EXPERIMENT NO: 1

Introduction to MySQL

AIM:
To study about MySQL database
OBJECTIVES
THEORY
MySQL, is one of the most popular Open Source SQL database management systems.
MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses.
MySQL is developed, marketed, and supported by MySQL AB, which is a Swedish company.
MySQL is becoming so popular because of many good reasons:

• MySQL is released under an open-source license. So, you have nothing to pay to use it.
• MySQL is a very powerful program in its own right. It handles a large subset of the functionality of the most expensive and powerful database packages.
• MySQL uses a standard form of the well-known SQL data language.
• MySQL works on many operating systems and with many languages including PHP, PERL, C, C++, JAVA, etc.
• MySQL works very quickly and works well even with large data sets.
• MySQL is very friendly to PHP, the most appreciated language for web development.
• MySQL supports large databases, up to 50 million rows or more in a table.
• MySQL is customizable.

Structured Query Language (SQL)

DATA DEFINITION LANGUAGE (DDL) COMMANDS

Database Queries:

Before creating any tables, MySQL requires you to create a database by executing the CREATE DATABASE command.

● Create a database
```
CREATE DATABASE <database name>
```
● Delete a database
```
DROP DATABASE <database name>
```
● Select the database
```
USE <database name>
```
● List all databases
```
SHOW databases;
```
● Rename a database
```
ALTER DATABASE <database name> RENAME <new database name>
```

Table Queries:
● To Create a table
```
CREATE TABLE <tablename> (<fieldname>< fieldtype>(<fieldsize>) , ...)
```
● List all tables in the current database
```
SHOW tables;
```

● Show table format with column names and data types

```
DESCRIBE <table name>
```

● Modify the structure of table

```
ALTER TABLE <table name> <alter specifications>
ALTER TABLE <table name> DROP COLUMN <column name>
ALTER TABLE <table name> ADD COLUMN <column name> datatype>(<size>)
```

● Delete the table

```
DROP TABLE <table name>
```

DATA MANIPULATION LANGUAGE(DML) COMMANDS

●INSERT

```
INSERT INTO <tablename> VALUES (value1, value2, ..., value n).
```

● UPDATE

```
UPDATE <table> SET <field1> = <value1> AND <field2> = <value2> WHERE
<conditions>
```

● DELETE

```
DELETE FROM <table> WHERE <condition>
```

● SELECT

a) Retrieve from all columns

```
SELECT * FROM <table>
```

b) Retrieve from selected columns

```
SELECT <column 1>, <column 2> FROM <table>
```

c) Retrieve unique values

```
SELECT DISTINCT <column name> FROM <table>
```

d) Retrieve data satisfying a given condition

```
SELECT <columns> FROM <tables> WHERE <condition>
```

Aggregate functions

● **Max**

```
Select Max (<Column name>) from <table name>;
```

● **Min**

```
Select Min (<Column name>) from <table name>;
```

● **Count**

```
Select Count (<Column name>) from <table name>;
```

● **Sum**

```
Select Distinct (Sum (<Column name>)) from <table name>;
```

● **Avg**

```
Select Distinct (avg (<Column name>)) from <table name>;
```

RESULT

The MySQL database is studied.

EXPERIMENT NO: 02

Car_insurance Database

AIM:

Consider the following car_insurance database. Write the queries for the following
(i) create the database
(ii) select the current database
(iii) Create the following tables. Where the primary keys are underlined.
a. Person (<u>driver-id</u>, name, address)
b. Car (<u>license</u>, model, year)
c. Accident (<u>report-number</u>, date, location)
d. Owns (<u>driver-id</u>, <u>license</u>)
e. Participated (<u>driver-id</u>, <u>car</u>, <u>report-number</u>, damage_amount)
(iv) Add a new accident to the database
(v) Delete the Toyota belonging to "Simanto"
(vi) Find the total number of people who owned cars that were involved in accidents in 2012.
(vii) Update the damage amount for the car with license number "DHAKA2000" in the accident with report number "AR2197" to 50000/-

OBJECTIVES

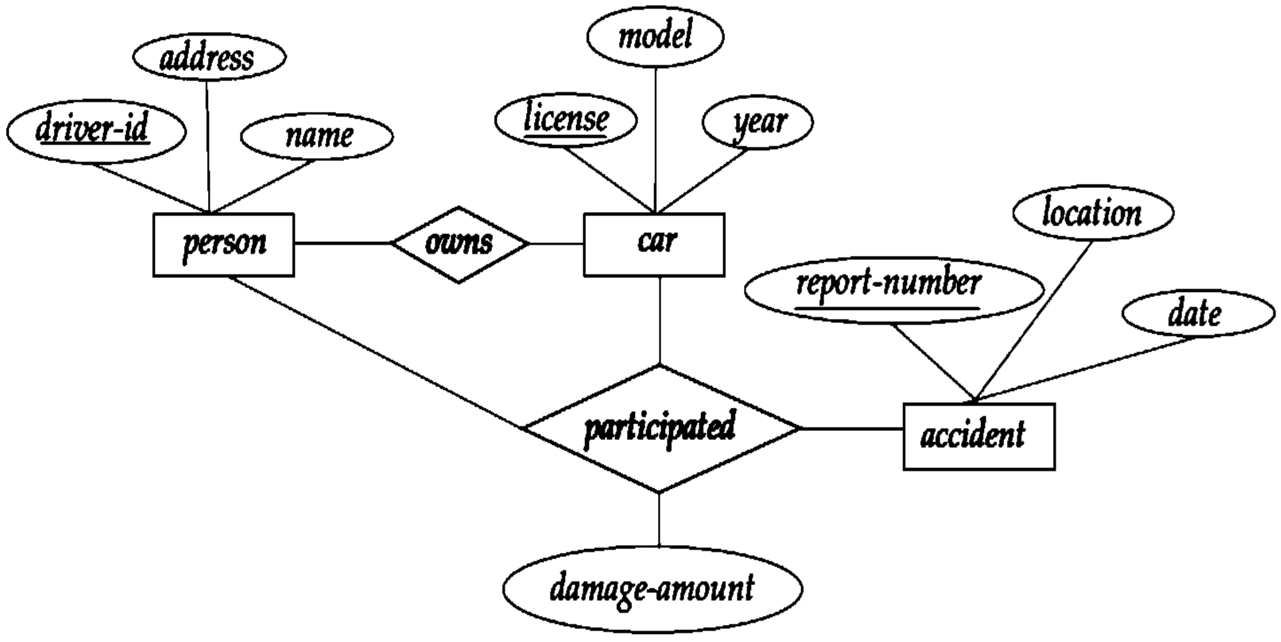
To understand car_insurance database in MySQL.

THEORY

Database Name

Car_insurance

Entity-Relationship (E-R) Diagram:



Data-Definition Language (DDL)

```
CREATE DATABASE IF NOT EXISTS car_insurance;
USE car_insurance;
CREATE TABLE accident (
     ‘report_no’ VARCHAR (45) NOT NULL PRIMARY KEY,
     ‘date’ TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
     ‘location’ VARCHAR (45)
) ;
CREATE TABLE car (
     ‘license’ VARCHAR (45) NOT NULL PRIMARY KEY,
     ‘model’ VARCHAR (45),
     ‘year’ VARCHAR (45)
);
CREATE TABLE  person (
     ‘driver_id’ VARCHAR (45) NOT NULL PRIMARY KEY,
     ‘name’ VARCHAR (45),
     ‘address’ VARCHAR (45)
);
CREATE TABLE  owns (
     ‘driver_id’ varchar (45) NOT NULL,
     ‘license’ varchar (45) NOT NULL,
     PRIMARY KEY (driver_id, license),
     FOREIGN KEY (driver_id) REFERENCES person (driver_id),
     FOREIGN KEY (license) REFERENCES car (license)
);
```

Data-Manipulation Language (DML)

```
INSERT INTO person (driver_id, name, address) VALUES
('d101','Rahim','Lalbagh'),
('d102','Karim','Jigatola'),
('d103','Shimanto','Dhanmondi'),
('d104','Saiful','Mohammadpur');

INSERT INTO car (license, model, year) VALUES
('DHAKA1000','Toyota','2012'),
('DHAKA2000','BMW','2012'),
('DHAKA3000','Toyota','2012'),
('DHAKA4000','Corolla','2012');

INSERT INTO accident (report_number, date, location) VALUES
('AR2195','2012-01-01','Shahbagh'),
('AR2196','2012-05-10','Kurmitola'),
('AR2197','2012-08-18','Sobhanbagh'),
('AR2198','2012-09-22','Sciencelab');

INSERT INTO owns (driver_id, license) VALUES
('d102','DHAKA2000'),
('d103','DHAKA3000'),
('d104','DHAKA4000');
```

```
INSERT INTO
participated(driver_id,license,report_number,damage_amount)VALUES
('d101','DHAKA1000','AR2195','1000'),
('d102','DHAKA2000','AR2197','2000'),
('d103','DHAKA3000','AR2198','3000');
```

Query

(iv) Add a new accident to the database

```
INSERT INTO accident VALUES (4007, '2012-05-01', 'Dhanmondi');
```

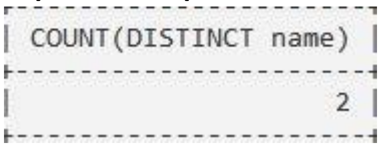
(v) Delete the Toyota belonging to "Simanto".

```
DELETE car
WHERE model = 'Toyota' AND license IN
(SELECT license FROM person p, owns o
WHERE p.name = 'Simanto' AND p.driver_id = o.driver_id)
```

(vi) Find the total number of people who owned cars that were involved in accidents in 2012.

```
SELECT COUNT (DISTINCT name)
FROM accident, participated, person
WHERE accident.report_number = participated.report_number
AND participated.driver_id = person.driver_id
AND date BETWEEN DATE '2012-00-00' AND DATE '2012-12-31'
```

Expected Output:



```
+----------------------+
| COUNT(DISTINCT name) |
+----------------------+
|                    2 |
+----------------------+
```

(vii) Update the damage amount for the car with license number “DHAKA2000” in the accident with report number “AR2197” to 50000/-

```
UPDATE participated
SET damage_amount = 50000
WHERE report_number = "AR2197" AND driver_id IN
(SELECT driver_id FROM owns
WHERE license = "DHAKA2000")
```

RESULT

The car_insurance database queries have been executed successfully.

EXPERIMENT NO: 03

Bank Database

AIM:

Consider the following bank database. Write the queries for the following

(i) create the database

(ii) select the current database

(iii) Create the following tables. Where the primary keys are underlined.

a. Branch (<u>branch-name</u>, <u>branch-city</u>, assets)

b. Customer (<u>customer-name</u>, customer_street, customer_city)

c. Loan (<u>loan-number</u>, <u>branch-name</u>, amount)

d. Borrower (<u>customer-name</u>, loan-number)

e. Account (<u>account-number</u>, branch_name, balance)

f. Depositor (<u>customer-name</u>, account_number)

(iv) Find all customers who have an account but no loan at the bank.

(v) Delete all loan amount between 5000/- and 15000/-.

(vi) Add a record in the database using a form.

(vii) Display your result of query (a) on a report.

OBJECTIVES

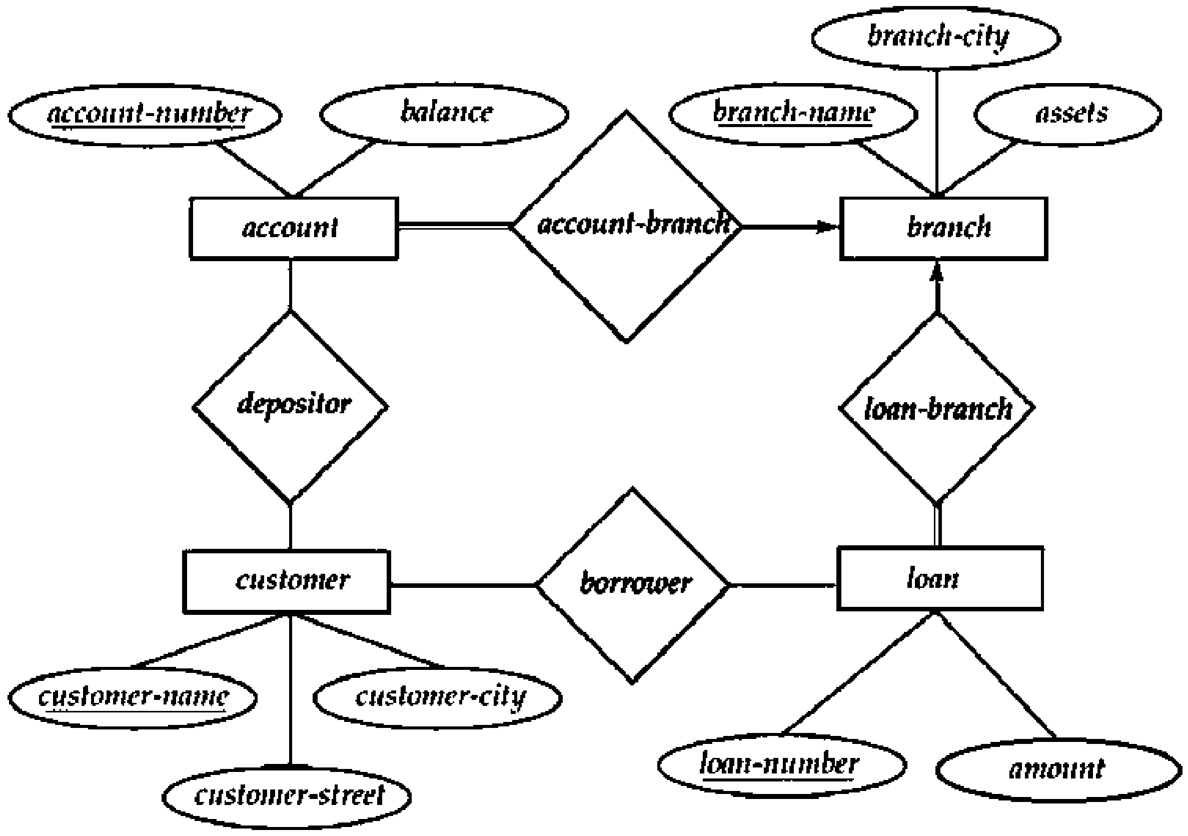
To understand Bank database in MySQL.

THEORY

Database Name

Bank

Entity-Relationship (E-R) Diagram:



Data-Definition Language (DDL)

```
CREATE DATABASE bank_database2;

USE bank_database2;

CREATE TABLE branch (

     branch_name varchar (45) not null,
     branch_city varchar (45) not null,
     assets int not null,
     PRIMARY KEY (branch_name, branch_city)
     );
CREATE TABLE customer (
     customer_name varchar (45) not null PRIMARY KEY,
     customer_street varchar (45) not null,
     customer_city varchar (45) not null
     );
CREATE TABLE loan (
     loan_number varchar (45) not null,
     branch_name varchar (45) not null,
     amount double not null,
     PRIMARY KEY (loan_number, branch_name),
     FOREIGN KEY (branch_name)
     REFERENCES branch(branch_name) ON DELETE CASCADE
     );
CREATE TABLE borrower (
     customer_name varchar (45) not null PRIMARY KEY,
     loan_number varchar (45) not null,
     FOREIGN KEY (customer_name)
     REFERENCES customer(customer_name) ON DELETE CASCADE,
     FOREIGN KEY (loan_number)
     REFERENCES loan(loan_number) ON DELETE CASCADE
     );
CREATE TABLE account (
     account_number varchar (45) not null PRIMARY KEY,
     branch_name varchar (45) not null,
     balance double not null,
     FOREIGN KEY (branch_name)
     REFERENCES branch(branch_name) ON DELETE CASCADE
     );
CREATE TABLE depositor (
     customer_name varchar (45) not null PRIMARY KEY,
     account_number varchar (45) not null,
     FOREIGN KEY (customer_name)
     REFERENCES customer(customer_name) ON DELETE CASCADE,
     FOREIGN KEY (account_number)
     REFERENCES account(account_number) ON DELETE CASCADE
     );
```

Data-Manipulation Language (DML)

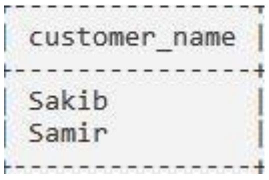
```
INSERT INTO branch(branch_name,branch_city,assets) VALUES
('Rupali','Dhaka','1000000'),
('Pubali','Dhaka','2000000'),
('Sonali','Dhaka','3000000'),
('Ogroni','Dhaka','3000000'),
('Islami','Dhaka','4000000');

INSERT INTO customer(customer_name,customer_street,customer_city) VALUES
('Rahim','Zigatola','Dhaka'),
('Karim','Dhanmondi','Dhaka'),
('Sakib','Motijheel','Dhaka'),
('Samir','Malibagh','Dhaka'),
('Sohan','Kakrail','Dhaka');

INSERT INTO loan(loan_number,branch_name,amount) VALUES
('L101','Sonali','10000'),
('L102','Rupali','15000'),
('L103','Sonali','3000'),
('L104','Pubali','5000'),
('L105','Islami','5000');

INSERT INTO Borrower(customer_name,loan_number) VALUES
('Rahim','L101'),
('Karim','L102'),
('Sohan','L103');

INSERT INTO account(account_number,branch_name,balance) VALUES
('A101','Sonali','50000'),
('A103','Islami','80000'),
('A104','Ogroni','50000'),
('A105','Pubali','10000');

INSERT INTO depositor(customer_name,account_number) VALUES
('Sakib','A103'),
('Samir','A104'),
('Rahim','A105'),
('Karim','A101');
```

Query

(iv) Find all customers who have an account but no loan at the bank.

```
SELECT customer_name FROM depositor
EXCEPT
SELECT customer_name FROM borrower;
```

Expected output:



```
+---------------+
| customer_name |
+---------------+
| Sakib         |
| Samir         |
+---------------+
```

(v) Delete all loan amount between 5000/- and 15000/-.

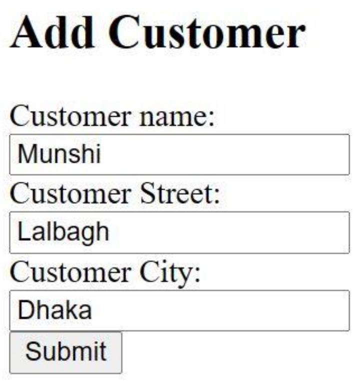
```
DELETE FROM loan WHERE amount BETWEEN 5000 AND 15000;
```

(vi) Add a record in the database using a form.

html:

```
<!DOCTYPE html>
<html>
<body>
    <h2>Add Customer</h2>
    <form action="input.php" method="POST">
        Customer name:<br>
        <input type="text" name="customer_name"><br>
        Customer Street:<br>
        <input type="text" name="customer_street"><br>
        Customer City:<br>
        <input type="text" name="customer_city"><br>
        <input type="submit">
    </form>
</body>
</html>
```

Expected output:



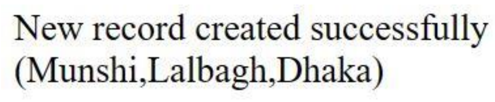
Add Customer

Customer name:
Munshi
Customer Street:
Lalbagh
Customer City:
Dhaka
Submit

Php:

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "bank_database2";
// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
$customer_name = $_POST['customer_name'];
$customer_street = $_POST['customer_street'];
$customer_city = $_POST['customer_city'];
$sql = "INSERT INTO customer (customer_name, customer_street,
customer_city)
VALUES ('$customer_name', '$customer_street', '$customer_city')";
if ($conn->query($sql) === TRUE) {
  echo "New record created successfully"."<br>";
  echo "($customer_name,$customer_street,$customer_city)";
} else {
  echo "Error: " . $sql . "<br>" . $conn->error;
}
}
$conn->close();

?>
```

Expected output:



```
New record created successfully
(Munshi,Lalbagh,Dhaka)
```

RESULT

The bank database queries have been executed successfully.

EXPERIMENT NO: 04

Employee Database

AIM:

Consider the following employee database. Write the queries for the following
(i) create the database
(ii) select the current database
(iii) Create the following tables. Where the primary keys are underlined.

a. Employee (<u>employee-id</u>, employee_name, street, city)
b. Works (<u>employee-id</u>, company_name, salary)
c. Company (<u>company-name</u>, city)
d. Manager (<u>employee-id</u>, manager_name)

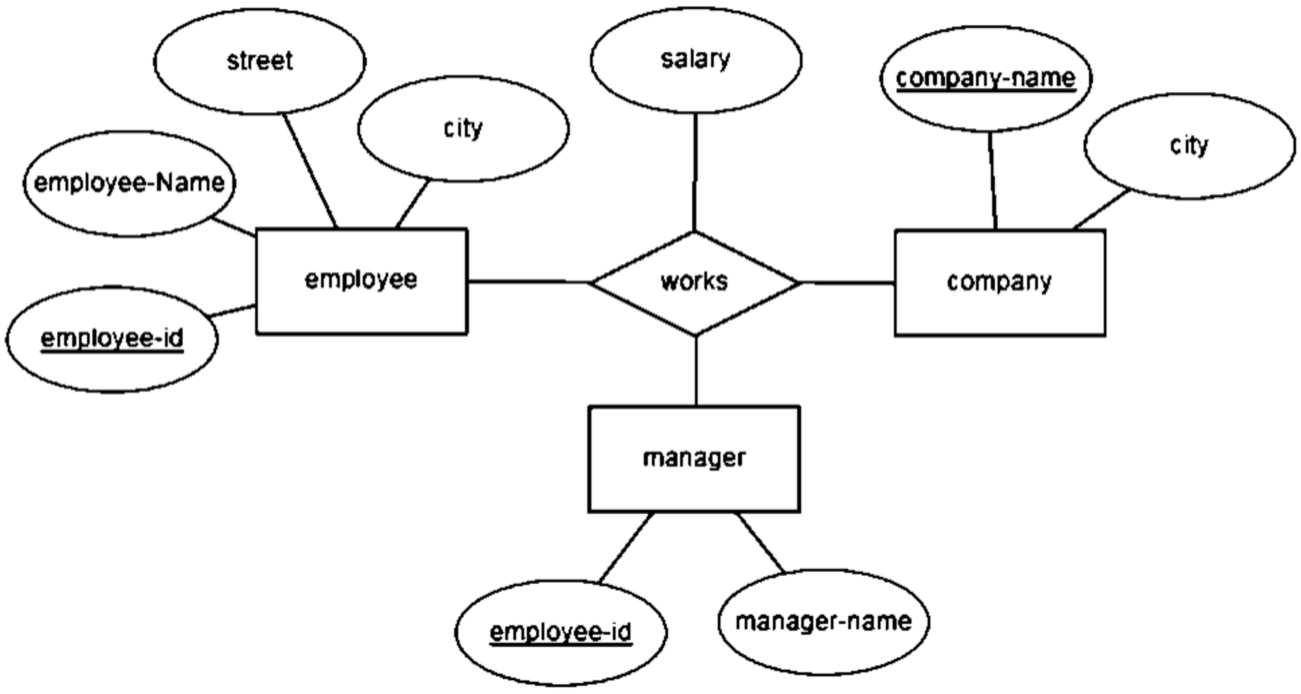
(iv) Find the company that has the most employees.
(v) Find the average salaries at each company.
(vi) Find all employees who live in Dhaka city, but their company is not in Dhaka.

OBJECTIVES
To understand employee database in MySQL.
THEORY

Database Name

Employee

Entity-Relationship (E-R) Diagram:



Data-Definition Language (DDL)

```
CREATE DATABASE IF NOT EXISTS employee;
USE employee;

CREATE TABLE employee(
employee_id varchar(45) not null PRIMARY KEY,
employee_name varchar(45) not null,
street varchar(45) not null,
city varchar(45) not null
);

CREATE TABLE works (
employee_id varchar (45) not null PRIMARY KEY,
company_name varchar (45) not null,
salary double not null,
FOREIGN KEY (employee_id)
REFERENCES employee(employee_id) ON DELETE CASCADE
);

CREATE TABLE company (
company_name varchar (45) not null PRIMARY KEY,
city varchar (45) not null
);

CREATE TABLE manager (
employee_id varchar(45) not null PRIMARY KEY,
manager_name varchar(45) not null,
FOREIGN KEY (employee_id)
REFERENCES employee(employee_id) ON DELETE CASCADE
);
```

Data-Manipulation Language (DML)

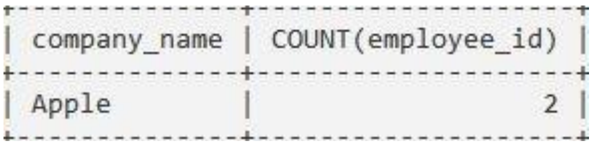
```
INSERT INTO employee(employee_id,employee_name,street,city) VALUES
('101','Rahim','Dhanmondi','Dhaka'),
('102','Karim','Motijheel','Dhaka'),
('103','Shimanto','Veramara','Khustia'),
('104','Saiful','begomgonj','Noakhali'),
('105','Sayem','Khilgaon','Dhaka');

INSERT INTO works(employee_id,company_name,salary) VALUES
('101','Samsung','10000'),
('102','Apple','12000'),
('103','Vivo','8000'),
('104','Xiomi','9000'),
('105','Apple','10000');


INSERT INTO company(company_name,city) VALUES
('Samsung','Dinajpur'),
('Apple','Bogura'),
('Vivo','Khustia'),
('Xiomi','Noakhali');

INSERT INTO manager(employee_id,manager_name) VALUES
('102','Karim'),
('104','Saiful');
```

Query

(iv) Find the company that has the most employees.

```
SELECT company_name,COUNT(employee_id)
FROM works
GROUP BY company_name DESC LIMIT 1;
```

Expected output:

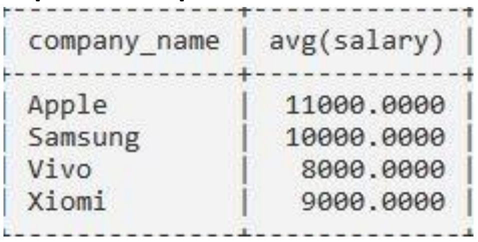


```
+--------------+--------------------+
| company_name | COUNT(employee_id) |
+--------------+--------------------+
| Apple        |                  2 |
+--------------+--------------------+
```

(v) Find the average salaries at each company.

```
SELECT company_name,avg(salary)
FROM works
GROUP BY company_name;
```

Expected output:

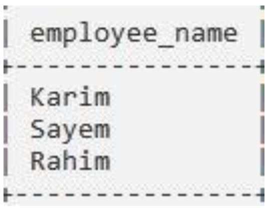


```
+--------------+-------------+
| company_name | avg(salary) |
+--------------+-------------+
| Apple        |  11000.0000 |
| Samsung      |  10000.0000 |
| Vivo         |   8000.0000 |
| Xiomi        |   9000.0000 |
+--------------+-------------+
```

(vi) Find all employees who live in Dhaka city, but their company is not in Dhaka.

```
SELECT employee_name
FROM company c,employee e,works w
WHERE e.employee_id = w.employee_id AND c.company_name = w.company_name
AND e.city = 'Dhaka' and c.city != 'Dhaka';
```

Expected output:



```
| employee_name |
+---------------+
| Karim         |
| Sayem         |
| Rahim         |
+---------------+
```

RESULT

The employee database queries have been executed successfully.

EXPERIMENT NO: 05

Banking database

AIM:

Consider the following banking database. Write the queries for the following
(i) create the database
(ii) select the current database
(iii) Create the following tables. Where the primary keys are underlined.

a. Branch (branch-name, branch_city, assets)
b. Customer (customer-name, customer_street, customer_city)
c. Loan_account (loan-number, branch_name, amount)
d. Borrower (customer-name, loan_number)
e. Saving_Account (account-number, branch_name, balance)
f. Depositor (customer-name, account_number)

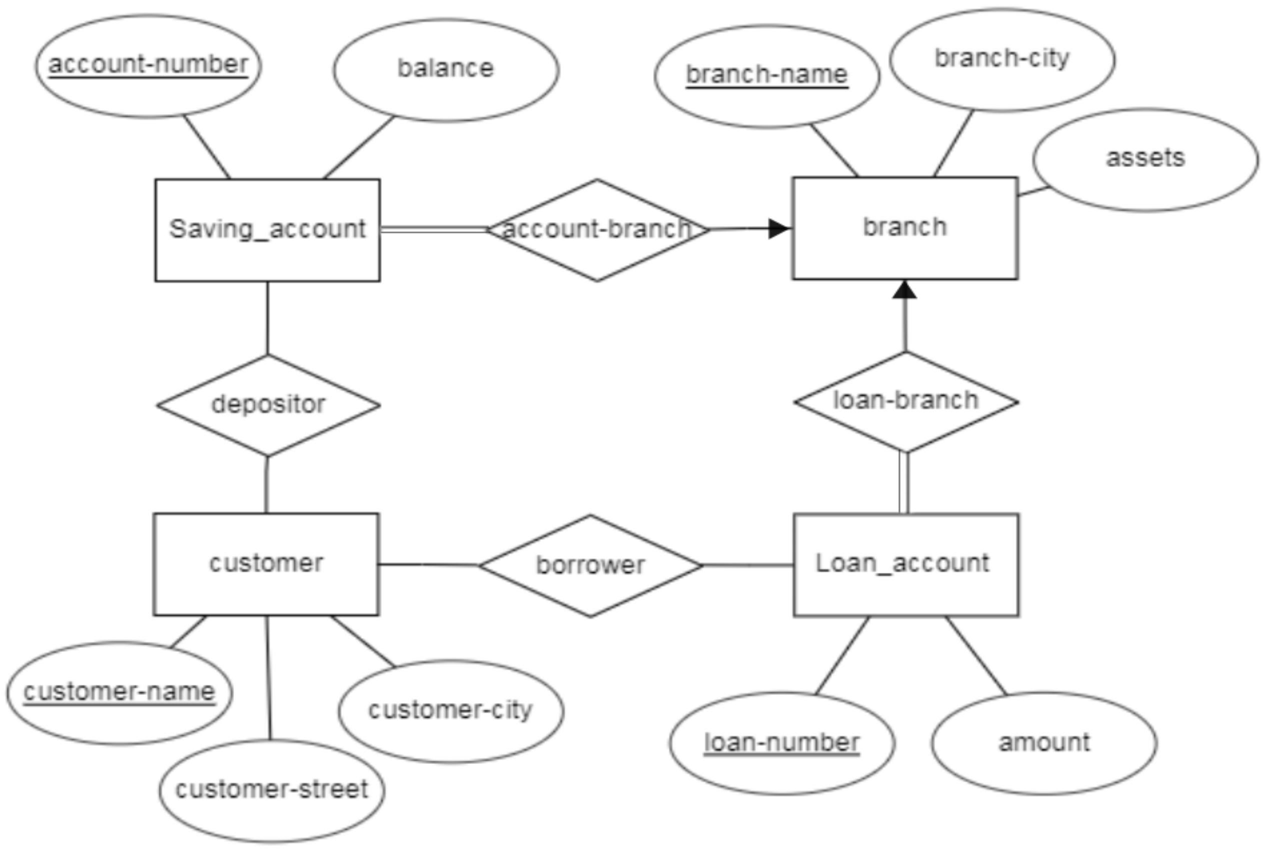
(iv) Find all customers of the bank who have both loan and saving account.
(v) Find all average account balance at each branch.
(vi) Deduct 1% service charge from saving account balance that have a both loan and a saving account otherwise deduct 2% service charge from saving account balance.

OBJECTIVES
To understand banking database in MySQL.
THEORY

Database Name

Banking

Entity-Relationship (E-R) Diagram:



Data-Definition Language (DDL)

```
CREATE DATABASE IF NOT EXISTS bank_database;
USE bank_database;

CREATE TABLE branch(
branch_name varchar (45) not null PRIMARY KEY,
branch_city varchar (45) not null,
assets int not null
);

CREATE TABLE customer (
customer_name varchar (45) not null PRIMARY KEY,
customer_street varchar (45) not null,
customer_city varchar (45) not null
);

CREATE TABLE loan_account (
loan_number varchar (45) not null PRIMARY KEY,
branch_name varchar (45) not null,
amount double not null,
FOREIGN KEY (branch_name)
REFERENCES branch(branch_name) ON DELETE CASCADE
);

CREATE TABLE borrower (
customer_name varchar (45) not null PRIMARY KEY,
loan_number varchar (45) not null,
FOREIGN KEY (customer_name)
REFERENCES customer(customer_name) ON DELETE CASCADE,
FOREIGN KEY (loan_number)
REFERENCES loan_account(loan_number) ON DELETE CASCADE
);

CREATE TABLE saving_account (
account_number varchar (45) not null PRIMARY KEY,
branch_name varchar (45) not null,
balance double not null,
FOREIGN KEY (branch_name)
REFERENCES branch(branch_name) ON DELETE CASCADE
);

CREATE TABLE depositor (
customer_name varchar (45) not null,
account_number varchar (45) not null,
PRIMARY KEY (customer_name, account_number),
FOREIGN KEY (customer_name)
REFERENCES customer(customer_name) ON DELETE CASCADE,
FOREIGN KEY (account_number)
REFERENCES saving_account(account_number) ON DELETE CASCADE
);
```

Data-Manipulation Language (DML)

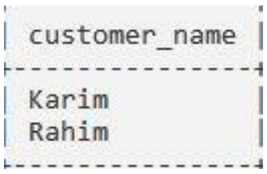
```
INSERT INTO branch(branch_name,branch_city,assets) VALUES
('Rupali','Dhaka','10000'),
('Pubali','Dhaka','20000'),
('Sonali','Dhaka','30000'),
('Ogroni','Dhaka','30000'),
('Islami','Dhaka','40000');

INSERT INTO customer(customer_name,customer_street,customer_city) VALUES
('Rahim','Zigatola','Dhaka'),
('Karim','Dhanmondi','Dhaka'),
('Sakib','Motijheel','Dhaka'),
('Samir','Malibagh','Dhaka'),
('Sohan','Kakrail','Dhaka');

INSERT INTO loan_account(loan_number,branch_name,amount) VALUES
('L101','Sonali','100'),
('L102','Rupali','150'),
('L103','Sonali','300'),
('L104','Pubali','50'),
('L105','Islami','50');

INSERT INTO Borrower(customer_name,loan_number) VALUES
('Rahim','L101'),
('Karim','L102'),
('Sohan','L103');

INSERT INTO saving_account(account_number,branch_name,balance) VALUES
('A101','Sonali','500'),
('A103','Islami','800'),
('A104','Ogroni','500'),
('A105','Pubali','100');

INSERT INTO Depositor(customer_name,account_number) VALUES
('Sakib','A103'),
('Samir','A104'),
('Rahim','A105'),
('Karim','A101');
```

Query

(iv) Find all customers of the bank who have both loan and saving account.

```
SELECT customer_name FROM depositor
INTERSECT
SELECT customer_name FROM borrower;
```

Expected output:

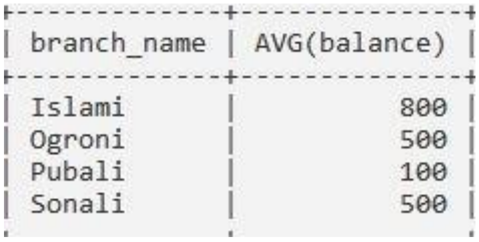


```
| customer_name |
+---------------+
| Karim         |
| Rahim         |
+---------------+
```

(v) Find all average account balance at each branch.

```
SELECT branch_name, AVG(balance)
FROM saving_account
GROUP BY branch_name;
```

Expected output:



```
+-------------+--------------+
| branch_name | AVG(balance) |
+-------------+--------------+
| Islami      |          800 |
| Ogroni      |          500 |
| Pubali      |          100 |
| Sonali      |          500 |
```

(vi) Deduct 1% service charge from saving account balance that have a both loan and a saving account otherwise deduct 2% service charge from saving account balance.

```
UPDATE saving_account
SET balance =
CASE
  WHEN
       account_number IN
       (SELECT depositor.account_number
       FROM depositor, borrower
       WHERE depositor.customer_name = borrower.customer_name)
  THEN
       balance-((balance*1)/100)
  ELSE
       balance-((balance*2)/100)
END;
```

RESULT

The banking database queries have been executed successfully.

EXPERIMENT NO: 06

Employee Database

AIM:

Consider the following employee database. Write the queries for the following
(i) create the database
(ii) select the current database
(iii) Create the following tables. Where the primary keys are underlined.

a. Employee (employee_name, street, city)
b. Works (<u>employee-name</u>, <u>company-name</u>, salary)
c. Company (<u>company-name</u>, city)
d. Manages (<u>employee-name</u>, manages_name)

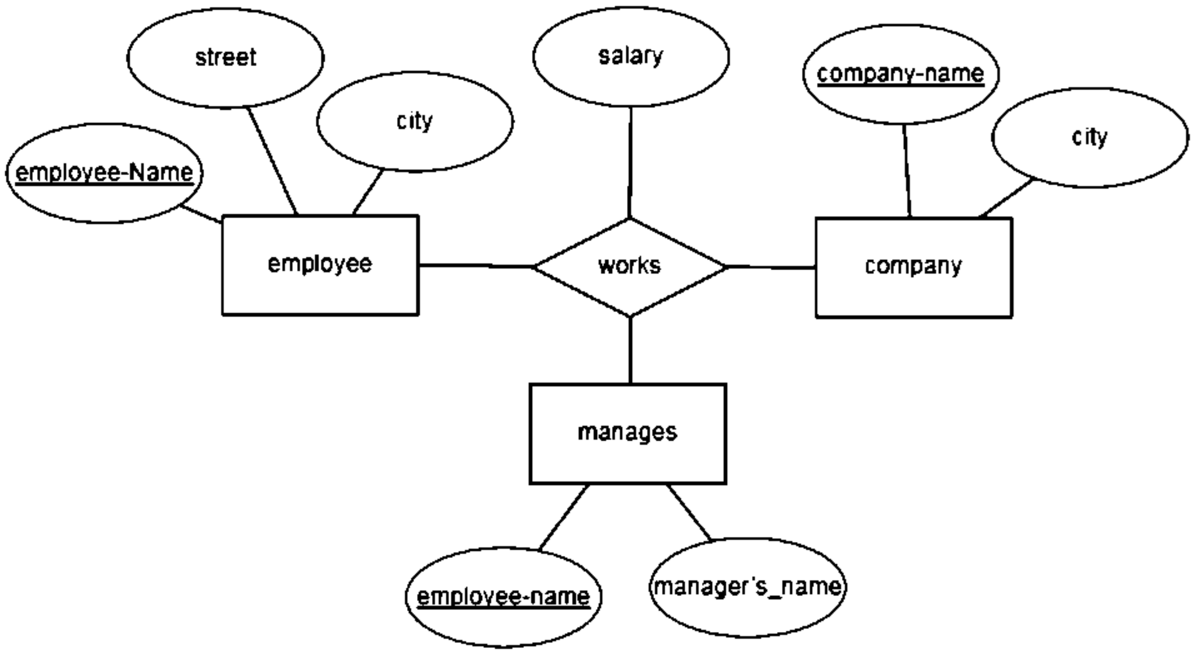
(iv) Finds the names, cities and salaries of all employees who work for PubaliBankLtd.
(v) Find the total salaries of each company.
(vi) Add and record in the database using a form.
(vii) Display your result of query (a) on a report.

OBJECTIVES
To understand banking database in MySQL.
THEORY

Database Name

Employee

Entity-Relationship (E-R) Diagram:



Data-Definition Language (DDL)

```
CREATE DATABASE IF NOT EXISTS employee;
USE employee;

CREATE TABLE employee(
employee_name varchar (45) not null PRIMARY KEY,
street varchar (45) not null,
city varchar (45) not null
);

CREATE TABLE company(
company_name varchar (45) not null PRIMARY KEY,
city varchar (45) not null
);

CREATE TABLE works(
employee_name varchar(45) not null,
company_name varchar(45) not null,
salary double not null,
PRIMARY KEY (employee_name,company_name,
FOREIGN KEY (employee_name)
REFERENCES employee(employee_name) ON DELETE CASCADE,
FOREIGN KEY (company_name)
REFERENCES company(company_name) ON DELETE CASCADE
);

CREATE TABLE manager(
employee_name varchar (45) not null PRIMARY KEY,
manager_name varchar (45) not null,
FOREIGN KEY (employee_name)
REFERENCES employee(employee_name) ON DELETE CASCADE
);
```

Data-Manipulation Language (DML)

```
INSERT INTO employee(employee_name,street,city) VALUES
('Rahim','Dhanmondi','Dhaka'),
('Karim','Motijheel','Dhaka'),
('Shimanto','Veramara','Khustia'),
('Saiful','begomgonj','Noakhali'),
('Sayem','Khilgaon','Dhaka'),
('Sohan','Jatrabari','Dhaka'),
('Piyas','Nobabbari','Bogura'),
('Siyam','Jindabazar','Cumilla'),
('Nuhas','Daudkandi','Cumilla'),
('Raheb','Lalbagh','Dhaka'),
('Riham','Azimpur','Dhaka'),
('Jamil','Narayanganj','Dhaka');


INSERT INTO company(company_name,city) VALUES
```

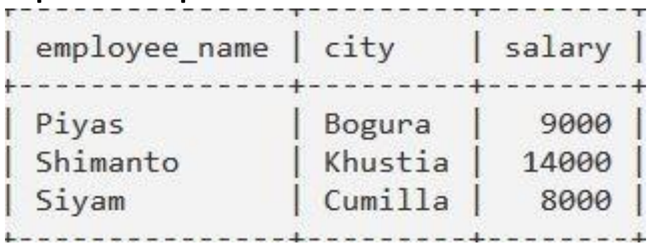
```
('PubaliBankLtd','Dhaka'),
('SonaliBankLtd','Bogura'),
('RupaliBankLtd','Khulna'),
('IslamiBankLtd','Cumilla');


INSERT INTO works(employee_name,company_name,salary) VALUES
('Rahim','SonaliBankLtd','10000'),
('Karim','RupaliBankLtd','12000'),
('Shimanto','PubaliBankLtd','14000'),
('Saiful','IslamiBankLtd','12000'),
('Sayem','IslamiBankLtd','8000'),
('Sohan','IslamiBankLtd','10000'),
('Piyas','PubaliBankLtd','9000'),
('Siyam','PubaliBankLtd','8000'),
('Nuhas','RupaliBankLtd','10000'),
('Raheb','RupaliBankLtd','9000'),
('Riham','SonaliBankLtd','8000'),
('Jamil','SonaliBankLtd','8000');


INSERT INTO manager(employee_name,manager_name) VALUES
('Riham','Rahim'),
('Jamil','Rahim'),
('Raheb','Karim'),
('Nuhas','Karim'),
('Siyam','shimanto'),
('Piyas','Shimanto'),
('Sohan','Saiful'),
('Sayem','Saiful');
```

Query

(iv) Finds the names, cities and salaries of all employees who work for PubaliBankLtd.

```
SELECT employee.employee_name, employee.city, works.salary
FROM employee, works
WHERE works.employee_name = employee.employee_name AND
works.company_name = 'PubaliBankLtd';
```

Expected output:

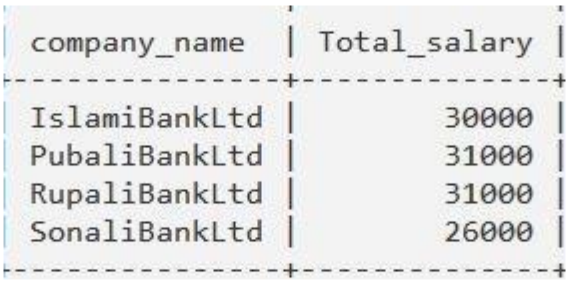


```
+---------------+---------+--------+
| employee_name | city    | salary |
+---------------+---------+--------+
| Piyas         | Bogura  |   9000 |
| Shimanto      | Khustia |  14000 |
| Siyam         | Cumilla |   8000 |
+---------------+---------+--------+
```

(v) Find the total salaries of each company.

```
SELECT company_name, SUM(salary)
AS Total_salary FROM works
GROUP BY company_name;
```

Expected output:



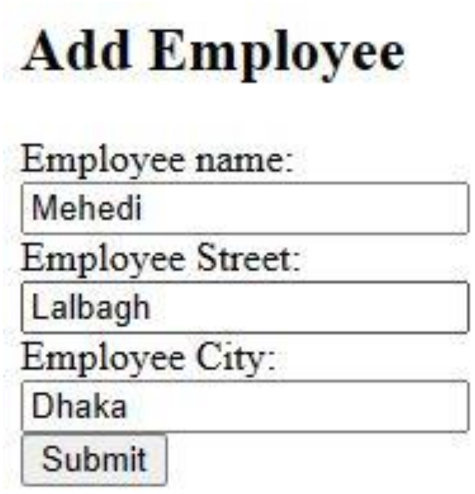
```
| company_name   | Total_salary |
+----------------+--------------+
| IslamiBankLtd  |        30000 |
| PubaliBankLtd  |        31000 |
| RupaliBankLtd  |        31000 |
| SonaliBankLtd  |        26000 |
+----------------+--------------+
```

(vi) Add and record in the database using a form.

Html:

```
<!DOCTYPE html>
<html>
<body>
    <h2>Add Employee</h2>
    <form action="input.php" method="POST">
        Employee name:<br>
        <input type="text" name="employee_name"><br>
        Employee Street:<br>
        <input type="text" name="street"><br>
        Employee City:<br>
        <input type="text" name="city"><br>
        <input type="submit">
    </form>
</body>
</html>
```

Expected output:



Add Employee

Employee name: Mehedi

Employee Street: Lalbagh

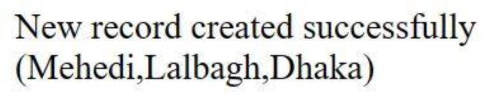
Employee City: Dhaka

Php:

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "employee6";
// Create connection
```

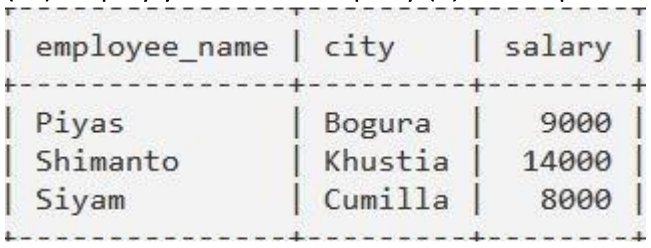
```
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
$employee_name = $_POST['employee_name'];
$street = $_POST['street'];
$city = $_POST['city'];
$sql = "INSERT INTO employee(employee_name,street,city)
VALUES ('$employee_name', '$street', '$city')";
if ($conn->query($sql) === TRUE) {
  echo "New record created successfully"."<br>";
  echo "($employee_name,$street,$city)";
} else {
  echo "Error: " . $sql . "<br>" . $conn->error;
}
}
$conn->close();
?>
```

Expected output:



```
New record created successfully
(Mehedi,Lalbagh,Dhaka)
```

(vii) Display your result of query (a) on a report.



```
+---------------+---------+--------+
| employee_name | city    | salary |
+---------------+---------+--------+
| Piyas         | Bogura  |   9000 |
| Shimanto      | Khustia |  14000 |
| Siyam         | Cumilla |   8000 |
+---------------+---------+--------+
```

RESULT

The employee database queries have been executed successfully.