/\* Module-1 \*/

-- 1. Create a database named training

CREATE DATABASE training;

USE training;

/\*

2. Create a table ‘demography’ with the following columns inside training database

CustID integer auto-increment

Name varchar(50)

Age integer

Gender varcahr(1)

\*/

CREATE TABLE demography

(

cust\_id INT AUTO\_INCREMENT,

name VARCHAR(50),

age INT,

gender VARCHAR(1),

PRIMARY KEY (cust\_id)

);

-- 3. Insert the following values into the table ‘demography’

-- Name=’John’, Age=25, Gender=’M’

INSERT INTO demography

(

cust\_id,

name,

age,

gender

)

VALUES

(

1,

'John',

25,

'M'

);

/\*

4. Insert the following values into the table ‘demography’ using a single query

Name = ’Pawan’, Age = 26, Gender=’M’

Name = ‘Hema’, Age=31, Gender=’F’

\*/

INSERT INTO demography

(name,age,gender)

VALUES( 'Pawan', 26, 'M'),

( 'Hema', 31, 'F');

/\*

5. Insert the following value into the table ‘demography’

Name = ‘Rekha’, Gender=’F’

\*/

INSERT INTO demography

(name,gender)

VALUES ( 'Rekha', 'F');

/\*

6. Retrieve all rows and columns from table ‘demography’

\*/

SELECT

\*

FROM

demography;

/\*

7. Update the age to NULL for Name = ‘John’. Note that the NULL used here is MySQL NULL and not string NULL.

\*/

UPDATE

demography

SET

age = NULL

WHERE

name = "John";

/\*

8.Retrieve all the rows from table ‘demography’ where Age is NULL.

\*/

SELECT

\*

FROM

demography

WHERE

age IS NULL;

/\*

9. Delete all the rows from the table ‘demography’.

\*/

DELETE FROM demography;

/\*

10. Drop the table ‘demography'.

\*/

DROP TABLE demography;

/\* Module - 2 \*/

/\*

1. Retrieve the account ID, customer ID, and available balance for all accounts

whose status equals 'ACTIVE' and whose available balance is greater than $2,500.

\*/

SELECT

account\_id, cust\_id, avail\_balance

FROM

account

WHERE

status = 'active'

AND avail\_balance > 2500;

/\*

2. Construct a query that retrieves all accounts opened in 2002.

\*/

SELECT

\*

FROM

account

WHERE

open\_date >= '2002-01-01';

/\*

3. Retrieve the account ID, available balance and pending balance for all accounts where available balance is not equal to pending balance.

\*/

SELECT

account\_id,

avail\_balance,

pending\_balance

FROM

account

WHERE

avail\_balance <> pending\_balance;

/\*

4. Retrieve account ID, Product code for the account ID’s 1,10,23,27

\*/

SELECT

account\_id, product\_cd

FROM

account

WHERE

account\_id IN (1 , 10, 23, 27);

/\*

5. Retrieve account ID, available balance from all those accounts whose available balance is in between 100 and 200.

\*/

SELECT

account\_id, avail\_balance

FROM

account

WHERE

avail\_balance BETWEEN 100 AND 200;

-- Module - 3

/\*

1. Construct a query that counts the number of rows in the account table

\*/

SELECT

COUNT(\*)

FROM

account;

/\*

2. Retrieve the first two rows from account table

\*/

SELECT

\*

FROM

account

LIMIT 2;

/\*

3. ⦁ Retrieve the third and fourth row from account table

\*/

SELECT

\*

FROM

account

LIMIT 2, 2;

/\*

4. Retrieve year of birth, month of birth, day of birth, weekday of birth for all the individuals from the table individual

\*/

SELECT

EXTRACT(YEAR FROM birth\_date) AS Year,

EXTRACT(MONTH FROM birth\_date) AS Month,

EXTRACT(DAY FROM birth\_date) AS day

FROM

individual;

/\*

5. Write a query that returns the 17th through 25th characters of the string 'Please find the substring in this string'.

\*/

SELECT

SUBSTR('Please find the substring in this string',

17,

(25 - 17 + 1)) AS SUBSTR;

/\*

6. ⦁ Write a query that returns the absolute value and sign (-1, 0, or 1)

of the number -25.76823.Also return the number rounded to the nearest hundredth.

\*/

SELECT

ROUND(ABS(- 25.76823), 2) AS absolute\_value,

SIGN(- 25.76823) AS SIGN;

/\*

7. Write a query that adds 30 days to the current date.

\*/

SELECT

DATE\_ADD(CURDATE(), INTERVAL 30 DAY) as 'CURRENT\_DATE+30';

/\*

8. Retrieve the first three letters of first name and last three letters of last name from the table individual.

\*/

SELECT

SUBSTR(fname, 1, 3) AS first\_name,

SUBSTR(lname, LENGTH(lname) - 3, 3) AS last\_name

FROM

individual;

/\*

9. Retrieve the first names in Upper case from individual whose first name consists of five characters.

\*/

SELECT

UPPER(fname)

FROM

individual

WHERE

LENGTH(fname) = 5;

/\*

10. Retrieve the maximum balance and average balance from the account table for customer ID = 1.

\*/

SELECT

MAX(avail\_balance) AS MAX,

AVG(avail\_balance) AS average

FROM

account

GROUP BY cust\_id

HAVING cust\_id = 1;

-- Module - 4

/\*

1. Construct a query to count the number of accounts held by each customer.

Show the customer ID and the number of accounts for each customer.

\*/

SELECT

cust\_id,

COUNT(account\_id) AS 'Number of Accounts'

FROM

account

GROUP BY cust\_id;

/\*

2. Modify the previous query to fetch only those customers who has more than two accounts.

\*/

SELECT

cust\_id, COUNT(account\_id) AS 'Number of Accounts'

FROM

account

GROUP BY cust\_id

HAVING COUNT(account\_id) > 2;

/\*

3. Retrieve first name and date of birth from individual and sort them from youngest to oldest.

\*/

SELECT

fname,

birth\_date

FROM

individual

ORDER BY birth\_date DESC;

/\*

4. From the account table, retrieve the year of account opening (year part of open\_date)

and average available balance (avail\_balance) present in the accounts that are opened

in each calendar year. Retrieve the year only if the average balance is greater than 200.

Also sort the results based on calendar year.

\*/

SELECT DISTINCT

EXTRACT(YEAR FROM open\_date) AS 'YEAR OF OPENING ACCOUNT',

AVG(avail\_balance) AS AVERAGE\_BALANCE

FROM

account

GROUP BY EXTRACT(YEAR FROM open\_date)

HAVING AVG(avail\_balance) > 200

ORDER BY EXTRACT(YEAR FROM open\_date);

/\*

5. Retrieve the product code and maximum pending balance for the product codes (CHK, SAV, CD)

present in account table.

\*/

SELECT

product\_cd, MAX(pending\_balance)

FROM

account

GROUP BY product\_cd

HAVING product\_cd IN ('CHK' , 'SAV', 'CD');

-- Module - 5

/\*

1. Retrieve first name, title and department name by joining tables employee and department

using department id.

\*/

SELECT

employee.fname, employee.title, department.name

FROM

employee

INNER JOIN

department

WHERE

employee.dept\_id = department.dept\_id;

/\*

2. Left join table product with table product\_type (product\_type left join product)

to retrieve product\_type.name and product.name from the tables.

\*/

SELECT

product\_type.name,

product.name

FROM

product\_type

LEFT JOIN

product ON product.product\_type\_cd = product\_type.product\_type\_cd;

/\*

3. Using inner join, Retrieve the full employee name (fname followed by a space and then lname),

Superior name (using superior\_emp\_id) from the employee table.

\*/

SELECT

CONCAT(e1.fname, ' ', e1.lname) AS 'Full Employee Name',

CONCAT(e2.fname, ' ', e2.lname) AS 'Superior Name'

FROM

employee e1

INNER JOIN

employee e2 ON e1.superior\_emp\_id = e2.emp\_id;

/\*

4. Using subquery, retrieve the fname and lname of the

employees whose superior is ‘Susan Hawthorne’ from employee

\*/

SELECT

fname, lname

FROM

employee

WHERE

superior\_emp\_id =

(SELECT

emp\_id

FROM

employee

WHERE

fname = 'Susan' AND lname = "Hawthorne"

)

;

/\*

5. In employee table, retrieve the superior names (fname and lname) present in department 1.

A person is superior if he/she is superior to atleast one person in the given department.

Use sub-query concept

\*/

SELECT

fname, lname

FROM

employee

WHERE

emp\_id IN (SELECT

emp\_id

FROM

employee

GROUP BY superior\_emp\_id

HAVING COUNT(emp\_id) > 1);