GURU NANAK INSTITUTIONS TECHNICAL CAMPUS (AUTONOMOUS)

School of Engineering & Technology

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

DATA BASE MANAGEMENT SYSTEMS LAB

Lab Manual

[Course Code: 22PC0CS12]

For the Academic year 2023-24 II B.Tech. II Semester

Guru Nanak Institutions Technical Campus (Autonomous)
Ibrahimpatnam, R R District – 501 506 (T. S.)

LAB MANUAL FOR THE ACADEMIC YEAR 2023-24

Name of the Lab	:	**Data Base Management Systems Lab**
Lab Course Code	:	**22PC0CS12**
Year & Semester	:	**II Year II Sem**
Branch	:	**CSE**
No. of Hours	:	**2 Practical Hours per Week**
No. of Credits	:	**1.5**
Document No.	:	**GNITC/CSE/DBMS LAB/R-22 (Autonomous)**
Date of Revision	:	**22th March 2024**
Date of Issue	:	
Prepared By	:	**Er. Bint ul Huda**
Verified by	:	**Dr. M.V Narayana (HOD CSE Special Batch)**

INDEX

LAB OBJECTIVE

- Introduce ER data model, database design and normalization
- Learn SQL basics for data definition and data manipulation

LAB OUTCOMES

Upon successful completion of this Lab, students will be able to:

CO1: Design database schema for a given application and apply normalization.
CO2: Acquire skills in using SQL Commands for data Definition and data manipulation.
CO3: Develop solutions for database applications using procedures, cursors and triggers.

Introduction about Data Base Management Systems Lab

- There are 65 systems (Acer) installed in this Lab. Their configurations are as follows:
- Hardware / Software's installed: Intel® CORE™ i3-3240 CPU@3.40GHZ RAM:4GB / C, C++ Compiler
- Systems are provided for students in the 1:1 ratio.
- Systems are assigned numbers and same system is allotted for students when they do the lab.
- All systems are configured in DUAL BOOT mode i.e., Students can boot from Windows XP or Linux as per their lab requirement. This is very useful for students because they are familiar with different Operating Systems so that they can execute their programs in different programming environments.
- Each student has a separate login for database access MySQL client version is installed in all systems. On the server, account for each student has been created. This is very useful because students can save their work (scenarios', PL / SQL programs, data related projects, etc) in their own accounts. Each student work is safe and secure from other students

STANDARD OPERATING PROCEDURE – SOP

a) Explanation on today's experiment by the concerned faculty using PPT covering the following aspects:

1) Name of the experiment
2) Aim
3) Software/Hardware requirements
4) Theory related to the aim
5) Commands with suitable Options
6) Creating Database
7) Altering database
8) Writing of DDL and DML commands by the students
9) Querying and executing of the SQL queries
10) Dropping database 120 mins.

b) Writing of Data Base Management Systems Experiments by the students

c) Execution of the Experiments

Writing of the experiment in the Observation Book

The students will write the today's experiment in the Observation book as per the following format:

a) Name of the experiment
b) Aim
c) Software/Hardware required
d) Theory
e) Commands with suitable Options
f) Creating Database
    i) Altering database
    ii) Querying
    iii) Dropping database
g) Results for different Queries
h) Viva-Voce Questions and Answers
i) Errors observed (if any) during compilation/execution
j) Signature of the Faculty

Guide Lines to Students in Lab

Disciplinary to be maintained by the students in the Lab:

- ➢ Students are required to carry their lab observation book and record book with completed experiments while entering the lab
- ➢ Students must use the equipment with care
- ➢ Students are not allowed to use their cell phones/pen drives/CDs in labs
- ➢ Students need to maintain proper dress code along with ID Card
- ➢ Students are supposed to occupy the computers allotted to them and are not supposed to talk or make noise in the lab
- ➢ Students, after completion of each experiment they need to be updated in observation notes and same to be updated in the record
- ➢ Lab records need to be submitted after completion of experiment and get it corrected with the concerned lab faculty
- ➢ If a student is absent for any lab, they need to be completed the same experiment in the free time before attending next lab.

Steps to perform experiments in the lab by the student

Step1: Students have to write the date, aim, and Software & Hardware requirements for that Experiment in the observation book.

Step2: Students have to listen and understand the experiment explained by the faculty and note down the important points in the observation book.

Step3: Students need to write procedure/algorithm in the observation book.

Step4: Analyze and Develop/implement the logic of the program by the student in respect platform

Step5: After approval of logic of the experiment by the faculty then the experiment has to executed on the system.

Step6: After successful execution the results are to be shown to the faculty and noted the same in the observation book.

Step7: Students need to attend the Viva-Voce on that experiment and write the same in the observation book.

Step8: Update the completed experiment in the record and submit to the concerned faculty in-charge.

Instructions to maintain the record

- Before start of the first lab, students have to buy the record and bring the record to the lab.
- Regularly (Weekly) update the record after completion of the experiment and get it corrected with concerned lab in-charge for continuous evaluation.
- In case the record is lost inform the same day to the faculty in charge and get the new record within 2 days the record has to be submitted and get it corrected by the faculty.
- If record is not submitted in time or record is not written properly, the evaluation marks will be deducted.

Awarding the marks for day-to-day evaluation

Total marks for day-to-day evaluation are 10 Marks as per GNITC. The distribution is as follows:

Record	3 Marks
Program Write up	5 Marks
Result and Viva-Voce	2 Marks

Allocation of Marks for Lab Internal

Total marks for lab internal are 40 Marks as per GNITC.

The distribution of 40 Marks is as follows:

Average of day-to-day evaluation marks:	10 Marks
Internal Lab Exam	: 10 Marks
Viva-Voce	: 10 Marks
Lab Project	: 10 Marks.

Allocation of Marks for Lab External

Total marks for lab external are 60 Marks as per GNITC. The distribution of 60 Marks is as follows:

Procedure	20 Marks
Observation / Programs / Calculations	15 Marks
Results & Inference / Output	15 Marks
Viva-Voce	10 Marks

LIST OF EXPERIMENTS

S. No.	Name of the Program
1	Concept design with E-R Model
2	ER To Relational Model
3	Normalization
4	Practicing DDL Commands
5	Practicing DML Commands
6	Querying using ANY, ALL, IN, EXISTS, NOT EXISTS, UNION, INTERSECT
7	Querying using aggregate functions, Group by, Having and Creation and dropping Views.
8	Triggers (Creation of insert trigger, Delete Trigger, Update trigger)
9	Procedures
10	Cursors
	ADDITIONAL EXPERIMENTS
A1	Design and implement queries on Tables (Emp, Dept)
A2	Design and implement queries on Library Data base Management

EXPERIMENT 1: CONCEPT DESIGN WITH E-R MODEL

AIM: Analyzing a system to implement the concepts of E-R Model through Bus Management System.

CONCEPTS OF ER MODEL:

- Entities and its types
- Cardinalities for each relationship.
- Identify strong entities and weak entities (if any).
- Indicate the type of relationships (total/partial).
- Try to incorporate generalization, aggregation, specialization etc. wherever required.

Note: *The student is required to submit a document by drawing the* E-R' Diagram *to the* lab teacher.

RECOMMENDED HARDWARE / SOFTWARE REQUIREMENTS:

- Hardware Requirements: Intel Based desktop PC with minimum of 166 MHZ or faster processor with at least 1GB RAM and 500 MB free disk space.
- MySQL 5.6.1

PRE-REQUISITES: Student must know the concepts of ER MODEL.

PROCEDURE:

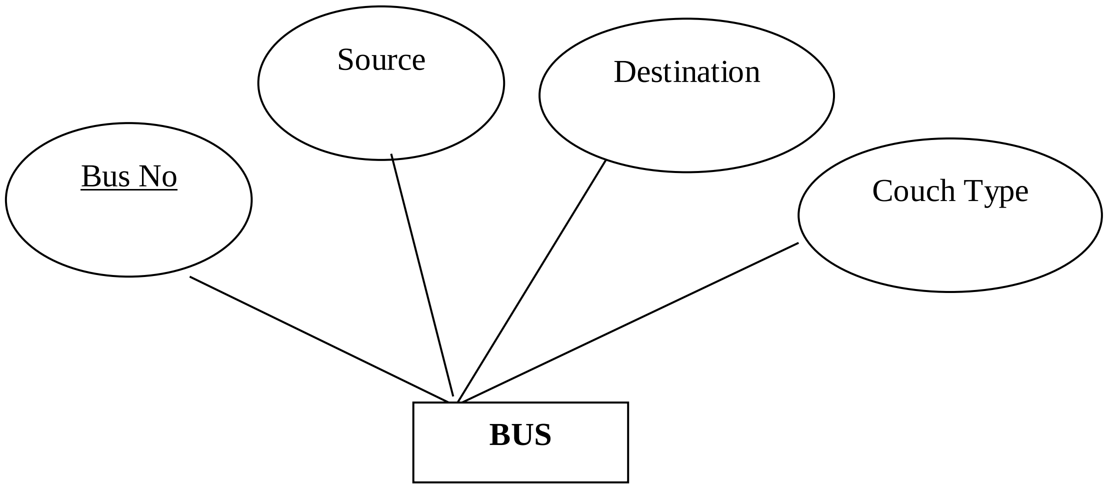
After identifying the system, students will

1. Identify the entities
2. Identify the attributes for the taken entities.
3. Establish the relationship between the entities.
4. Identify the cardinalities and types of relationships for the mentioned relationships.
5. Identify the class hierarchy (Generalization/ Specialization) and aggregation if any.
6. Show all the above representations in one diagram to generate an E-R Diagram.

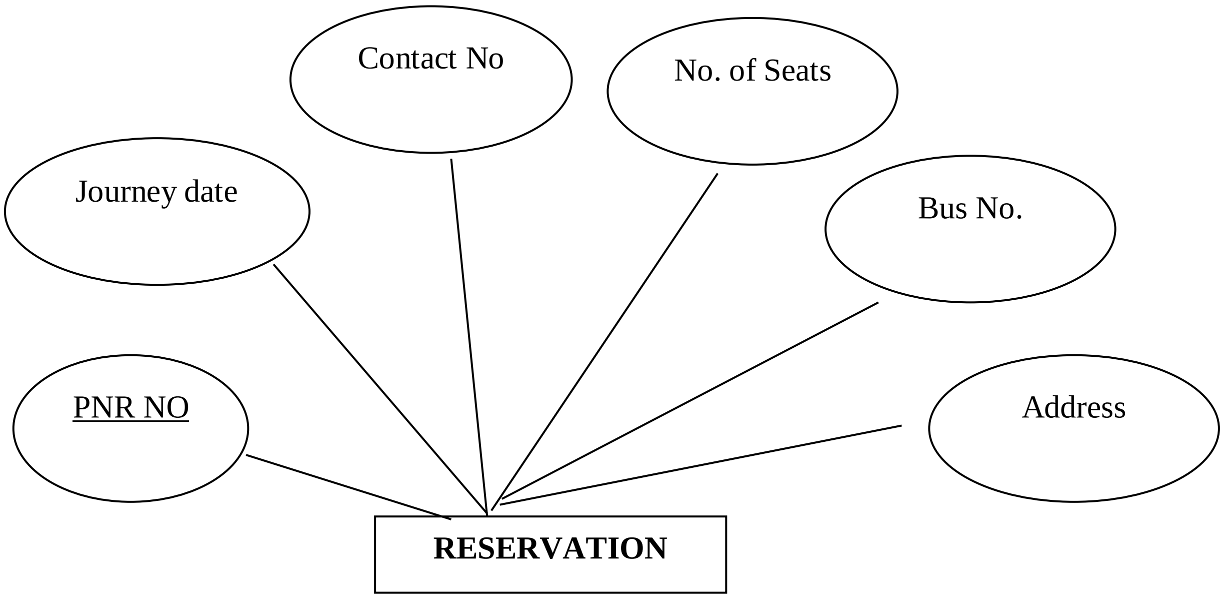
The following are the entities

i. Bus
ii. Reservation
iii. Ticket
iv. Passenger
v. Cancellation

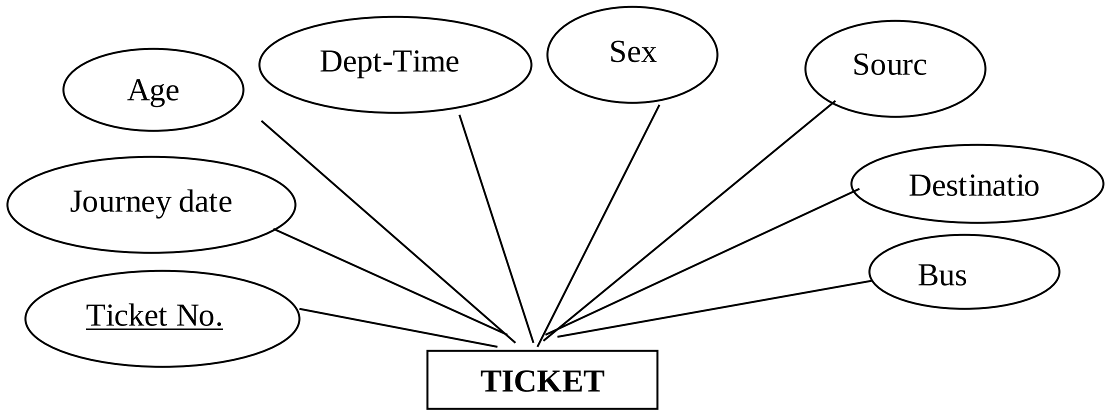
Bus: (Entity)



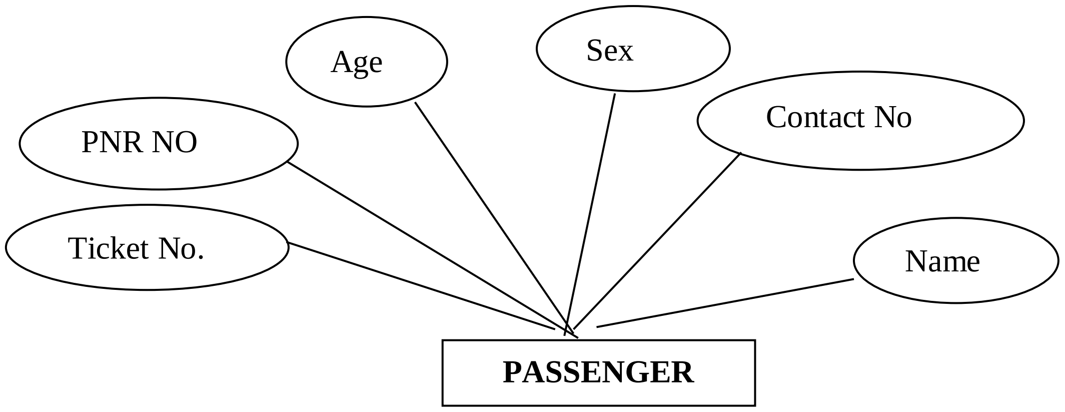
Reservation: (Entity)



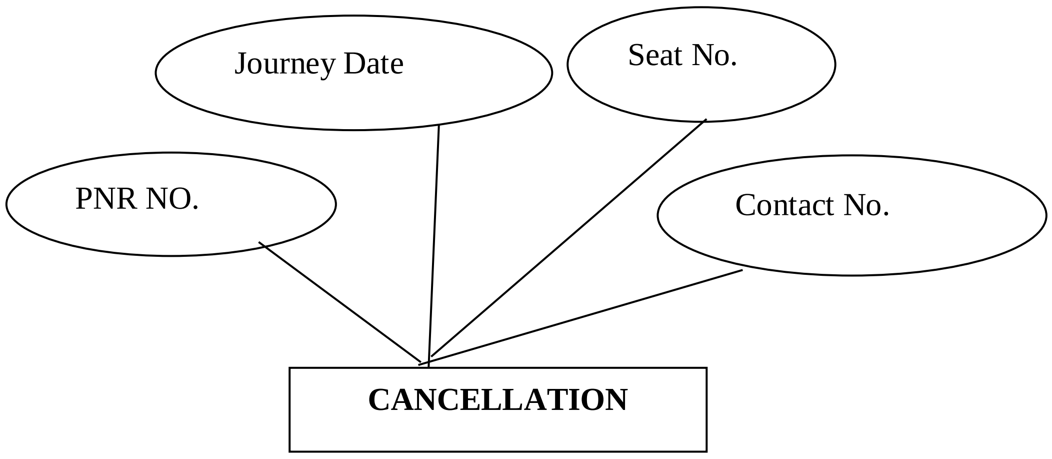
Ticket: (Entity)



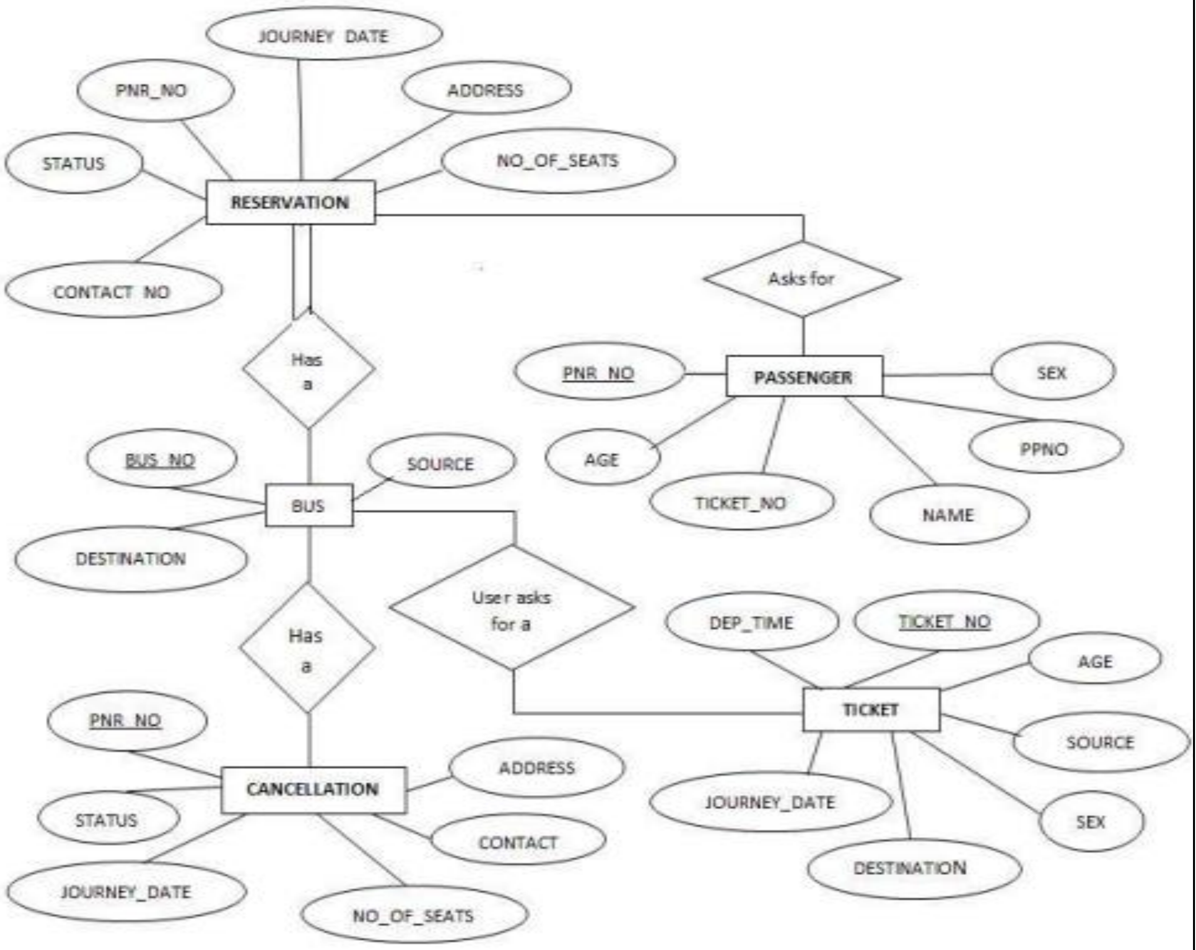
Passenger: (Entity)



Cancellation: (Entity)



Concept Design With E-R Model:



RESULT: Student gains the ability to describe the data requirements for a new information system and implement them through an E-R Model.

VIVA- VOCE

1. Distinguish database with database management system?
2. Define entity and how it is different from an attribute?
3. Explain Participation Constraints?
4. Differentiate generalization and specialization?
5. Describe degree of cardinality?

EXPERIMENT 2: ER TO RELATIONAL MODEL

AIM: To Convert the concepts of ER model (Graphical Notation) into Relational Model (Table) database.

Hints:

- Represent attributes as columns in tables or as tables based on the requirement.
- Different types of attributes (Composite, Multi-valued, and Derived) have different way of representation.
- Represent all the entities (Strong, Weak) in tabular fashion.
- Represent relationships in a tabular fashion. There are different ways of representing, relationships as tables based on the cardinality.

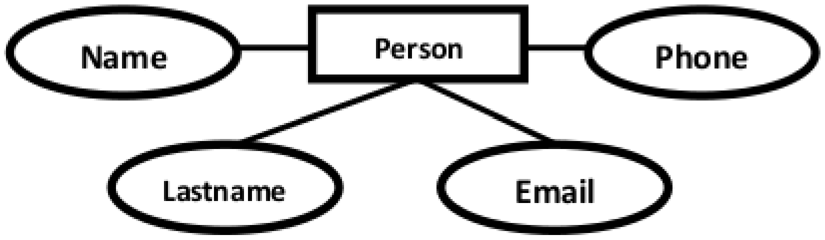
RECOMMENDED HARDWARE/ SOFTWARE REQUIREMENTS:

- Hardware Requirements: Intel Based desktop PC with minimum of 166 MHZ or faster processor with at least 1GB RAM and 500 MB free disk space.
- MySQL 5.6.1

PRE-REQUISITES: Student must know about the conversion of ER to Relational Model.

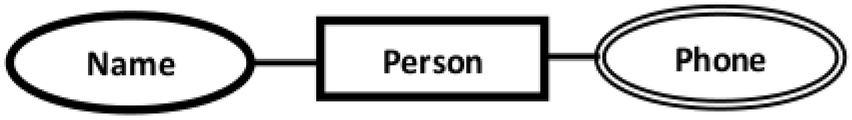
Conversion:

Entities and Simple Attributes:



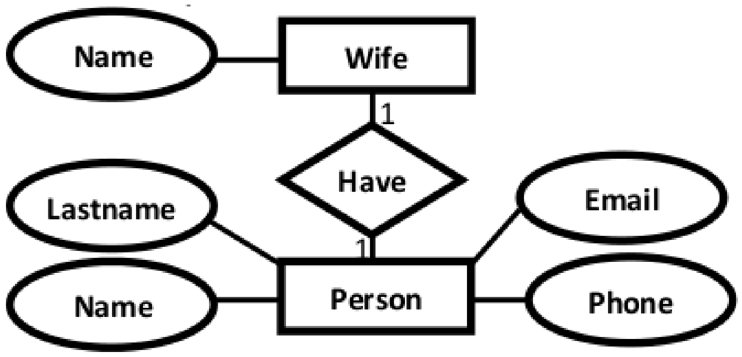
Persons (personid , name, lastname, email)

Multi-Valued Attributes:



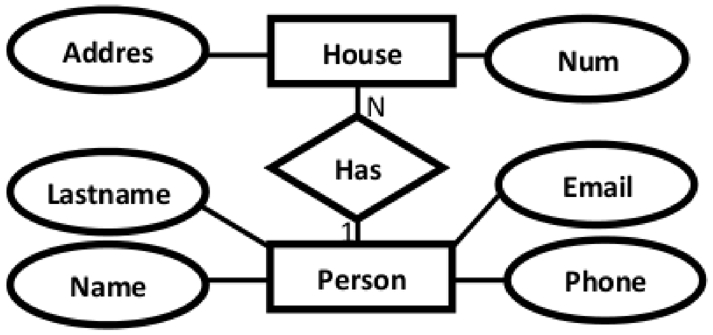
Persons (personid, name, lastname, email), Phones (phoneid , personid, phone)

1:1 Relationships



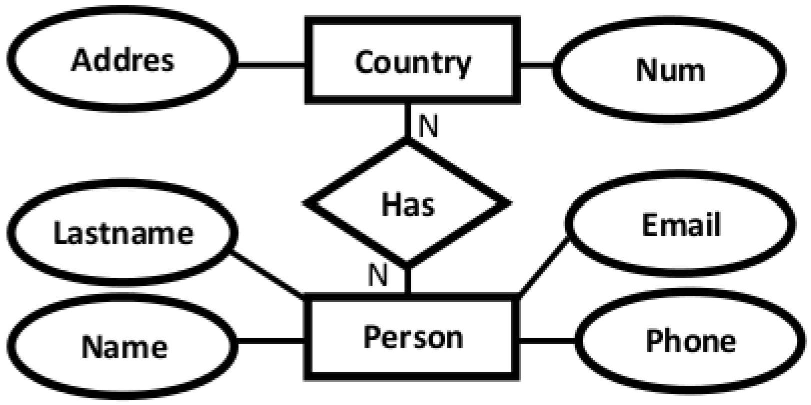
Persons(personid , name, lastname, email , wifeid), Wife (wifeid , name)
Persons(personid , name, lastname, email), Wife (wifeid , name , personid)

1:N Relationships

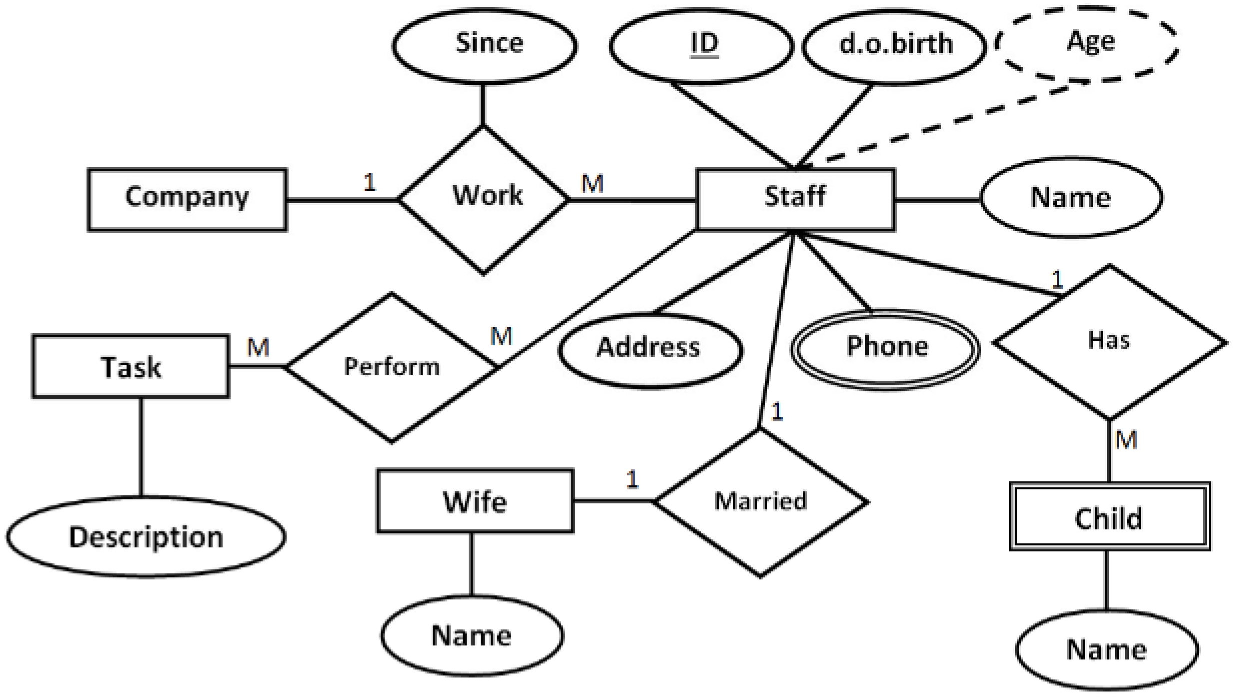


Persons(personid , name, lastname, email), House (houseid , num , address, personid)

N:N Relationships



Persons(personid , name, lastname, email)
Countries (countryid , name, code) , HasRelat (hasrelatid , personid , countryid)



The relational schema for the ER Diagram is given below as:
Company(CompanyID , name , address)
Staff(StaffID , dob , address , WifeID)
Child(ChildID , name , StaffID)
Wife (WifeID , name)
Phone(PhoneID , phoneNumber , StaffID)
Task (TaskID , description)
Work(WorkID , CompanyID , StaffID , since)
Perform(PerformID , StaffID , TaskID)

RESULT: The student gains the knowledge on converting the E-R Model concepts in to a relational database.

VIVA-VOCE:

1. Show the representation of data in relational model?
2. State the use of CASCADE constraint?
3. Explain Query optimization?
4. Differentiate relation schema from relation?
5. Define view and how it is related to data independence?

EXPERIMENT 3: NORMALIZATION

AIM: Implementing the normalization techniques up to 3NF to avoid data redundancy in the database design.

RECOMMENDED HARDWARE/ SOFTWARE REQUIREMENTS:

- Hardware Requirements: Intel Based desktop PC with minimum of 166 MHZ or faster processor with at least 1GB RAM and 500 MB free disk space.
- MySQL 5.6.1

PRE-REQUISITES: Student must know about different types of normal forms.

NORMALIZATION: Database normalization is a technique for designing relational database tables to minimize duplication of information and, in so doing, to safeguard the database against certain types of logical or structural problems, namely data anomalies. A table that is sufficiently normalized is less vulnerable to problems of anomalies.

Normalization is a process of converting a relation to be standard form by decomposition a larger relation into smaller efficient relation that depicts a good database design.

1NF: A Relation scheme is said to be in 1NF if the attribute values in the relation are atomic. Muti –valued attributes are not permitted.

2NF: A Relation scheme is said to be in 2NF, if every non-key attribute is fully functionally dependent on primary Key.

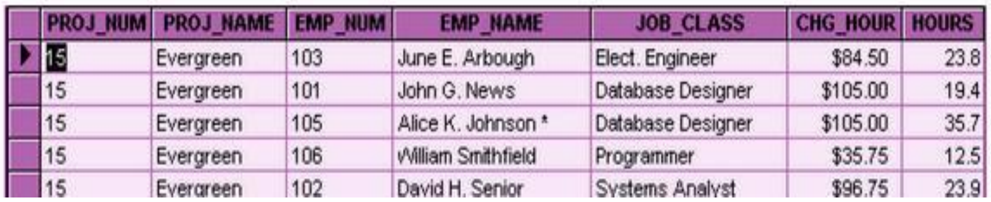
3NF: A Relation scheme is said to be in 3NF, if it does not have transitivity dependencies. A Relation is said to be 3NF if every determinant is a key for each & every functional dependency.

Example: Let us consider the following case of a Construction Company

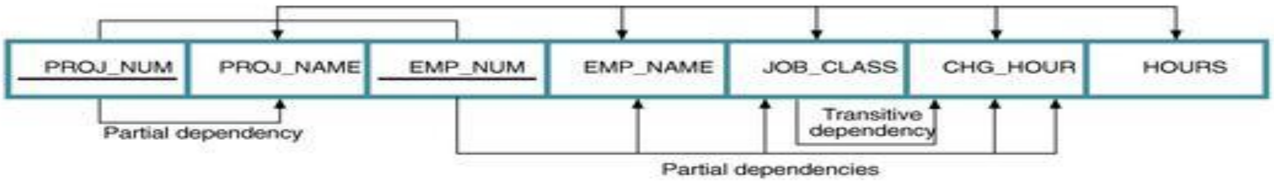
- Building project -- Project number, Name, Employees assigned to the project.
- Employee -- Employee number, Name, Job classification
- The company charges its clients by billing the hours spent on each project. The hourly billing rate is dependent on the employee’s position.
- Periodically, a report is generated.

Conversion to First Normal Form:

A relational table must not contain repeating groups.

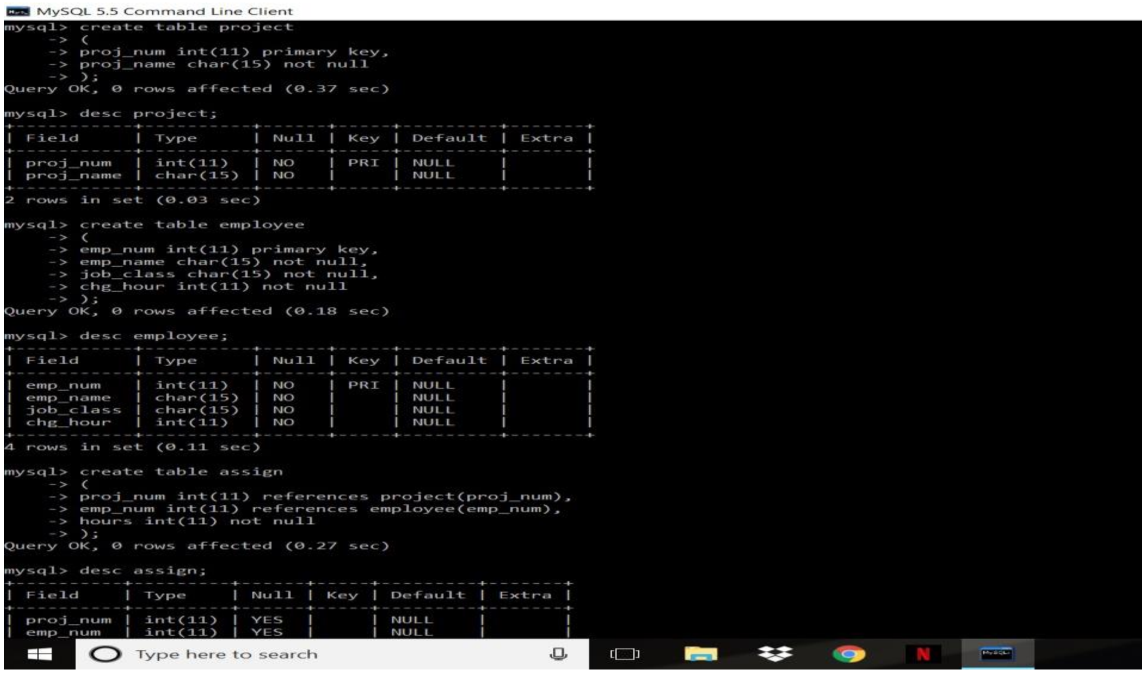


PROJ_NUM	PROJ_NAME	EMP_NUM	EMP_NAME	JOB_CLASS	CHG_HOUR	HOURS
15	Evergreen	103	June E. Arbough	Elect. Engineer	$84.50	23.8
15	Evergreen	101	John G. News	Database Designer	$105.00	19.4
15	Evergreen	105	Alice K. Johnson *	Database Designer	$105.00	35.7
15	Evergreen	106	William Smithfield	Programmer	$35.75	12.5
15	Evergreen	102	David H. Senior	Systems Analyst	$96.75	23.9

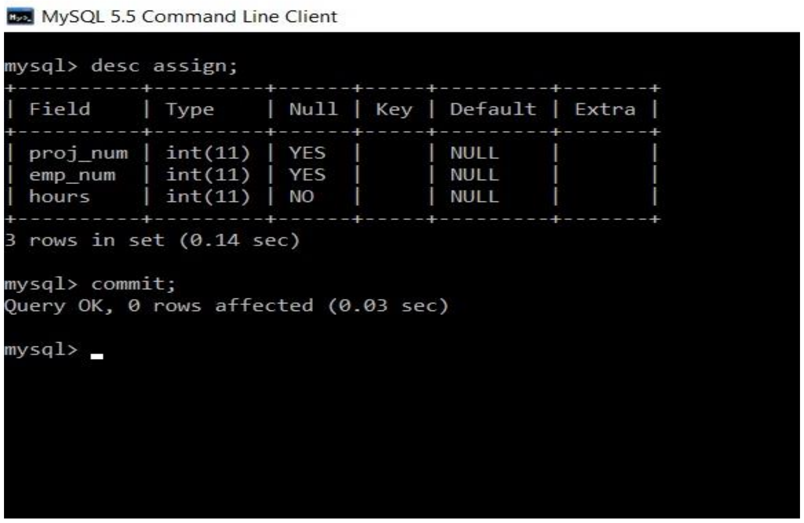


Conversion of Second Normal Form:

- It is in 1NF and
- It includes no partial dependencies; that is, no attribute is dependent on only a portion of the primary key
  PROJECT (PROJ_NUM, PROJ_NAME),
  EMPLOYEE (EMP_NUM, EMP_NAME, JOB_CLASS, CHG_HOUR),
  ASSIGN (PROJ_NUM, EMP_NUM, HOURS)



```
mysql> create table project
    -> (
    -> proj_num int(11) primary key,
    -> proj_name char(15) not null
    -> );
Query OK, 0 rows affected (0.37 sec)

mysql> desc project;
+-----------+----------+------+-----+---------+-------+
| Field     | Type     | Null | Key | Default | Extra |
+-----------+----------+------+-----+---------+-------+
| proj_num  | int(11)  | NO   | PRI | NULL    |       |
| proj_name | char(15) | NO   |     | NULL    |       |
+-----------+----------+------+-----+---------+-------+
2 rows in set (0.03 sec)

mysql> create table employee
    -> (
    -> emp_num int(11) primary key,
    -> emp_name char(15) not null,
    -> job_class char(15) not null,
    -> chg_hour int(11) not null
    -> );
Query OK, 0 rows affected (0.18 sec)

mysql> desc employee;
+-----------+----------+------+-----+---------+-------+
| Field     | Type     | Null | Key | Default | Extra |
+-----------+----------+------+-----+---------+-------+
| emp_num   | int(11)  | NO   | PRI | NULL    |       |
| emp_name  | char(15) | NO   |     | NULL    |       |
| job_class | char(15) | NO   |     | NULL    |       |
| chg_hour  | int(11)  | NO   |     | NULL    |       |
+-----------+----------+------+-----+---------+-------+
4 rows in set (0.11 sec)

mysql> create table assign
    -> (
    -> proj_num int(11) references project(proj_num),
    -> emp_num int(11) references employee(emp_num),
    -> hours int(11) not null
    -> );
Query OK, 0 rows affected (0.27 sec)

mysql> desc assign;
+----------+---------+------+-----+---------+-------+
| Field    | Type    | Null | Key | Default | Extra |
+----------+---------+------+-----+---------+-------+
| proj_num | int(11) | YES  |     | NULL    |       |
| emp_num  | int(11) | YES  |     | NULL    |       |
```


MySQL 5.5 Command Line Client

```
mysql> desc assign;
+----------+---------+------+-----+---------+-------+
| Field    | Type    | Null | Key | Default | Extra |
+----------+---------+------+-----+---------+-------+
| proj_num | int(11) | YES  |     | NULL    |       |
| emp_num  | int(11) | YES  |     | NULL    |       |
| hours    | int(11) | NO   |     | NULL    |       |
+----------+---------+------+-----+---------+-------+
3 rows in set (0.14 sec)

mysql> commit;
Query OK, 0 rows affected (0.03 sec)

mysql>
```

Conversion of Third Normal Form:

A table is in 3NF if:

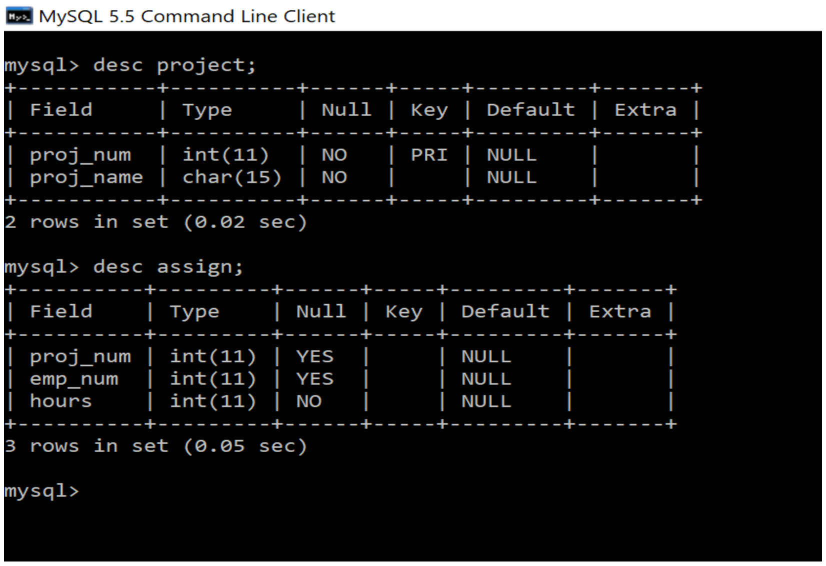
- It is in 2NF and
- It contains no transitive dependencies

PROJECT (PROJ_NUM, PROJ_NAME)

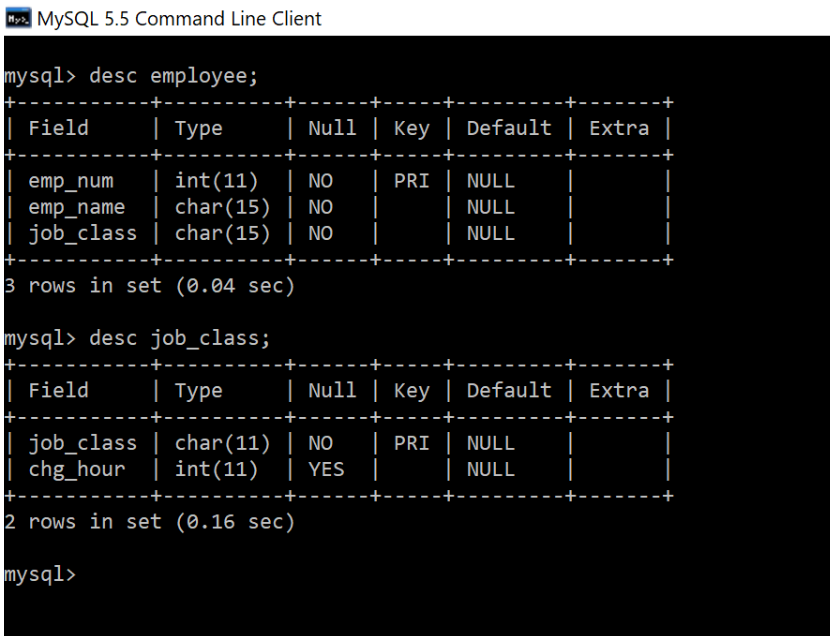
ASSIGN (PROJ_NUM, EMP_NUM, HOURS)

EMPLOYEE (EMP_NUM, EMP_NAME, JOB_CLASS)

JOB (JOB_CLASS, CHG_HOUR)


MySQL 5.5 Command Line Client

```
mysql> desc project;
+-----------+----------+------+-----+---------+-------+
| Field     | Type     | Null | Key | Default | Extra |
+-----------+----------+------+-----+---------+-------+
| proj_num  | int(11)  | NO   | PRI | NULL    |       |
| proj_name | char(15) | NO   |     | NULL    |       |
+-----------+----------+------+-----+---------+-------+
2 rows in set (0.02 sec)

mysql> desc assign;
+----------+---------+------+-----+---------+-------+
| Field    | Type    | Null | Key | Default | Extra |
+----------+---------+------+-----+---------+-------+
| proj_num | int(11) | YES  |     | NULL    |       |
| emp_num  | int(11) | YES  |     | NULL    |       |
| hours    | int(11) | NO   |     | NULL    |       |
+----------+---------+------+-----+---------+-------+
3 rows in set (0.05 sec)

mysql>
```


MySQL 5.5 Command Line Client

```
mysql> desc employee;
+-----------+----------+------+-----+---------+-------+
| Field     | Type     | Null | Key | Default | Extra |
+-----------+----------+------+-----+---------+-------+
| emp_num   | int(11)  | NO   | PRI | NULL    |       |
| emp_name  | char(15) | NO   |     | NULL    |       |
| job_class | char(15) | NO   |     | NULL    |       |
+-----------+----------+------+-----+---------+-------+
3 rows in set (0.04 sec)

mysql> desc job_class;
+-----------+----------+------+-----+---------+-------+
| Field     | Type     | Null | Key | Default | Extra |
+-----------+----------+------+-----+---------+-------+
| job_class | char(11) | NO   | PRI | NULL    |       |
| chg_hour  | int(11)  | YES  |     | NULL    |       |
+-----------+----------+------+-----+---------+-------+
2 rows in set (0.16 sec)

mysql>
```

RESULT: The Student is able to perform normalization on tables to produce redundant free database design.

VIVA-VOCE:

1. Define Normalization?
2. Outline Fully Functional dependency?
3. Define partial dependency?
4. Recall join dependency?
5. Define multi valued dependency?

EXPERIMENT 4: PRACTICING DDL COMMANDS

AIM: To Practice DDL commands by creating a database.

RECOMMENDED HARDWARE/ SOFTWARE REQUIREMENTS:

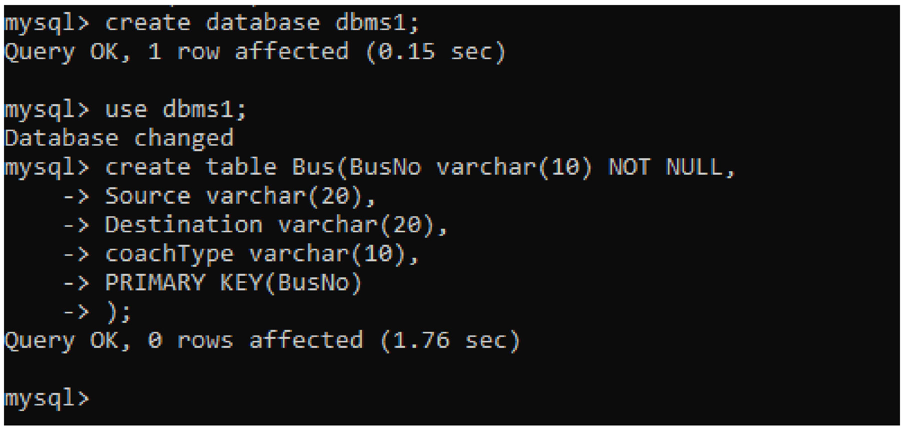
- Hardware Requirements: Intel Based desktop PC with minimum of 166 MHZ or faster processor with at least 1GB RAM and 500 MB free disk space.
- MySQL 5.6.1

PRE-REQUISITES: Student must know DDL commands

DDL Commands:

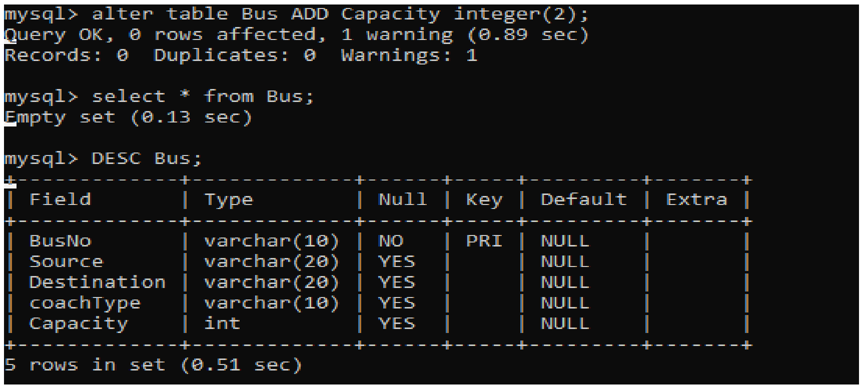
1. **Create** -It is used to create the database and a table
2. **Alter** -it is used to alter the table and also a database.
3. **Drop** -it is used to drop the database instance
4. **Truncate**-it is used to delete table in a database instance

CREATE



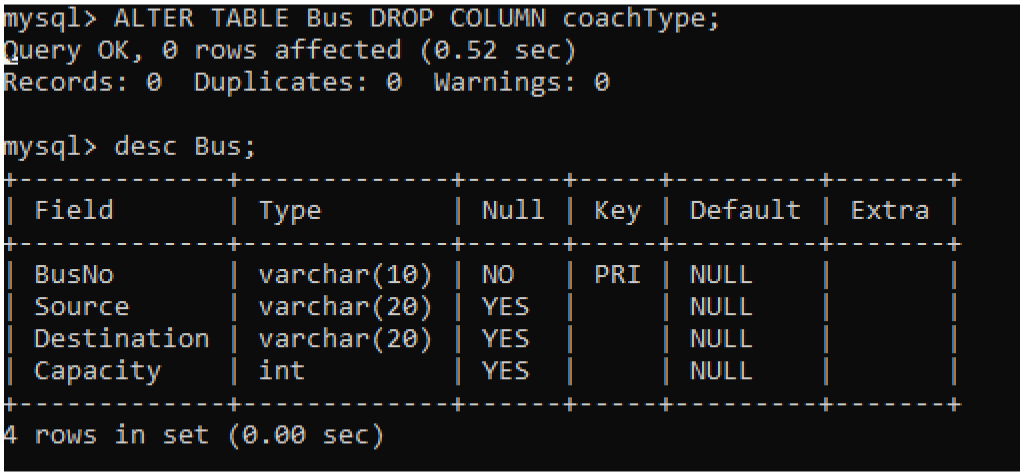
```
mysql> create database dbms1;
Query OK, 1 row affected (0.15 sec)

mysql> use dbms1;
Database changed
mysql> create table Bus(BusNo varchar(10) NOT NULL,
    -> Source varchar(20),
    -> Destination varchar(20),
    -> coachType varchar(10),
    -> PRIMARY KEY(BusNo)
    -> );
Query OK, 0 rows affected (1.76 sec)

mysql>
```

ALTER



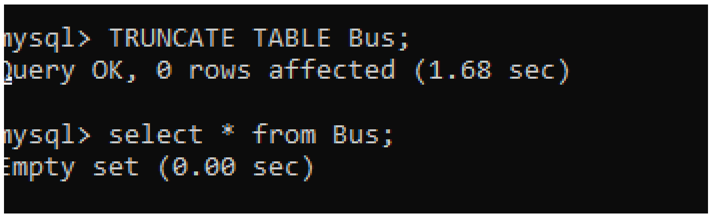
```
mysql> alter table Bus ADD Capacity integer(2);
Query OK, 0 rows affected, 1 warning (0.89 sec)
Records: 0  Duplicates: 0  Warnings: 1

mysql> select * from Bus;
Empty set (0.13 sec)

mysql> DESC Bus;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| BusNo       | varchar(10) | NO   | PRI | NULL    |       |
| Source      | varchar(20) | YES  |     | NULL    |       |
| Destination | varchar(20) | YES  |     | NULL    |       |
| coachType   | varchar(10) | YES  |     | NULL    |       |
| Capacity    | int         | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
5 rows in set (0.51 sec)
```

DROP



```
mysql> ALTER TABLE Bus DROP COLUMN coachType;
Query OK, 0 rows affected (0.52 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc Bus;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| BusNo       | varchar(10) | NO   | PRI | NULL    |       |
| Source      | varchar(20) | YES  |     | NULL    |       |
| Destination | varchar(20) | YES  |     | NULL    |       |
| Capacity    | int         | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

TRUNCATE



```
mysql> TRUNCATE TABLE Bus;
Query OK, 0 rows affected (1.68 sec)

mysql> select * from Bus;
Empty set (0.00 sec)
```

RESULT: The student is able to work in the MYSQL environment and gains the knowledge on DDL Commands.

VIVA VOCE:

1. Distinguish SQL from MYSQL?
2. Differentiate between drop and truncate?
3. Abbreviation of SQL?
4. Show the syntax to add a record to table?
5. A. Define Commit

   B. Define Schema

EXPERIMENT 5: PRACTICING DML COMMANDS

AIM: To perform the Database modifications by using DML Commands

RECOMMENDED HARDWARE/ SOFTWARE REQUIREMENTS:

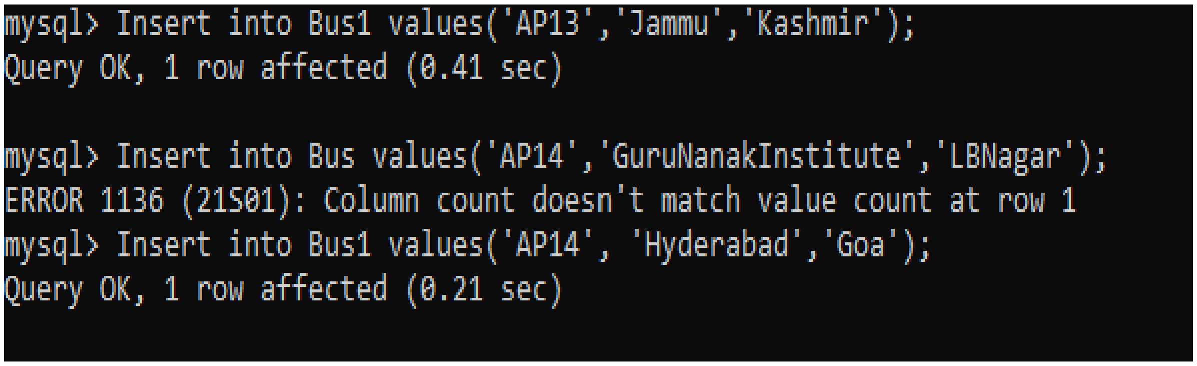
- Hardware Requirements: Intel Based desktop PC with minimum of 166 MHZ or faster processor with at least 64MB RAM and 100 MB free disk space.
- MySQL 5.6.1

PRE-REQUISITES: Students must know DML commands

DML Commands:

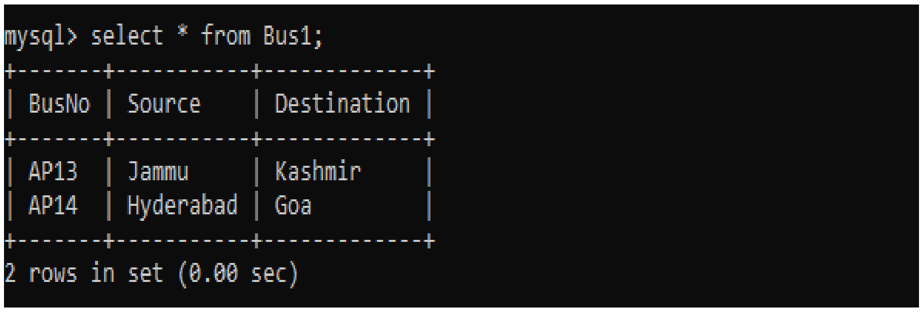
1. SELECT – retrieve data from a database
2. INSERT - insert data into a table
3. UPDATE - updates existing data within a table
4. DELETE – deletes all records from a table, the space for the records remains

INSERT



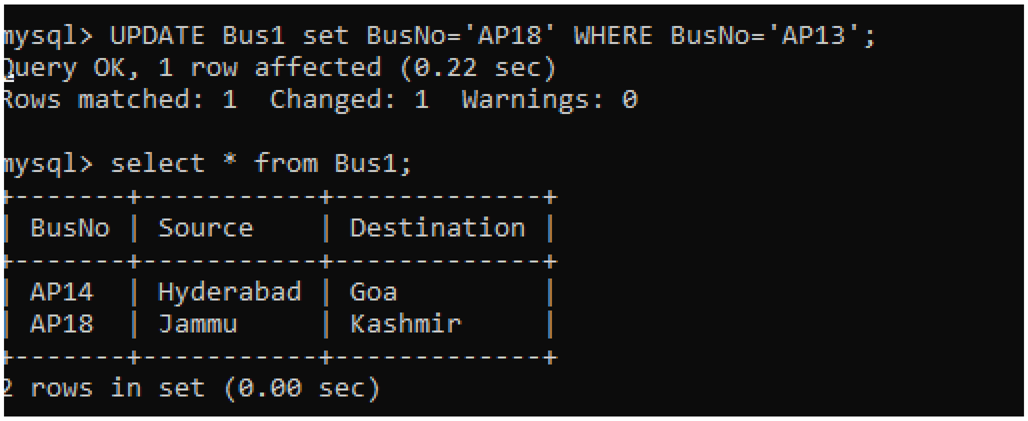
```
mysql> Insert into Bus1 values('AP13','Jammu','Kashmir');
Query OK, 1 row affected (0.41 sec)

mysql> Insert into Bus values('AP14','GuruNanakInstitute','LBNagar');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> Insert into Bus1 values('AP14', 'Hyderabad','Goa');
Query OK, 1 row affected (0.21 sec)
```

SELECT



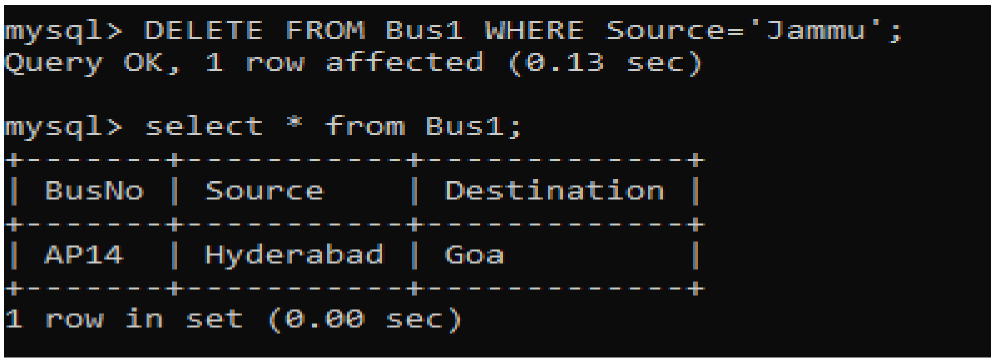
```
mysql> select * from Bus1;
+-------+-----------+-------------+
| BusNo | Source    | Destination |
+-------+-----------+-------------+
| AP13  | Jammu     | Kashmir     |
| AP14  | Hyderabad | Goa         |
+-------+-----------+-------------+
2 rows in set (0.00 sec)
```

UPDATE



```
mysql> UPDATE Bus1 set BusNo='AP18' WHERE BusNo='AP13';
Query OK, 1 row affected (0.22 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from Bus1;
+-------+-----------+-------------+
| BusNo | Source    | Destination |
+-------+-----------+-------------+
| AP14  | Hyderabad | Goa         |
| AP18  | Jammu     | Kashmir     |
+-------+-----------+-------------+
2 rows in set (0.00 sec)
```

DELETE



```
mysql> DELETE FROM Bus1 WHERE Source='Jammu';
Query OK, 1 row affected (0.13 sec)

mysql> select * from Bus1;
+-------+-----------+-------------+
| BusNo | Source    | Destination |
+-------+-----------+-------------+
| AP14  | Hyderabad | Goa         |
+-------+-----------+-------------+
1 row in set (0.00 sec)
```

RESULT: The Student gains the knowledge on DML Commands like Insert, Update, Delete and Select.

VIVA-VOCE:

1. Why do we use SQL Constraints?
2. Differentiate between delete and truncate?
3. State some properties of RDMS?
4. Define SQL delete statement?
5. Define Check Constraint?

EXPERIMENT 6: QUERYING

AIM: Performing querying using *ANY*, ALL, IN, Exists, NOT EXISTS, UNION, INTERSECT, Constraints etc.

RECOMMENDED HARDWARE/ SOFTWARE REQUIREMENTS:

- Hardware Requirements: Intel Based desktop PC with minimum of 166 MHZ or faster processor with at least 64MB RAM and 100 MB free disk space.
- MySQL 5.6.1

PRE-REQUISITES: Student must know about the RDBMS- Basic forms of SQL

OPERATORS:

1.**UNION***:*
UNION is used to combine results of two or more **SELECT** statements.it eliminates the duplicate rows *from* result set.

2.INTERSECT:
Intersect operation is used to combine two select statements, but it only returns the records which are **COMMON** from both **SELECT** statements

3.MINUS/EXCEPT:
The minus operation combines results of two SELECT statements and return only those in the final result, which belongs to the first set of the result

4.IN:
The in-operator helps to connect inner query to outer query and also allows to test whether a value is in a given set of elements.

5.NOT-IN:
The not-in operator is used as opposite of IN operator

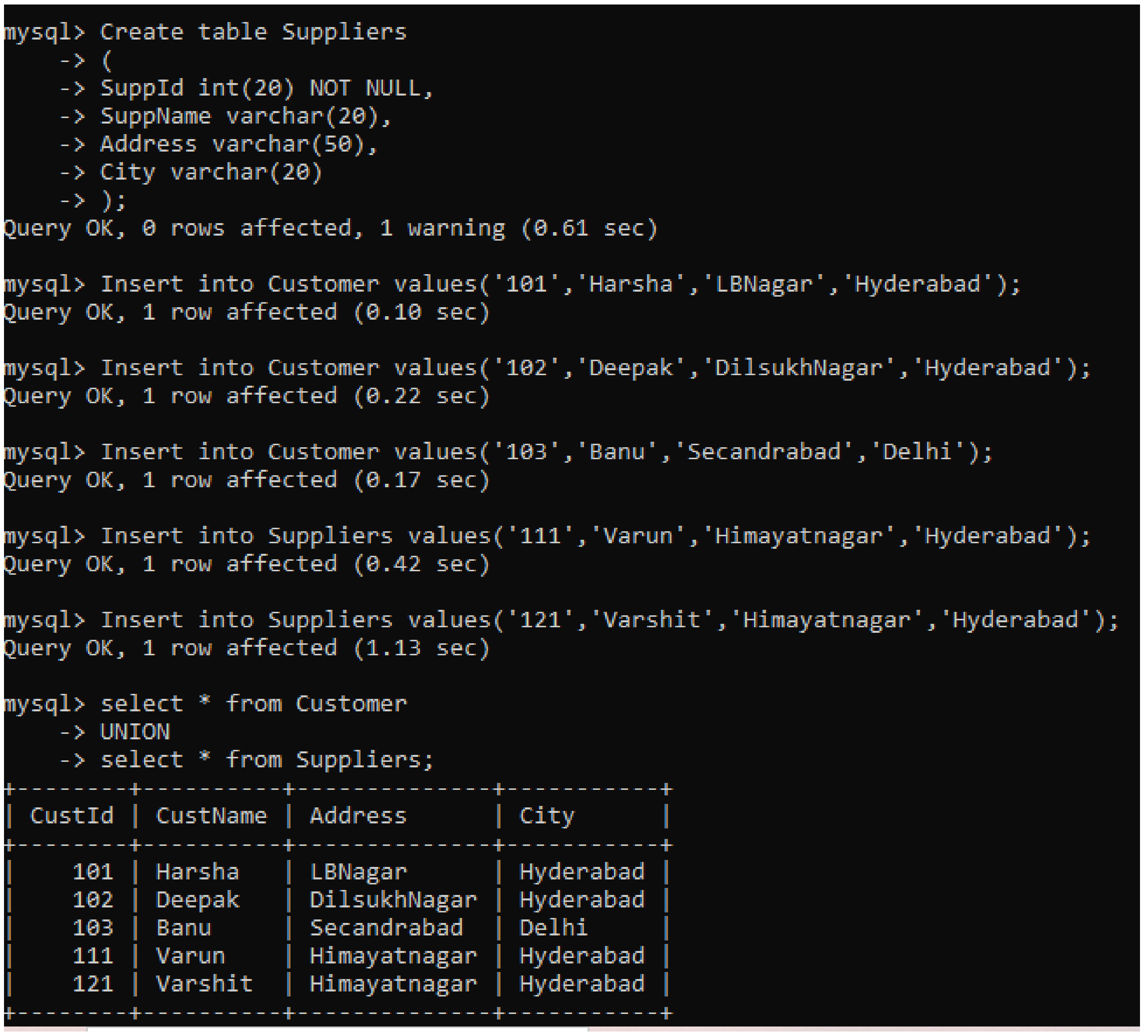
6.EXISTS:
The exists operator is used to search for the presence of a row in a specified table that meets a certain criterion. It allows us to test whether a set is non empty or not.

7.**NOT EXISTS**:
It is used opposite to EXISTS OPERATOR.

8.ANY:
It compares a value to any applicable value in the list As per the condition

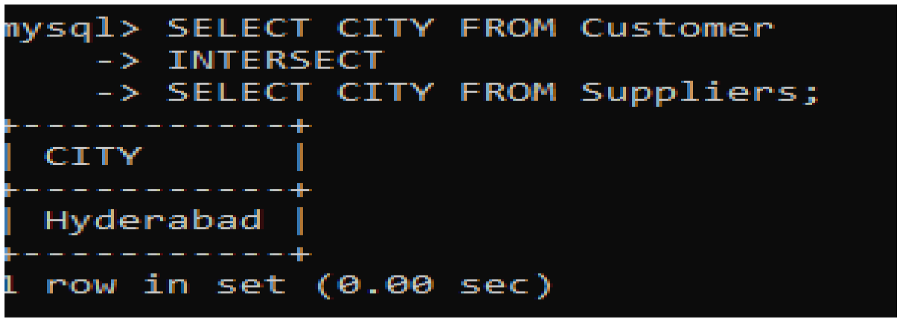
9.ALL:
ALL operator is used to select all tuples of SELECT statements.

UNION



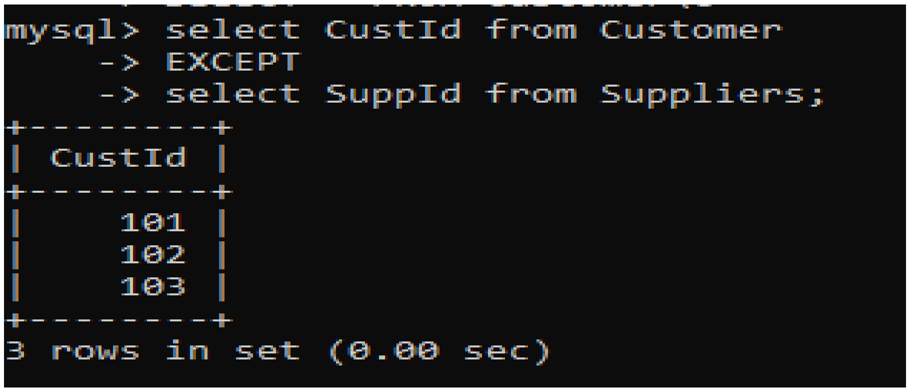
```
mysql> Create table Suppliers
    -> (
    -> SuppId int(20) NOT NULL,
    -> SuppName varchar(20),
    -> Address varchar(50),
    -> City varchar(20)
    -> );
Query OK, 0 rows affected, 1 warning (0.61 sec)

mysql> Insert into Customer values('101','Harsha','LBNagar','Hyderabad');
Query OK, 1 row affected (0.10 sec)

mysql> Insert into Customer values('102','Deepak','DilsukhNagar','Hyderabad');
Query OK, 1 row affected (0.22 sec)

mysql> Insert into Customer values('103','Banu','Secandrabad','Delhi');
Query OK, 1 row affected (0.17 sec)

mysql> Insert into Suppliers values('111','Varun','Himayatnagar','Hyderabad');
Query OK, 1 row affected (0.42 sec)

mysql> Insert into Suppliers values('121','Varshit','Himayatnagar','Hyderabad');
Query OK, 1 row affected (1.13 sec)

mysql> select * from Customer
    -> UNION
    -> select * from Suppliers;
+--------+----------+--------------+-----------+
| CustId | CustName | Address      | City      |
+--------+----------+--------------+-----------+
|    101 | Harsha   | LBNagar      | Hyderabad |
|    102 | Deepak   | DilsukhNagar | Hyderabad |
|    103 | Banu     | Secandrabad  | Delhi     |
|    111 | Varun    | Himayatnagar | Hyderabad |
|    121 | Varshit  | Himayatnagar | Hyderabad |
+--------+----------+--------------+-----------+
```

INTERSECT

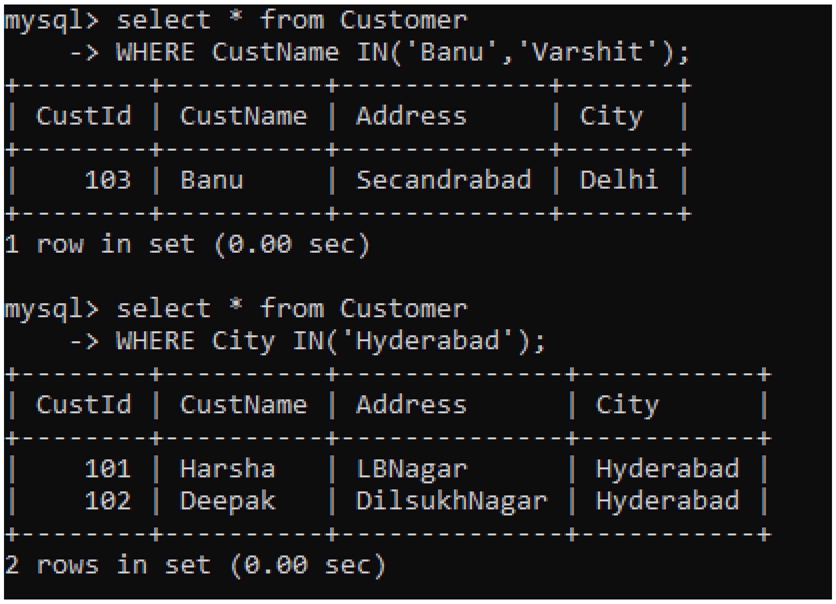


```
mysql> SELECT CITY FROM Customer
    -> INTERSECT
    -> SELECT CITY FROM Suppliers;
+-----------+
| CITY      |
+-----------+
| Hyderabad |
+-----------+
1 row in set (0.00 sec)
```

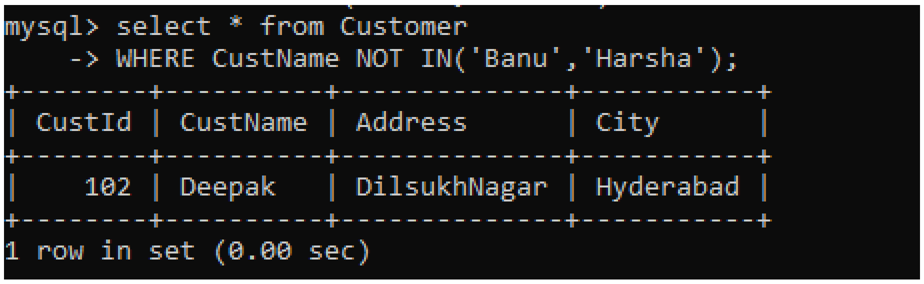
EXCEPT


```
mysql> select CustId from Customer
    -> EXCEPT
    -> select SuppId from Suppliers;
+--------+
| CustId |
+--------+
|    101 |
|    102 |
|    103 |
+--------+
3 rows in set (0.00 sec)
```

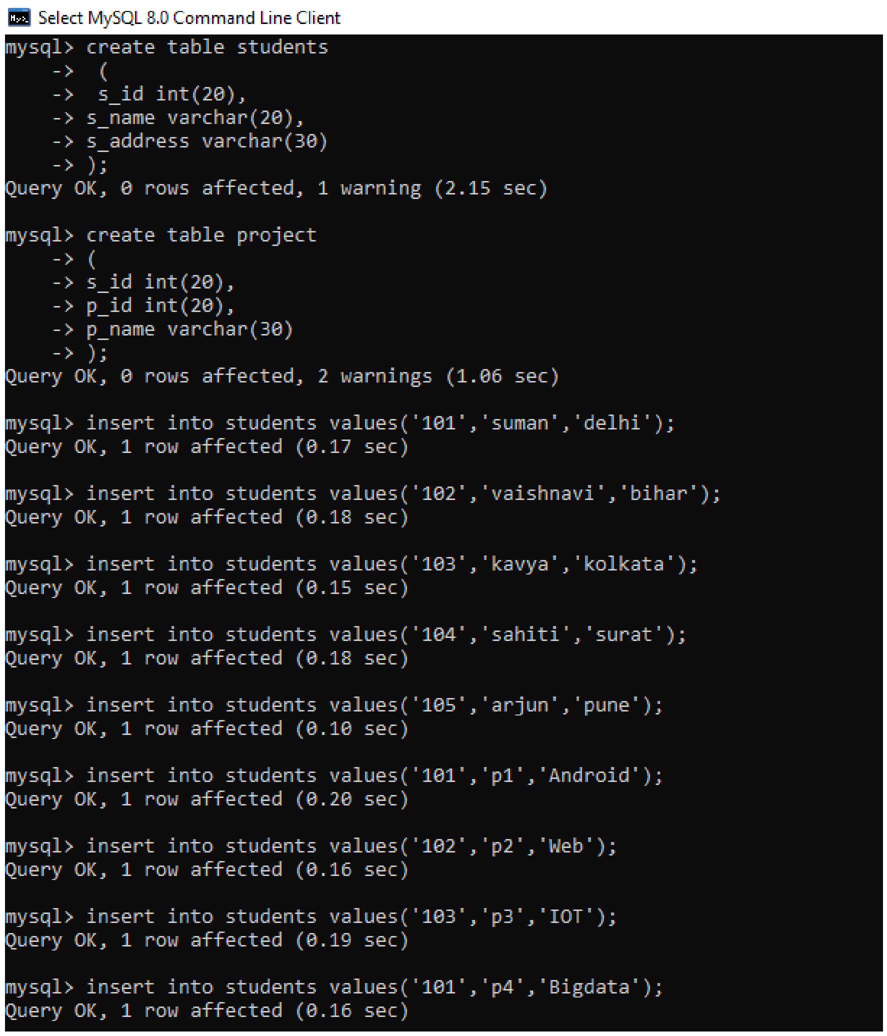
IN-OPERATOR


```
mysql> select * from Customer
    -> WHERE CustName IN('Banu','Varshit');
+--------+----------+-------------+-------+
| CustId | CustName | Address     | City  |
+--------+----------+-------------+-------+
|    103 | Banu     | Secandrabad | Delhi |
+--------+----------+-------------+-------+
1 row in set (0.00 sec)

mysql> select * from Customer
    -> WHERE City IN('Hyderabad');
+--------+----------+--------------+-----------+
| CustId | CustName | Address      | City      |
+--------+----------+--------------+-----------+
|    101 | Harsha   | LBNagar      | Hyderabad |
|    102 | Deepak   | DilsukhNagar | Hyderabad |
+--------+----------+--------------+-----------+
2 rows in set (0.00 sec)
```

NOT-IN OPERATOR

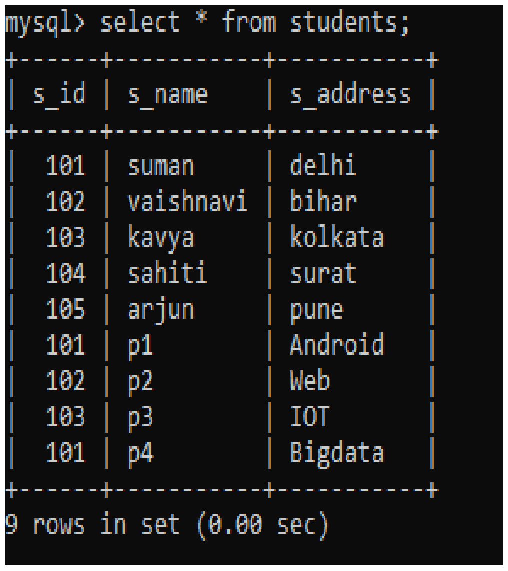

```
mysql> select * from Customer
    -> WHERE CustName NOT IN('Banu','Harsha');
+--------+----------+--------------+-----------+
| CustId | CustName | Address      | City      |
+--------+----------+--------------+-----------+
|    102 | Deepak   | DilsukhNagar | Hyderabad |
+--------+----------+--------------+-----------+
1 row in set (0.00 sec)
```

EXISTS AND NOT EXISTS

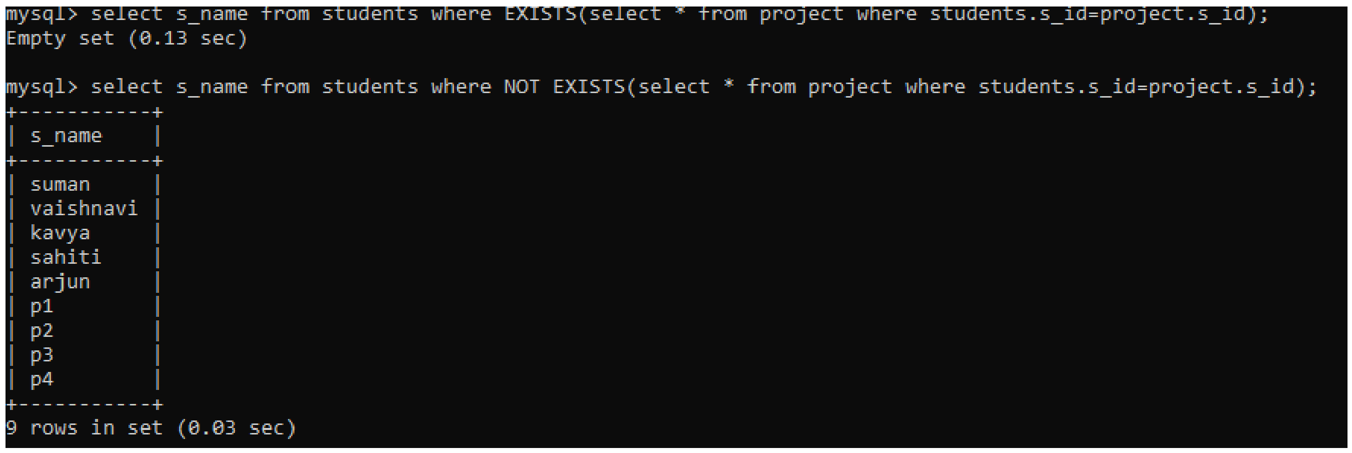


Select MySQL 8.0 Command Line Client

```
mysql> create table students
    -> (
    -> s_id int(20),
    -> s_name varchar(20),
    -> s_address varchar(30)
    -> );
Query OK, 0 rows affected, 1 warning (2.15 sec)

mysql> create table project
    -> (
    -> s_id int(20),
    -> p_id int(20),
    -> p_name varchar(30)
    -> );
Query OK, 0 rows affected, 2 warnings (1.06 sec)

mysql> insert into students values('101','suman','delhi');
Query OK, 1 row affected (0.17 sec)

mysql> insert into students values('102','vaishnavi','bihar');
Query OK, 1 row affected (0.18 sec)

mysql> insert into students values('103','kavya','kolkata');
Query OK, 1 row affected (0.15 sec)

mysql> insert into students values('104','sahiti','surat');
Query OK, 1 row affected (0.18 sec)

mysql> insert into students values('105','arjun','pune');
Query OK, 1 row affected (0.10 sec)

mysql> insert into students values('101','p1','Android');
Query OK, 1 row affected (0.20 sec)

mysql> insert into students values('102','p2','Web');
Query OK, 1 row affected (0.16 sec)

mysql> insert into students values('103','p3','IOT');
Query OK, 1 row affected (0.19 sec)

mysql> insert into students values('101','p4','Bigdata');
Query OK, 1 row affected (0.16 sec)
```

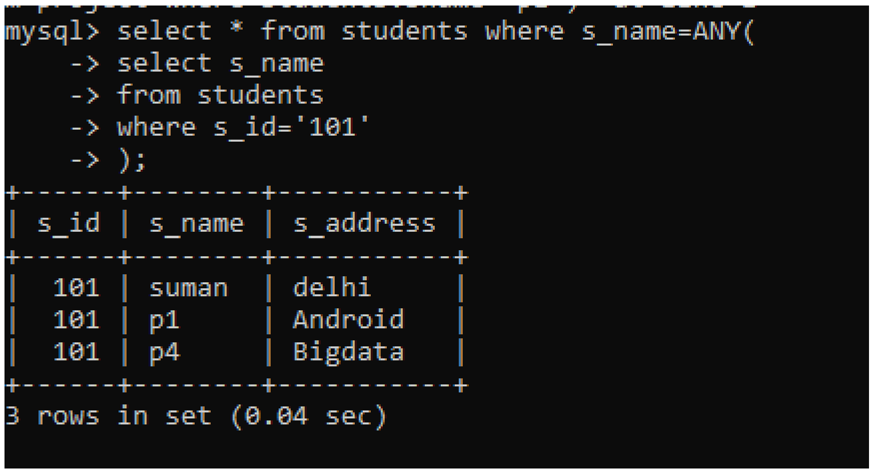


```
mysql> select * from students;
+------+-----------+-----------+
| s_id | s_name    | s_address |
+------+-----------+-----------+
|  101 | suman     | delhi     |
|  102 | vaishnavi | bihar     |
|  103 | kavya     | kolkata   |
|  104 | sahiti    | surat     |
|  105 | arjun     | pune      |
|  101 | p1        | Android   |
|  102 | p2        | Web       |
|  103 | p3        | IOT       |
|  101 | p4        | Bigdata   |
+------+-----------+-----------+
9 rows in set (0.00 sec)
```



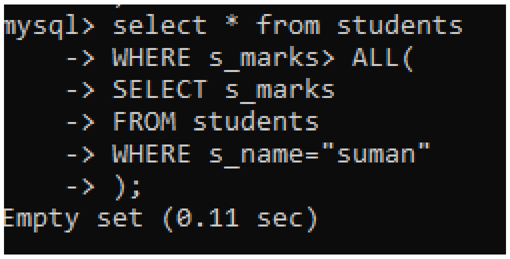
```
mysql> select s_name from students where EXISTS(select * from project where students.s_id=project.s_id);
Empty set (0.13 sec)

mysql> select s_name from students where NOT EXISTS(select * from project where students.s_id=project.s_id);
+-----------+
| s_name    |
+-----------+
| suman     |
| vaishnavi |
| kavya     |
| sahiti    |
| arjun     |
| p1        |
| p2        |
| p3        |
| p4        |
+-----------+
9 rows in set (0.03 sec)
```

ANY



```
mysql> select * from students where s_name=ANY(
    -> select s_name
    -> from students
    -> where s_id='101'
    -> );
+------+--------+-----------+
| s_id | s_name | s_address |
+------+--------+-----------+
|  101 | suman  | delhi     |
|  101 | p1     | Android   |
|  101 | p4     | Bigdata   |
+------+--------+-----------+
3 rows in set (0.04 sec)
```

ALL



```
mysql> select * from students
    -> WHERE s_marks> ALL(
    -> SELECT s_marks
    -> FROM students
    -> WHERE s_name="suman"
    -> );
Empty set (0.11 sec)
```

RESULT: The Student is able execute the Queries by using the above operators.

VIVA- VOCE

1. Specify the result of String functions?
2. Specify the result of Date functions?
3. Infer the result of conversion function?
4. Define Concatenation?
5. Differentiate between LTRIM and RTRIM?

EXPERIMENT 7 : QUERYING (CONTINUED …)

AIM: Performing the querying using Aggregate functions (COUNT, SUM, AVG, and MAX and MIN), GROUP BY, HAVING and Creation and dropping of Views.

RECOMMENDED HARDWARE/ SOFTWARE REQUIREMENTS:

- Hardware Requirements: Intel Based desktop PC with minimum of 166 MHZ or faster processor with at least 1GB RAM and 500 MB free disk space.
- MySQL 5.6.1

PRE-REQUISITES: Student must know about the RDBMS-SQL

AGGREGATE FUNCTIONS:

a) AVG: Retrieve average value of a column.

b) SUM: Retrieve the sum of all unique values in a column

c) COUNT: Retrieve the count of a column

d) MAX: Retrieve the maximum value of a column

e) MIN: Retrieve the minimum value of a column

CLAUSES:

a)ORDER BY:

The ORDER BY clause sorts the results of a query in ascending or descending order.

b)GROUP BY:

Sometimes we want to apply aggregate functions to groups of rows

c)HAVING:

HAVING is like *a* WHERE *clause except that it applies to the results of a GROUP BY query*.

d)VIEW:

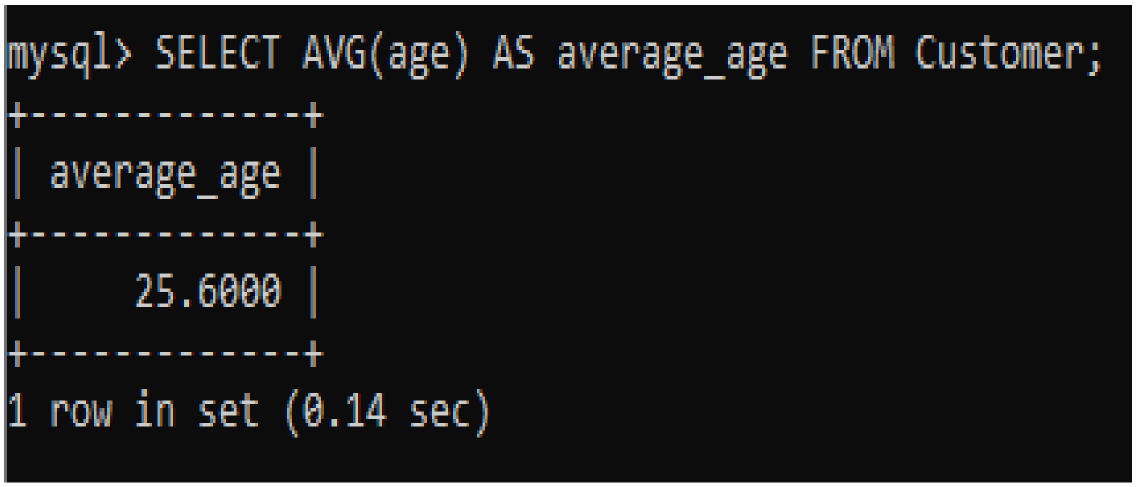
A VIEW is a table whose rows are not explicitly stored in the database but are computed as needed from ‘view definition’.

A VIEW is a computed table by taking reference from base tables.

AGGREGATE FUNCTIONS:

AVG:

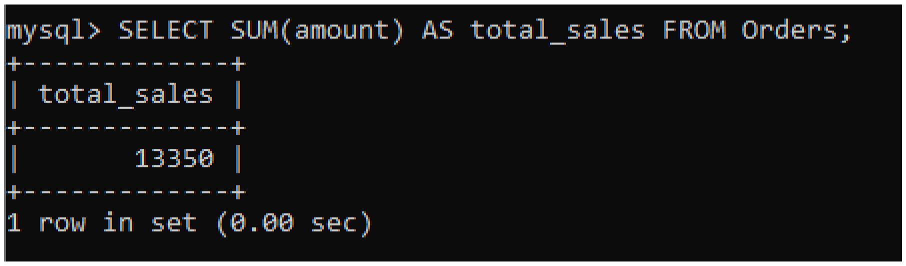
```
mysql> create table Customer
    -> (
    -> cust_id integer(20),
    -> fname varchar(30),
    -> lname varchar(30),
    -> age int(10),
    -> country varchar(30)
    -> );
mysql> INSERT INTO Customer VALUES('1','John','Doe','31','USA');
mysql> INSERT INTO Customer VALUES('2','Robinson','Luna','22','USA');
mysql> INSERT INTO Customer VALUES('3','David','Robinson','22','UK');
mysql> INSERT INTO Customer VALUES('4','John','Ronald','25','UK');
mysql> INSERT INTO Customer VALUES('5','Betty','Doe','28','UAE');
```



```
mysql> SELECT AVG(age) AS average_age FROM Customer;
+-------------+
| average_age |
+-------------+
|     25.6000 |
+-------------+
1 row in set (0.14 sec)
```

SUM

```
create table Orders
    -> (
    -> order_id integer(20),
    -> item varchar(30),
    ->  amount integer(40),
    -> cust_id integer(40)
    -> );
mysql> INSERT INTO Orders VALUES('1','Keyboard','400','4');
mysql>  INSERT INTO Orders VALUES('2','Mouse','300','4');
mysql>  INSERT INTO Orders VALUES('3','Monitor','12000','3');
mysql>  INSERT INTO Orders VALUES('4','Keyboard','400','1');
mysql>  INSERT INTO Orders VALUES('5','Mousepad','250','2');
```



```
mysql> SELECT SUM(amount) AS total_sales FROM Orders;
+-------------+
| total_sales |
+-------------+
|       13350 |
+-------------+
1 row in set (0.00 sec)
```

Count

```
mysql>  create table Customer
    -> (
    -> cust_id integer(20),
    ->  fname varchar(30),
    ->  lname varchar(30),
    -> age int(10),
    -> country varchar(30)
    -> );
```

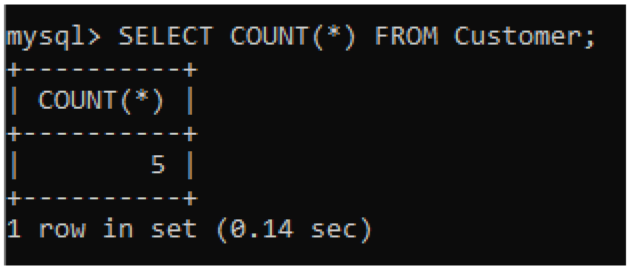
mysql> INSERT INTO Customer VALUES('1','John','Doe','31','USA');

mysql> INSERT INTO Customer VALUES('2','Robinson','Luna','22','USA');

mysql> INSERT INTO Customer VALUES('3','David','Robinson','22','UK');

mysql> INSERT INTO Customer VALUES('4','John','Ronald','25','UK');

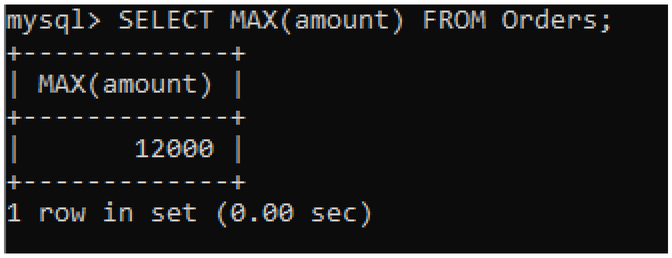
mysql> INSERT INTO Customer VALUES('5','Betty','Doe','28','UAE');



```
mysql> SELECT COUNT(*) FROM Customer;
+----------+
| COUNT(*) |
+----------+
|        5 |
+----------+
1 row in set (0.14 sec)
```

MAX

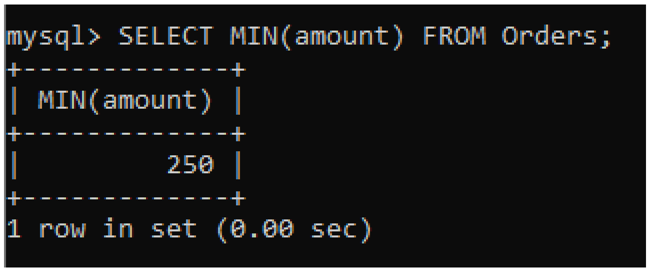
```
create table Orders
    -> (
    -> order_id integer(20),
    -> item varchar(30),
    ->  amount integer(40),
    -> cust_id integer(40)
    -> );
mysql> INSERT INTO Orders VALUES('1','Keyboard','400','4');
mysql>  INSERT INTO Orders VALUES('2','Mouse','300','4');
mysql>  INSERT INTO Orders VALUES('3','Monitor','12000','3');
mysql>  INSERT INTO Orders VALUES('4','Keyboard','400','1');
mysql>  INSERT INTO Orders VALUES('5','Mousepad','250','2');
```



```
mysql> SELECT MAX(amount) FROM Orders;
+-------------+
| MAX(amount) |
+-------------+
|       12000 |
+-------------+
1 row in set (0.00 sec)
```

MIN

```
create table Orders
   -> (
   -> order_id integer(20),
   -> item varchar(30),
   ->  amount integer(40),
   -> cust_id integer(40)
   -> );
mysql> INSERT INTO Orders VALUES('1','Keyboard','400','4');
mysql>  INSERT INTO Orders VALUES('2','Mouse','300','4');
mysql>  INSERT INTO Orders VALUES('3','Monitor','12000','3');
mysql>  INSERT INTO Orders VALUES('4','Keyboard','400','1');
mysql>  INSERT INTO Orders VALUES('5','Mousepad','250','2');
```



```
mysql> SELECT MIN(amount) FROM Orders;
+-------------+
| MIN(amount) |
+-------------+
|         250 |
+-------------+
1 row in set (0.00 sec)
```

GROUP BY

```
mysql>  create table Customer
   -> (
   -> cust_id integer(20),
   ->  fname varchar(30),
   ->  lname varchar(30),
   -> age int(10),
   -> country varchar(30)
   -> );
```

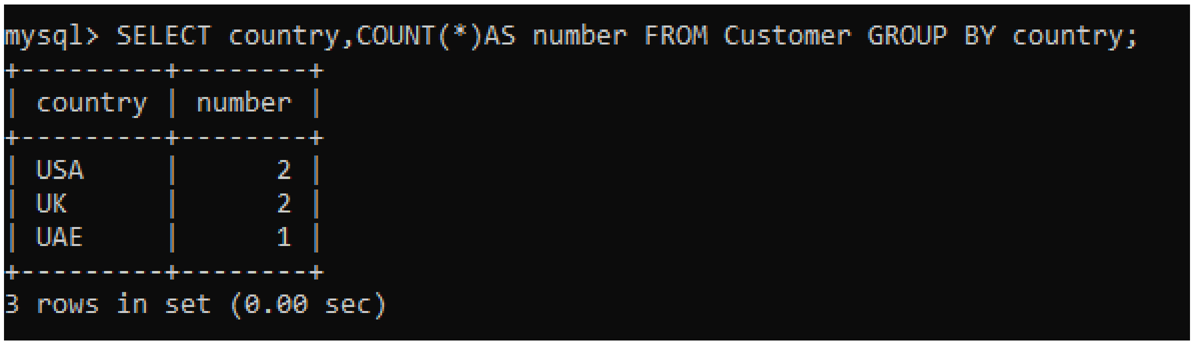
mysql> INSERT INTO Customer VALUES('1','John','Doe','31','USA');

mysql> INSERT INTO Customer VALUES('2','Robinson','Luna','22','USA');

mysql> INSERT INTO Customer VALUES('3','David','Robinson','22','UK');

mysql> INSERT INTO Customer VALUES('4','John','Ronald','25','UK');

mysql> INSERT INTO Customer VALUES('5','Betty','Doe','28','UAE');

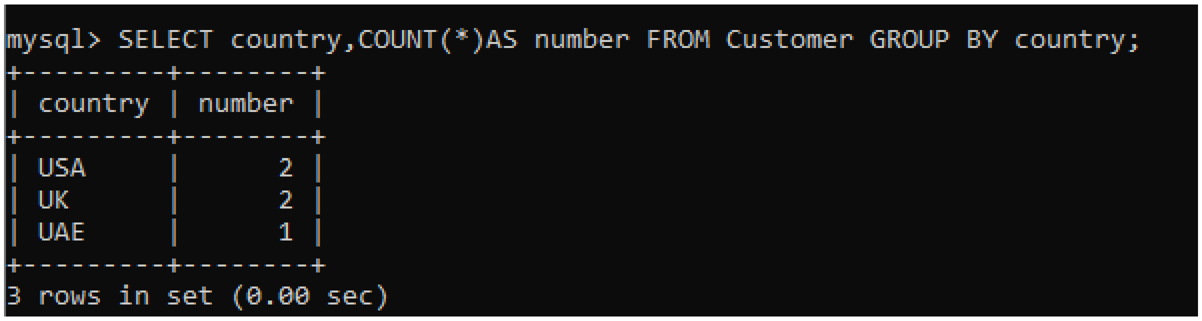


```
mysql> SELECT country,COUNT(*)AS number FROM Customer GROUP BY country;
+---------+--------+
| country | number |
+---------+--------+
| USA     |      2 |
| UK      |      2 |
| UAE     |      1 |
+---------+--------+
3 rows in set (0.00 sec)
```

ORDER BY

```
mysql>  create table Customer
    -> (
    -> cust_id integer(20),
    ->  fname varchar(30),
    ->  lname varchar(30),
    -> age int(10),
    -> country varchar(30)
    -> );
```

mysql> INSERT INTO Customer VALUES('1','John','Doe','31','USA');
mysql> INSERT INTO Customer VALUES('2','Robinson','Luna','22','USA');
mysql> INSERT INTO Customer VALUES('3','David','Robinson','22','UK');
mysql> INSERT INTO Customer VALUES('4','John','Ronald','25','UK');
mysql> INSERT INTO Customer VALUES('5','Betty','Doe','28','UAE');



```
mysql> SELECT country,COUNT(*)AS number FROM Customer GROUP BY country;
+---------+--------+
| country | number |
+---------+--------+
| USA     |      2 |
| UK      |      2 |
| UAE     |      1 |
+---------+--------+
3 rows in set (0.00 sec)
```

HAVING

mysql> create table Customer

-> (

-> cust_id integer(20),

-> fname varchar(30),

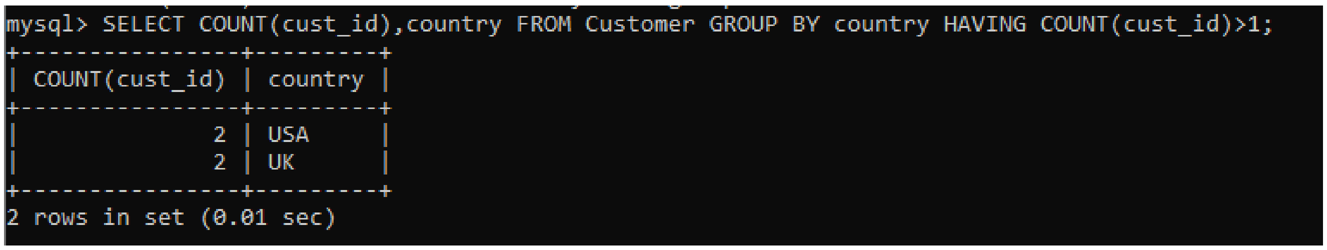
-> lname varchar(30),

-> age int(10),

-> country varchar(30)

->);

mysql> INSERT INTO Customer VALUES('1','John','Doe','31','USA');
mysql> INSERT INTO Customer VALUES('2','Robinson','Luna','22','USA');
mysql> INSERT INTO Customer VALUES('3','David','Robinson','22','UK');
mysql> INSERT INTO Customer VALUES('4','John','Ronald','25','UK');
mysql> INSERT INTO Customer VALUES('5','Betty','Doe','28','UAE');

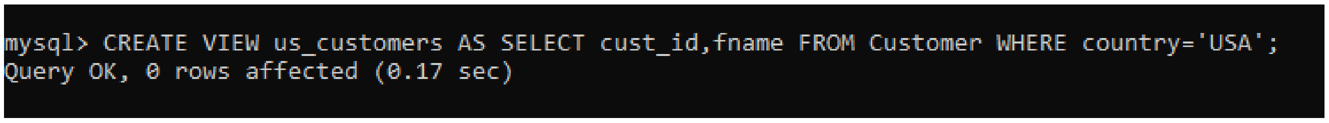


```
mysql> SELECT COUNT(cust_id),country FROM Customer GROUP BY country HAVING COUNT(cust_id)>1;
+----------------+---------+
| COUNT(cust_id) | country |
+----------------+---------+
|              2 | USA     |
|              2 | UK      |
+----------------+---------+
2 rows in set (0.01 sec)
```

VIEWS

CREATE VIEW

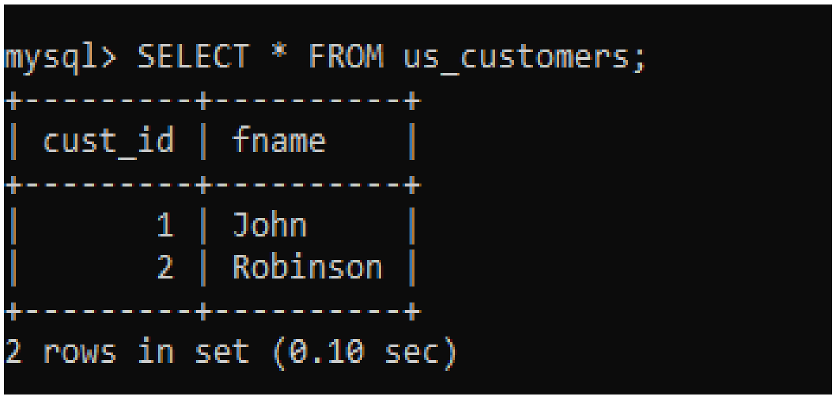
```
mysql>  create table Customer
    -> (
    -> cust_id integer(20),
    ->  fname varchar(30),
    ->  lname varchar(30),
    -> age int(10),
    -> country varchar(30)
    -> );
mysql> INSERT INTO Customer VALUES('1','John','Doe','31','USA');
mysql> INSERT INTO Customer VALUES('2','Robinson','Luna','22','USA');
mysql> INSERT INTO Customer VALUES('3','David','Robinson','22','UK');
mysql> INSERT INTO Customer VALUES('4','John','Ronald','25','UK');
mysql> INSERT INTO Customer VALUES('5','Betty','Doe','28','UAE');
```



```
mysql> CREATE VIEW us_customers AS SELECT cust_id,fname FROM Customer WHERE country='USA';
Query OK, 0 rows affected (0.17 sec)
```

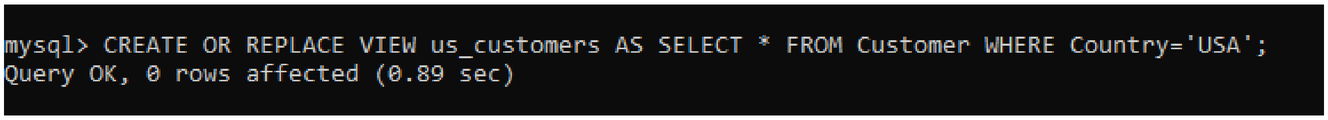
A VIEW NAMED us_customers is created from the customers’ table.

Now to select the customers who live in USA run following:



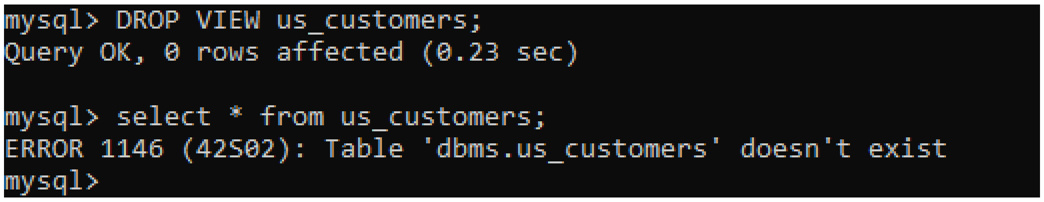
```
mysql> SELECT * FROM us_customers;
+---------+----------+
| cust_id | fname    |
+---------+----------+
|       1 | John     |
|       2 | Robinson |
+---------+----------+
2 rows in set (0.10 sec)
```

UPDATING VIEW



```
mysql> CREATE OR REPLACE VIEW us_customers AS SELECT * FROM Customer WHERE Country='USA';
Query OK, 0 rows affected (0.89 sec)
```

DELETING VIEW



```
mysql> DROP VIEW us_customers;
Query OK, 0 rows affected (0.23 sec)

mysql> select * from us_customers;
ERROR 1146 (42S02): Table 'dbms.us_customers' doesn't exist
mysql>
```

EXPERIMENT 8: TRIGGERS

AIM: To Implement the concept of triggers -Insert, Update, Delete

RECOMMENDED HARDWARE/ SOFTWARE REQUIREMENTS:

- Hardware Requirements: Intel Based desktop PC with minimum of 166 MHZ or faster processor with at least 64MB RAM and 100 MB free disk space.
- MySQL 5.6.1

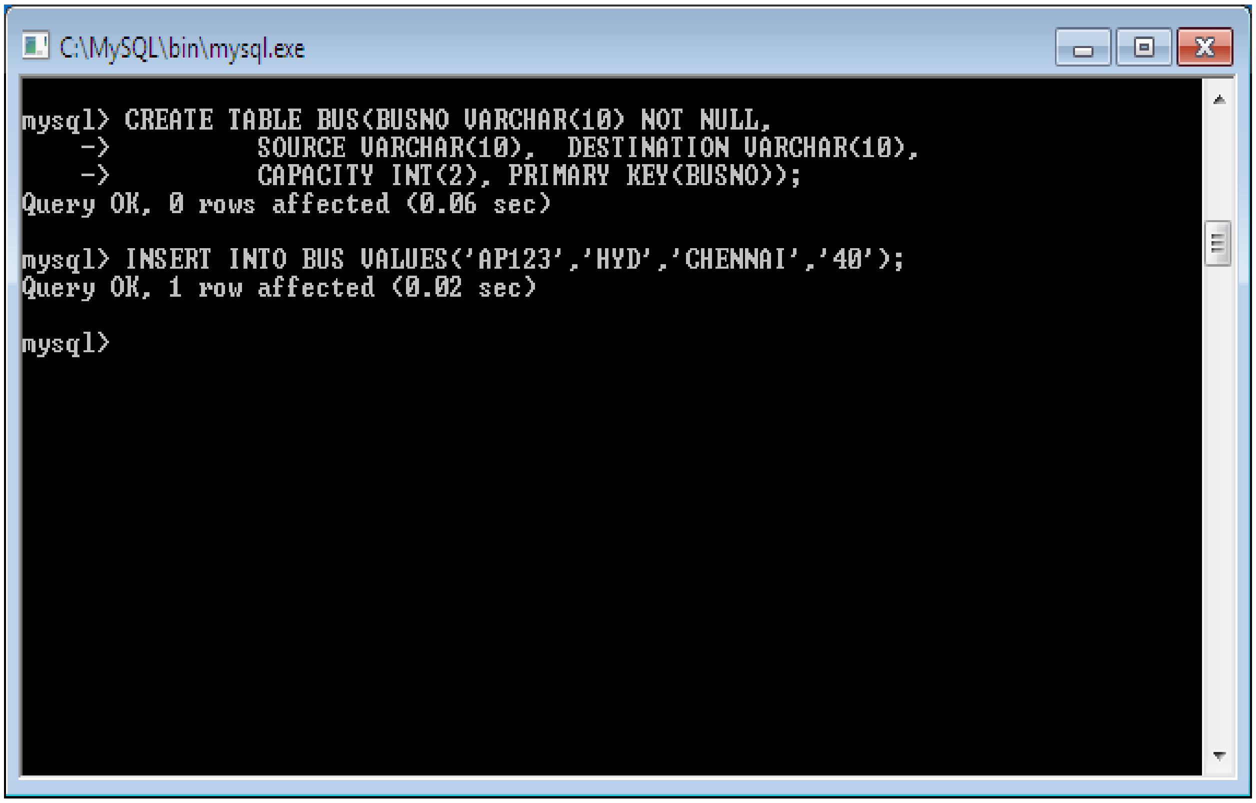
PRE-REQUISITES: Student must know about the Relational Database SQL-Triggers.

1. Create a table with the schema Bus(busno, source, destination, capacity)

```
MySQL>CREATE TABLE BUS(BUSNO VARCHAR(10) NOT NULL,
        SOURCE VARCHAR(10),  DESTINATION VARCHAR(10),
        CAPACITY INT(2), PRIMARY KEY(BUSNO));
```

2. Insert values

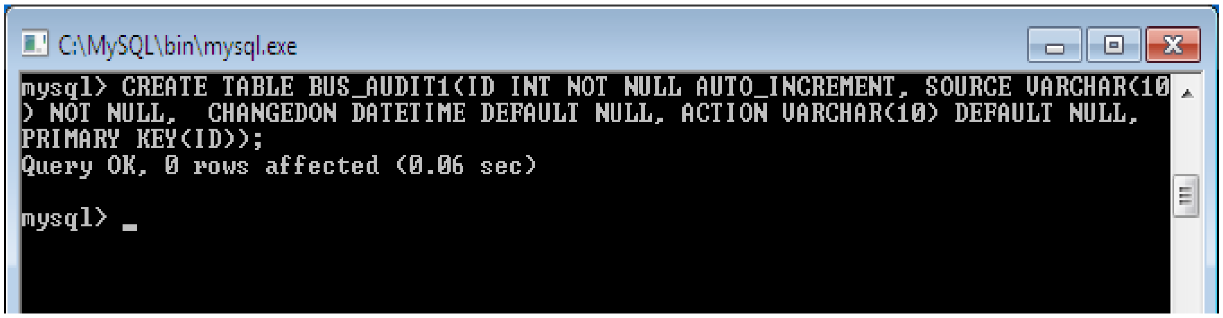
```
 MySQL>INSERT INTO BUS VALUES('AP123','HYD','CHENNAI','40');
```



```
mysql> CREATE TABLE BUS(BUSNO VARCHAR(10) NOT NULL,
    ->          SOURCE VARCHAR(10),  DESTINATION VARCHAR(10),
    ->          CAPACITY INT(2), PRIMARY KEY(BUSNO));
Query OK, 0 rows affected (0.06 sec)

mysql> INSERT INTO BUS VALUES('AP123','HYD','CHENNAI','40');
Query OK, 1 row affected (0.02 sec)

mysql>
```

3. Create an Audit table for the bus to track the actions on the table using Triggers concept. (Schema : Bus_Audit1(<u>ID</u>, Source, Changedon, Action))

CREATE TABLE BUS_AUDIT1(ID INT NOT NULL AUTO_INCREMENT, SOURCE VARCHAR(10) NOT NULL, CHANGEDON DATETIME DEFAULT NULL, ACTION VARCHAR(10) DEFAULT NULL, PRIMARY KEY(ID));

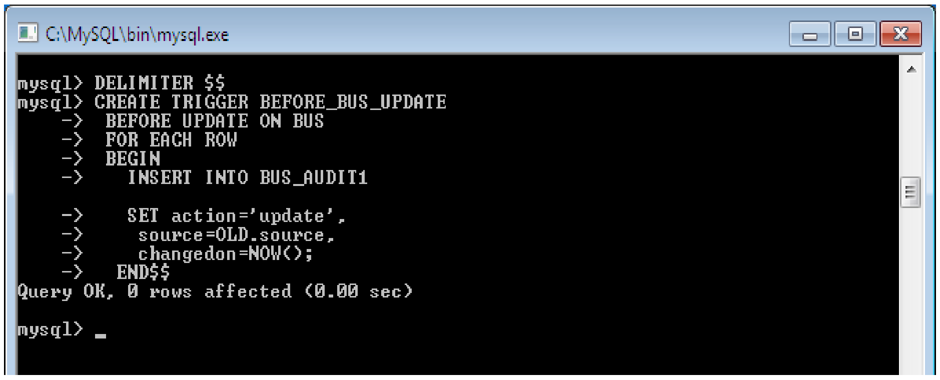


C:\MySQL\bin\mysql.exe

```
mysql> CREATE TABLE BUS_AUDIT1(ID INT NOT NULL AUTO_INCREMENT, SOURCE VARCHAR(10
) NOT NULL,  CHANGEDON DATETIME DEFAULT NULL, ACTION VARCHAR(10) DEFAULT NULL,
PRIMARY KEY(ID));
Query OK, 0 rows affected (0.06 sec)

mysql> _
```

4. Creating UPDATE Trigger:

```
DELIMITER $$
CREATE TRIGGER BEFORE_BUS_DELETE
 BEFORE DELETE  ON BUS
 FOR EACH ROW
 BEGIN
   INSERT INTO BUS_AUDIT1
   SET action='delete',
    source=OLD.source,
    changedon=NOW();
  END$$
```

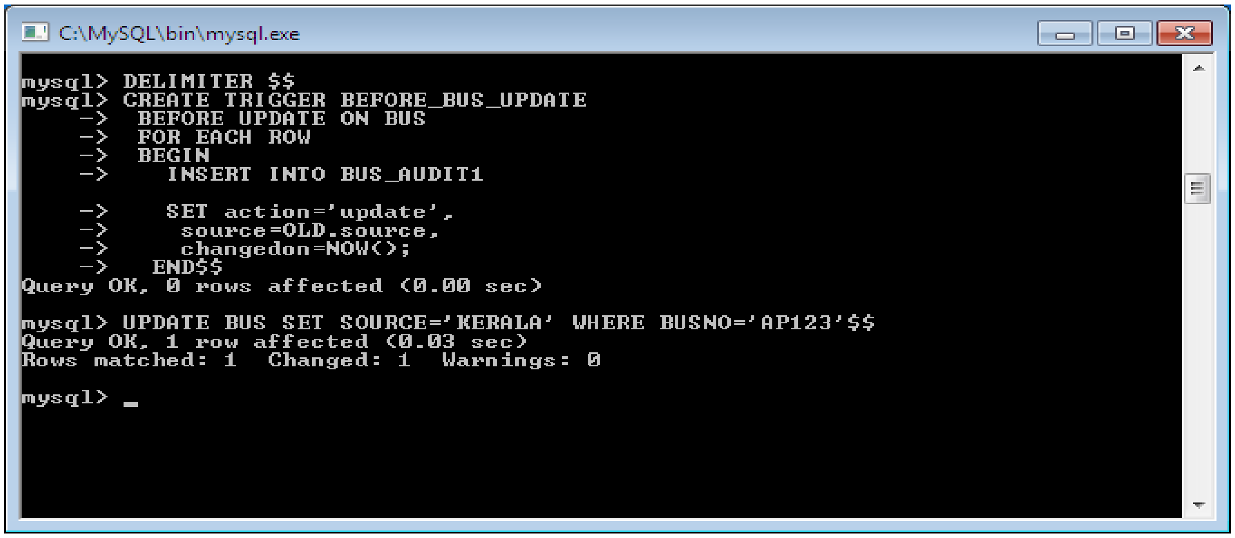


C:\MySQL\bin\mysql.exe

```
mysql> DELIMITER $$
mysql> CREATE TRIGGER BEFORE_BUS_UPDATE
    ->  BEFORE UPDATE ON BUS
    ->  FOR EACH ROW
    ->  BEGIN
    ->    INSERT INTO BUS_AUDIT1

    ->    SET action='update',
    ->     source=OLD.source,
    ->     changedon=NOW();
    ->   END$$
Query OK, 0 rows affected (0.00 sec)

mysql> _
```

5. Perform an UPDATE operation on the bus table:

MySQL>UPDATE BUS SET SOURCE='KERALA' WHERE BUSNO='AP123'$$



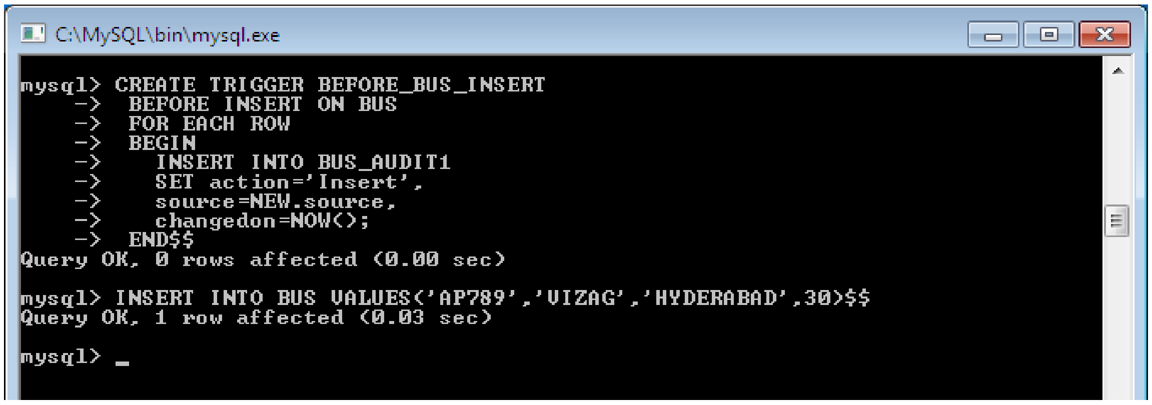

```
mysql> DELIMITER $$
mysql> CREATE TRIGGER BEFORE_BUS_UPDATE
    ->  BEFORE UPDATE ON BUS
    ->  FOR EACH ROW
    ->  BEGIN
    ->    INSERT INTO BUS_AUDIT1

    ->    SET action='update',
    ->     source=OLD.source,
    ->     changedon=NOW();
    ->   END$$
Query OK, 0 rows affected (0.00 sec)

mysql> UPDATE BUS SET SOURCE='KERALA' WHERE BUSNO='AP123'$$
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> _
```

6. Creating INSERT Trigger:

```
CREATE TRIGGER BEFORE_BUS_INSERT

 BEFORE INSERT ON BUS

 FOR EACH ROW

 BEGIN

   INSERT INTO BUS_AUDIT1

   SET action='Insert',

   source=NEW.source,

   changedon=NOW();

 END$$
```

7. Perform an INSERT operation on bus:

INSERT INTO BUS VALUES('AP789','VIZAG','HYDERABAD',30)$$




```
mysql> CREATE TRIGGER BEFORE_BUS_INSERT
    -> BEFORE INSERT ON BUS
    -> FOR EACH ROW
    -> BEGIN
    ->   INSERT INTO BUS_AUDIT1
    ->   SET action='Insert',
    ->   source=NEW.source,
    ->   changedon=NOW();
    -> END$$
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO BUS VALUES('AP789','VIZAG','HYDERABAD',30)$$
Query OK, 1 row affected (0.03 sec)

mysql> _
```

8. Create DELETE Trigger:

```
CREATE TRIGGER BEFORE_BUS_DELETE
 BEFORE DELETE ON BUS
 FOR EACH ROW
 BEGIN
   Insert into bus_audit1
   SET action='delete’,
   source=old.source,
   changedon=NOW();
 END$$
```

9. Perform DELETE operation on bus:

DELETE FROM BUS WHERE SOURCE=’HYDERABAD’$$

OUTPUT:

Select * from bus_audit1$$

SNo	Source	Changedon	Action
1	Banglore	2014:03:23 12:51:00	Insert
2	Kerela	2014:03:25:12:56:00	Update
3	Hyderabad	2014:04:26:12:59:02	Delete

RESULT: The Student is able to work on Triggers to create an active database.

VIVA- VOCE

1. DefineTRIGGER?
2. List the types of triggers?
3. List the trigger timings?
4. Is it possible to create a trigger on views?
5. Outline row and statement trigger?

EXPERIMENT -9: PROCEDURES

AIM: Creating and Executing Stored procedures.

RECOMMENDED HARDWARE/ SOFTWARE REQUIREMENTS:

- Hardware Requirements: Intel Based desktop PC with minimum of 166 MHZ or faster processor with at least 64MB RAM and 100 MB free disk space.
- MySQL 5.6.1

PRE-REQUISITES: Student must know about the Relational Database SQL-Procedures

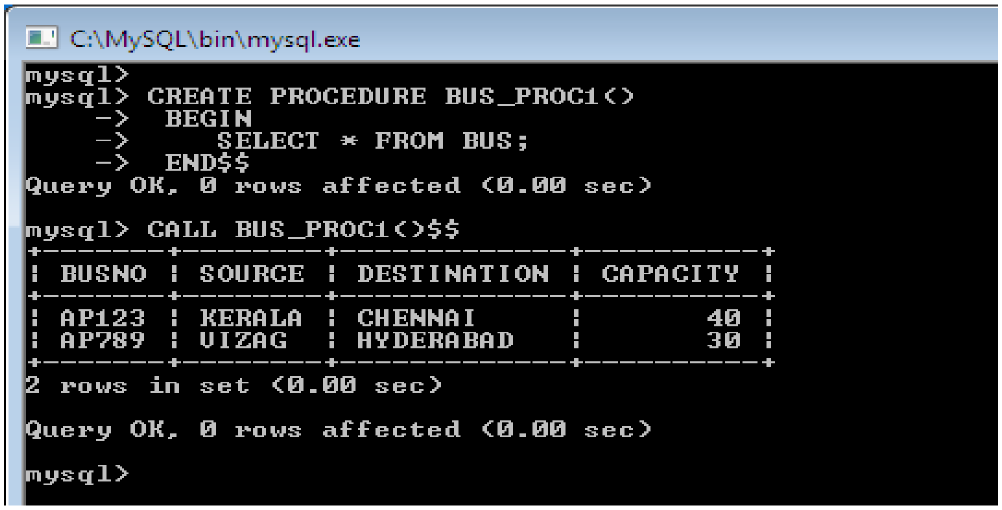
PROCEDURE: A stored procedure is a set of Structured Query Language (SQL) statements with an assigned name, which are stored in a relational database management system as a group, so it can be reused and shared by multiple programs.

Ex1: Executing a Simple procedure called BUS_PROC1(), When we execute it will display all the data from “bus” table.

```
Delimiter $$
CREATE PROCEDURE BUS_PROC1()
 BEGIN
   SELECT * FROM BUS;
   SELECT * FROM BUS_AUDIT1;
DESC BUS;
 END$$
```

OUTPUT:

```
CALL BUS_PROC1()$$
```


C:\MySQL\bin\mysql.exe

```
mysql>
mysql> CREATE PROCEDURE BUS_PROC1()
    ->   BEGIN
    ->      SELECT * FROM BUS;
    ->   END$$
Query OK, 0 rows affected (0.00 sec)

mysql> CALL BUS_PROC1()$$
+-------+--------+-------------+----------+
| BUSNO | SOURCE | DESTINATION | CAPACITY |
+-------+--------+-------------+----------+
| AP123 | KERALA | CHENNAI     |       40 |
| AP789 | VIZAG  | HYDERABAD   |       30 |
+-------+--------+-------------+----------+
2 rows in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql>
```

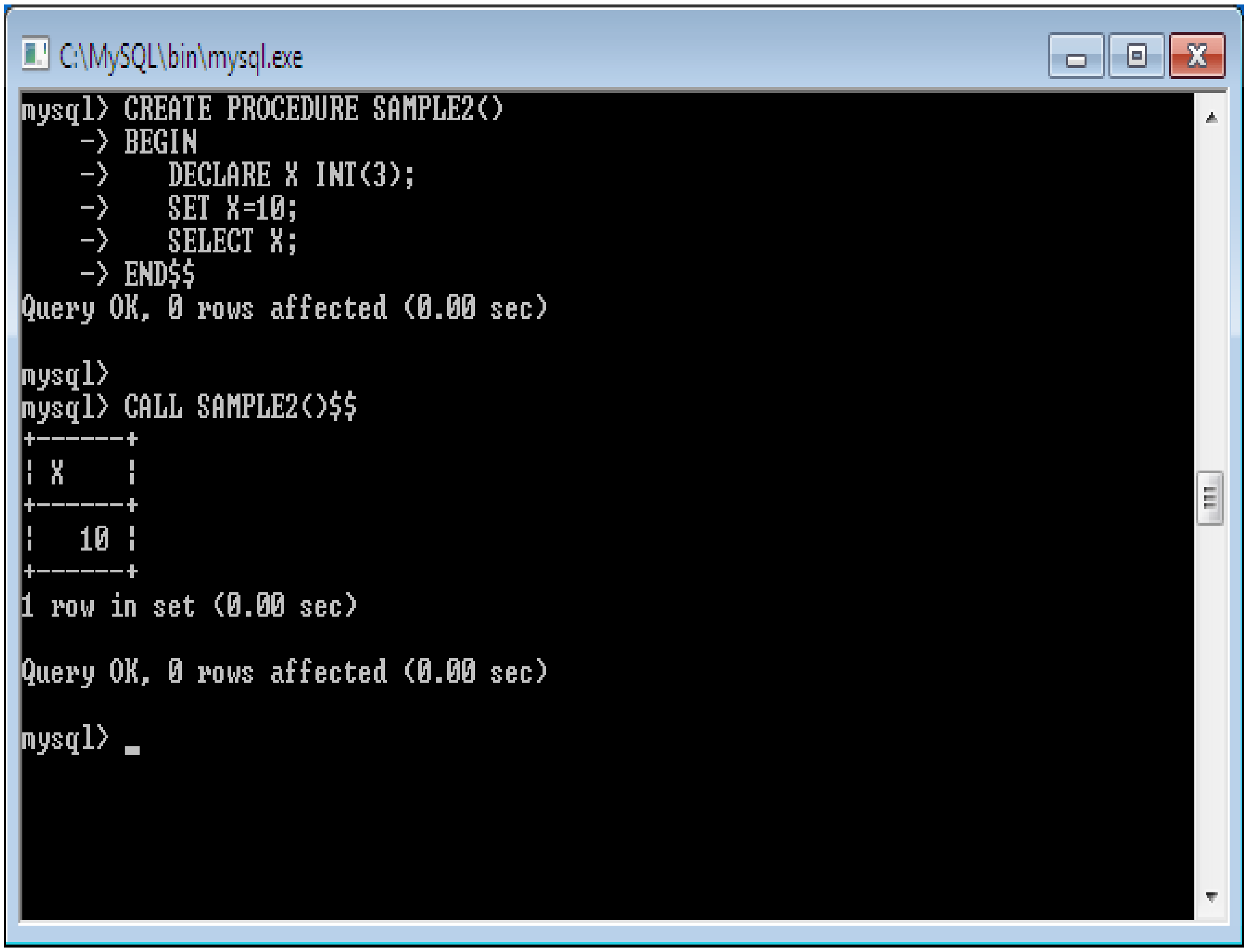
Ex2: Executing the procedure to show the declaration of local variables in a stored procedure.

Local variables are declared within stored procedures and are only valid between Begin and END. Block where they are declared. Local variables can have any SQL data type.

```
CREATE PROCEDURE SAMPLE2()
BEGIN
  DECLARE X INT(3);
  SET X=10;
  SELECT X;
END$$
```

OUTPUT:

```
CALL SAMPLE2()$$
```


C:\MySQL\bin\mysql.exe

```
mysql> CREATE PROCEDURE SAMPLE2()
    -> BEGIN
    ->    DECLARE X INT(3);
    ->    SET X=10;
    ->    SELECT X;
    -> END$$
Query OK, 0 rows affected (0.00 sec)

mysql>
mysql> CALL SAMPLE2()$$
+-------+
| X     |
+-------+
|   10 |
+-------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql> _
```

Ex3: Executing the Procedure parameter- OUT

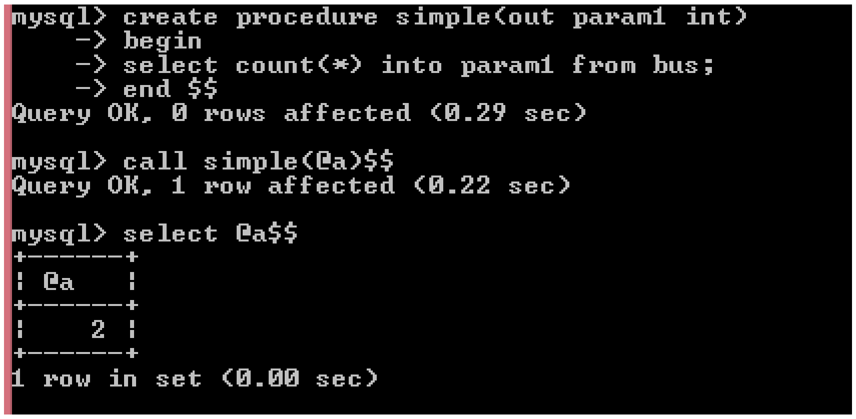
The following example shows a simple stored procedure that uses an OUT parameter.

```
CREATE PROCEDURE SIMPLE_PROC(OUT PARAM INT)
                BEGIN
                  SELECT COUNT(*) INTO PARAM FROM BUS;
                END$$
```

In the body of the procedure, the parameter will get the count value from the table bus. After calling the procedure the work OUT tells the DBMS that the values goes out from the procedure. Here param1 is the name of the output parameter and we have passed its value to a session variable named @a, in the call statement.

OUTPUT:

```
→CALL SIMPLE_PROC(@a)$$
Query ok, 1 row affected (0.22 sec)
→ SELECT @a$$
```



```
mysql> create procedure simple(out param1 int)
    -> begin
    -> select count(*) into param1 from bus;
    -> end $$
Query OK, 0 rows affected (0.29 sec)

mysql> call simple(@a)$$
Query OK, 1 row affected (0.22 sec)

mysql> select @a$$
+------+
| @a   |
+------+
|    2 |
+------+
1 row in set (0.00 sec)
```

RESULT: The Student is able to work on Stored Procedures.

VIVA VOCE:

1. Define stored procedure?
2. When would you use stored procedure or functions ?
3. State external procedures?
4. Recall input parameter and how it is different from OUT parameter?
5. Show how to use Stored Procedures

EXPERIMENT 10: CURSORS

AIM: To declare MySQL cursor in stored procedure to iterate through a result set returned by a SELECT statement.

RECOMMENDED HARDWARE/ SOFTWARE REQUIREMENTS :

- Hardware Requirements: Intel Based desktop PC with minimum of 166 MHZ or faster processor with at least 64MB RAM and 100 MB free disk space.
- MySQL 5.6.1

PRE-REQUISITES: Student must know about the Relational SQL-Cursors

CURSOR: To handle a result set inside a stored procedure, you use a cursor. A cursor allows you to iterate a set of rows returned by a query and process each row accordingly.

MySQL cursor is read-only, non-scrollable and asensitive.

- **Read-only**: you cannot update data in the underlying table through the cursor.
- **Non-scrollable**: you can only fetch rows in the order determined by the SELECT statement. You cannot fetch rows in the reversed order. In addition, you cannot skip rows or jump to a specific row in the result set.
- **Asensitive**: there are two kinds of cursors: asensitive cursor and insensitive cursor. An asensitive cursor points to the actual data, whereas an insensitive cursor uses a temporary copy of the data. An asensitive cursor performs faster than an insensitive cursor because it does not have to make a temporary copy of data. However, any change that made to the data from other connections will affect the data that is being used by an asensitive cursor, therefore, it is safer if you do not update the data that is being used by an asensitive cursor. MySQL cursor is asensitive.

Working with MySQL cursor:

Step1: Declare a cursor by using the DECLARE statement:

```
DECLARE cursor_name CURSOR FOR SELECT_statement;
```

The cursor declaration must be after any variable declaration. If you declare a cursor before variables declaration, MySQL will issue an error. A cursor must always be associated with a SELECT statement.

Step2: Open the cursor by using the OPEN statement. The OPEN statement initializes the result set for the cursor; therefore, you must call the OPEN statement before fetching rows from the result set.

```
1 OPEN cursor_name;
```

Step:3: FETCH statement to retrieve the next row pointed by the cursor and move the cursor to the next row in the result set.

```
1 FETCH cursor_name INTO variables list;
```

After that, you can check to see if there is any row available before fetching it.Declare a NOT FOUND handler to handle the situation when the cursor could not find any row.

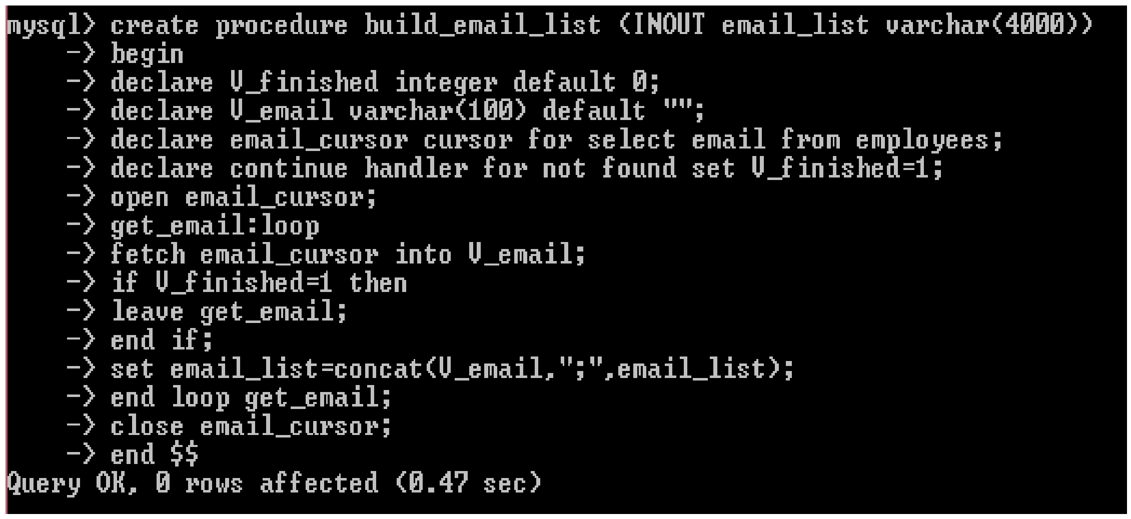
```
1 DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished = 1;
```

Step:4: CLOSE statement to deactivate the cursor and release the memory associated with it as follows:

```
1 CLOSE cursor_name;
```

Example: Developing a stored procedure that builds an email list of all employees in the employees table in the sample database.

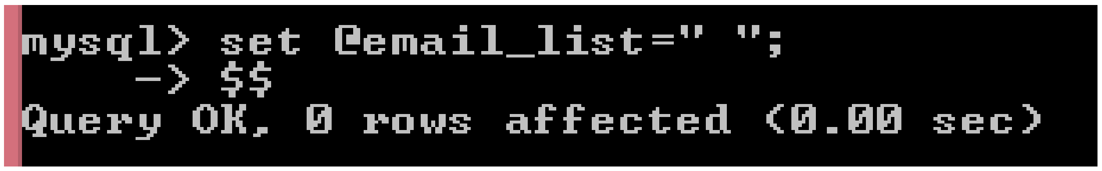
```
DELIMITER $$
CREATE PROCEDURE build_email_list (INOUT email_list varchar(4000))
BEGIN
 DECLARE v_finished INTEGER DEFAULT 0;
 DECLARE v_email varchar(100) DEFAULT "";
 -- declare cursor for employee email
DEClARE email_cursor CURSOR FOR
SELECT email FROM employee;
 -- declare NOT FOUND handler
DECLARE CONTINUE HANDLER
       FOR NOT FOUND SET v_finished = 1;
 OPEN email_cursor;
 get_email: LOOP
 FETCH email_cursor INTO v_email;
 IF v_finished = 1 THEN
LEAVE get_email;
END IF;
 -- build email list
SET email_list = CONCAT(v_email,";",email_list);
 END LOOP get_email;
 CLOSE email_cursor;
 END$$
```



```
mysql> create procedure build_email_list (INOUT email_list varchar(4000))
    -> begin
    -> declare V_finished integer default 0;
    -> declare V_email varchar(100) default "";
    -> declare email_cursor cursor for select email from employees;
    -> declare continue handler for not found set V_finished=1;
    -> open email_cursor;
    -> get_email:loop
    -> fetch email_cursor into V_email;
    -> if V_finished=1 then
    -> leave get_email;
    -> end if;
    -> set email_list=concat(V_email,";",email_list);
    -> end loop get_email;
    -> close email_cursor;
    -> end $$
Query OK, 0 rows affected (0.47 sec)
```

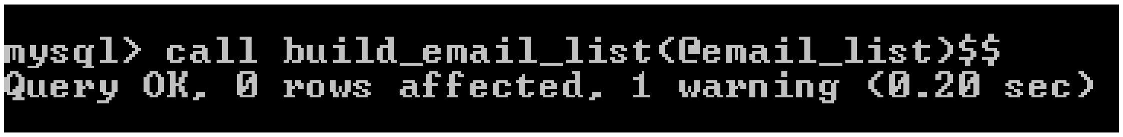
You can test the build_email_list stored procedure using the following script:

SET @email_list = ""$$



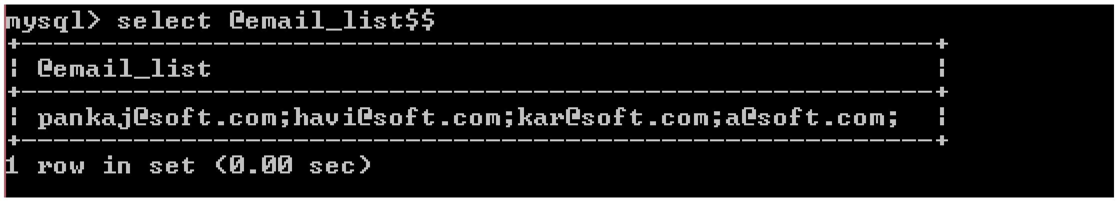
```
mysql> set @email_list=" ";
    -> $$
Query OK, 0 rows affected (0.00 sec)
```

CALL build_email_list(@email_list)$$



```
mysql> call build_email_list(@email_list)$$
Query OK, 0 rows affected, 1 warning (0.20 sec)
```

SELECT @email_list$$



```
mysql> select @email_list$$
+----------------------------------------------------------+
| @email_list                                              |
+----------------------------------------------------------+
| pankaj@soft.com;havi@soft.com;kar@soft.com;a@soft.com;   |
+----------------------------------------------------------+
1 row in set (0.00 sec)
```

RESULT: The Student is able to work on Cursors.

VIVA VOCE:

1. Define a cursor?
2. List the types of cursor?
3. State the use of parameterized cursor?
4. State the use of cursor variable?
5. Define normal cursor?

ADDITIONAL EXPERIMENTS

1. Design and implement queries on Tables (Emp,Dept)

A) AIM: To create the following relations using appropriate SQL statements:

Create table for various relation

DEPT (<u>DEPTNO: NUMBER(2)</u>, DNAME: VARCHAR2(10), LOC: VARCHAR2(8))

EMP(<u>EMPNO:NUMBER(4)</u>, ENAME: VARCHAR2(9),JOB: VARCHAR2(9),MGR:NUMBER(4), HIREDATE:DATE,SAL:NUMBER(7,2),COMM.:NUMBER(7,2),DEPTNO:NUMBER(2))

SALGRADE (GRADE:NUMBER(1),LOSAL:NUMBER(4),HISAL:NUMBER(4))

- **Define a constraint on EMP relation that will ensure that every employee earns atmost Rs.10000/-**
- **Define constraint on EMP relation such that deptno will be foreign key to DEPT relation**
- **Define dept's relation so that every department is guaranteed to have some name.**

```
SQL>CREATE TABLE DEPT (DEPTNO NUMBER (2) PRIMARY KEY, DNAME VARCHAR2 (10) NOT NULL, LOC VARCHAR2 (8));
```

Table created.

```
SQL>CREATE TABLE EMP(EMPNO NUMBER(4) PRIMARY KEY,ENAME VARCHAR2(9),JOB VARCHAR2(9),MGR NUMBER(4),HIREDATE DATE,SAL NUMBER(7,2) CHECK(SAL<=10000),COMM NUMBER(7,2),DEPTNO NUMBER(2) ,FOREIGN KEY(DEPTNO) REFERENCES DEPT);
```

Table created

```
SQL>CREATE TABLE SALGRADE (GRADE NUMBER (1), LOSAL NUMBER (4), HISAL NUMBER (4));
```

Table created.

To insert the following data into appropriate relations:

MULTI ROW INSERTIONS

1) SQL> INSERT INTO DEPT (DEPTNO, DNAME, LOC)

```
VALUES(&DEPTNO,'&DNAME','&LOC');
Enter value for deptno: 10
Enter value for dname: ACCOUNTING
Enter value for loc: NEWYORK
old   1: INSERT INTO DEPT VALUES(&DEPTNO,'&DNAME','&LOC')
new   1: INSERT INTO DEPT VALUES(10,'ACCOUNTING','NEWYORK')
```

1 row created.

2) SQL> INSERT INTO EMP(EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO)

```
VALUES(&EMPNO,’&ENAME’,’&JOB’,&MGR,’&HIREDATE’,&SAL,&COMM,&DEPT
NO);
Enter value for empno: 7499
Enter value for ename: ALLEN
Enter value for job: SALESMAN
Enter value for mgr: 7698
Enter value for hiredate: 20-FEB-81
Enter value for sal: 1600
Enter value for comm: 300
Enter value for deptno: 30
old   1: INSERT INTO
EMP(EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO)
new   1: INSERT INTO EMP(7499,’ALLEN’,’SALESMAN’,7698,’20-FEB-81’,1600,300,30)
```

1 row created.

3) SQL> INSERT INTO SALGRADE (GRADE,LOSAL,HISAL)

```
VALUES(&GRADE,&LOSAL,&HISAL);
Enter value for grade: 5
Enter value for losal: 3001
```

```
        Enter value for hisal: 9999
        old   1: INSERT INTO SALGRADE (GRADE,LOSAL,HISAL)
        VALUES(&GRADE,&LOSAL,&HISAL)
        new   1: INSERT INTO SALGRADE (GRADE,LOSAL,HISAL) VALUES(5,3001,9999)
        1 row created.
```

SINGLE ROW INSERTIONS

```
  SQL> INSERT INTO DEPT(DEPTNO,DNAME,LOC) VALUES(20,'RESEARCH','DALLAS')
        1 row created.
SQL> INSERT INTO EMP(EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO)
 VALUES(7521,’WARD’,’SALESMAN’,7698,’22-FEB-81’,1250,500,30);
        1 row created.
SQL>  INSERT INTO SALGRADE (GRADE,LOSAL,HISAL) VALUES(4,2001,3000);
        1 row created.
```

b) AIM: To Alter the tables using appropriate SQL statements:

1) Alter the size of dname of DEPT table

```
 SQL> ALTER TABLE DEPT MODIFY (DNAME VARCHAR2 (20));
 Table altered.
```

2) Alter the SALGRADE table by adding constraint unique to the field grade

```
SQL> ALTER TABLE SALGRADE ADD CONSTRAINT CONS_GRADE UNIQUE (GRADE);
 Table altered.
```

3) Alter the SALGRADE table by dropping constraint unique to the field grade

```
 SQL> ALTER TABLE SALGRADE DROP CONSTRAINT CONS_GRADE;
 Table altered.
```

c)AIM: Dropping the tables using appropriate SQL statements

1) Drop the EMP table

```
    SQL> DROP TABLE EMP;
    Table dropped.
```

2) Drop the DEPT table

```
    SQL>   DROP TABLE DEPT;
    Table dropped.
```

Working of Different Functions on Relation(single line and group functions).

- AGGREGATE FUNCTIONS
- STRING FUNCTIONS
- NUMBER FUNCTIONS
- DATE FUNCTIONS
- CONVERSION FUNCTIONS

a)AIM: Queries using aggregate functions(COUNT,SUM,AVG,MIN,MAX) GROUP BY,HAVING .

1) Find the number of rows in the EMP table.

```
SQL> SELECT COUNT(*) FROM EMP;
                 COUNT(*)
                ----------
                  14
```

2) List the numbers of jobs.

```
SQL>  SELECT COUNT(DISTINCT(JOB)) AS TOTALJOBS FROM EMP;
        TOTALJOBS
        ----------
          5
```

3) Find total salary of the EMP table

```
SQL> SELECT SUM(SAL) AS TOTALSALARY FROM EMP;
TOTALSALARY
-----------
 29025
```

4) List maximum sal, minimum sal, average sal of EMP table

```
SQL>  SELECT MAX(SAL),MIN(SAL),AVG(SAL) FROM EMP;
  MAX(SAL)   MIN(SAL)   AVG(SAL)
---------- ---------- ---------------------
  5000          800        2073.21429
```

5) List the numbers of people and average salary in deptno 30.

```
SQL>  select count(*),avg(sal) from emp where deptno=30;
 COUNT(*)              AVG(SAL)
--------------     -------------
  6                         1666.66667
```

6) List maximum sal and minimum sal in the designations SALESMAN and CLERK.

```
SQL>  SELECT COUNT(*),MAX(SAL),MIN(SAL),AVG(SAL) FROM EMP WHERE JOB
IN('SALESMAN','CLERK');
  COUNT(*)   MAX(SAL)   MIN(SAL)   AVG(SAL)
---------- ---------- ---------- ---------------------------
        8               1600            800      1218.75
```

7) Calculate total salary bill for each department.

```
SQL>SELECT DEPTNO,SUM(SAL) AS TOTALSAL FROM EMP GROUP BY DEPTNO;
    DEPTNO                        TOTALSAL
-------------------               ----------
       10                         8750
       20                         10875
       30                          9400
```

8) List max sal, min sal and average sal of depts. 10,30.

```
SQL> SELECT DEPTNO,MIN(SAL),MAX(SAL),AVG(SAL) FROM EMP WHERE DEPTNO
IN(10,30) GROUP BY DEPTNO;

    DEPTNO   MIN(SAL)   MAX(SAL)   AVG(SAL)
   ------- ---------- ---------- --------------------------
      10         1300       5000     2916.66667
      30          950       2850     1566.66667
```

9) Find all departments which having more than 3 employees.

```
SQL> SELECT  DEPTNO,COUNT(*) FROM EMP GROUP BY DEPTNO HAVING COUNT(*)>3;
   DEPTNO             COUNT(*)
     ----------     ----------
      20                    5
      30                     6
```

10) Display the jobs where the minimum salary is greater than or equal   to 3000.

```
SQL> SELECT JOB,MIN(SAL) FROM EMP GROUP BY JOB HAVING MIN(SAL)>=3000;
JOB        MIN(SAL)
---------      ----------
ANALYST        3000
PRESIDENT       5000
```

b) AIM: Queries using string functions
(Concatenation, lpad, rpad, ltrim, rtrim, lower, upper, initcap, length, substr and instr)

1) Display the output for all departments in the following manner:
Department number 10 with name Accounting is situated in New York.

```
SQL> SELECT 'Department number '||DEPTNO||' with name '||INITCAP(DNAME)||' is situated in
'||INITCAP(LOC) AS CONCATENATEDSTRING FROM DEPT;
                              CONCATENATEDSTRING
------------------------------------------------------------------------------------------
Department number 10 with name Accounting is situated in New York
Department number 20 with name Research is situated in Dallas
Department number 30 with name Sales is situated in Chicago
Department number 40 with name Operations is situated in Boston
```

2) Display ‘*”s before the employee name.

```
SELECT LPAD (ENAME, 9,’*’) FROM EMP;
LPAD (ENAM
---------
****SMITH
****ALLEN
*****WARD
****JONES
14 rows selected.
```

3) Display ‘*”s after the employee name.

```
 SELECT RPAD (ENAME, 9,’*’) FROM EMP;
 RPAD (ENAM
---------
 SMITH****
ALLEN****
FORD*****
MILLER***
14 rows selected.
```

4) Left trim of character ‘s’ from employee names of department number 20.

```
SQL> SELECT LTRIM(ENAME,'S') FROM EMP WHERE DEPTNO=20;
LTRIM(ENAM
------------------
MITH
JONES
```

5) Right trim of character ‘s’ from employee names of department number 20.

```
SQL> SELECT RTRIM(ENAME,'S') FROM EMP WHERE DEPTNO=20;
RTRIM(ENAM
----------
SMITH
JONE
SCOTT
ADAM
```

6) List employee names with all capital letters, with all small letters and with first letter only as capital of department number 10.

```
SQL>SELECT ENAME,UPPER(ENAME),LOWER(ENAME),INITCAP(ENAME) FROM EMP
        WHERE DEPTNO=10;
```

```
ENAME       UPPER(ENAM)          LOWER(ENAM       INITCAP(EN
----------  ---------- ----------       ------------------    --------------------
CLARK       CLARK                clark              Clark
KING        KING                 king               King
MILLER      MILLER               miller             Miller
```

7) List employee names with length of the name sorted on length for department number 30.

```
SQL>SELECT ENAME, LENGTH (ENAME) FROM EMP WHERE DEPTNO=30 ORDER BY
        LENGTH(ENAME);
ENAME     LENGTH(ENAME)
---------- ------------------------
WARD            4
ALLEN           5
BLAKE           5
JAMES           5
MARTIN          6
TURNER          6
6 rows selected.
```

8) Display the first 4 letters of job of EMP table.

```
SQL> SELECT DISTINCT(SUBSTR(JOB,1,4)) AS JOB FROM EMP;
JOB
----
CLER
SALE
```

```
MANA

ANAL

PRES

5 rows selected.
```

9) Display ename and return the position of character 'S' in ename.

```
SQL>SELECT ENAME,INSTR(ENAME,'S') FROM EMP WHERE DEPTNO=20;

ENAME      INSTR(ENAME,'S')

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

SMITH                  1

JONES                  5

SCOTT                  1
```

C) AIM: Queries using string functions

(To_number, LEAST, GREATEST, TRUNC)

1) Find the least value of the following series:

9,3,56,89,23,1,0,-2,12,34,9,22

```
SQL> SELECT LEAST(9,3,56,89,23,1,0,-2,12,34,9,22) AS LOWEST FROM DUAL;

   LOWEST

   ----------

        -2
```

2) Find the greatest value of the following series:

9,3,56,89,23,1,0,-2,12,34,9,22

```
SQL> SELECT GREATEST(9,3,56,89,23,1,0,-2,12,34,9,22) AS HIGHEST FROM DUAL;

HIGHEST

------------
```

```
        89
```

3)Trunk of the number 567.231656 by 3.

```
SQL> SELECT TRUNC(567.231656,3) FROM DUAL;

TRUNC(567.231656,3)
```

4)Add '100.00' to the salary of every employee in EMP table

```
SQL>SELECT ENAME,SAL+TO_NUMBER('100.00') AS SALARY FROM EMP;

ENAME        SALARY
----------   ----------
SMITH             900
ALLEN            1700
WARD             1350
```

d) AIM: Queries using date functions (Sysdate,next_day,add_months,last_day,months_between, least,greatest,trunk,round,to_char,to_date)

1)List employee names having an experience more than 24 years.

```
SQL>  SELECT ENAME,ROUND(MONTHS_BETWEEN(SYSDATE,HIREDATE)/12) EXP
FROM EMP WHERE ROUND(MONTHS_BETWEEN(SYSDATE,HIREDATE)/12)>24;

ENAME          EXP

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

SMITH          28

ALLEN          28

WARD           28
```

2)Find the first 'SUN'day of employees after join in the organization of EMP table.

```
SQL>SELECT NEXT_DAY (HIREDATE,'SUN') AS HOLIDAY FROM EMP;

HIREDATE        HOLIDAY
```

```
--------------      ---------------
17-DEC-80     21-DEC-80
20-FEB-81     22-FEB-81
22-FEB-81     01-MAR-81
02-APR-81     05-APR-81
```

3)Display hiredate and reviewdate from EMP table, consider reviewdate As 1year from the hiredate for the deptno ‘20’.

```
SQL> SELECT HIREDATE,ADD_MONTHS(HIREDATE,12) AS REVIEWDATE FROM EMP
WHERE DEPTNO=20;

HIREDATE      REVIEWDAT
-------------       ---------------
17-DEC-80     17-DEC-81
02-APR-81      02-APR-82
```

4)Display last day of joining month of employees of deptno ‘10’ from EMP table.

```
SQL> SELECT HIREDATE,LAST_DAY(HIREDATE) AS LASTDAY  FROM EMP WHERE
         DEPTNO=10;

HIREDATE   LASTDAY
-------------   -------------
09-JUN-81   30-JUN-81
17-NOV-81  30-NOV-81
23-JAN-82    31-JAN-82
```

e) AIM: Queries using conversion functions (to_char,to_number,to_date)
1) Display the names and hire dates of the employees of deptno 20. Format hire date as ‘12/03/84’.

```
SQL>SELECT ENAME, TO_CHAR (HIREDATE,’DD/MM/YY’) AS HIREDATE FROM EMP
WHERE DEPTNO=20;
```

```
ENAME     HIREDATE
----------------------------
SMITH      17/12/80
JONES      02/04/81
SCOTT      19/04/87
ADAMS      23/05/87
FORD       03/12/81
```

2) Display empno, employee name, job, salary of the employees. Show the salary with thousand separators.

```
SQL>SELECT EMPNO, ENAME,JOB,TO_CHAR(SAL,’$9,999’)AS SALARY FROM EMP;
    EMPNO     ENAME        JOB           SALARY
  ----------  ----------   ---------     ----------------
     7369     SMITH        CLERK         $800
     7499     ALLEN        SALESMAN      $1,600
```

3) List number of employees joined year wise.

```
SQL> SELECT TO_CHAR(HIREDATE,'YY')AS YY,COUNT(*) FROM EMP  GROUP BY
TO_CHAR(HIREDATE,'YY');
YY   COUNT(*)
-- ----------------
80       1
81      10
82       1
87       2
```

4) List employees who joined between Apr 81 and Apr 82.

```
SQL>  SELECT ENAME,TO_CHAR(HIREDATE,'MON YY') AS HIREDATE FROM EMP
WHERE TO_DATE(HIREDATE) BETWEEN TO_DATE('01-APR-81') AND
 TO_DATE('30-APR-82');
```

```
ENAME      HIREDATE
---------- ----------------
JONES         APR 81
MARTIN        SEP 81
```

2. Consider the following schema for a Library Database:

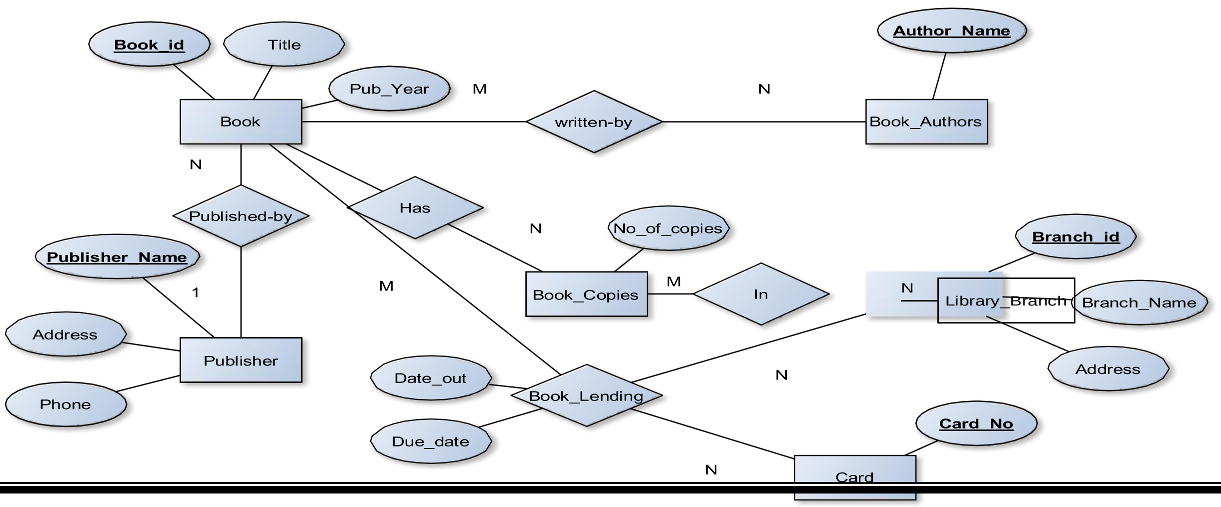
BOOK (*Book_id, Title, Publisher_Name, Pub_Year*)
BOOK_AUTHORS (Book_id, Author_*Name*) **PUBLISHER (*Name, Address, Phone*)**
BOOK_COPIES (*Book_id, Branch_id, No-of_Copies*)

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 Jun 2017**
3. **Delete a book in BOOK table. Update the contents of other tables to reflect this data manipulation operation.**
4. **Partition the BOOK table based on year of publication. Demonstrate its working with a simple query.**
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

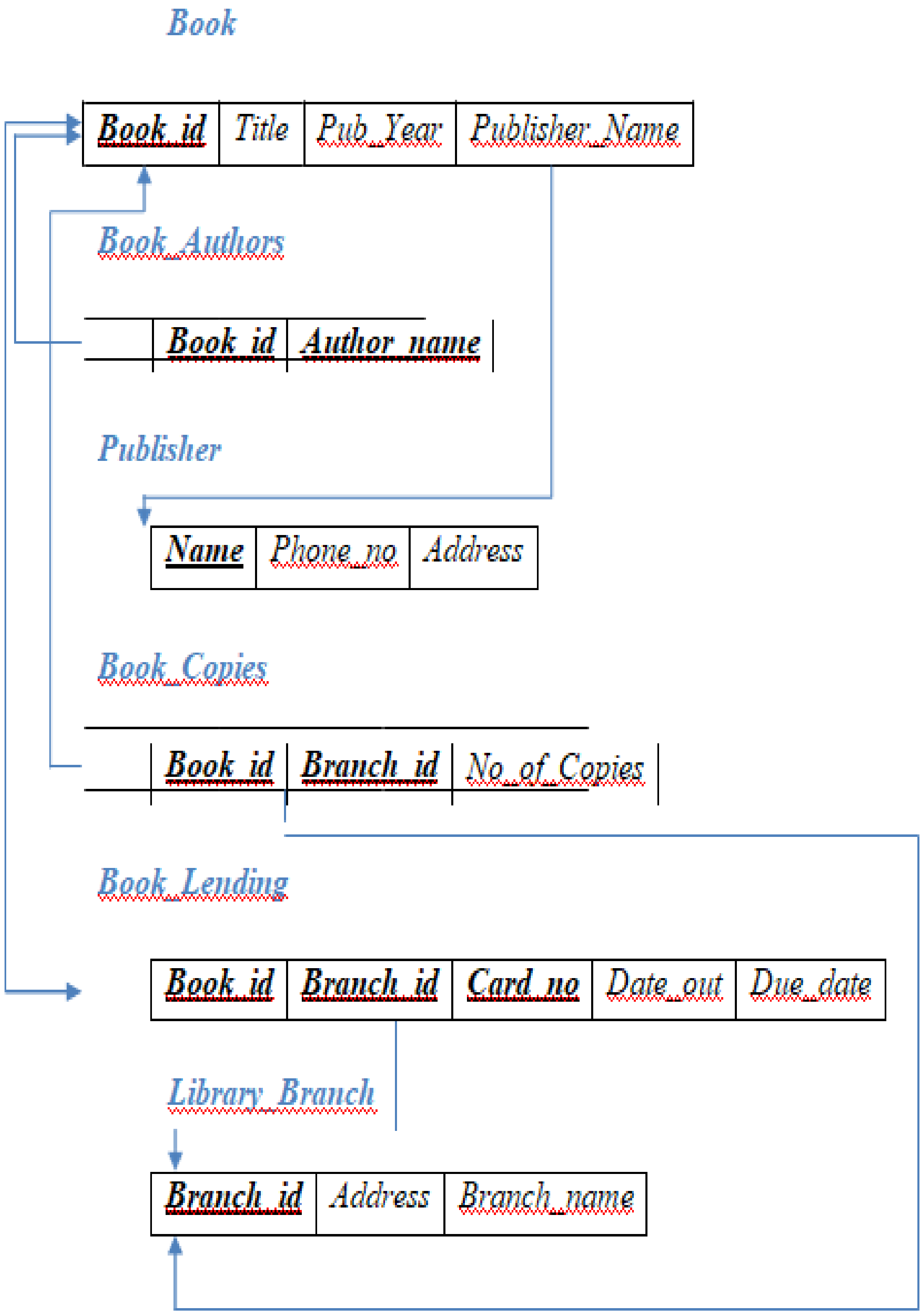


TABLE CREATION

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

CREATE TABLE BOOK
(BOOK_ID INTEGER
PRIMARY KEY, TITLE
VARCHAR2 (20),
PUB_YEAR VARCHAR2 (20),
PUBLISHER_NAME REFERENCES PUBLISHER (NAME) ON DELETE CASCADE);
CREATE TABLE
BOOK_AUTHORS
(AUTHOR_NAME
VARCHAR2 (20),
BOOK_ID REFERENCES BOOK (BOOK_ID) ON DELETE
CASCADE, PRIMARY KEY (BOOK_ID,
AUTHOR_NAME));

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

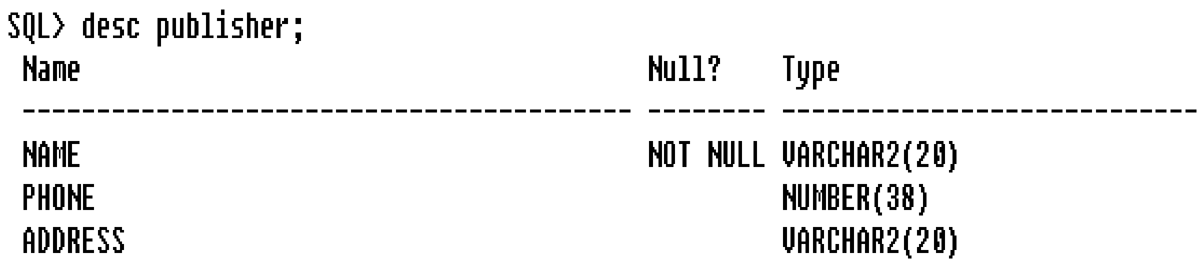
CREATE TABLE
BOOK_COPIES
(NO_OF_COPIES
INTEGER,
BOOK_ID REFERENCES BOOK (BOOK_ID) ON DELETE CASCADE,
BRANCH_ID REFERENCES LIBRARY_BRANCH (BRANCH_ID)
ON DELETE CASCADE,
PRIMARY KEY (BOOK_ID, BRANCH_ID));

CREATE TABLE CARD
(CARD_NO INTEGER PRIMARY KEY);

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

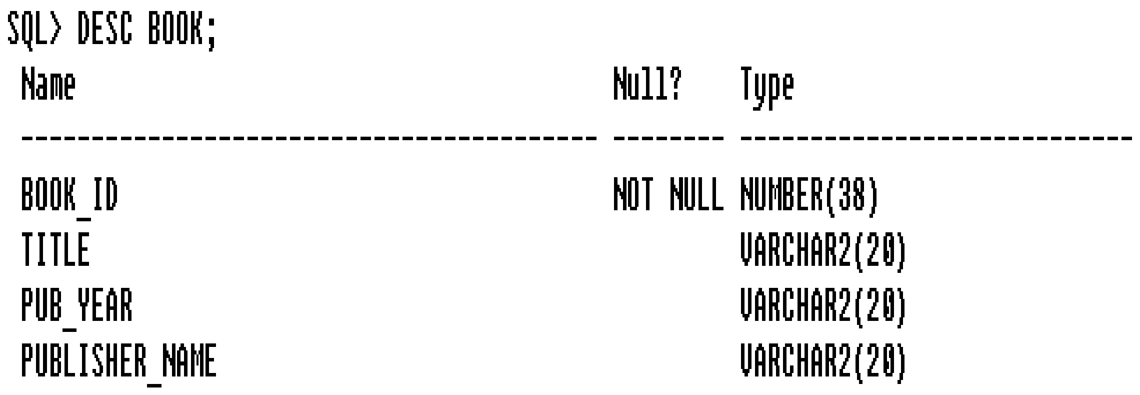
TABLE DESCRIPTIONS

DESC PUBLISHER;



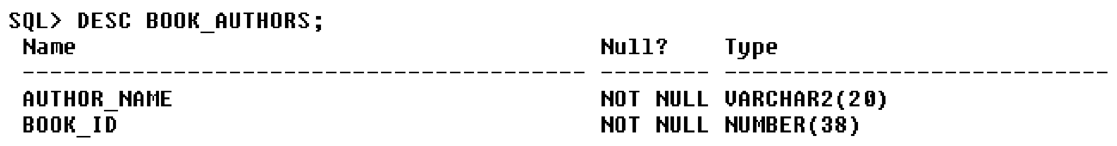
```
SQL> desc publisher;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 NAME                                      NOT NULL VARCHAR2(20)
 PHONE                                              NUMBER(38)
 ADDRESS 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

DESC BOOK;



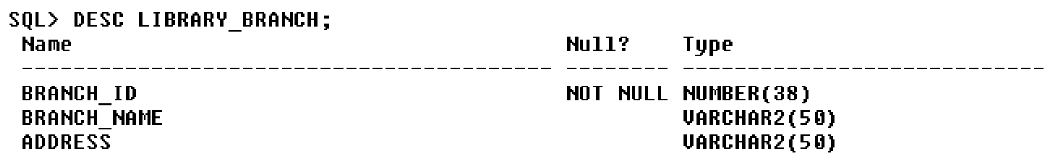
```
SQL> DESC BOOK;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 BOOK_ID                                   NOT NULL NUMBER(38)
 TITLE                                              VARCHAR2(20)
 PUB_YEAR     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

DESC BOOK_AUTHORS;



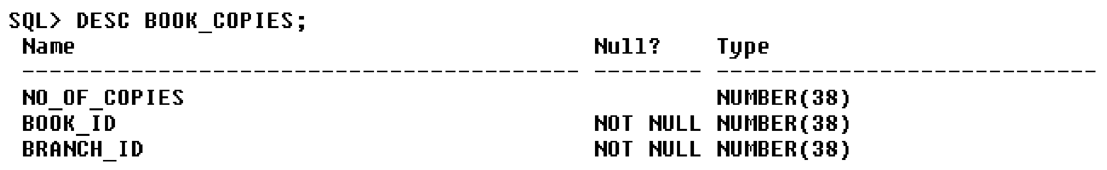
```
SQL> DESC BOOK_AUTHORS;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 AUTHOR_NAME                               NOT NULL VARCHAR2(20)
 BOOK_ID                                   NOT NULL NUMBER(38)
```

DESC LIBRARY_BRANCH;



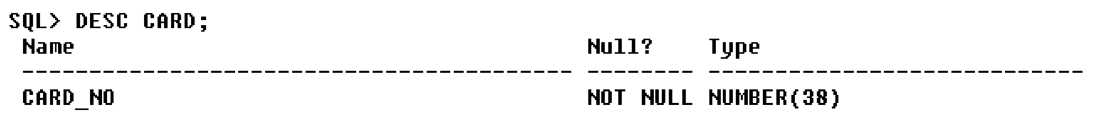
```
SQL> DESC LIBRARY_BRANCH;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 BRANCH_ID                                 NOT NULL NUMBER(38)
 BRANCH_NAME                                        VARCHAR2(50)
 ADDRESS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

DESC BOOK_COPIES;



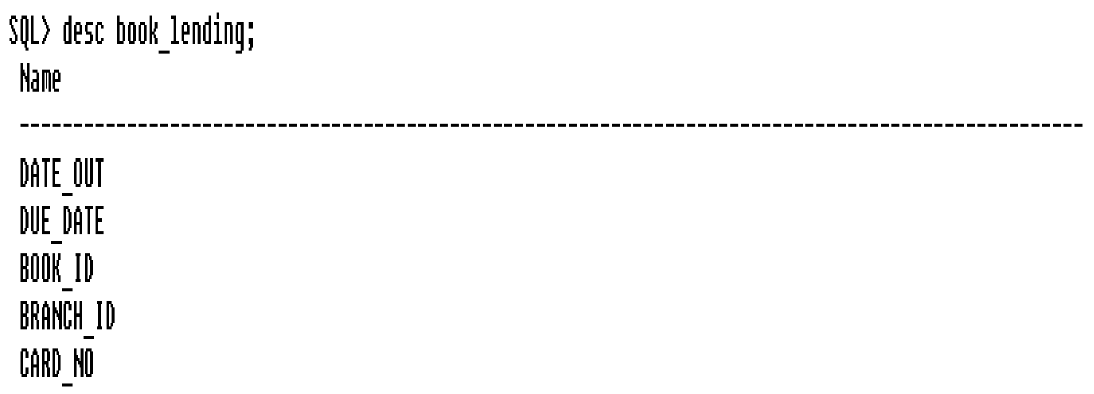
```
SQL> DESC BOOK_COPIES;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 NO_OF_COPIES                                       NUMBER(38)
 BOOK_ID                                   NOT NULL NUMBER(38)
 BRANCH_ID                                 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‘); INSERT INTO PUBLISHER VALUES (_GRUPO PLANETA‘, 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);

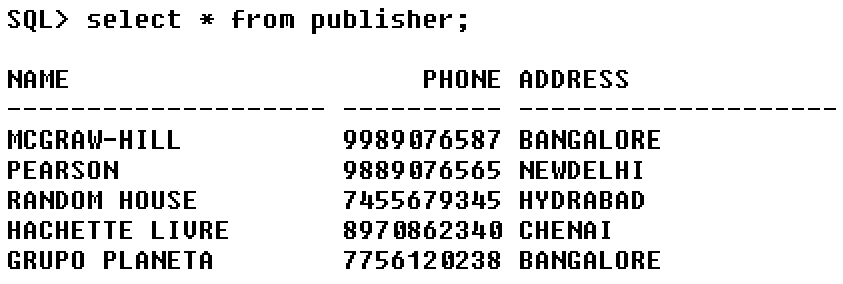
INSERT INTO LIBRARY_BRANCH VALUES (10,‘RR NAGAR‘,‘BANGALORE‘); INSERT INTO LIBRARY_BRANCH VALUES (11,‘RNSIT‘,‘BANGALORE‘);
INSERT INTO LIBRARY_BRANCH VALUES (12,‘RAJAJI NAGAR‘, ‘BANGALORE‘); INSERT INTO LIBRARY_BRANCH VALUES (13,‘NITTE‘,‘MANGALORE‘);
INSERT INTO LIBRARY_BRANCH VALUES (14,‘MANIPAL‘,‘UDUPI‘);

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);

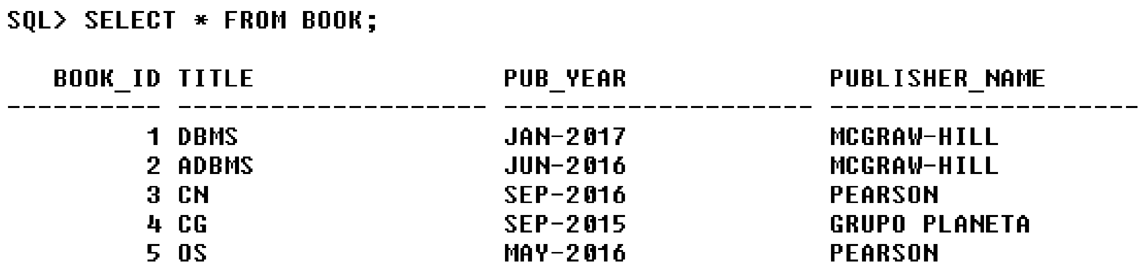
INSERT INTO BOOK_COPIES VALUES (3, 4, 11);

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);

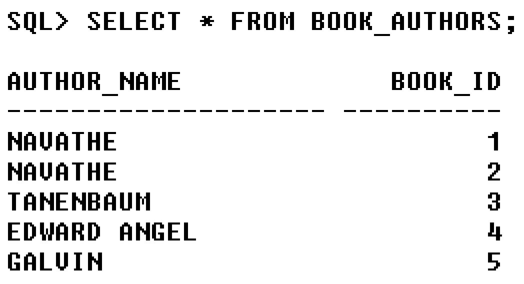
INSERT INTO BOOK_LENDING VALUES (‘01-JAN-17‘,‘01-JUN-17‘, 1, 10, 101);
INSERT INTO BOOK_LENDING VALUES (‘11-JAN-17‘,‘11-MAR-17‘, 3, 14, 101);
INSERT INTO BOOK_LENDING VALUES (‘21-FEB-17‘,‘21-APR-17‘, 2, 13, 101);
INSERT INTO BOOK_LENDING VALUES (‘15-MAR-17‘,‘15-JUL-17‘, 4, 11, 101);
INSERT INTO BOOK_LENDING 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
```

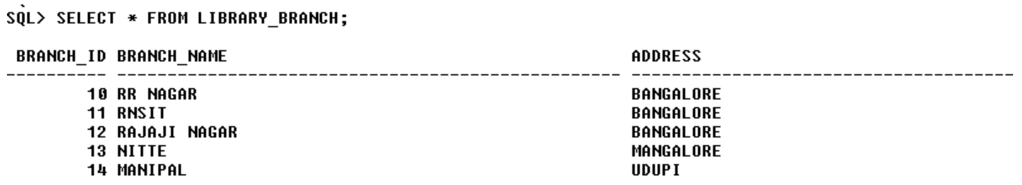


```
SQL> SELECT * FROM BOOK;

   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
```

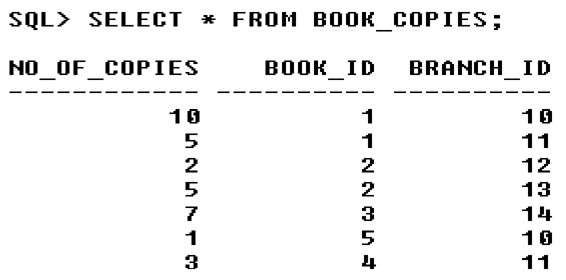


```
SQL> SELECT * FROM BOOK_AUTHORS;

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

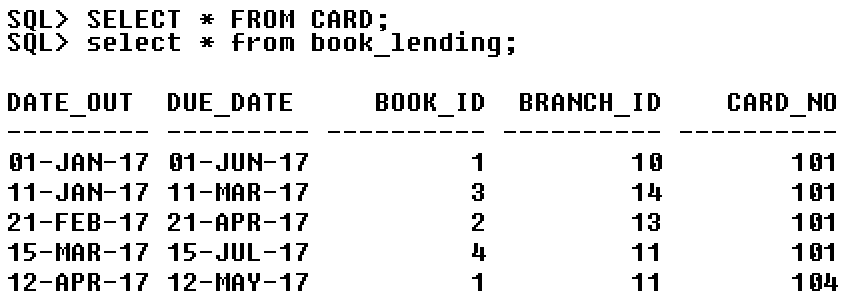
;



```
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
```



```
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
```

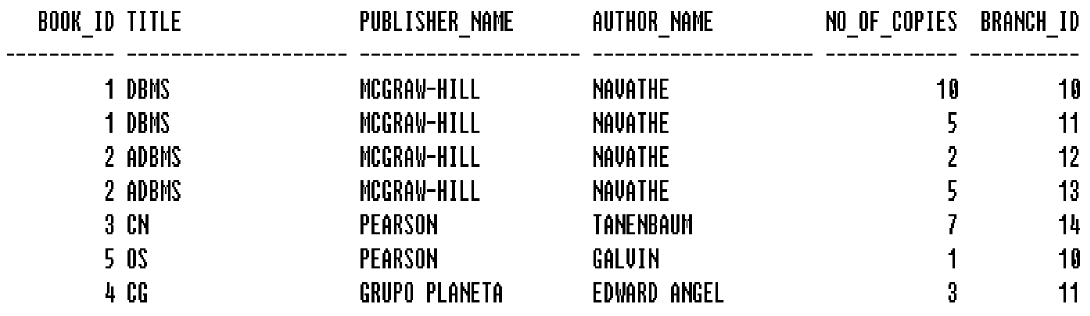


```
SQL> SELECT * FROM CARD;
SQL> select * from book_lending;

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.**

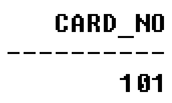
```
SELECT       B.BOOK_ID, B.TITLE, B.PUBLISHER_NAME,
      A.AUTHOR_NAME, C.NO_OF_COPIES, L.BRANCH_ID
      FROM BOOK B, BOOK_AUTHORS A, BOOK_COPIES C,
      LIBRARY_BRANCH L WHERE B.BOOK_ID=A.BOOK_ID
      AND B.BOOK_ID=C.BOOK_ID
      AND L.BRANCH_ID=C.BRANCH_ID;
```



```
   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
```

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

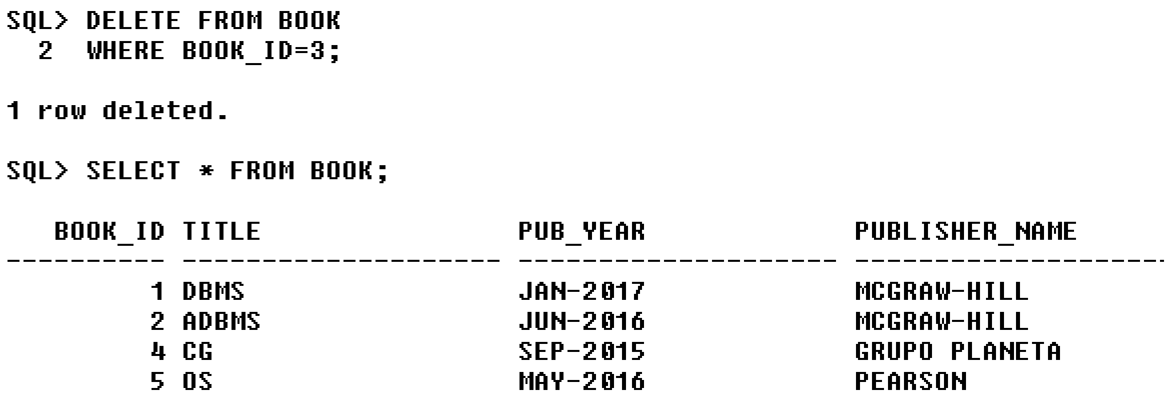
```
SELECT CARD_NO FROM BOOK_LENDING
WHERE DATE_OUT BETWEEN '01-JAN-2017' AND '01-JUL-2017' GROUP BY
CARD_NO
HAVING COUNT (*)>3;
```



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

B. Delete a book in BOOK table. Update the contents of other tables to reflect this data manipulation operation.

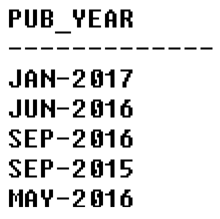
```
DELETE FROM BOOK WHERE BOOK_ID=3;
```



```
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
```

C. Partition the BOOK table based on year of publication. Demonstrate its working with a simple query.

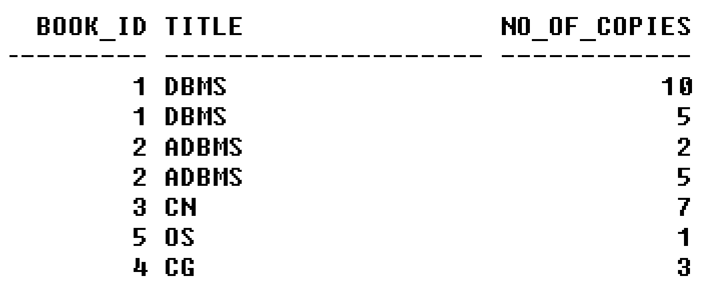
```
CREATE VIEW
V_PUBLICATION AS SELECT
PUB_YEAR
FROM BOOK;
```



```
PUB_YEAR
-------------
JAN-2017
JUN-2016
SEP-2016
SEP-2015
MAY-2016
```

D. 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
```

TEXT BOOKS / REFERENCE BOOKS:

T/R	BOOK TITLE/AUTHORS/PUBLICATION
T1	Database Management Systems, Raghurama Krishnan, Johannes Gehrke, Tata Mc Graw Hill, 3rd Edition
T2	Database System Concepts, Silberschatz, Korth, Mc Graw hill, 5th Edition
R1	Database Systems design, Implementation, and Management, Peter Rob & Carlos Coronel 7th Edition.
R2	Fundamentals of Database Systems, Elmasri Navrate Pearson Education
R3	Introduction to Database Systems, C.J.Date Pearson Education
R4	Oracle for Professionals, The X Team, S.Shah and V. Shah, SPD.
R5	Database Systems Using Oracle: A Simplified guide to SQL and PL/SQL,Shah,PHI.