# 

# Tribhuvan University

# Faculty of Humanities and Social Sciences

DATABASE LABSHEETS

A PROJECT REPORT

Submitted to

Department of Computer Application

Shahid Smarak College

*In partial fulfillment of the requirements for the Bachelors in Computer Application*

Submitted by: -

Amir Maharjan

**Internal supervisor**

# Ramu Khadkathoki

# External Supervisor

**Table of content**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **SN** | **Lab experiment**  **Unit** | **Title of lab experiment** | **Date** | **Page**  **No** | **Signature** | **Remarks** |
| **1.** | **DDL & DML** | **Problem**  **1 – 20** | **2023-05-25** | **1 - 13** |  |  |
| **2.** | **DDL & DML** | **Problem**  **20 – 40** | **2023-05-26** | **13- 23** |  |  |
| **3.** | **DDL & DML** | **Problem**  **20 - 60** | **2023-05-27** | **23 – 37** |  |  |
| **4.** | **Views** | **Problem 61 – 70** | **2023-05-28** | **38 – 41** |  |  |
| **5.** | **Stored Procedures** | **Problem 71 – 75** | **2023-05-28** | **42 – 49** |  |  |
| **6.** | **Triggers** | **Problem 75 - 80** | **2023-05-29** | **50 - 53** |  |  |

# Chapter 1: DDL & DML

# Problem - 1

Write SQL Query to create following table (Student).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Fileds | Datatype | Null | Key | Default | Check | Extra |
| student\_id | int(11) | No | Primary |  |  | Auto\_Increment |
| name | varchar(50) | No |  |  |  | Unique |
| address | varchar(100) | No |  | Birtamode |  |  |
| class\_id | int(11) | No | Foreign |  |  |  |
| section | varchar(50) | Yes |  |  |  |  |
| Age | Int(11) | No |  | 16 | Age>=15 |  |

Note: Foreign key references to (Class) Table.

# Objective

To create a table with primary key, foreign key and default constraints.

# Requirement

* Hardware
* Processor: i5 12 Gen
* Memory: 8 GB RAM
* Storage: 500 GB HDD and 256 SSD
* Input Devices: Keyboard and Mouse
* Output Deices: Monitor and Printer
* Software
* Operating System: Windows 10
* Application: Oracle
* Virtual Machine: N / A
* Extension: N / A
* Utilities: N / A

# Procedure

**Creating Classroom table**

CREATE TABLE Classroom(

class\_id number(10) not null primary key,

class\_floor number(10) not null,

class\_teacher varchar(255) not null

);

**Creating Student Table**

CREATE TABLE STUDENT (

student\_id number(11) not null primary key,

name varchar(50) not null unique,

address varchar(100) DEFAULT 'Birtamode' not null,

class\_id number(11) not null,

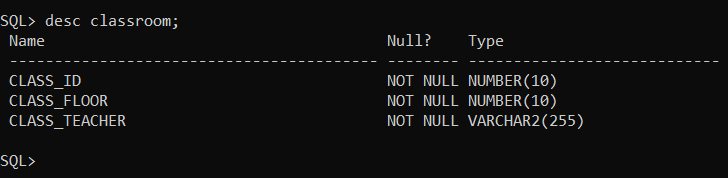
section varchar(50),

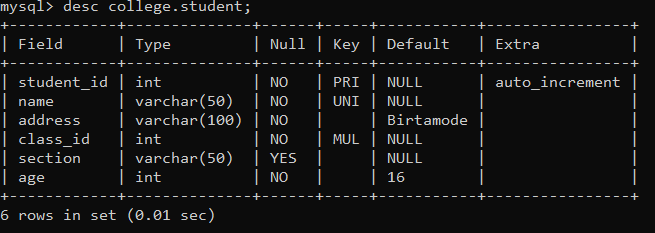
age number(11) DEFAULT '16' CHECK( age>=16 ) not null,

foreign key ( class\_id ) references classroom( class\_id )

);

# Result





# Problem - 2

# Write SQL query to drop primary key from above table.

# Objective

# To drop primary key from Student table

# Procedure

# ALTER TABLE student

# DROP PRIMARY KEY;

# Result

# 

# Problem - 3

# Write SQL query to drop foreign key from above table

# Objective

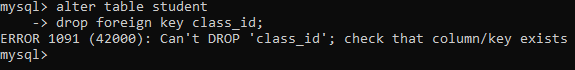
# To drop foreign key from Student Table

# Procedure

# ALTER TABLE student

# DROP FOREIGN KEY class\_id;

# Result



# Problem - 4

Write SQL query to set student id as primary key.

# Objective

# To set student id as primary key in student table.

# Procedure

# ALTER TABLE student

# ADD PRIMARY KEY ( student\_id );

# Result

# 

# Problem - 5

# Write SQL query to set class id as foreign key

# Objective

# To set class id as foreign key in student table.

# Procedure

# ALTER TABLE student

# ADD FOREIGN KEY ( class\_id ) REFERENCES classroom( class\_id );

# Result

# 

# Problem - 6

# Write SQL query to remove unique constraint from name.

# Objective

# To remove unique constraint from name in student table.

# Procedure

# ALTER TABLE student

# DROP CHECK name;

# Result

# 

# Problem - 7

# Write SQL query to remove default constraint from age.

# Objective

# To remove default constraint from student table.

# Procedure

# ALTER TABLE student

# ALTER city DROP DEFAULT

# Result

# 

# Problem - 8

# Write SQL query to add unique constraint to section.

# Objective

# To add unique constraint in student table.

# Procedure

# ALTER TABEL student

# ADD UNIQUE ( section )

# Result

# 

# Problem - 9

# Write SQL query to add default value 18 to age.

# Objective

# To add default value 18 to age in student table.

# Procedure

# ALTER TABLE student

# ALTER AGE SET DEFUALT ‘18’

# Result

# 

# Problem - 10

# Write SQL query to change column name address to location.

# Objective

# To change column name address to location in student table.

# Procedure

# ALTER TABLE student

# RENAME COLUMN address TO location;

# Result

# 

# Problem - 11

# Write SQL query to add new column email and make it not null.

# Objective

# To add new column email (not null) in student table.

# Procedure

# ALTER TABLE student

# ADD email varchar( 100 ) not null;

# Result

# 

# Problem - 12

# Write SQL query to remove column section from above table.

# Objective

# To remove column section from student table.

# Procedure

# ALTER TABLE student

# DROP COLUMN section

# Result

# 

# Problem - 13

# Write SQL query to add new column contact and make data type as integer 14.

# Objective

# To add new column contact ( integer 14 ) in student table.

# Procedure

# ALTER TABLE student

# ADD contact int( 14 );

# Result

# 

# Problem - 14

# Write SQL query to change data type of column contact to varchar and make it unique.

# Objective

# To modify column contact to varchar and make it unique.

# Procedure

# ALTER TABLE student

# MODIFY contact varchar( 200 ) unique;

# Result

# 

# Problem - 15

# Write SQL query to change default value of address to Kathmandu.

# Objective

# To change default value of location(address) to Kathmandu.

# Procedure

# ALTER TABLE student

# ALTER location SET DEFAULT ‘Kathmandu’;

# Result

# 

# Problem - 16

# Insert five set of records in above table.

# Objective

# To insert five records in student table.

# Procedure

# INSERT INTO student ( value1, value2 … value n) VALUES ( data1, data2 … data n );

# Result

# 

# Problem - 17

# Write SQL query to update name and address of student whose student\_id is 5.

# Objective

# To update name and address (location) of student whose student\_id is 5.

# Procedure

# UPDATE student

# SET name = ‘Ajesh’, location = ‘Kirtipur’

# WHERE student\_id = 5;

# Result

# 

# Problem - 18

# Write SQL query to delete all the records of student have age greater than 20.

# Objective

# To delete all records of student that have age greater than 20.

# Procedure

# DELETE FROM student WHERE age > 20

# Result

# 

# Problem - 19

# Write SQL query to update age of student having address kirtipur.

# Objective

# To update age of students have address(location) kirtipur.

# Procedure

# UPDATE student

# Set age = 22

# WHERE location = kirtipur;

# Result

# 

# Problem - 20

# Write SQL query to delete all records of student have student\_id 1.

# Objective

# To delete all records of students of having student\_id 1.

# Procedure

# DELETE FROM student WHERE student\_id = 1;

# Result

# 

# Problem - 21

# Write SQL query to select all records of student.

# Objective

# To select all records of student.

# Procedure

# SELECT \* FROM student;

# Result

# 

# Problem - 22

# Write SQL query to select all records of student having student id 3.

# Objective

# To select all records of students having student id 3.

# Procedure

# SELECT \* FROM student WHERE student\_id = 3;

# Result

# 

# Problem - 23

# Write SQL query to select name and address of students whose age is greater than 21.

# Objective

# To select name and address(location) of students whose age is greater than 21.

# Procedure

# SELECT name, location FROM student WHERE age > 21.

# Result

# 

# Problem - 24

# Write SQL query to select student\_id and name of students whose address in Birtamode.

# Objective

# To select student\_id and name of students have address in Birtamode.

# Procedure

# SELECT student\_id, name FROM student WHERE location = ‘Birtamode’;

# Result

# 

# Problem - 25

# Write SQL query to select records of students whose class\_id is 3 and address is Kathmandu.

# Objective

# To select records of students whose class\_id is 3 and address(location) is Kathmandu

# Procedure

# SELECT \* FROM student WHERE class\_id = 3 AND location = ‘Kathmandu’;

# Result

# 

# Problem - 26

# Write SQL query to select maximum age from above table.

# Objective

# To select maximum age from student table.

# Procedure

# SELECT MIN(age) FROM student;

# Result

# 

# Problem - 27

# Write SQL Query to select minimum age of students whose address(location) is Kathmandu.

# Objective

# To select minimum age of students whose location is Kathmandu.

# Procedure

# SELECT MIN(age) FROM student WHERE location = ‘Kathmandu’;

# Result

# 

# Problem - 28

# Write SQL query to find total number of students having class\_id 3 and age greater than 19.

# Objective

# To select total number of student having class\_id 3 and age greater than 19.

# Procedure

# SELECT COUNT (student\_id) FROM student WHERE class\_id = 3 AND age > 19

# Result

# 

# Problem - 29

# Write SQL query to find average age of students whose class\_id is 4 and section is B.

# Objective

# To find average age of students whose class\_id is 4 and section is B.

# Procedure

# SELECT AVG(age) FROM student WHERE class\_id = 3 AND section = “B”

# Result

# 

# Problem - 30

# Write SQL query to select students whose address starts with letter “B”.

# Objective

# To select students whose address starts with letter “B”.

# Procedure

# SELECT \* FROM students WHERE location LIKE ‘B%’;

# Result

# 

# Problem - 31

# Write SQL query to count those students whose name ends with letter “R”.

# Objective

# To count those students whose name ends with letter “R”.

# Procedure

# SELECT COUNT(name) FROM student WHERE name LIKE ‘%r’;

# Result

# 

# Problem - 32

# Write SQL query to select name and age of students whose address is Kathmandu or Kirtipur.

# Objective

# To select name and age of students whose address is Kathmandu or Kirtipur

# Procedure

# SELECT \* FROM student where location = ‘Kirtipur’ OR location = ‘Kirtipur’.

# Result

# 

# Problem - 33

# Write SQL query to select sum of age of students having id 1,2, and 3.

# Objective

# To sum of age of students having id 1,2, and 3.

# Procedure

# SELECT SUM (age) FROM student where student\_id = 1 or student\_id = 2 or student\_id = 3;

# Result

# 

# Problem - 34

# Write SQL query to select students whose age is between 18 and 22.

# Objective

# To select students whose age is between 18 and 22.

# Procedure

# SELECT \* FROM student WHERE age BETWEEN 18 AND 22;

# Result

# 

# Problem - 35

# Write SQL query to select total students of each age group.

# Objective

# To select total students of each age group.

# Procedure

# SELECT COUNT(student\_id) from student group by age;

# Result

# 

# Problem - 36

# Write SQL query to select class\_id, name and maximum age of students studying in each class.

# Objective

# To select class\_id, name and maximum age of students studying in each class.

# Procedure

# SELECT class\_id, name, MAX(age) from student group by class\_id;

# Result

# Problem - 37

# Write SQL query to select student’s records by arranging in descending order on the basis of id.

# Objective

# To select student’s records by arranging in descending order on the basis of id.

# Procedure

# SELECT \* FROM student order by student\_id DESC;

# Result

# 

# Problem - 38

# Write SQL query to select student\_id and name of students whose age is greater than 20 after arranging records in alphabetical order on the basis of name.

# Objective

# To select student\_id and name of students whose age is greater than 20 after arranging records in alphabetical order on the basis of name.

# Procedure

# SELECT student\_id, name FROM student WHERE age > 20 ORDER BY name ASC

# Result

# 

# Problem - 39

# Write SQL query to select records of student whose age is maximum among all the students.

# Objective

# To select records of student whose age is maximum among all the students.

# Procedure

# SELECT MAX (age) FROM student;

# Result

# 

# Problem - 40

# Write SQL query to select student\_id and name of student whose student\_id is maximum among all the students.

# Objective

# To select student\_id and name of student whose student\_id is maximum among all the students.

# Procedure

# SELECT student\_id, name FROM student WHERE student\_id = (SELECT MAX (student\_id) from student);

# Result

# 

# Problem - 41

# Write SQL query to select name and age of student whose age is minimum than the average age of all student.

# Objective

# To select name and age of student whose age is minimum than the average age of all student.

# Procedure

# SELECT name, age FROM student WHERE age < (SELECT AVG (age) FROM student);

# Result

# 

# Problem - 42

# Write SQL query to list all the students except ‘birtamode’ and ‘kathmandu’ in asc order of age.

# Objective

# To list all the students except ‘birtamode’ and ‘kathmandu’ in asc order of age.

# Procedure

# SELECT \* FROM student EXCEPT SELECT \* FROM student WHERE location IN ( ‘Birtamode’, ‘Kathmandu’ );

# Result

# 

# Problem - 43

# Write SQL query to select students who do not belong to location ‘Kathmandu’

# Objective

# To select students who do not belong to location ‘Kathmandu’

# Procedure

# SELECT name, age FROM student EXCEPT SELECT \* FROM student WHERE location = ‘Kathmandu’;

# Result

# 

# Problem - 44

# Write SQL query to display the location of ‘Jess’.

# Objective

# To display the location of ‘Jess’.

# Procedure

# SELECT location FROM student WHERE name = ‘Jess’;

# Result

# 

# Problem - 45

# Write SQL query to display the total information of student table along with name and location of all the students having address ‘Birtamode’.

# Objective

# To display the total information of student table along with name and location of all the students having address(location) ‘Birtamode’.

# Procedure

# SELECT name, age FROM student EXCEPT SELECT \* FROM student WHERE location = ‘Kathmandu’;

# Result

# 

# Problem - 46

# Create table below with appropriate data type and constraints.

Employee

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Emp\_Id | Name | Address | Salary | Dept\_Id |

Department

|  |  |  |
| --- | --- | --- |
| Dept\_Id | Dept\_Name | Floor |

# Objective

# To create the above tables.

# Procedure

# Employee Table

# CREATE TABLE employee (

# Emp\_id int (11) not null primary key,

# Name varchar (255) not null,

# Address varchar (255) not null DEFAULT ‘Birtamode’,

# Salary int (11) not null DEFAULT ‘8000’,

# dept\_id int (11) not null,

# foreign key (dept\_id) references department(dept\_id)

# );

# Result

# 

# Department Table

# CREATE TABLE department (

# Dept\_id int (11) not null primary key,

# Dept\_name varchar (255) not null,

# Floor int (11) not null,

# );

# Result

# 

# Problem - 47

# Use all types of joins to select employee id, name and department name of employees.

# Objective

# To use all types of joins.

# Procedure

# Inner Join

# SELECT \* FROM employee INNER JOIN department ON employee.dept\_id = department.dept\_id

# Left Join

# SELECT \* FROM employee LEFT JOIN department ON employee.dept\_id = department.dept\_id

# Right Join

# SELECT \* FROM employee RIGHT JOIN department ON employee.dept\_id = department.dept\_id

# Cross Join

# SELECT \* FROM employee CROSS JOIN department;

# Result

# 

# 

# 

# 

# Problem - 48

# Select name and address of employees whose salary is between 10000 and 20000

# Objective

# To select name and address of employees whose salary is between 10000 and 20000

# Procedure

# SELECT name, address FROM student WHERE salary BETWEEN 10000 AND 20000;

# Result

# 

# Problem - 49

# Select employee id, employee name and department name of employees working in first floor.

# Objective

# To Select employee id, employee name and department name of employees working in first floor.

# Procedure

# SELECT employee\_id, name and department name FROM employee INNER JOIN department ON employee.dept\_id = department.dept\_id WHERE floor = ‘first’;

# Result

# 

# Problem - 50

# Select all records of department which are in second floor.

# Objective

# To select all records of department which are in second floor.

# Procedure

# SELECT \* FROM department WHERE floor = ‘second’;

# Result

# 

# Problem - 51

# Select name, address and department name of employees which are from Birtamode

# Objective

# To select name, address and department name of employees which are from Birtamode

# Procedure

# SELECT name, address, dept\_name FROM employee INNER JOIN department ON employee.dept\_id = department.dept\_id WHERE address= ‘Birtamode’;

# Result

# 

# Problem - 52

# Select employee\_id and name of employees having salary more than 10000 and from Kathmandu.

# Objective

# To select employee\_id and name of employees having salary more than 10000 and from Kathmandu.

# Procedure

# SELECT employee\_id, name FROM employee WHERE salary > 10000 AND address = ‘Kathmandu’;

# Result

# 

# Problem - 53

# Select name, department name and floor of employee whose name start with letter ‘R’ and age is greater than 30.

# Objective

# To select name, department name and floor of employee whose name start with letter ‘R’ and age is greater than 30.

# Procedure

# SELECT name, dept\_name, floor FROM employee INNER JOIN department ON employee.dept\_id = department.dept\_id WHERE name LIKE ‘R%’ AND age > 30;

# Result

# 

# Problem - 54

# Select employee\_id and department name of employees whose floor is ‘first’ by arranging in ascending order on the basis of salary.

# Objective

# To employee\_id and department name of employees whose floor is ‘first’ by arranging in ascending order on the basis of salary.

# Procedure

# SELECT emp\_id, dept\_name FROM employee INNER JOIN department on employee.dept\_id = department.dept\_id WHERE floor = ‘first’ ORDER BY emp\_id, dept\_name ASC;

# Result

# 

# Problem - 55

# Select total number of employees working in each department

# Objective

# To select total number of employees working in each department.

# Procedure

# SELECT COUNT(name), dept\_name FROM employee INNER JOIN department ON employee.dept\_id = department.dept\_id GROUP BY dept\_id;

# Result

# 

# Problem - 56

# Select maximum salary of employee working in each floor and whose department is ‘Finance’.

# Objective

# To select maximum salary of employee working in each floor and whose department is ‘Finance’.

# Procedure

# SELECT MAX(salary), dept\_name FROM employee INNER JOIN department ON employee.dept\_id = department.dept\_id WHERE dept\_name = ‘Finance’ GROUP BY dept\_name;

# Result

# 

# Problem - 57

# Select name and department name of employees whose salary is greater than average salary of all employees.

# Objective

# To select name and department name of employees whose salary is greater than average salary of all employees.

# Procedure

# SELECT name, dept\_name FROM employee INNER JOIN department ON employee.dept\_id = department.dept\_id WHERE salary > (SELECT AVG(salary) FROM employee);

# Result

# 

# Problem - 58

# Select name and address of employee whose salary is between 20000 and 30000 and floor is ‘second’.

# Objective

# To select name and address of employee whose salary is between 20000 and 30000 and floor is ‘second’.

# Procedure

# SELECT name, address FROM employee INNER JOIN department on employee.dept\_id = department.dept\_id WHERE salary BETWEEN 20000 AND 30000 AND floor = ‘second’;

# Result

# 

# Problem - 59

# Select name and department name employee whose age is minimum.

# Objective

# To select name and department name employee whose age is minimum.

# Procedure

# SELECT name, dept\_name FROM employee INNER JOIN department ON employee.dept\_id = department.dept\_id WHERE (SELECT MIN(age) FROM employee);

# Result

# 

# Problem - 60

# Select sum of salary of all employees whose name ends with letter ‘s’ and department is ‘Account’

# Objective

# To select sum of salary of all employees whose name ends with letter ‘s’ and department is ‘Account’

# Procedure

# SELECT SUM(salary) from employee INNER JOIN department ON employee.dept\_id = department.dept\_id WHERE name LIKE ‘%s’ AND dept\_name = ‘Account’;

# Result

# 

# Conclusion

# DDL are processes that define the data itself. CREATE, ALTER, DROP, TRUNCATE, etc. fall under the DDL commands. is used to create or modify the database objects like tables, views etc. These commands deal with defining a schema and adding a table description to it.

# DML are the processes that helps to manipulate the data that are defined via the DDL.SELECT, UPDATE, DELETE, etc. fall under the DML commands. is responsible for performing CRUD (Create Read Update and Delete) operations on the data of the database table. Data Manipulation Language (DML) allows you to modify the database tables by inserting, modifying, and deleting their data.

Therefore, the things I learned during this lab sheet are: creation of database and table, use of keys such as primary key and foreign key, other DDL commands like altering a table, dropping a table and truncating a table, selection of records, deletion of records, updating of records or table, clauses such as where, order by, group by, join, etc. were learned during DML.

# Lab Sheet – 2

# Chapter 2: View

# Problem - 61

Consider the following table,

**Employee**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Emp\_Id | Name | Address | Salary | Dept\_Id |

Create a view named Birtamode Employees that shows Employee id, name and address of employees having address Birtamode.

# Objective

# To Create a view named Birtamode Employees that shows Employee id, name and address of employees having address Birtamode

# Procedure

# CREATE VIEW Birtamode AS

# SELECT emp\_id, name, address FROM employee

# WHERE address = ‘Birtamode’;

# SELECT \* FROM Birtamode;

# Result

# 

# Problem - 62

# Create a view named ‘Our Salary’ that shows employee name and salary whose salary is greater than 20000.

# Objective

# To create a view named ‘Our Salary’ that shows employee name and salary whose salary is greater than 20000.

# Procedure

# CREATE VIEW [Our Salary] AS

# SELECT name, salary FROM employee WHERE salary > 20000;

# SELECT \* FROM [Our Salary];

# Result

# 

# Problem - 63

# Write SQL query to drop view Our salary.

# Objective

# To drop view Our salary.

# Procedure

# DROP VIEW Salary.

# Result

# 

# Problem - 64

# Write SQL query to add column salary on view Birtamode Employees.

# Objective

# To add column salary on view Birtamode.

# Procedure

# CREATE OR REPLACE VIEW Birtamode AS

# SELECT emp\_id, name, address, salary FROM employee WHERE address = ‘Birtamode’;

# SELECT \* FROM Birtamode;

# Result

# 

# Problem - 65

# Write SQL query to drop view Our salary.

# Objective

# To drop view Our salary.

# Procedure

# DROP VIEW Salary.

# Result

# 

# Conclusion

# Therefore, views in SQL are kind of virtual tables. A view also has rows and columns as they are in a real table in the database. Views are generally used to focus, simplify, and customize the perception each user has of the database. Views can be used as security mechanisms by letting users access data through the view, without granting the users permissions to directly access the underlying base tables of the view.

In this unit, I learned how to create a view, drop a view, select records from the view, select records with the use of clauses such as where, order by, etc.

# Chapter 3: Stored Procedure

# Problem - 66

# Write SQL query to create stored procedure named SelectRecords that selects all recrods from employees table.

# Objective

# To create stored procedure named SelectRecords that selects all recrods from employees table.

# Procedure

# DELIMITER && CREATE PROCEDURE SelectRecords () BEGIN

# SELECT \* FROM employee; END &&

# CALL SelectRecords;

# Result

# 

# Problem - 67

# Write SQL query to create stored procedure named MyEmployees that select employees records of a particular address.

# Objective

# To create stored procedure named MyEmployees that select employees records of a particular address.

# Procedure

# DELIMITER && CREATE PROCEDURE MyEmployees() BEGIN

# SELECT \* FROM employee WHERE address = ‘Kathmandu’; END &&

# CALL MyEmployees;

# Result

# 

# Problem - 68

# Write SQL query to create stored procedure named MyEmployees that select employees records of a particular address and department id.

# Objective

# To create stored procedure named MyEmployees that select employees records of a particular address and department id.

# Procedure

# DELIMITER && CREATE PROCEDURE MyEmployees1() BEGIN

# SELECT \* FROM employee WHERE address = ‘Kathmandu’ AND dept\_id = 1; END &&

# CALL MyEmployees1;

# Result

# 

# Problem - 69

# Write SQL query to drop above stored procedure SelectRecords.

# Objective

# To drop above stored procedure SelectRecords.

# Procedure

# DROP PROCEDURE SelectRecords;

# Result

# 

# 

# Problem - 70

# Write SQL query to insert records in above table using stored procedures.

# Objective

# To insert records in above table using stored procedures.

# Procedure

# DELIMITER &&

# CREATE PROCEDURE MyEmployees1(

# In emp\_id int(11),

# In name varchar(255),

# In address varchar(255),

# In salary int(11),

# In dept\_id int(11),

# In age int(11) )

# BEGIN

# INSERT INTO employee VALUES (emp\_id, name, address, salary, dept\_id, age );

# END &&

# CALL MyEmployees1 (8, ‘Razer Mantis’, ‘Kirtirpur, 100000, 3, 50);

# SELECT \* FROM employee;

# Result

# 

# Problem - 71

# Write SQL query to delete record of employee whose employee id is given in parameter using stored procedure

# Objective

# To delete record of employee whose employee id is given in parameter using stored procedure

# Procedure

# DELIMITER &&

# CREATE PROCEDURE DeleteRecords (

# In id int(11) )

# BEGIN

# DELETE FROM employee WHERE emp\_id = id;

# END &&

# CALL DeleteRecords ( 5 );

# CALL SelectRecords; OR SELECT \* FROM employee;

# Result

# 

# Problem - 72

# Write SQL query to update name and address of employee on the basis of salary. (Here name, address, and salary is given in parameter).

# Objective

# To update name and address of employee on the basis of salary.

# Procedure

# DELIMITER &&

# CREATE PROCEDURE UpdateRecords (

# In name int(11),

# In address varchar(255),

# In salary int(11) )

# BEGIN

# UPDATE employee SET name = name, addresss = address WHERE salary = salary;

# END &&

# CALL UpdateRecords (‘Pringles’, ‘Kirtipur’, 30000);

# CALL SelectRecords; OR SELECT \* FROM employee;

# Result

# 

# Problem - 73

# Write SQL query to display maximum salary returned by stored procedure

# Objective

# To display maximum salary returned by stored procedure

# Procedure

# DELIMITER &&

# CREATE PROCEDURE MaxSalary()

# BEGIN

# SELECT MAX (salary) FROM employee;

# END &&

# CALL MaxSalary();

# Result

# 

# Problem - 74

# Write SQL query to display average salary of employees returned by stored procedures

# Objective

# To display average salary of employees returned by stored procedures

# Procedure

# DELIMITER &&

# CREATE PROCEDURE AverageSalary()

# BEGIN

# SELECT AVG (salary) FROM employee;

# END &&

# CALL AverageSalary ();

# Result

# 

# Conclusion

# Therefore, 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 (RDBMS) as a group, so it can be reused and shared by multiple programs. Stored procedures help group one or multiple SQL statements for reuse under a common name, encapsulating common business logic within the database itself.

# In this unit, I learned that stored procedures are very much similar to function from programming languages. New keywords such as delimiter, begin, end, etc. were learned.

# Chapter 4: Trigger & Index

# Problem - 75

# Write SQL query to create a trigger named MyTrigger.

# Objective

# To create a trigger named MyTrigger.

# Procedure

# CREATE TRIGGER MyTrigger BEFORE INSERT ON employee FOR EACH ROW set new.total = new.salary = new.dept\_id;

# Result

# 

# Problem - 76

# Write SQL query to drop above trigger.

# Objective

# To drop trigger mytrigger.

# Procedure

# DROP TRIGGER mytrigger;

# Result

# 

# 

# Problem - 77

# Write SQL query to create index named MyIndex on name of employee.

# Objective

# To create index named MyIndex on name of employee.

# Procedure

# CREATE INDEX MyIndex ON employee (name);

# Result

# 

# Problem - 78

# Write SQL query to create index named MyIndex1 on address and salary of employee.

# Objective

# To create index named MyIndex1 on address and salary of employee.

# Procedure

# CREATE INDEX MyIndex ON employee (name);

# Result

# 

# Problem - 79

# Write SQL query to drop index MyIndex.

# Objective

# To create drop index MyIndex.

# Procedure

# ALTER TABLE employee DROP INDEX myindex.

# Result

# 

# Problem - 80

# Write SQL query to drop index MyIndex1.

# Objective

# To create index MyIndex.

# Procedure

# ALTER TABLE employee DROP INDEX MyIndex.

# Result

# 

# Conclusion

# In conclusion, database is easy to learn if the concepts are properly understood. During this lab experiment / lab sheet I learned about the importance of database and how easily the data / information/ records can be added / retrieved / deleted.

# In database stored procedures are like function that implement the DRY (Don’t Repeat Yourself) principle, we have triggers that are like if statements. Triggers activate only when certain criteria are met, much like the if statements. DBMS becomes even easier if we have knowledge of basic programming concepts like if statements, function, date and time functions, etc.

# In this unit, I learned how to create trigger, drop trigger, create index on multiple columns, create index on single column, drop index, etc.