse.
Server Administration
Or click to manage a database server instance.
Local instance MySQL55
User: root Host: localhost:3306
Local MySQL55
Local Type: Windows

Click on **Users and Privileges**. Then click on **Add Account**. Enter a **login name** for the new user, type **localhost** and a new **password** as shown. Click **Apply** to create the new user account.



User Accounts

User	From Host
<anonymo...	localhost
backup	localhost
root	localhost
root	127.0.0.1
root	::1
newuser	%

Details for account newuser@%

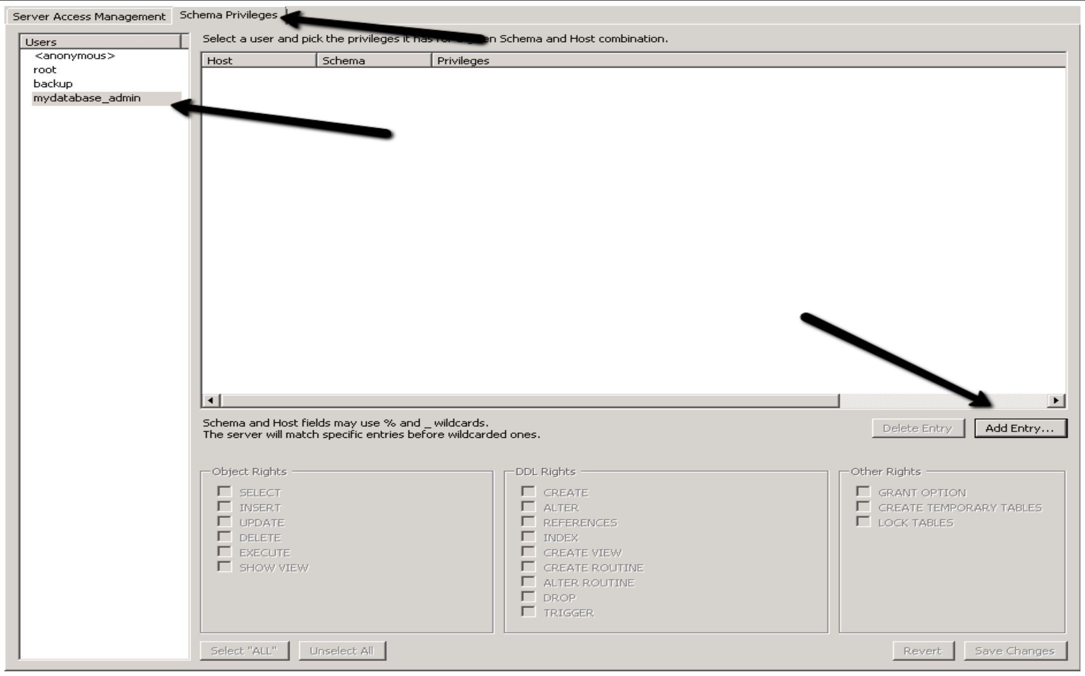
Login Name: mydatabase_admin — You may create multiple accounts with the same name to connect from different hosts.

Limit Connectivity to Hosts Matching: localhost — % and _ wildcards may be used

Password: ******** — Type a password to reset it.

Confirm Password: ******** — Enter password again to confirm.

To assign privileges for this user to access a specific database, click on the **Schema Privileges** tab. **Click the user account** from the list of users on the left. Click the **Add Entry** button.



Server Access Management | Schema Privileges

Users
<anonymous>
root
backup
mydatabase_admin

Select a user and pick the privileges it has for a given Schema and Host combination.

Host	Schema	Privileges

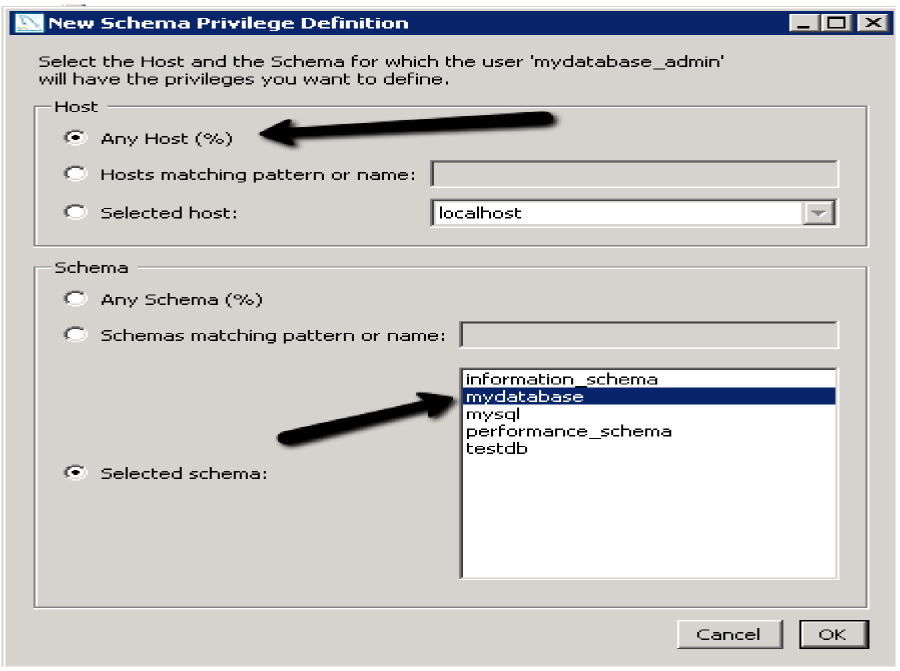
Schema and Host fields may use % and _ wildcards.
The server will match specific entries before wildcarded ones.

Object Rights: ☐ SELECT ☐ INSERT ☐ UPDATE ☐ DELETE ☐ EXECUTE ☐ SHOW VIEW

DDL Rights: ☐ CREATE ☐ ALTER ☐ REFERENCES ☐ INDEX ☐ CREATE VIEW ☐ CREATE ROUTINE ☐ ALTER ROUTINE ☐ DROP ☐ TRIGGER

Other Rights: ☐ GRANT OPTION ☐ CREATE TEMPORARY TABLES ☐ LOCK TABLES

Select the **Selected Schema** radio option, and choose your database schema from the list.



New Schema Privilege Definition

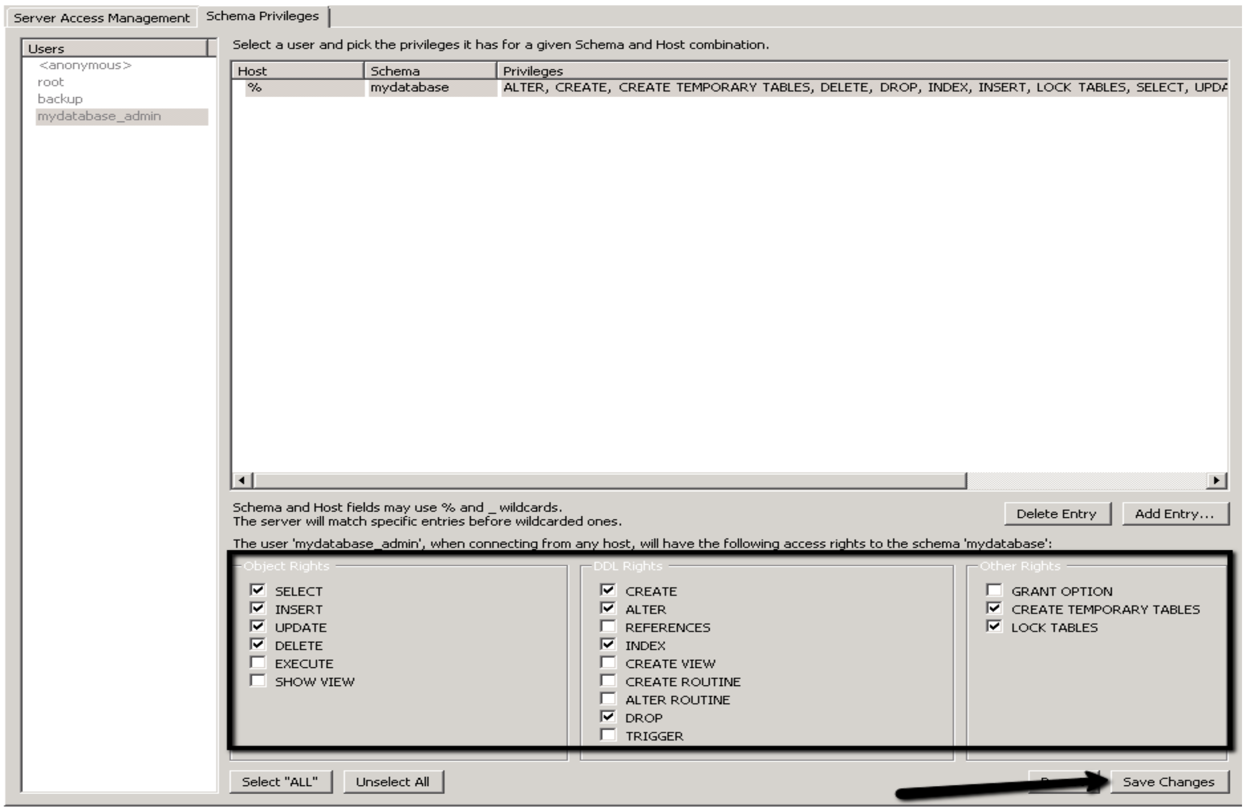
Select the Host and the Schema for which the user 'mydatabase_admin' will have the privileges you want to define.

Host
- [✓ Any Host (%)]
- Hosts matching pattern or name:
- Selected host: localhost

Schema
- Any Schema (%)
- Schemas matching pattern or name:
- [✓ Selected schema:] information_schema, **mydatabase**, mysql, performance_schema, testdb

Select the appropriate privileges to allow the user access to the selected database. Most modern website software will only require the permissions listed below. Click Save Changes to complete your new user setup.

Select, Insert, Update, Delete, Create, Alter, Index, Drop, Create Temporary Tables, Lock Tables



Users
<anonymous>
root
backup
mydatabase_admin

Select a user and pick the privileges it has for a given Schema and Host combination.

Host	Schema	Privileges
%	mydatabase	ALTER, CREATE, CREATE TEMPORARY TABLES, DELETE, DROP, INDEX, INSERT, LOCK TABLES, SELECT, UPDA

Schema and Host fields may use % and _ wildcards.
The server will match specific entries before wildcarded ones.

The user 'mydatabase_admin', when connecting from any host, will have the following access rights to the schema 'mydatabase':

Object Rights	DDL Rights	Other Rights
✓ SELECT	✓ CREATE	☐ GRANT OPTION
✓ INSERT	✓ ALTER	✓ CREATE TEMPORARY TABLES
✓ UPDATE	☐ REFERENCES	✓ LOCK TABLES
✓ DELETE	✓ INDEX	
☐ EXECUTE	☐ CREATE VIEW	
☐ SHOW VIEW	☐ CREATE ROUTINE	
	☐ ALTER ROUTINE	
	✓ DROP	
	☐ TRIGGER	

3. How to use MySql Workbench to run SQL Statements.

SQL Query Tab

The SQL query secondary tab opens by default when you make a connection to a server from the Home screen. It includes a query editor area and a toolbar. You can enter SQL statements directly into the query editor area. The statements entered can be saved to a file or snippet for later use. At any point, you can also execute the statements you have entered.

To save a snippet of code entered into the query editor, click **Save SQL to Snippets List** ( ) from the SQL query toolbar, enter a name (optional), and click OK. The following figure shows the main elements of a query tab.

SQL Editor - SQL Query Tab

Executing a SELECT query will display the associated result set in the SQL View panel, directly below the SQL Query panel. These cells are editable if MySQL Workbench is able to determine how, as for example they are editable if a Primary or Unique key exists within the result set. If not, MySQL Workbench will display a "read-only" icon at the bottom-right corner of the SQL View panel, and hovering the mouse cursor over this icon will provide a hint as to why it's not editable.

SQL Query Toolbar

The SQL query toolbar provides actions that enable you to create and manage queries. The following figure shows the set buttons in the toolbar, located within the SQL query tab.

Figure 8.3 SQL Query Toolbar

SQL query buttons (from left to right) include:

- ***Open a Script File in this Editor***: Loads content from a saved SQL script into the SQL editor.
- ***Save SQL Script to File***: Saves contents from the SQL editor into a file.
- ***Execute SQL Script***: Executes the selected portion of the query, or the entire query if nothing is selected.
- ***Execute Current SQL script***: Execute the statement under the keyboard cursor.
- ***Explain (All or Selection)***: Execute the EXPLAIN command on the query under the keyboard cursor.
  A result grid tab is also displayed when executing an EXPLAIN statement. Clicking it will execute the same query, as if **Execute SQL Script**was selected.
  Alternatively, the Visual Explain plan is already available for all executed queries. Select **Execution Plan** from the results tab to view it.
- ***Stop the query being executed***: Halts execution of the currently executing SQL script.
  Note

  The database connection will not be restarted, and open transactions will remain open.

- ***Toggle whether execution of SQL script should continue after failed statements***: If the red "breakpoint" circle is displayed, the script terminates on a statement that fails. If you click the button so that the green arrow is displayed, execution continues past the failed code, possibly generating additional result sets. In either case, any error generated from attempting to execute the faulty statement is recorded in the Output tab sheet.
  This behavior can also be set from the **SQL Execution** user preferences panel.
- ***Commit***: Commits the current transaction.
  Note

  All query tabs in the same connection share the same transactions. To have independent transactions, a new connection must be opened.

- ***Rollback***: Rolls back the current transaction.
  Note

  All query tabs in the same connection share the same transactions. To have independent transactions, a new connection must be opened.

- ***Toggle Auto-Commit Mode***: If selected, each statement will be committed independently.
  Note

  All query tabs in the same connection share the same transactions. To have independent transactions, a new connection must be opened.

Auto-commit is enabled by default, and this default behavior can be modified (disabled) under the **SQL Execution** user preferences panel.

- ***Set Limit for Executed Queries***: The default value is 1000, which appends "LIMIT 0, 1000" to SELECT queries.
  The default (1000) can be changed from the **SQL Execution** user preferences panel.
- ***Save Snippet***: Save the current statement or selection to the active snippet list.
- ***Beautify SQL***: Beautify/reformat the SQL script.
  By default, SQL keywords are changed to UPPER CASE. This functionality can be changed from the **SQL Editor** user preferences window.
- ***Find panel***: Show the Find panel for the editor.
- ***Invisible characters***: Toggle display of invisible characters, such as newlines, tabs, spaces.
  A new line is represented as ***[LF]***, a space as a single dot (.), and a tab as a right arrow.
- ***Wrapping***: Toggles the wrapping of long lines in the SQL editor.

Introduction to Data Types in MySQL.

(Character,Integer,Fixed,Floating,Date,Time,ENUM,SET,Large Objects).

Once you have identified all of the tables and columns that the database will need, you should determine each field's MySQL data type. When creating the database, as you will do in the next chapter, MySQL requires that you define what sort of information each field will contain. There are three primary categories, which is true for almost every database software:

- Text
- Numbers
- Dates and times

Within each of these, there are a number of variants—some of which are MySQL-specific—you can use. Choosing your column types correctly not only dictates what information can be stored and how, but also affects the database's overall performance. **Table 3.2** lists most of the available types for MySQL, how much space they take up, and a brief description.

Table 3.2 Here are most of the available column types for use with MySQL databases.

MySQL Datatypes		
Ty p e	**S i z e**	**D e s c r i p t i o n**
CHAR[Length]	Length bytes	A fixed-length field from 0 to 255 characters long.

VARCHAR(Length)	String length + 1 bytes	A fixed-length field from 0 to 255 characters long.
TINYTEXT	String length + 1 bytes	A string with a maximum length of 255 characters.
TEXT	String length + 2 bytes	A string with a maximum length of 65,535 characters.
MEDIUMTEXT	String length + 3 bytes	A string with a maximum length of 16,777,215 characters.
LONGTEXT	String length + 4 bytes	A string with a maximum length of 4,294,967,295 characters.
TINYINT[Length]	1 byte	Range of -128 to 127 or 0 to 255 unsigned.
SMALLINT[Length]	2 bytes	Range of -32,768 to 32,767 or 0 to 65535 unsigned.
MEDIUMINT[Length]	3 bytes	Range of -8,388,608 to 8,388,607 or 0 to 16,777,215 unsigned.
INT[Length]	4 bytes	Range of -2,147,483,648 to 2,147,483,647 or 0 to 4,294,967,295 unsigned.
BIGINT[Length]	8 bytes	Range of -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 or 0 to 18,446,744,073,709,551,615 unsigned.
FLOAT	4 bytes	A small number with a floating decimal point.
DOUBLE[Length, Decimals]	8 bytes	A large number with a floating decimal point.
DECIMAL[Length, ecimals]	Length + 1 or Length + 2 bytes	A DOUBLE stored as a string, allowing for a fixed decimal point.
DATE	3 bytes	In the format of YYYY-MM-DD.
DATETIME	8 bytes	In the format of YYYY-MM-DD HH:MM:SS.
TIMESTAMP	4 bytes	In the format of YYYYMMDDHHMMSS; acceptable range ends inthe year 2037.
TIME	3 bytes	In the format of HH:MM:SS
ENUM	1 or 2 bytes	Short for enumeration, which means that each column can haveone of several possible values.
SET	1, 2, 3, 4, or 8 bytes	Like ENUM except that each column can have more than one ofseveral possible values.

EXPERIMENT:2

AIM: Examples on i)DDL Commands: CREATE , ALTER, DROP and TRUNCATE a Table ii) Implementation of Constraints PRIMARY KEY, FOREIGN KEY, CHECK,NOT NULL, UNIQUE.

i.) Examples on DDL Commands. CREATE , ALTER, DROP, and TRUNCATE a Table.

DESCRIPTION: Data Definition Language (**DDL**) is a standard for **commands** that define the different structures in a database. **DDL statements** create, modify, and remove database objects such as tables, indexes, and users. Common **DDL statements** are CREATE, ALTER, DROP and TRUNCATE.

a) **Create:**

Syntax:

CREATE TABLE<*tablename> (column_1 datatype(size), column_2 datatype(size)…colum_n datatype(size));*

Example:

create table student(sname varchar(20),sidint(5),sclassint(1),sphoneint(10));

b) Alter:

The Alter table command is used to add, delete, modify columns, add or drop various constraints on an existing table. In other words, Alter command is used to change the structure of the table.

- ➔ **Syntax to alter table and add column:**
  ALTER TABLE *table_name*ADD *column_name datatype*;

  Example:

  Alter table student add slnamevarchar(30);

- ➔ **Syntax to alter table and delete a column:**
  ALTER TABLE*table_name*DROP COLUMN *column name;*

Example:

Alter table student drop column section;

➔ **Syntax to alter table and modify the column:**
ALTER TABLE *table_name*MODIFY COLUMN *column_name datatype*;

Example:
Alter table student modify column sidvarchar(10);

c) Drop:
This statement is used to remove a table definition and all the data, indexes, triggers, constraints and permission specifications for that table.

Syntax:
DROP TABLE<table_name>

Example:
Drop table student;

d) Truncate:
This command is used to delete complete data from table. However, the structure of table is retained.

Syntax:
TRUNCATE TABLE <table_name>

Design the following Schema Diagram :

Employee

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

Department

Dname	Dnumber	Mgr_ssn	Mgr_Start_date

Dept_locations

Dnumber	Dlocation

Project

Pname Pnumber Plocation Dnum

Works_On

Essn Pno Hours

Dependent

Essn Dependent_name Sex Bdate Relationship

EMPLOYEE SCHEMA:

employee:

create table employee(fname varchar2(10), Minit varchar2(4) ,lname varchar2(10), ssn number(10) primarykey ,bdate date ,address varchar2(10) , sex varchar2(10) ,salary number(10),dno number(10));

Table created

desc emp;

```
Name          Null?           Type

Fname                         varchar2(10)
Minit                         varchar2(10)
Lname                         varchar2(10)
Ssn              Not Null     number(10)
Bdate                         date
Address                       varchar2(10)
Sex                           varchar2(10)
Salary                        number(10)
Mssn                          number(10)
Dno                           number(10)
```

Department:

create table department (dname number(10), dnum number(10) primarykey, mssn number(20), strt_date date);

Table Created

desc department;

```
        Name              Not Null?                 Type
        dname                                       varchar2(10)
        dnum              Not Null                  number(10)
        Mssn                                        number(10)
        Date                                        date
```

dept_locations:

create table depl(dno number(10), dlocation varchar2(10) primarykey);

Table created

desc depl;

```
      Name            Null?                        Type
      Dno             Not Null                     number(10)
      dlocation       Not Null                     varchar2(20)
```

Project:

create table project (pname varchar2(10) , pno number(2) primarykey ,ploc varchar2(10), numd number(10));

Table created

desc project;

```
          Name            Null?                    Type
          Pname                                    Varchar2(20)
          Pno             Not Null                 number(2)
          Ploc                                     varchar2(20)
          Numd                                     number(2)
```

works_on:

create table works_on(essn number(10), pnum number(10), hours number(10) primarykey);

Table created

desc works_on;

```
Name            Not Null?                   Type
Essn            Not Null                    Number(10)
pnum            Not Null                    Number(10)
Hours                                       Number(10)
```

Dependent:

create table dependent (ssn number(10),depname varchar2(20),sex varchar2(2),bdate date, rs varchar2(20) primarykey);

Table Created

desc dependent;

```
Name            Null?                       Type
Ssn             Not Null                    Number(10)
Depname         Not Null                    varchar2(10)
sex                                         varchar2(10)
Bdate                                       date
Rs                                          varchar2(20)
```

Cmd: alter

Syntax:

- add new columns-alter table tablename add(new colname1 datatype(size),…..….....colname datatype(size));
- modify the existing- table tablename modify(colname new datatype(new size));
- adding constraints-
    1. alter table tablename add (primarykey (colname));

2. alter table tablename add(foreignkey(colname) references tablename(colname));

Use: modifies the structure of database.

Employee:

alter table emp modify(ssn primarykey, dno references dept(dnum));

Department:

alter table dept modify(mssn references emp(ssn),dnum primarykey));

Department_location:

alter table depl modify(foreignkey (dnum) references dept(dnum));

Project:

alter table project modify (foreignkey(numd) references dept(dnum));

Works_on:

alter table works_on modify(foreignkey(essn) references emp(ssn), foreignkey(pnum) references project(pno));

Dependent:

alter table dependent modify (foreignkey(ssn) references emp(ssn));

cmd: insert

Syntax: insert into tablename values(colname1, colname2, colname3,…..);

Use: This command is used to insert all the values into the table.

Employee:

insert into emp values(‘John’,’B‘,’Smith’,123456789,‘09-Jan-65‘,‘731 Forden Houston TX‘,’ M‘, 30000,333445555,5);

insert into emp values(‘ Franklin’,’T ‘,’Wong ‘,333445555, ‘ 08-Dec-55‘, ‘638 Voss Houston ‘,’M ‘,40000, 888665555,5);

insert into emp values (‘ Alicia’,’J ‘,’Zelaya ‘,999887777,’14-Jan-68 ‘,’332 Castle Spring TX ‘F‘,25000,987654321,4,);

insert into emp values (‘Jennifer ’,’S ‘,’Wallace‘, 987654321, ‘20-Jun-41’,’291 Berry Belliare TX‘, 43000,888665555,4);

insert into emp values(‘Ramesh’,‘K’, ‘Narayan’,666884444 ,’15-Sep-62’,‘975 Fire Oak Humble TX’,’M’ , 38000 ,33445555 , 5);

insert into emp values(‘Joyce’ ,’ A ‘, ‘English’ ,453453453 ,’31-Jul-72’ , ‘5631 Rice Houston TX ‘, ‘F’ ,25000 ,333445555,5);

insert into emp values(‘Ahmad’ , ‘V’ , ‘Jabbar’ , 987987987 , ‘29-Mar-69’ , ‘980 Dallas Houston TX’ ,’ M ‘, 25000 ,987654321 , 4);

insert into emp values(‘ James’ ,’ E’, ‘Borg’ ,888665555, ‘10- Nov-37 ‘, ‘450 stone Houston TX’ ,55000, null ,1);

Fname	Min it	Lname	Ssn	Bdate	Address	sex	sal	Mssn	Dno
John	B	Smith	123456789	09Jan65	731Forden	M	30000	33344	5
					Houston TX			555	
					638 Voss				
Franklin	T	Wong	33344555	08Dec55	Houston TX	M	40000	88866	5
					321 Castle			555	
					Spring TX				
Alicia	J	Zelaya	99988777	19Jan68		F	25000	98765	4
					291 Berry			4321	
					Bellarie TX			88866	
Jennifer	S	Wallce	987654321	20Jun41	975 FireOak	F	43000	555	4
					Humble				
					5631 Rice			33344	
Ramesh	K	Naraya	66688444	15Sep62	Houston TX	M	38000	555	5
		n			980 Dallas				
					Houston TX			33344	
Joyce	A	English	453453453	31Jun72	450 Stone	F	25000	555	5
					Houston TX				
		Jabbar						98765	

Ahmad	V		987987987	29Mar69		M	25000	4321	4
James	E	Borg	888665555	10Nov37		M	25000	NULL	1

Department:

insert into dep values (‘research’ ,5,333445555,’22may88’);

insert into dep values(‘administration’4,987654321,01Jan95);

insert into dep values(‘head quarters’ 1, 88866555, 19Jun81);

Dname	Dno	Mssn	Sdate
Research	5	333445555	22-May-88
Administration	4	987654321	01-Jan-95
Head Quarters	1	888665555	19-Jun-81

Dept_loc:

insert into depl values(1,’Houston’);

insert into depl values(4,’Stafford’);

insert into depl values(5,’Bellarie’);

insert into depl values(5,’Sugar land’);

insert into depl values(5,’Houston’);

Dno	Dloc
1	Houston
4	Stafford
5	Bellarie
5	Sugar land

5	Houston

Project:

insert into project values(‘product X’,1,’Bellarie’,5);

insert into project values(‘product Y’,2,’Sugarland’,5);

insert into project values(‘product Z’,3,’Houston’,5);

insert into project values(‘computerization’,10,’Stafford’,4);
insert into project values(‘reorganization’,20,’Houston’,1);

insert into project values(‘new benefits’,30,’Stafford’,4);

Pname	Pno	Ploc	NumD
Product X	1	Bellarie	5
Product Y	2	Sugarland	5
Product Z	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
New Benefits	30	Stafford	4

Works_on:

insert into works_on values(123456789,1,32.5);

insert into works_on values(123456789,2,7.5);

insert into works_on values(666884444,3,40.0);

insert into works_on values(453453453,2,20.0);

insert into works_on values(453453453,2,20.0);

insert into works_on values(333445555,2,10.0);

insert into works_on values(333445555,3,10.0);

insert into works_on values(333445555,10,10.0);

insert into works_on values(333445555,2,20.0);

```
insert into works_on values(999887777,30,30);

insert into works_on values(999887777,10,10);

insert into works_on values(987654321,10,35);

insert into works_on values(987654321,30,5);

insert into works_on values(987654321,30,20);

insert into works_on values(987654321,20,15);

insert into works_on(888665555,20,null);
```

Essn	Pnum	Hours
123456789	1	32.5
12345678	2	7.5
666884444	3	40
453453453	1	20
453453453	2	20
333445555	2	10
333445555	3	10
333445555	10	10
333445555	20	10
999887777	30	30
999887777	10	10
987654321	10	10
987654321	30	35
987654321	30	5
987654321	20	20
888665555	20	15

Dependent:

```
insert into dependent values( 333445555,’Alicia’,’F’,’05-Apr-86’,’Daughter’);

insert into dependent values(333445555,’Theodore’,’M’,25-Oct-83’,’Son’);

insert into dependent values(333445555,’Joy’,’F’,03-may-58’,’Spouse’);
```

insert into dependent values(987654321,’Abner’,’M’,’28-Feb-42’,’Spouse’);

insert into dependent values(123456789,’Alice’,’F’,’30-Dec-86’,’Daughter’);

insert into dependent values(123456789,’Elizabeth’,’F’,’05-may-67,’Spouse’);

Ssn	Dependent_Name	Sex	Bdate	Relationship
333445555	Alice	F	05-apr-86	Daughter
333445555	Theodore	M	25-oct-83	son
333445555	Joy	F	03-may-58	spouse
987654321	Abner	M	28-feb-42	spouse
123456789	Michael	M	04-jan-88	son
123456789	Alice	F	30-dec-88	Daughter
123456789	Elizabeth	F	05-may-67	spouse

Cmd: drop

Syntax: drop table tablename;

Use: destroying of all tables is done using this command.

Employee:

drop table emp;

Department:

drop table dept;

Department_loc:

drop table depl;

Project:

drop table project;

Works_on:

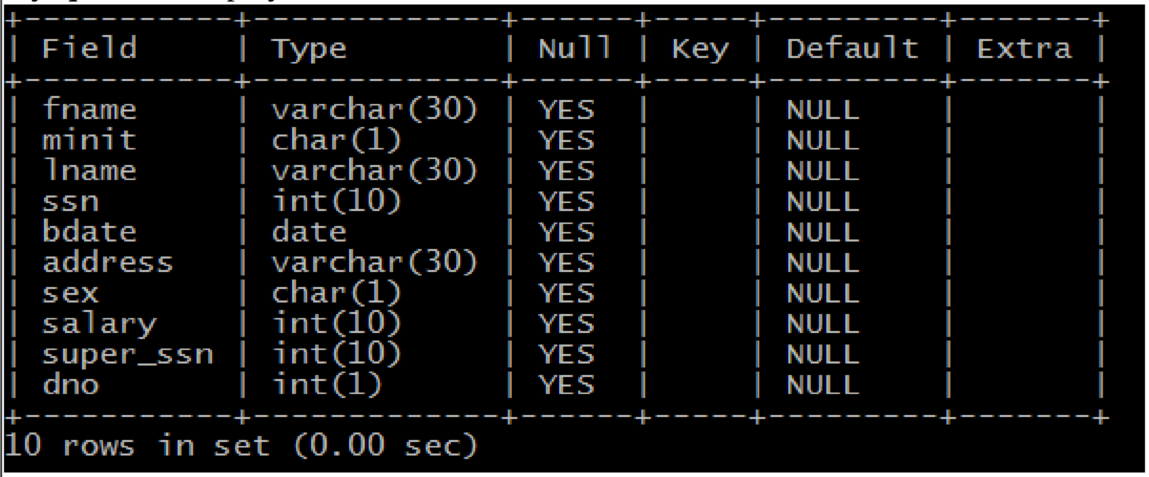
drop table works_on;

Dependent:

drop table dependent;

mysql>>alter table employee drop column dept_location;
query OK, 0 rows affected (0.60 sec)
Records: 0   Duplicates: 0   Warnings: 0

mysql>>desc employee;



```
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| fname     | varchar(30) | YES  |     | NULL    |       |
| minit     | char(1)     | YES  |     | NULL    |       |
| lname     | varchar(30) | YES  |     | NULL    |       |
| ssn       | int(10)     | YES  |     | NULL    |       |
| bdate     | date        | YES  |     | NULL    |       |
| address   | varchar(30) | YES  |     | NULL    |       |
| sex       | char(1)     | YES  |     | NULL    |       |
| salary    | int(10)     | YES  |     | NULL    |       |
| super_ssn | int(10)     | YES  |     | NULL    |       |
| dno       | int(1)      | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
10 rows in set (0.00 sec)
```

mysql>> truncate table employee;
query OK, 0 rows affected (0.29 sec)

ii. Examples on Implementation of Constraints PRIMARY KEY,FOREIGN KEY,CHECK,NOT NULL,UNIQUE.

DESCRIPTION: Constraints are the rules that are used to limit the type of data that can go into a table, to maintain accuracy and integrity of the data inside table. There are two types of constraints: **Column level constraints** and **Table level constraints** depending upon the range up to which their conditions hold. Constraints are used to make sure that the integrity of data is maintained. The following are various constraints:

a) Primary Key: Primary key constraint is used to uniquely identify each record in a database. A primary key must contain a unique value and it must not contain a null value. Primary key is usually used to index data inside a table.

➔ **Syntax: (to add primary key constraint while creating a table)**

Create table <tablename> (column_1 datatype(size) Primary key, column_2 datatype(size)…column_n datatype(size));

Example:

Create table student (sidint(5) primary key, sname varchar(30), sdept varchar(4));

➔ **Syntax: (to add primary key constraint using alter table command)**

Alter table <tablename> add primary key (column_name)

Example:

Alter table employee add primary key (eid);

) Foreign Key:

oreign Key is used to establish relation between two tables. It is used to ensure **referential integrity** etween tables. The foreign key is usually a column that is linked with the primary key of parent table. nd once a column is declared to be as foreign key, it doesn't allow inclusion of details which are not resent in the primary key of parent table. In this way it ensures referential integrity.

➔ **Syntax: (to add foreign key using alter table command)**

Alter table <tablename> add foreign key (column_child table) references parent table (column_parent table);

Example:

Alter table employeebonus add foreign key(esuperid) references employee(eid);

c) Not Null:

NOT NULL constraint restricts a column from having a NULL value. Once **NOT NULL** constraint is applied to a column, you cannot pass a null value to that column. It enforces a column to contain a proper value.

➔ **Syntax:**

Create table <table-name>(column_1 datatype(size) NOT NULL, column_2 datatype(size)…column_n datatype(size));

Example:

Create table student (sidint(5) NOT NULL, sname varchar(30), sdept varchar(3));

d) Unique:
Unique constraint prevents the user from adding duplicate values in the column.

➔ **Syntax: (to add unique constraint while creating a table)**

CREATE TABLE tablename (column_1 datatype(size) NOT NULL,column_2 datatype(size)… column_n datatype(size), UNIQUE(columnname));

Example:

CREATE TABLE Persons (IDint NOT NULL, LastName varchar(25) NOT NULL,FirstName varchar(25),Age int,UNIQUE (ID));

➔ **Syntax: (to add unique constraint using alter command)**

Alter table <table-name> add unique (column-name);

Example:
Alter table persons add unique (ID);

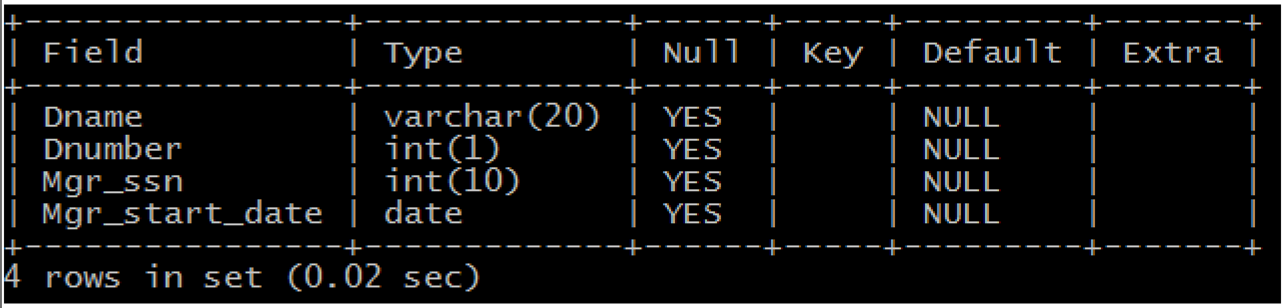
e) Check:
CHECK constraint is used to restrict the value of a column within a range. It performs check on the values, before storing them into the database. It is like condition checking before saving data into a column.
Example to apply check constraint while creating the table:
Create table student(sidint(5) NOT NULL Check(sid>0), sname varchar(30) NOT NULL, age int(2));

Example to apply check constraint using alter command:
Alter table student add check (sid>0);

mysql>> create table department(Dname varchar(20), Dnumberint(1), Mgr_ssnint(10,Mgr_start_date date);

Query OK, 0 rows affected (0.29 sec)

mysql>>desc department;



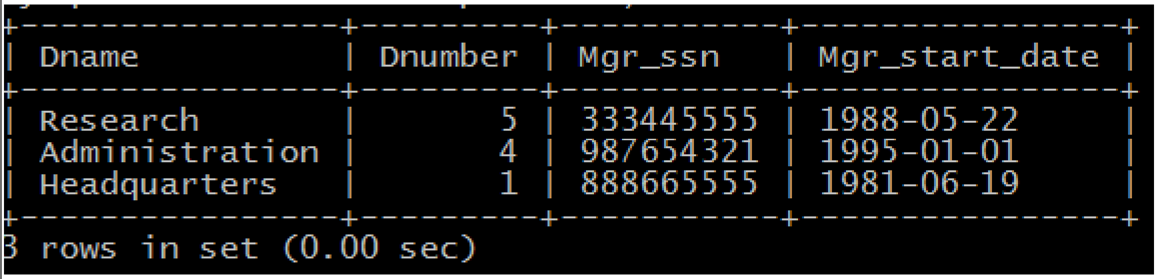
```
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| Dname           | varchar(20) | YES  |     | NULL    |       |
| Dnumber         | int(1)      | YES  |     | NULL    |       |
| Mgr_ssn         | int(10)     | YES  |     | NULL    |       |
| Mgr_start_date  | date        | YES  |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
4 rows in set (0.02 sec)
```

mysql>>insert into department values(“ Research”,5,333445555,”1988-05-22”);
Query OK, 1 row affected (0.07 sec)

mysql>>insert into department values(“Administration”,4,987654321,”1955-01-01”);
Query OK, 1 row affected (0.03 sec)

mysql>>insert into department values(“Headquarters”,1,888665555,”1981-06-19”);
Query OK, 1 row affected (0.08 sec)

mysql>>select * from department;

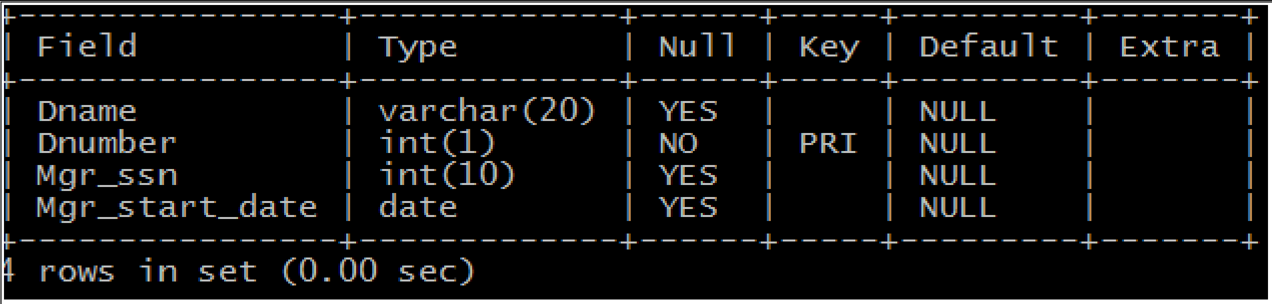


```
+----------------+---------+-----------+----------------+
| Dname          | Dnumber | Mgr_ssn   | Mgr_start_date |
+----------------+---------+-----------+----------------+
| Research       |       5 | 333445555 | 1988-05-22     |
| Administration |       4 | 987654321 | 1995-01-01     |
| Headquarters   |       1 | 888665555 | 1981-06-19     |
+----------------+---------+-----------+----------------+
3 rows in set (0.00 sec)
```

mysql>>create table dept_locations (Dnumberint(1), Dlocation varchar(20));
Query OK, 0 rows affected (0.32 sec)

mysql>> alter table department add primary key(Dnumber);
Query OK, 0 rows affected (0.64 sec)
Records: 0   Duplicates:0   Warnings:0

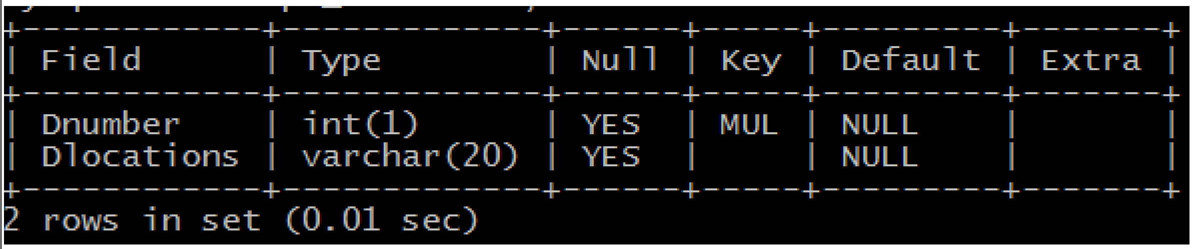
mysql>>desc department;



```
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| Dname           | varchar(20) | YES  |     | NULL    |       |
| Dnumber         | int(1)      | NO   | PRI | NULL    |       |
| Mgr_ssn         | int(10)     | YES  |     | NULL    |       |
| Mgr_start_date  | date        | YES  |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

mysql>> alter table dept_locations add foreign key (Dnumber) references department (Dnumber);
Query OK, 0 rows affected (0.97 sec)
Records: 0   Duplicates: 0   Warnings: 0

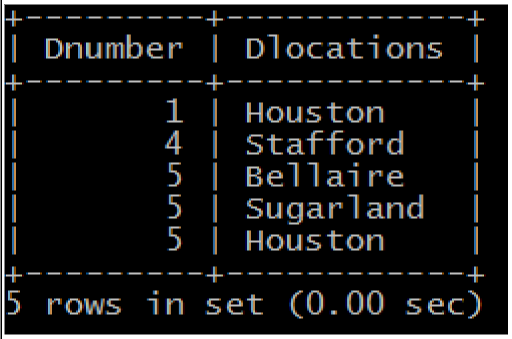
mysql>descdept_locations;



```
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| Dnumber    | int(1)      | YES  | MUL | NULL    |       |
| Dlocations | varchar(20) | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
2 rows in set (0.01 sec)
```

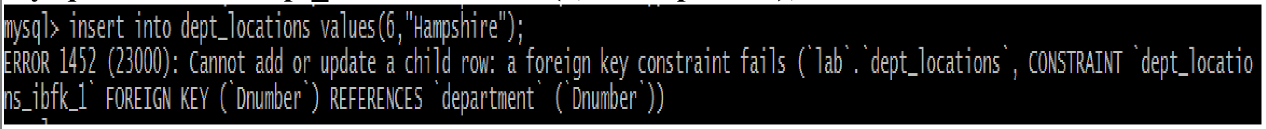
mysql>insert into dept_locations values(1,”Houston”);
Query OK, 1 row affected (0.07 sec)
mysql> insert into dept_locations values(4,”Stafford”);
Query OK, 1 row affected (0.05 sec)
mysql>insert into dept_locations values(5,”Bellaire”);
Query OK, 1 row affected (0.07 sec)

mysql>select * from dept_locations;



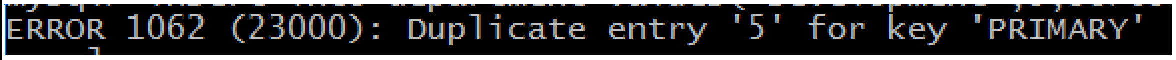
```
+---------+------------+
| Dnumber | Dlocations |
+---------+------------+
|       1 | Houston    |
|       4 | Stafford   |
|       5 | Bellaire   |
|       5 | Sugarland  |
|       5 | Houston    |
+---------+------------+
5 rows in set (0.00 sec)
```

mysql>insert into dept_locations values (6,” Hampshire”);



```
mysql> insert into dept_locations values(6,"Hampshire");
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails (`lab`.`dept_locations`, CONSTRAINT `dept_locatio
ns_ibfk_1` FOREIGN KEY (`Dnumber`) REFERENCES `department` (`Dnumber`))
```

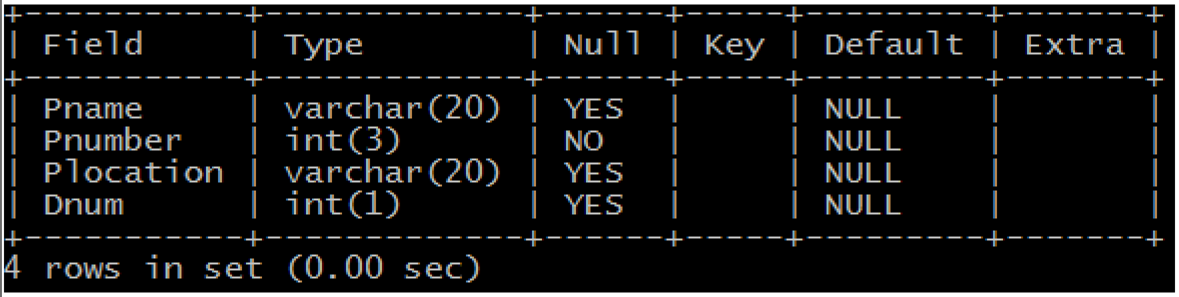
mysql>insert into department values(“Development”,5,987654322,”1995-02-01”);



```
ERROR 1062 (23000): Duplicate entry '5' for key 'PRIMARY'
```

mysql>create table project (Pname varchar(20), Pnumberint(3) NOT NULL, Plocation varchar(30), Dnumint(1));
Query OK, 0 rows affected (0.32 sec)

mysql>desc project;



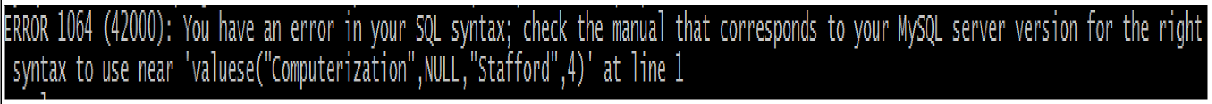
```
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| Pname     | varchar(20) | YES  |     | NULL    |       |
| Pnumber   | int(3)      | NO   |     | NULL    |       |
| Plocation | varchar(20) | YES  |     | NULL    |       |
| Dnum      | int(1)      | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

mysql>insert into project values (“ProductX”,1,”Bellaire”,5);
Query OK, 1 row affected (0.05 sec)

mysql> insert into project values("ProductY",2,"Sugarland",5);
Query OK, 1 row affected (0.09 sec)

mysql> insert into project values("ProductZ",3,NULL,5);
Query OK, 1 row affected (0.06 sec)

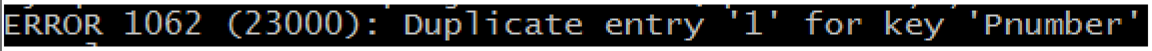
mysql> insert into project valuese("Computerization",NULL,"Stafford",4);



```
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right
 syntax to use near 'valuese("Computerization",NULL,"Stafford",4)' at line 1
```

mysql>alter table project add unique (Pnumber);
Query OK, 0 rows affected (0.79 sec)
Records: 0   Duplicates: 0   Warnings: 0

mysql>insert into project values (“ ProductA”,1,”Houston”,3);



```
ERROR 1062 (23000): Duplicate entry '1' for key 'Pnumber'
```

EXPERIMENT -3

AIM: Examples on

i. DML Commands. INSERT, UPDATE, DELETE.
ii. DCL Commands. COMMIT, ROLLBACK and SAVEPOINT.

DESCRIPTION:

DML: SELECT, INSERT, UPDATE, DELETE

. **Cmd: Update**

Syntax: Update table set colname=expression, colname=exp;

Use: It updates all the columns given based upon expression value.

1. Update the salary of employees with ssn '999887777' to 28000

```
>update emp set sal=28000 where ssn=999887777.

1row updated.
```

2.Update the dep_no of employee '999887777' ssn to 1.

```
>update dno set ssn=1 where ssn=999887777;

1 row updated.
```

3. Show that the resulting salaries of every emp working on the project 'productX' is given a 10% rise.

```
>select sal+0.1*sal hike, dno from emp, project where emp.dno=project.numd and pname='productX';

HIKE    DNO
--------  ----------
33000      5
44000      5
41800      5
```

```
27500       5
```

4.Update salary of all employees to 40000 .

```
>Update emp set sal=40000;

1 row updated.
```

5.Change the location and controlling dept.no of project no 10 to Bellarie and 5.

```
>Update project set ploc='Bellarie',numd=5 where project.pno=10;

1 row updated.
```

Cmd: Select

Syntax:

- Select colname1,colname2,…..colnamen from tablename;
- Select * from tablename;
- Select colname1,colname2,….colnamen from tablename where cond;

1.Retrieve bdate and address of all employees whose name is 'John B Smith'

```
>Select bdate , address from emp where fname='John' and min='B' and lname='Smith';
```

<u>Output:</u>

```
Bdate             Address

09-jan-65        731 Forden,Houston,TX.
```

2.Retrieve the name and address of all employees whose dept no is 5.

```
>Select fname , min, address from emp where dno=5;

Output:

Fname Minit      lname           Address

John               B            smith          731 forden,Houston,TX
```

```
Franklin      T      Wong        638 voss, Houston,TX
Ramesh        K      Narayan     975 fireoak ,Humble,TX
Joyce         A      English     5631 Rice,Houston,TX
```

3.For each employee, retrieve the emp's fname and lname and his /her immediate supervisor.

Select emp.fname empf ,emp.ssn ,emp.lname empl ,emp2.fname emp2f , emp2.ssn emp2ssn , emp2.lname emp2l from emp ,emp2 where emp.ssn=emp2.ssn;

Output:

```
empf          ssn           empl        emp2f ssn2         emp2l

John          123456789   smith         Franklin      333445555  Wong

Franklin      333445555  wong     james 888665555     borg

Alicia        999887777  zelaya         Jennifer      987654321  wallace

Jennifer      987654321     wallace     james         888665555   borg

Ramesh        666884444 narayan   faranklin     333445555     wong

Joyce         453453453     english English       333445555     wong

Ahmed987987987     jabber       Jennifer      987654321     wallace
```

4.Retrieve the sal of all employees .

Select salary from emp;

Output:

```
Sal
30000
40000
25000
43000
38000
25000
25000
25000
```

Cmd: delete

Syntax:

- delete from tablename;
- delete from tablename where search condition;

Use: To delete the particular row or all the columns data.

1. delete the employee tuple in ssn ‘999887777 ‘.

delete from emp where ssn=9998887777;

1 row deleted.

2. delete only those rows that have lname Smith.

delete from emp where hours=’Smith’;

1 row deleted.

3. delete the rows from works_on table where hours is 32.5.

delete from works_on where hours=32.5;

1 row deleted.

4. delete the rows from project where pname is productX.

delete from project where pname=’productX’;

1 row deleted.

Cmd: in

Syntax: select colname1, colname2 ….colnamen where colname in (condition);

Use: To retrieve the columns present in many conditions

1. Retrieve the social security numbers of all employees who works on project numbers 1, 2 or 3.

select ssn from emp ,project where emp.dno=project.numd and pno in(1,2,3);

output:

ssn

```
123456789

123456789

123456789

333445555

333445555

333445555

666884444

666884444

666884444

453453453

453453453

453453453
```

2. Retrieve the emp's tuple where employee lname is smith or Wallace.

select fname,lname from emp where lname in ('smith' , 'wallace');

output:

```
fname                          lname

john                                 smith

jennifer                             Wallace
```

ii.DCL: COMMIT WORK, ROLLBACK,SAVEPOINT Commands.

. Aim: Insert data into table using COMMIT, ROLLBACK, SAVE POINT, in PL/SQL block

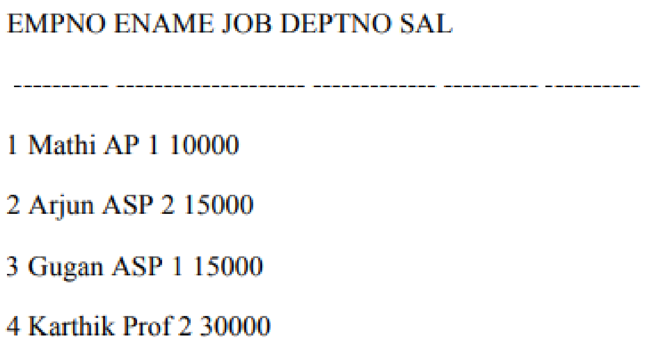
Description: COMMIT command is used to make changes permanently save to database during the current transaction

ROLLBACK command is used to execute at the end of current transaction and undo any changes made since the begin transaction.

SAVE POINT saves current point with the unique name in the transaction.

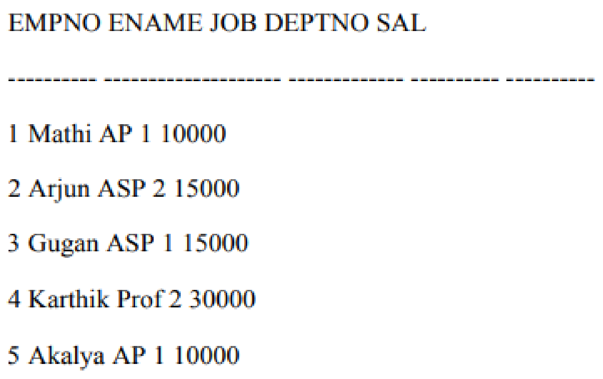
i. Write a query to implement the save point

```
Ans: SQL> SAVEPOINT S1; Savepoint created.

SQL> select * from emp;
```



```
EMPNO ENAME JOB DEPTNO SAL
---------- -------------------- ------------- ---------- ----------
1 Mathi AP 1 10000
2 Arjun ASP 2 15000
3 Gugan ASP 1 15000
4 Karthik Prof 2 30000
```

```
SQL> INSERT INTO EMP VALUES(5,'Akalya','AP',1,10000);

1 row created.

SQL> select * from emp;
```



```
EMPNO ENAME JOB DEPTNO SAL
---------- -------------------- ------------- ---------- ----------
1 Mathi AP 1 10000
2 Arjun ASP 2 15000
3 Gugan ASP 1 15000
4 Karthik Prof 2 30000
5 Akalya AP 1 10000
```

ii. Write a query to implement the rollback

Ans: SQL> rollback s1;

SQL> select * from emp;

```
EMPNO ENAME JOB DEPTNO SAL
---------- -------------------- ------------- ---------- ----------
 1 Mathi AP 1 10000
2 Arjun ASP 2 15000
3 Gugan ASP 1 15000
4 Karthik Prof 2 30000
```

iii. Write a query to implement the commit

Ans: SQL> COMMIT; Commit complete.

Code: set auto commit on;

```
 declare
 begin
         insert  into areas values(10,100);
         insert  into areas values(20,200);
 save point A
         insert into areas values(30,300);
         insert into areas values(40,400);
save point B
         insert into areas values(50,500);
rollback to B
end;
/
```

::procedure is successfully completed.

EXPERIMENT -4

Examples on retrieving data from a single table using

i)SELECT statement

ii) SELECT statement with where clause(Comparison Operators, AND, OR, NOT, IN, BETWEEN,LIKE)

iii) ORDER BY clause(sort by column name)

iv) LIMIT clause

Cmd: AND

Syntax: select colname1, colname2……. Colnamen from tablename where cond1 and cond2;

Use: This selects the combined result from both conditions

1. Retrieve details from works_on table whose ssn is 333445555 and pno is 10.

select* from works_on where essn=333445555 and pnum=10 ;

Output:

```
Essn            pnum              hrs

333445555          10                   10
```

Cmd: OR

Syntax: select colname1, colname2.…. colnamen from tablename where cond1 or cond2;

Use: This selects the tuples which are common in both cond1, cond2

1. Retrieve the emp's whose minit is A or B or K

Select fname,lname,minit from emp where min='A' or min= 'B' or min='K';

Output:

```
Fname      lname       minit

John          smith         B
```

Ramesh	narayan	K
Joyce	English	A

Cmd:BETWEEN

Syntax: between value1 or/and value

Use: It get the values between two ranges.

1. Retrieve all employees of dep whose salary is between 30000 and 40000.

select ssn,salary from emp where emp.dno=5 and sal between 30000 and 40000;

Output:

Ssn	salary
123456789	30000
333445555	40000
666884444	38000

Cmd: LIKE

Syntax:

- %: matching any string
- -: underscore for any character

Use: it allows comparison of one string value with another string value which is not identical . This can be achieved by using wild card characters like %,-.\

1. Retrieve all employees whose address is' Houston,TX'.

select ssn,address from emp where address like '%Houston,TX';

Output:

Ssn	address
123456789	731,fordan,Houston,tx

```
333445555            638,vossHouston,tx

43434343             531,riseHouston,tx

888665555            450,stoneHouston,tx
```

2. Find all the employees who are born during 1950's.

select fname,ssn from emp where to_char (bdate,'yy) like '5%';

Output:

```
Fname              ssn

John           123456789

Alicia         999887777

Ahmad          987987987
```

Cmd:ORDERBY

Syntax: select colname1 , colname2……..colnamen from tablename orderby colname1…..colnamen;

Use: To arrange in an alphabetical order and desending and ascending order also.

1. Retrieve a list of emp's and the projects they are working on ordered by dept and within each department ordered by alphabetically by lname, fname.

select fname, lname ,pno,numd from emp ,project where emp.dno=project.numd order by dno,fname,lname;

Output:

```
Fname              lname

Alicia             zelaya

James                      borg

Jennifer                   Wallace

Jennifer                   Wallace

Ahmad              jabbar
Ahmad               jabbar

Franklin                 wong

Franklin                 wong

Franklin                 wong

John                     smith

John                     smith

John                     smith

Joyce                    english

Joyce                    english

Joyce                    English

Ramesh                   narayan

Ramesh                   narayan

Ramesh                   narayan
```

2. Retrieve department number from department table ordered by mgr_start_date

select dnum from department order by sdate;

Output:

```
Dnum

1

5

4
```

Cmd: DISTINCT

Syntax: select distinct * from tablename;

Use: It retrives distinct set of values from the table.

1. Retrieve distinct salaries from employee table

select distinct sal from emp;

Output:

<u>Sal</u>

30000

40000

25000

43000

38000

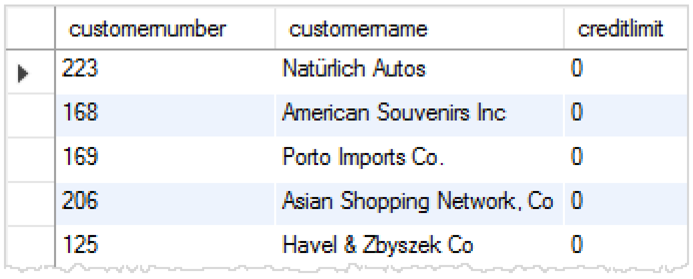
55000

LIMIT clause

LIMIT clause to constrain the number of rows returned by a query.

The LIMIT clause is used in the SELECT statement to constrain the number of rows to return. The LIMIT clause accepts one or two arguments. The values of both arguments must be zero or positive integers.

```
SELECT
    customerNumber,
    customerName,
    creditLimit
FROM
    customers
ORDER BY creditLimit DESC
LIMIT 5;
```



customernumber	customername	creditlimit
223	Natürlich Autos	0
168	American Souvenirs Inc	0
169	Porto Imports Co.	0
206	Asian Shopping Network, Co	0
125	Havel & Zbyszek Co	0

In this example:

- First, the ORDER BY clause sorts the customers by credits in low to high.
- Then, the LIMIT clause returns the first 5 rows.

EXPERIMENT -5

AIM: Examples on Functions in MySQL: String, Numeric,Date, Time and Other Functions.

i.) To implement conversion, date, string functions

- **Instr:**gets the appearance of particular character in the string

  Eg: select instr('student','t',1,32) from dual;

- **Concatenation:** combines two strings

  Syntax: select concat (<str1>,<str2>) from dual;

  Eg: select concat ('pvp','sit') from dual;

- **Least:** It gives the least value of set of values

  Syntax: select least(<number1>,<number2>) from dual;

  Eg:select least(12,2) from dual;

- **Greatest:** it gives greatest values among set of values

  Syntax: select greatest (<num1>,<num2>) from dual;

  Eg: select greatest(1,2,34) from dual;

- **Truncate(trunc):**truncate the value to a special decimal number

  Syntax: select trunk(<num>,<decimal>) from dual;

  Eg: select trunk(153.82,1) from dual;

- **Round:** it rounds the value
  Syntax: select round (<num>,[rounded to places]) from dual;
  Eg: select round (51, 234, [2]) from dual;

Output:

1) INSTR('STUDENT','T',1,2)
   7
2) CONCAT

PVPSIT

3) LEAST(10,5,15,25)

   5

4) GREATEST(10,5)

   10

5) TRUNC(153.81.1)

   153.8

6) ROUND(5.152)

   5

ii.CONVERSION FUNCTIONS:

- **TO_CHAR(N,[FMT]):** converts a value of number date type to a value of char data type using option format string.

  Select to_char(1000,'$9,999');

- **TO_NUMBER(CHAR):** converts char value containing no to number type.

  select to_number(24) from dual;

- **TO_DATE(CHAR,FMT):** converts char field to date filed.

  select to_date('18-07-2010','dd-mm-yy') from dual;

iii.STRING FUNCTIONS:

- **LPAD:** returns char1,leftpadded to length with sequence of character in char2.

  select lpad('pvp','5,'*');

- **RPAD:** returns right padded to length n with sequence of character in char2

  Select rpad('pvp',5,'*') from dual;

Output:

- TO_CHAR (1000,9,999)

  1,000

- TO_NUMBER(24)

  24

- TO_DATE

  18-JUN-10

- LPAD

  **PVP

- RPAD

  PVP**

- **LTRIM:** ltrim(char,[set])-removes chars from left of char with initial characters removes up to 1st char

  Select ltrim (‘siddhartha’,’sid’); from dual;

- **RTRIM:** returns chars after right chars removed after char not in set.

  select rtrim(‘siddhartha’,’artha’); from dual;

- **LOWER:** return string with all lower case letters

  select lower(name) from sales;

- **UPPER:** returns uppercase letters

  select upper(name) from sales;

- **INITCAP:** returns initial letter as capital letter

  select initcap(name) frm sales;

- **LENGTH:** length(char)-returns length of chars select length(name) from client;

Output:

- LTRIM

  Hartha

- RTRIM

  Sidd

- LOWER(NAME)

  Kiran

  Maish

  Ashish

- UPPER(NAME)

  KIRAN
  MANISH
  RAVI
  ASHISH

- INITCAP(NAME)

  Kiran
  Manish
  Ravi
  Ashish

- LENGTH

  12
  15
  15
  13

SUBSTR: Extracts substring from given nvalue

select substr(name,7) from client;

DATE FUNCTIONS:

- **SYSDATE**: returns sysdate from dual;

select sysdate from dual;

- **NEXT_DAY:** Returns the date of str week day named by char

  select next_day(sysdate,’sat’) from dual;

- **ADD_MONTHS:** returns add_months(d,n) returns date after adding the no of months specified within the function
- **LAST_DAY:**Returns last_day of the month specified within function

  select last_day(sys_date) from dual;
- **MONTHS_BETWEEN:** returns months between d1 and d2

  select months_between (sysdate,’13-aug-2015’) from dual;

Output:

- SYSDATE

  07-SEP-2010

- NEXT_DAY

  18-JUL-15

- ADD_MONTHS

  13-NOV-15

- LAST_DAY

  31-JUL-15

- MONTHS_BETWEEN

EXPERIMENT -6

AIM: Examples on Summary Queries: Queries using Aggregate functions,GROUP By and Having Clauses, ROLLUP Operator.

Aggregate functions: These are also called group functions; they include min, max, count, avg and sum.

Count: This function count no. of rows

1. Count the total number of employees whose salaries exceed 400000 in each department but only for department where more than 3 employees work

select count (*),dnum from emp,dep where emp.dn0=dnum and emp.sal>40000 group by dnum having count(*)>0;

Output:

```
Count(*)     dnum

1  1
1  4
```

2. For each department that has more than 3emps ,retrieve the depy. No. and no. of its emps who are more than 30000

select count(*),dnum from emp,dept where emp.no=dept.dnum and emp.sal>30000 group by dnum having count(*)>1

Output:

```
Count(*)     dnum

      1      5
```

3. Retrive the total no.of employees in the company

select count(ssn) from emp;

Output:

```
Count(ssn)
8
```

4. Retrive the no. of emps in research dept.

select count(dno) from emp,dept where emp.dno=dept.dnum and dname='research';

Output:

<u>Count(dno)</u>

4

5. Count the no. of distinct salary values in the database employee

select count(distinct sal) from emp;

Output<u>:COUNT(DISTINCT SAL)</u>

```
                    7
```

<u>SUM, MIN, MAX,AVG:</u>

1. Find the sum of salaries of all employees the maximum salary,minimum salary and the avg salary

>select sum(sal).max(sal),min(sal),avg(sal) from emp;

2. Find the sum of salaries of the research dept as well as the max,min,avg sal in dept

>selectsun(sal),max(sal),min(sal),avg(sal), from emp dept where emp.dno=dept.dnum and dname='reasearch';

Output:
```
 1) SUM (SAL)    MAX (SAL)     MIN (SAL)    AVG (SAL)
          ------------------------------------------------------------------
              284000          55000             25000             35500
          2)  SUM (SAL)    MAX (SAL)    MIN (SAL)    AVG (SAL)
          --------------------------------------------------------------------
              133000                40000             25000         33250
```

Group by having:

1.for each department ,retrieve the dno ,no of emp's in the dept and there avg salary.

Select dno ,count(ssn) ,avg(sal) from dept,emp where emp.dno=dept.dno group by dno;

Output:

```
Dno          count(ssn)          avg(salary)

1                  2             41500

5                  4             33250

4                  2             34000
```

2.for each project retrieve the pno,pname,&no of employes who works on that project from the project table

Select pno,pname,count(essn) from project,workson where project.pno=workson.pnum group by pname,pno;

Output:

```
Pno          pname          count(essn)

3            productZ             2

10           computerization      3

2            productY        `    3

20           Reorganisation       3

1            productX             2

30           NewBenifits          3
```

3.for each project,on which more than 2 employees work,retrieve the pno ,pname,no of emp's who work on project

Select pno,pname,count(essn) from project,workson where project.pno=workson.pno group by pname,pno having count(*)>2;

Output:

```
Pno     pname              count(essn)

10           computerisation      3

2            productY             3

20           Reorganisation       3

30           newbenifits          3
```

4.for each project,retrieve the pno,name,no of emp's from dept 5 who work on the project

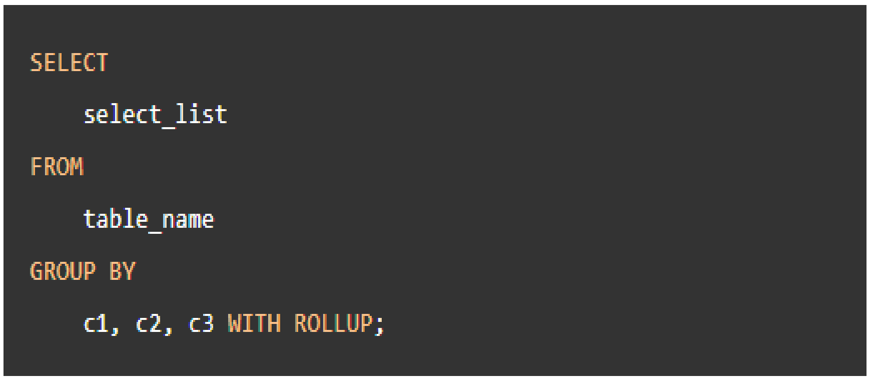
Select pno,pname,count(ssn) from project,workson,emp,dept where project.pno=workson.pno and emp.dno=dep.dnum and project.numd=emp.dnum and emp.dnum=5 group by pno,pname;

Output:

```
Pno    pname          count(ssn)

3      productZ            8

2      productY            12

1      productX            8
```

ROLLUP Operator:

The ROLLUP clause is an extension of the GROUP BY clause with the following syntax:

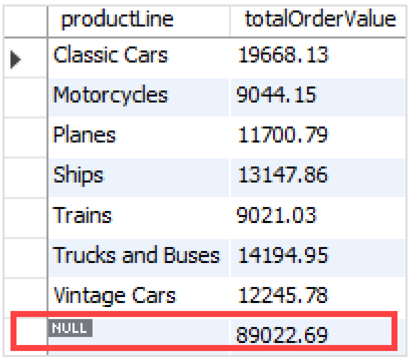


```
SELECT
    select_list
FROM
    table_name
GROUP BY
    c1, c2, c3 WITH ROLLUP;
```

The ROLLUP generates multiple grouping sets based on the columns or expressions specified in the GROUP BY clause. For example:

```
SELECT
    productLine,
    SUM(orderValue) totalOrderValue
FROM
    sales
GROUP BY
    productline WITH ROLLUP;
```

Here is the output:



productLine	totalOrderValue
Classic Cars	19668.13
Motorcycles	9044.15
Planes	11700.79
Ships	13147.86
Trains	9021.03
Trucks and Buses	14194.95
Vintage Cars	12245.78
NULL	89022.69

As clearly shown in the output, the ROLLUP clause generates not only the subtotals but also the grand total of the order values.

EXPERIMENT -7

AIM: EXAMPLES ON INNER JOIN ,OUTER JOIN USING ,NATURAL KEYWORDS

Join:

1. For every project located in Stafford , retrieve the project num controlling dept no ,manger last name , address and birth date using join.

Select pno, numd,lname,address,bdate from emp ,project where project.numd=emp.dno and project.ploc='Stafford';

Output:

```
Pno             numd        lname        address                 bdate
30                 4             wallace      291berry,Bellaire,Tx   20Jun-41
10                 4             wallace      291berry,Bellaire,Tx   20Jun-41
30                 4             jabbar       980dallasHouston,Tx  29March-69
10                 4             jabbar       980dallasHouston,Tx  29March-69
```

Natural Join:

1. Retrieve the names and address of all employees who works on research department using natural join.

Select fname , lname , address from emp e natural join dept d where e.dno=d.dnum and dname='research';

Output:

```
Fname`      lname        address
John                smith               731,fordan,Houston,TX
Franklin            wong                638,voss,Houston,TX
Ramesh              narayan             975,fireOak,Humble,TX
Joyce               English             531,rice,Houston,TX
```

Left Outer Join:

1. Retrieve the names of employees and supervisors using left outer join.

select e.fname , e.lname , s.fname , s.lname from emp e left outer join emp s on s.ssn=e.ssn;

Output:

```
Fname       lname        fname        lname
Franklin            wong                john                smith
James               borg                franklin            wong
```

```
Jennifer            Wallace             Alicia              zelaya
James               borg                Jennifer            Wallace
Franklin            wong                ramesh       narayan
Franklin            wong                joyce               English
Jennifer            Wallace             ahmad        jabbar
Ahmad        jabbar
Joyce               English
Alicia              zelaya
Ramesh              narayan
John                smith
```

Right outer Join:

1. Retrieve the names of employees and supervisor using right outer join.

Select e.fname, e.lname, s.lname, s.fname from emp e right outer join emp s on e.ssn=s.ssn;

Output:

```
Fname        lname         fname         lname
john                smith        franklin            wong
franklin            wong         james               borgAhmad           jabbar
                                 Joyce               English
                                 Alicia              zelaya
                                 Ramesh              narayan
                                 Joyce               smith


Alicia              zelaya  Jennifer            Wallace
Jennifer            Wallace      james               borg
Ramesh              narayan      franklin            wong
Joyce               English      frnklin      wong
Ahmad        jabbar        Jennifer            Wallace

                                 Ahmad        jabbar
                                 Joyce               English
                                 Alicia              zelaya
                                 Ramesh              narayan
                                 Joyce               smith
```

EXPERIMENT – 8

AIM: Examples on SUB/SUMMARY Queries Using IN, ANY, SOME,
ALL , EXISTS and NOT EXISTS functions

Retrieval of data from interrelated tables in the database in several ways

In: (nested queries)

1. select the project no fro projects that have an employee with lname smith as manager.

select pno from project where pno in (select pno from emp, dep, project where emp.dnum=emp.dno and ssn=mssn and lname='smith');

Output:

No rows selected

2. List the essn of all employees who works on same project, hours on some project that employees with essn 123456789 and 999887777 from works_on table.

Select essn from works_on where (pnum,hours) in (select pnum,hours) in (select pnum,hours from works_on where essn=123456789 and essn=999778888);

Output:

No rows selected

3. List the names of emp's whose salary is greater than the salary of all employees in dept '5'.

Select fname ,lname from emp where sal>all(select sal from emp where dno=5);

Output:

Lname	fname
Wallace	Jennifer
Borg	james

Correlated Nested Queries:

any:

List the names of employee whose salary is above average of their department.

Select fname, lname from emp, dep where sal > any (select avg (sal) from emp group by dno);

Output:

Lname	fname
Borg	james
Wallace	Jennifer
Wong	franklin
Narayan	ramesh

all:

Retrieve the name of emp whose salary is above salaries of all employees in the dept.

```
select fname, lname from emp where sal > all ( select sal from emp);
```

Output:

```
No rows selected
```

<u>Exists:</u>

1.List the names of employees who have atleast one dependent

```
Select fname from emp e where exists(select * from dependent d where e.ssn=d.ssn) and exists(select * from dependent dp where e.ssn=dp.msssn);
```

Output:

```
Fname
Franklin
Jennifer
```

2.retrive the names of employees whose dependent name is same as employee name

```
Select fname,lname from emp where exists(select * from dependent d where d.dname=e.fname and e.ssn=d.essn);
```

Output:

```
No rows selected.
```

<u>Not exists:</u>

1.retrive the names of employess who have no dependents

```
Select fname from emp e where not exists(select * from dependent d where e.ssn=d.ssn);
```

Output:

```
Fname
Alicia
Ramesh
Joyce
Ahamd
James
Wallace            Jennifer
Wong               franklin
Narayan            ramesh
```

EXPERIMENT -9

AIM: Examples on
i)Creating INDEXES and VIEWS
ii) INSERT,DELETE and DROP on VIEWS

CREATING INDEXES

1.Create index on ssn of employee table;

create index field_name on table_name(field_name);

Q: create index emp-ssn-ix on employee(ssn);

2.Create index on ssn and dno of employee table.

Q Create index emp-ssn-dno-ix on employee(ssn,dno);

DROPPING INDEXES

1.Drop index on employee table

Drop index employee-ssn-ix on employee;

Creation,dropping and updation through View.

View: It is a virtual table.if a table is derived from a another table then a view is created.

Cmd: View

Syntax:create view name of the view as select colname1,colname2…..colnamen from tablename;

1.create a view with emp details of emp's who are working in dep5.

Create view emp_view as select * from employee where dno=5;

Output:

View created

2.create a view to list the details of emp who r workg in dep5

Create view emp_views as select * from emp where dno=5;

Output:

View created

3.create a view to list the details of the dept with name ‘research’.

Create view dept_view as select * from emp,dept where emp.dno=dept.dnum and dname=’research’;

Output:

View created

4.create a view to list the names of emp’s the project names under where they are workg and the no of hrs

Create view ename as select lname ,fname,pname,hrs from emp,project,workson where emp.ssn=workson.essn and project.numd=emp.dnum and project.pno=workson.pnum;

Output:

View created

5.create a view to list the details of emp’s with dept ‘research’.

Create view dept_views as select * from emp,dept where emp.dno=dept.dnum and dname=’research’;

Output:

View created

6.create a view to the dept, to list the sum of the salaries of employees in that dept.

Create view dep_view as select dname,count(*) no of emp’s ,sum(sal) total from emp,dep where emp.dno=dept.dnum group by dname;

Output:

View created

7.retrive the lname,fname of the emp who workson the project ‘productX’ as the name.

Create view won_view as select fname,lname from emp,project where pname=’productX’ and project.numd=emp.dno;

Output:

View created

UPDATION OF VIEWS

1.update sal of emp’s where fname is ‘john’.

Update emp_view set sal=0 where fname=’john’;

Output:

1 row updated

DROPPING OF VIEWS

1.drop a view with emp details who are working in dept5.

Drop view emp_view;

Output:

View dropped

2.drop a view with the details of dept with name ‘reasearch’

Drop a view dept_view ;

Output:

View dropped

3.drop a view with project names ,the names of emp’s where there are working

And no of hrs.

Drop view enames_view;

Output:

View dropped

4.Drop a view with the details of dept with name ‘reasearch’

drop view edetails_view;

Output:

View dropped

5.Drop a view to list the sum of sal’s od emp’s in the dep

Drop view dept_view ;

Output:

View dropped

6.Drop a view with details of names who work on project

Drop view workson_view ;

Output:

View dropped

VIEW WITH CHECK OPTION:

1.Create a view with employee details who are working in departments with check option.

Create view emp-views as select ssn,salary,dno from employee where dno=5 with check option.

2.Create a view to list the details of department with name ‘RESEARCH’ using check option.

Create view dept-view as select fname,lname,dno, dnumber and dname=’Research’ with check option.

EXPERIMENT-10

AIM: Examples on
i.)Create and Call STORED PROCEDURE (IN,OUT,INOUT Parameters) , Drop a STORED PROCEDURE.
ii.)Create,call and Drop a FUNCTION.
iii.)Create and Drop a TRIGGER

Description: PL/SQL process procedural capabilities and it includes declaration section where we declare variables and these types which is optional and execution of code takes place between begin and end and exception handling section where we handle exceptions.

PROCEDURES:

```
show databases;
use c1;
show tables;
select * from employees;
```

Procedures:

1)Create a procedure to display salary of an employe using in , out parameteres .

```
Delimiter //

Create procedure sale6(in eno int,out salar int(10))

Begin

Select salary into salar from employees where ssn=eno;

End

//

Delimiter ;

call sale6(123456789,@salar);

select @salar;
```

output:

```
@salar
3000
```

2) Create a procedure to raise(10%) salary of an employee using in parameter

```
Delimiter //
Create procedure sale14(in eno int)
Begin
declare sal int;
Select salary into sal from employees where ssn=eno;
if sal<35000 then
set sal= sal+(sal*0.1) ;
select "salary updated " as salary_status;
select sal;
else
select "salary not updated " as salary_status;
end if;
End
//
Delimiter ;
call sale14(123456789);
```

output:

```
33000
```

3) . Display name of an employee whose ssn is 123456789.

```
Delimiter //
Create procedure p1(in eno int)
begin
declare name varchar(30) ;
Select fname into name from employees where ssn=eno;
Select name;

end

//

Delimiter ;

call p1(123456789);



output:

John

(or)

(Note: Display name of  student whose stu_id  is 101.)



Delimiter //
Create procedure tt(in eno int)
begin
declare name varchar(30) ;
Select fname into name from employees where ssn=eno;
Select name;
end
//
Delimiter ;
call tt(123456789);
*********************************************
Delimiter //
Create procedure tt33(in eno int)
Begin
declare sal int(10);
Select salary into sal from employees where ssn=eno;
Select sal;
End
//
Delimiter ;
call tt33(123456789);
*****************************
Delimiter //
Create procedure sale6(in eno int,out salar int(10))
Begin
Select salary into salar from employees where ssn=eno;
End
//
Delimiter ;
call sale6(123456789,@salar);
select @salar;
```

```
output:
@salar
3000

*************************
desc employees;
select * from employees;
*************************
Delimiter //
Create procedure tt33(in eno int)
Begin
declare sal int(10);
Select salary into sal from employees where ssn=eno;
Select sal;
End
//
Delimiter ;
call tt33(123456789);

output:
sal
3300

*****************************
Delimiter //
Create procedure sale14(in eno int)
Begin
declare sal int;
Select salary into sal from employees where ssn=eno;
if sal<35000 then
set sal= sal+(sal*0.1) ;
select "salary updated " as salary_status;
select sal;
else
select "salary not updated " as salary_status;
end if;
End
//
Delimiter ;
call sale14(123456789);

output:
33000



********************************************
select * from employees;

Delimiter //
```

```
Create procedure tt35()
Begin
Declare sal int;
Declare eid int;
Declare sta varchar(30) default "update";
Declare r_f tinyint default false;
Declare cur_sal cursor for select ssn,salary from employees where salary<40000;
Declare continue handler for not found set r_f=true;
Open cur_sal;
While r_f=false do
Fetch cur_sal into eid,sal;
Update employees set salary=sal+(sal*0.1) where ssn=eid;
Select sal;
Select eid,sta as salary_status;
End while;
Close cur_sal;
End
//
Delimiter ;
Call tt35();
```

output:
employee table values less than 3500 automatically increase

FUNCTIONS:

funtion1: Create a function to display sum of salaries of employees belonging to a particular department.

```
delimiter //
create function get_sal(deptno int) returns decimal(9,2)
begin
declare sum_sal decimal(9,2);
select sum(salary) into sum_sal from employee where dno=deptno;
return sum_sal;
end
//
delimiter ;
select get_sal(5)as dept_sal;
```

Output : 133000.00

Function 2:create a function to count the no.of employees belonging to department 4

```
delimiter //
create function get_count(deptno int) returns int
begin
declare ecount int;
select count(*) into ecount from employee where dno=deptno;
return ecount;
end
//
delimiter ;
select get_count(4) as no_of_emps;
```

Output : 3

----

function 3: create a function to display the name of employee for a given employee number

```
delimiter //
create function get_name(eno int) returns varchar(30)
begin
declare ename varchar(30);
select fname into ename from employee where ssn=eno;
return ename;
end
//
delimiter ;
select get_name(123456789) as eno;
```

Output: John

----

function 4: create a function to display the count of number of projects a given employee is working on

```
delimiter //
```

```
create function get_proj(eno int) returns int
begin
declare ecount int;
select count(Pno) into ecount from works_on where essn=eno;
return ecount;
end
//
delimiter ;
select get_proj(123456789)as projects;
```

Output: 2

TRIGGERS:

DEVELOP PROGRAMS USING BEFORE AND AFTER TRIGGERS, ROW AND STATEMENT TRIGGERS AND INSTEAD OF TRIGGERS

Aim: To implement triggers in the programs.

Implementation:

```
Create table schema(userid_id Varchar2(20),action Varchar2(20),log_date(date));

Select * from schema;

Create or replace trigger  trig after logon on schema

Begin

        Insert into schema values(user,’loggingin’,sysdate);

End;

/

Select * from schema;

Drop trigger trig;

Drop table schema;
```

//create view

```
Create view trig_view as select ssn,salary,count(dp_dno) total_dep from emp left outer join department on ssn=mssn group by (ssn,salary);
```

Select * from trig_view;

Update: error at line1:

ORA-01732 data manipulation operation not legal on this view

Trigger created

1 row updated

Commit completed

1row created

Commit complete

DELETE:

ORA-01732: data manipulation operation not legal on this view

//update command

Update trig_view set sal=50000 where ssn=333445555;

//trigger update

```
Create or replace trigger trig instead of update on trig_view for each row
Begin
Update emp set sal=50000 where ssn=old.ssn;
End;
/
```

//update command:

Update trig_view set sal=50000 where ssn=333445555;

//temp_emp

Insert into emp (ssn,sal,dno) values(123456789,10000,5);

//delete command

Delete from trgi_view where ssn=1234567890;

//trigger update

Create or replace trigger trig instead of delete on trig_view fro each roe

Begin delete from emp where ssn:=old.ssn;

End;

1row deleted

Commit compete

Trigger dropped

View dropped

//delete command

Delete from trig_view where ssn=1234567890;

Drop trigger trig;

Drop view trig_view;

EXPERIMENT 11

AIM: Case Study using real world database applications

Example Case Study: Consider the following schema for a LibraryDatabase:

BOOK (Book_id, Title, Publisher_Name, Pub_Year)

BOOK_AUTHORS (Book_id, Author_Name)

PUBLISHER (Name, Address, Phone)

BOOK_COPIES (Book_id, Branch_id, No-of_Copies)

BOOK_LENDING (Book_id, Branch_id, Card_No, Date_Out, Due_Date)

LIBRARY_BRANCH (Branch_id, Branch_Name, Address)

Write SQL queries to

1. Retrieve details of all books in the library – id, title, name of publisher, authors, number of copies in each branch,etc.

2. Get the particulars of borrowers who have borrowed more than 3 books, but from Jan 2017 to Jun2017

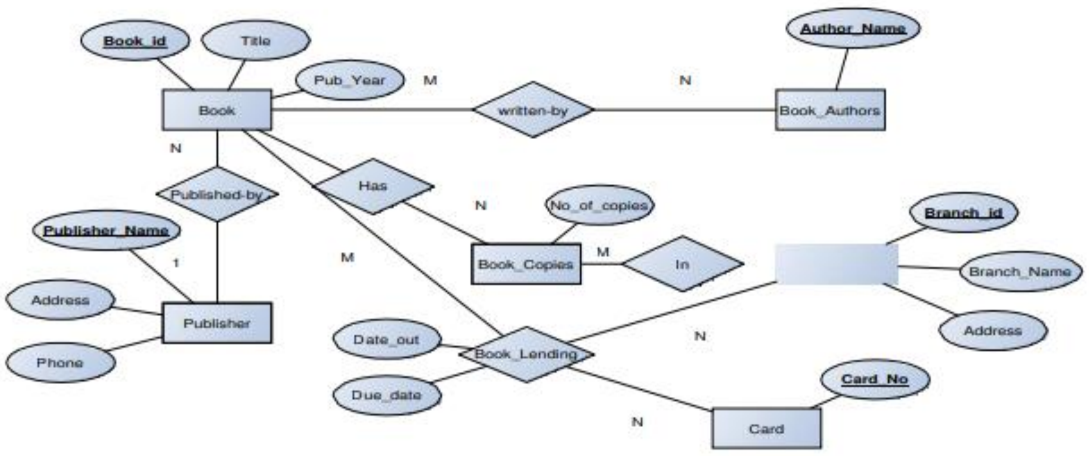
3. Delete a book in BOOK table. Update the contents of other tables to reflect this data manipulationoperation.

4. Partition the BOOK table based on year of publication. Demonstrate its working with a simplequery.

5. Create a view of all books and its number of copies that are currently available in the Library.

SOLUTION:

Entity-Relationship Diagram



Schema Diagram Book:

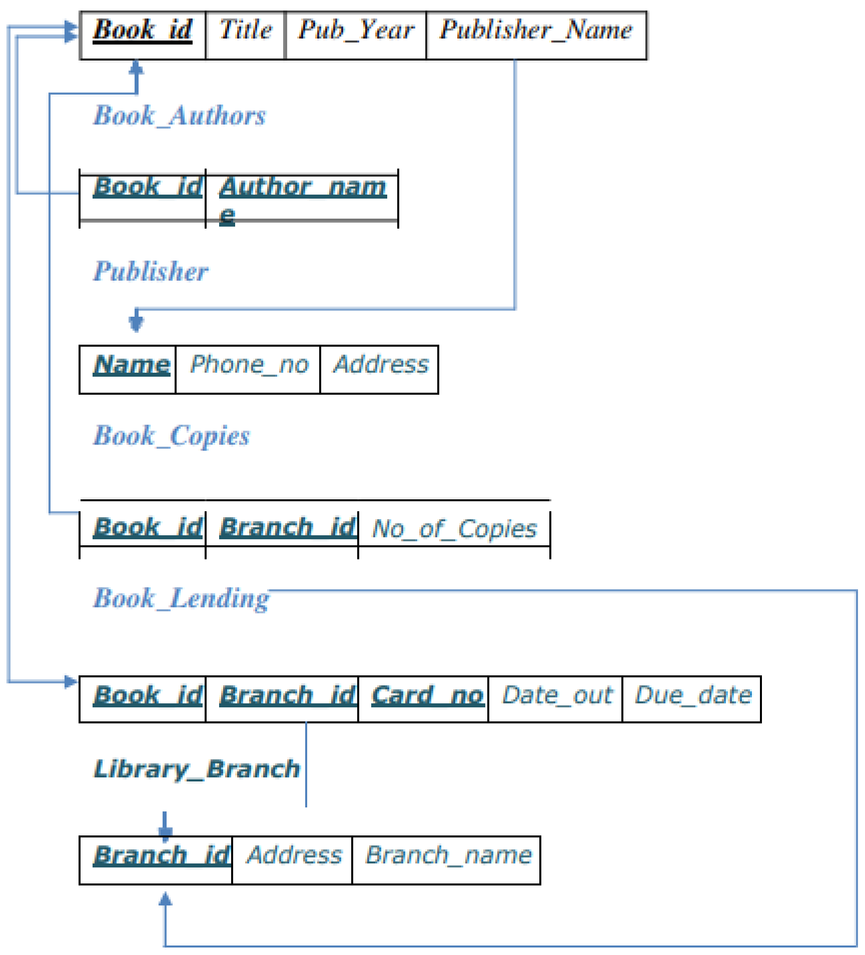


Table Creation:

CREATE TABLE PUBLISHER (NAME VARCHAR2 (20) PRIMARY KEY, PHONE INTEGER, ADDRESS VARCHAR2 (20));

CREATE TABLEBOOK (BOOK_ID INTEGER PRIMARYKEY, TITLE VARCHAR2(20), PUB_YEAR VARCHAR2 (20), PUBLISHER_NAME REFERENCES PUBLISHER (NAME));

CREATE TABLE BOOK_AUTHORS (AUTHOR_NAME VARCHAR2 (20), BOOK_ID REFERENCES BOOK (BOOK_ID), PRIMARY KEY (BOOK_ID, AUTHOR_NAME));

CREATE TABLE LIBRARY_BRANCH (BRANCH_ID INTEGER PRIMARY KEY, BRANCH_NAME VARCHAR2 (50), ADDRESS VARCHAR2 (50));

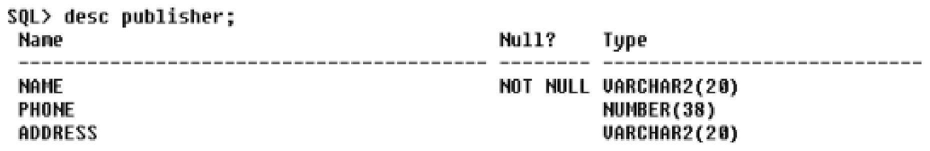
CREATE TABLE BOOK_COPIES (NO_OF_COPIES INTEGER, BOOK_ID REFERENCES BOOK (BOOK_ID), BRANCH_ID REFERENCES LIBRARY_BRANCH (BRANCH_ID), PRIMARY KEY (BOOK_ID, BRANCH_ID));

CREATE TABLE CARD (CARD_NO INTEGER PRIMARY KEY);

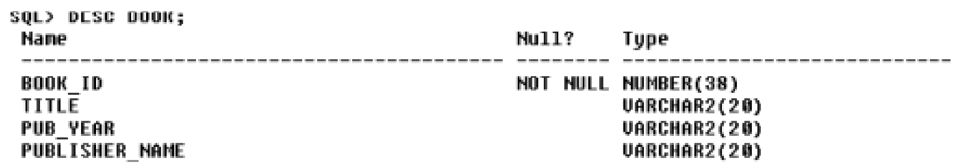
CREATE TABLE BOOK_LENDING (DATE_OUT DATE, DUE_DATE DATE, BOOK_ID REFERENCES BOOK (BOOK_ID), BRANCH_ID REFERENCES LIBRARY_BRANCH (BRANCH_ID), CARD_NO REFERENCES CARD (CARD_NO), PRIMARY KEY (BOOK_ID, BRANCH_ID, CARD_NO));

Table Descriptions

DESC PUBLISHER;

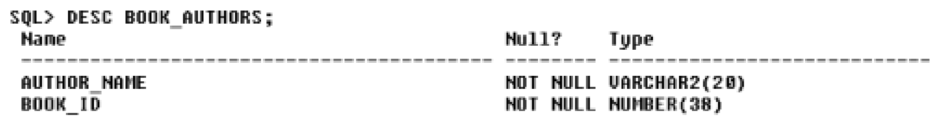


```
SQL> desc publisher;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 NAME                                      NOT NULL VARCHAR2(20)
 PHONE                                              NUMBER(38)
 ADDRESS                                            VARCHAR2(20)
```



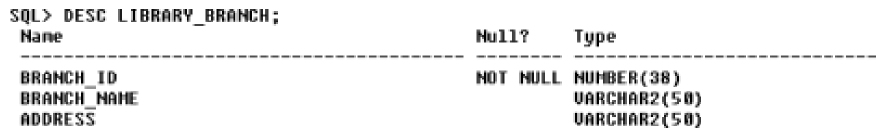
```
SQL> DESC BOOK;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 BOOK_ID                                   NOT NULL NUMBER(38)
 TITLE                                              VARCHAR2(20)
 PUB_YEAR                                           VARCHAR2(20)
 PUBLISHER_NAME                                     VARCHAR2(20)
```

DESC BOOK_AUTHORS;



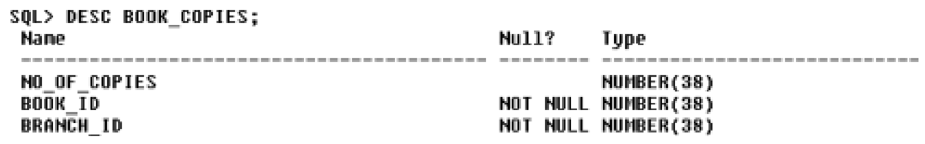
```
SQL> DESC BOOK_AUTHORS;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 AUTHOR_NAME                               NOT NULL VARCHAR2(20)
 BOOK_ID                                   NOT NULL NUMBER(38)
```

DESC LIBRARY_BRANCH;



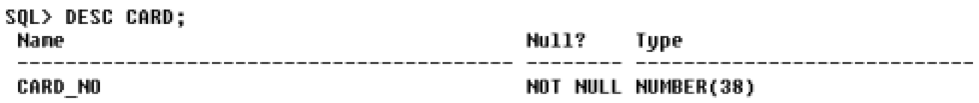
```
SQL> DESC LIBRARY_BRANCH;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 BRANCH_ID                                 NOT NULL NUMBER(38)
 BRANCH_NAME                                        VARCHAR2(50)
 ADDRESS                                            VARCHAR2(50)
```

DESC BOOK_COPIES;



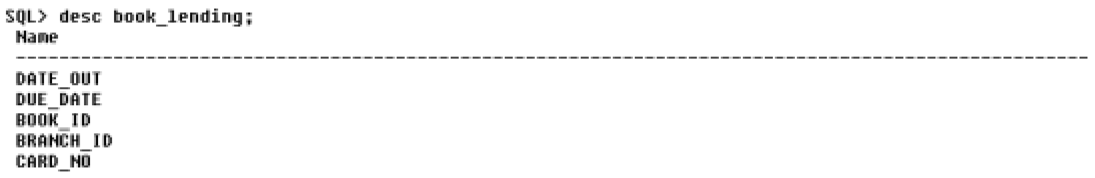
```
SQL> DESC BOOK_COPIES;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 NO_OF_COPIES                                       NUMBER(38)
 BOOK_ID                                   NOT NULL NUMBER(38)
 BRANCH_ID                                 NOT NULL NUMBER(38)
```

DESC CARD;



```
SQL> DESC CARD;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CARD_NO                                   NOT NULL NUMBER(38)
```

DESC BOOK_LENDING;



```
SQL> desc book_lending;
 Name
 -----------------------------------------------------------------------------------------------------
 DATE_OUT
 DUE_DATE
 BOOK_ID
 BRANCH_ID
 CARD_NO
```

Insertion of Values to Tables

INSERT INTO PUBLISHER VALUES (_MCGRAW-HILL‘, 9989076587, _BANGALORE‘);
INSERT INTO PUBLISHER VALUES (_PEARSON‘, 9889076565, _NEWDELHI‘);
INSERT INTO PUBLISHER VALUES (_RANDOM HOUSE‘, 7455679345, _HYDRABAD‘); INSERT
INTO PUBLISHER VALUES (_HACHETTE LIVRE‘, 8970862340, _CHENAI‘);
INSERTINTOPUBLISHERVALUES(_GRUPOPLANETA‘,7756120238,_BANGALORE‘);

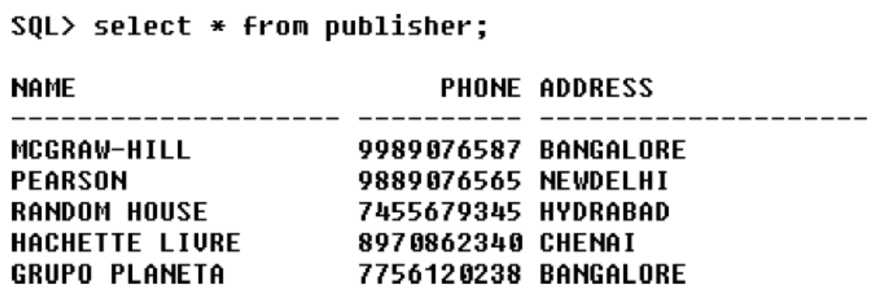
INSERT INTO BOOK VALUES (1,‘DBMS‘,‘JAN-2017‘, _MCGRAW-HILL‘); INSERT INTO
BOOK VALUES (2,‘ADBMS‘,‘JUN-2016‘, _MCGRAW-HILL‘); INSERT INTO BOOK
VALUES (3,‘CN‘,‘SEP-2016‘, _PEARSON‘);
INSERT INTO BOOK VALUES (4,‘CG‘,‘SEP-2015‘, _GRUPO PLANETA‘); INSERT
INTO BOOK VALUES (5,‘OS‘,‘MAY-2016‘, _PEARSON‘);

INSERT INTO BOOK_AUTHORS VALUES (‘NAVATHE‘, 1); INSERT INTO
BOOK_AUTHORS VALUES (‘NAVATHE‘, 2); INSERT INTO
BOOK_AUTHORS VALUES (‘TANENBAUM‘, 3); INSERT INTO
BOOK_AUTHORS VALUES (‘EDWARD ANGEL‘, 4); INSERT INTO
BOOK_AUTHORS VALUES (‘GALVIN‘, 5);

INSERT INTO LIBRARY_BRANCH VALUES (10,‘RR NAGAR‘,‘BANGALORE‘); INSERT
INTO LIBRARY_BRANCH VALUES (11,‘RNSIT‘,‘BANGALORE‘);
INSERT INTO LIBRARY_BRANCH VALUES (12,‘RAJAJI NAGAR‘, ‘BANGALORE‘); INSERT INTO
LIBRARY_BRANCH VALUES (13,‘NITTE‘,‘MANGALORE‘);
INSERT INTO LIBRARY_BRANCH VALUES (14,‘MANIPAL‘,‘UDUPI‘);

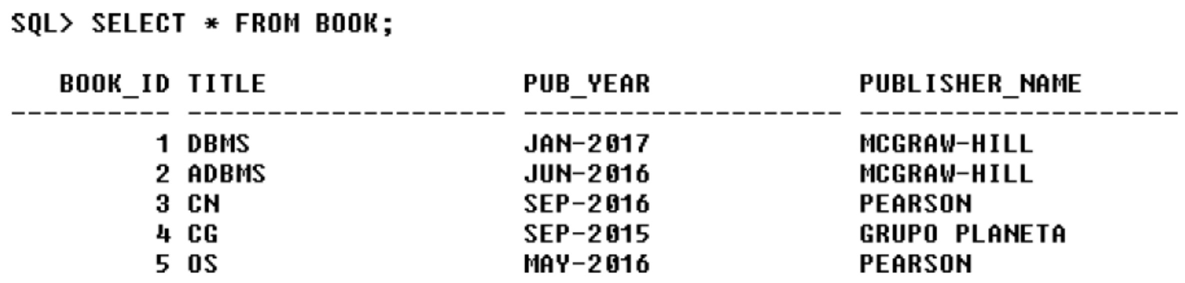
```
INSERT INTO BOOK_COPIES VALUES (10, 1, 10);
INSERT INTO BOOK_COPIES VALUES (5, 1,11);
INSERT INTO BOOK_COPIES VALUES (2, 2,12);
INSERT INTO BOOK_COPIES VALUES (5, 2,13);
INSERT INTO BOOK_COPIES VALUES (7, 3,14);
INSERT INTO BOOK_COPIES VALUES (1, 5,10);
INSERT INTO BOOK_COPIES VALUES (3, 4,11);

INSERT  INTO  CARD  VALUES  (100);
INSERT  INTO  CARD  VALUES  (101);
INSERT  INTO  CARD  VALUES  (102);
INSERT  INTO  CARD  VALUES  (103);
INSERT INTO CARD VALUES (104);
```

```
INSERT INTO BOOK_LENDING VALUES (‘01-JAN-17`,`01-JUN-17`, 1, 10, 101);
INSERT INTO BOOK_LENDING VALUES (‘11-JAN-17`,`11-MAR-17`, 3, 14, 101);
INSERT INTO BOOK_LENDING VALUES (‘21-FEB-17`,`21-APR-17`, 2, 13, 101);
INSERT INTO BOOK_LENDING VALUES (‘15-MAR-17`,`15-JUL-17`, 4, 11, 101);
INSERT INTO BOOK_LENDING VALUES (_12-APR-17`,`12-MAY-17`, 1, 11, 104);
SELECT * FROM PUBLISHER;
```



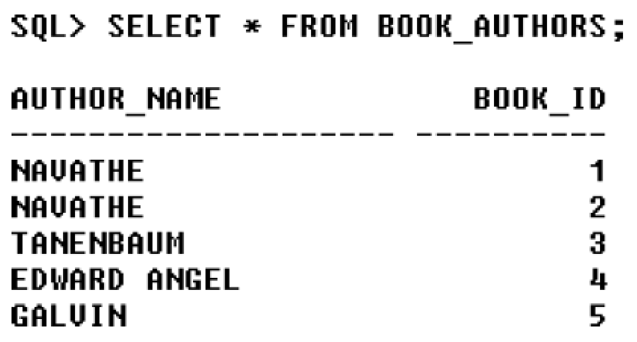
```
SQL> select * from publisher;

NAME                      PHONE ADDRESS
-------------------- ---------- --------------------
MCGRAW-HILL          9989076587 BANGALORE
PEARSON              9889076565 NEWDELHI
RANDOM HOUSE         7455679345 HYDRABAD
HACHETTE LIVRE       8970862340 CHENAI
GRUPO PLANETA        7756120238 BANGALORE
```

```
SELECT * FROM BOOK;
```



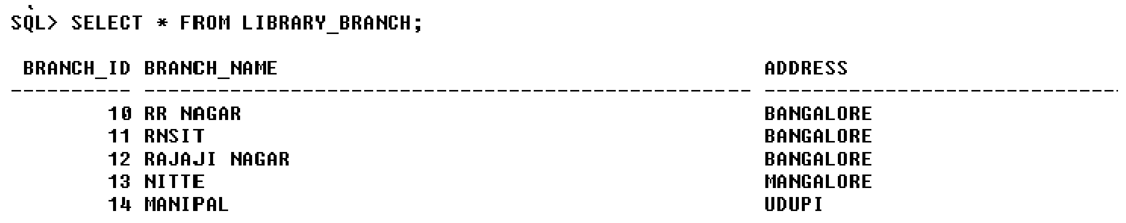
```
SQL> SELECT * FROM BOOK;

   BOOK_ID TITLE                PUB_YEAR             PUBLISHER_NAME
---------- -------------------- -------------------- --------------------
         1 DBMS                 JAN-2017             MCGRAW-HILL
         2 ADBMS                JUN-2016             MCGRAW-HILL
         3 CN                   SEP-2016             PEARSON
         4 CG                   SEP-2015             GRUPO PLANETA
         5 OS                   MAY-2016             PEARSON
```

SELECT * FROM BOOK_AUTHORS;



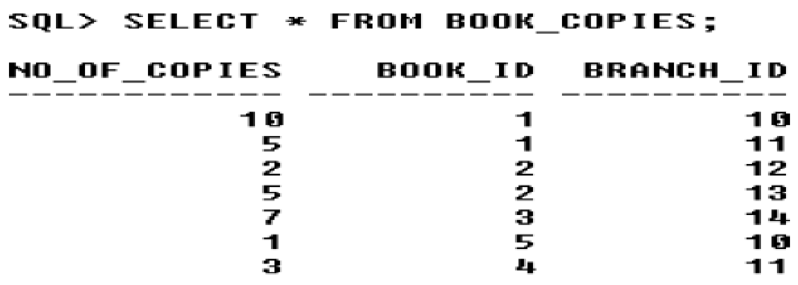
```
SQL> SELECT * FROM BOOK_AUTHORS;

AUTHOR_NAME                BOOK_ID
-------------------- ----------
NAVATHE                          1
NAVATHE                          2
TANENBAUM                        3
EDWARD ANGEL                     4
GALVIN                           5
```

SELECT * FROM LIBRARY_BRANCH;



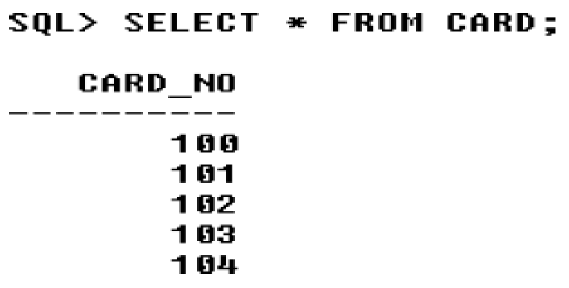
```
SQL> SELECT * FROM LIBRARY_BRANCH;

 BRANCH_ID BRANCH_NAME                                        ADDRESS
---------- -------------------------------------------------- ------------------------------
        10 RR NAGAR                                           BANGALORE
        11 RNSIT                                              BANGALORE
        12 RAJAJI NAGAR                                       BANGALORE
        13 NITTE                                              MANGALORE
        14 MANIPAL                                            UDUPI
```

SELECT * FROM BOOK_COPIES;



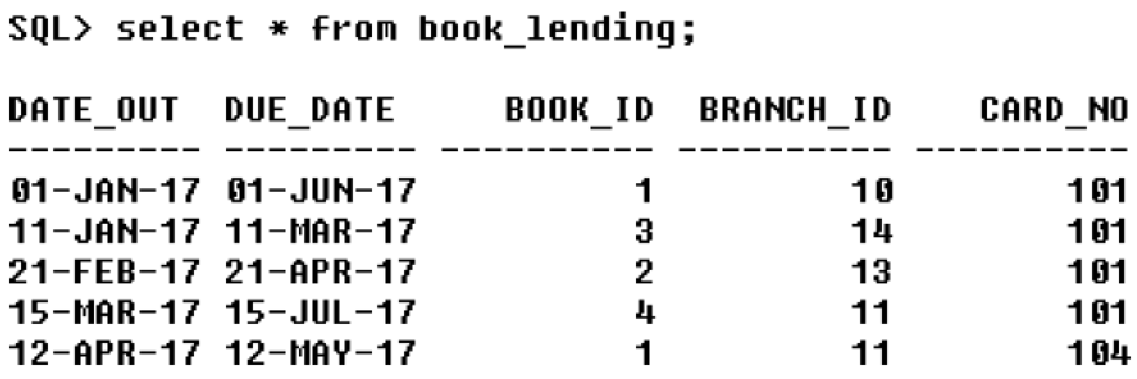
```
SQL> SELECT * FROM BOOK_COPIES;

NO_OF_COPIES    BOOK_ID  BRANCH_ID
------------ ---------- ----------
          10          1         10
           5          1         11
           2          2         12
           5          2         13
           7          3         14
           1          5         10
           3          4         11
```

SELECT * FROM CARD;



```
SQL> SELECT * FROM CARD;

   CARD_NO
----------
       100
       101
       102
       103
       104
```

SELECT * FROM BOOK_LENDING;

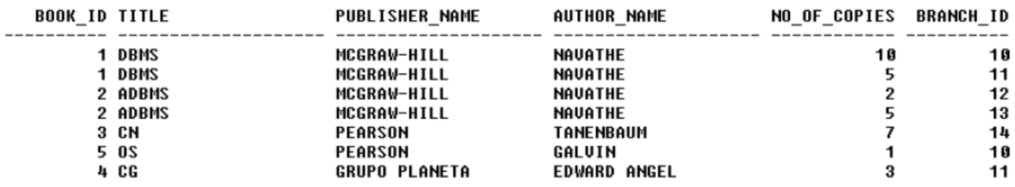


```
SQL> select * from book_lending;

DATE_OUT  DUE_DATE     BOOK_ID  BRANCH_ID    CARD_NO
--------- --------- ---------- ---------- ----------
01-JAN-17 01-JUN-17          1         10        101
11-JAN-17 11-MAR-17          3         14        101
21-FEB-17 21-APR-17          2         13        101
15-MAR-17 15-JUL-17          4         11        101
12-APR-17 12-MAY-17          1         11        104
```

Queries:

1. **Retrieve details of all books in the library – id, title, name of publisher, authors, number of copies in each branch,etc.**

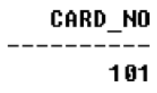
   SELECT B.BOOK_ID, B.TITLE, B.PUBLISHER_NAME, A.AUTHOR_NAME, C.NO_OF_COPIES,L.BRANCH_ID
   FROM BOOK B, BOOK_AUTHORS A, BOOK_COPIES C, LIBRARY_BRANCHL
   WHEREB.BOOK_ID=A.BOOK_ID
   AND B.BOOK_ID=C.BOOK_ID
   AND L.BRANCH_ID=C.BRANCH_ID;



```
   BOOK_ID TITLE                PUBLISHER_NAME       AUTHOR_NAME          NO_OF_COPIES  BRANCH_ID
---------- -------------------- -------------------- -------------------- ------------ ----------
         1 DBMS                 MCGRAW-HILL          NAVATHE                        10         10
         1 DBMS                 MCGRAW-HILL          NAVATHE                         5         11
         2 ADBMS                MCGRAW-HILL          NAVATHE                         2         12
         2 ADBMS                MCGRAW-HILL          NAVATHE                         5         13
         3 CN                   PEARSON              TANENBAUM                       7         14
         5 OS                   PEARSON              GALVIN                          1         10
         4 CG                   GRUPO PLANETA        EDWARD ANGEL                    3         11
```

1. **Get the particulars of borrowers who have borrowed more than 3 books, but from Jan 2017 to Jun2017.**

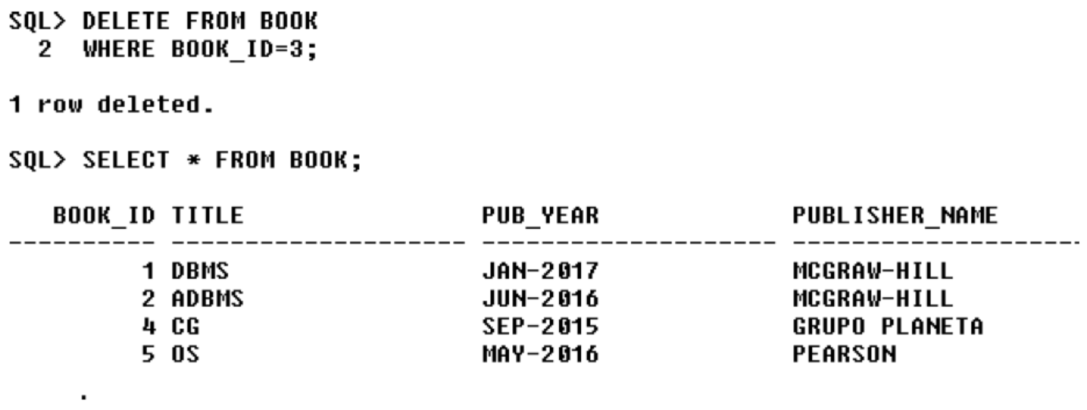
   SELECT CARD_NO FROM
   BOOK_LENDING
   WHERE DATE_OUT BETWEEN '01-JAN-2017' AND '01-JUL-2017'
   GROUP BY CARD_NO
   HAVING COUNT (*)>3;



```
   CARD_NO
----------
       101
```

2. **Delete a book in BOOK table. Update the contents of other tables to reflect this data manipulationoperation.**

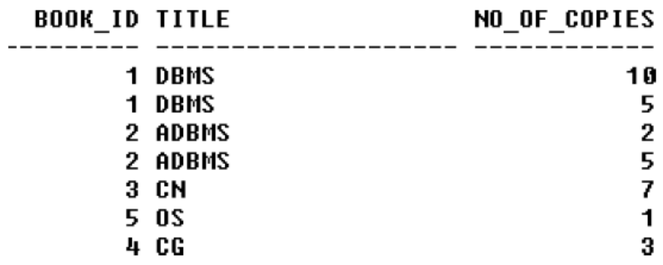
```
DELETE FROM BOOK
WHERE BOOK_ID=3;
```



```
SQL> DELETE FROM BOOK
  2  WHERE BOOK_ID=3;

1 row deleted.

SQL> SELECT * FROM BOOK;

   BOOK_ID TITLE                PUB_YEAR             PUBLISHER_NAME
---------- -------------------- -------------------- --------------------
         1 DBMS                 JAN-2017             MCGRAW-HILL
         2 ADBMS                JUN-2016             MCGRAW-HILL
         4 CG                   SEP-2015             GRUPO PLANETA
         5 OS                   MAY-2016             PEARSON
```

4. **Create a view of all books and its number of copies that are currently available in the Library.**

```
CREATE VIEW V_BOOKS AS
SELECT B.BOOK_ID, B.TITLE, C.NO_OF_COPIES
FROM BOOK B, BOOK_COPIES C, LIBRARY_BRANCH L
WHERE B.BOOK_ID=C.BOOK_ID
AND C.BRANCH_ID=L.BRANCH_ID;
```



```
  BOOK_ID TITLE                NO_OF_COPIES
--------- -------------------- ------------
        1 DBMS                           10
        1 DBMS                            5
        2 ADBMS                           2
        2 ADBMS                           5
        3 CN                              7
        5 OS                              1
        4 CG                              3
```

**********THE END******************