Coding Question (Java& Mysql)

By : Sakib Sadri

Set-1

1 . Create a class PesonalDetails having attributes (id, name, age, location, address etc), use appropriate access modifiers, print them in the main class.

```
package company;

public class PersonalDetails {
// Private attributes
private int id;
private String name;
private int age;
private String location;
private String address;
// Constructor
public PersonalDetails(int id, String name, int age, String location, String address) {
this.id = id;
this.name = name;
this.age = age;
this.location = location;
this.address = address;
}
// Getter methods to access private attributes
public int getId() {
return id;
}
public String getName() {
return name;
}
public int getAge() {
return age;
}
public String getLocation() {
return location;
}
public String getAddress() {
return address;
}

public static void main(String[] args) {
// Creating an instance of the PersonalDetails class
PersonalDetails person = new PersonalDetails(56, "Sakib Sadri", 20, "Bihar", "Fazilpur");
// Printing the personal details
System.out.println("Personal Details:");
System.out.println("ID: " + person.getId());
System.out.println("Name: " + person.getName());
System.out.println("Age: " + person.getAge());
```

```
System.out.println("Location: " + person.getLocation());
System.out.println("Address: " + person.getAddress());
}
}
```

OUTPUT :

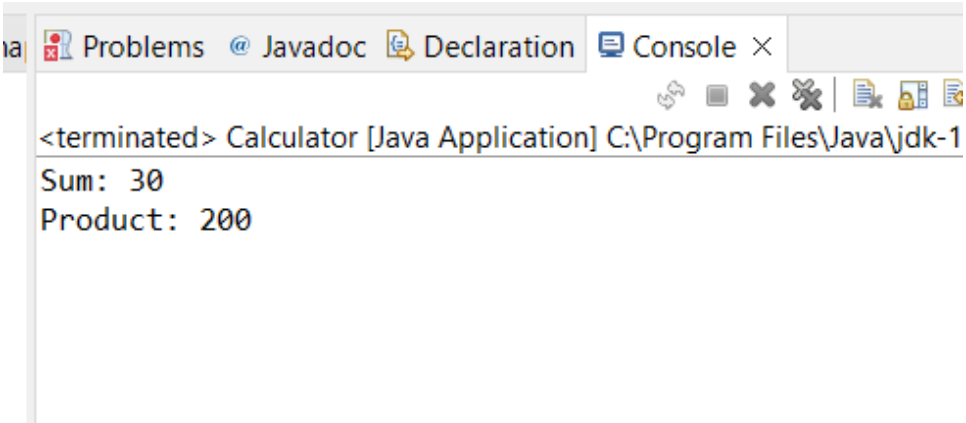
```
Problems  @ Javadoc  Declaration  Console ×
<terminated> PersonalDetails [Java Application] C:\Program Files\Java\jc
Personal Details:
ID: 56
Name: Sakib Sadri
Age: 20
Location: Bihar
Address: Fazilpur
```

2. Create a class having two methods addition and multiplication, pass two input parameters to the methods and return added/multiplied values.

```
package company;
public class Calculator {
// Method to perform addition
public int addition(int num1, int num2) {
return num1 + num2;
}
// Method to perform multiplication
public int multiplication(int num1, int num2) {
return num1 * num2;
}
public static void main(String[] args) {
// Create an instance of the Calculator class
Calculator calculator = new Calculator();
// Input values
int value1 = 10;
int value2 = 20;
// Perform addition and multiplication
int sum = calculator.addition(value1, value2);
int product = calculator.multiplication(value1, value2);
// Display the results
System.out.println("Sum: " + sum);
System.out.println("Product: " + product);
}
```

```
}
```

OUTPUT :



3 . Create a class having a method checkOddNumbers, the method will take an array of integer as input and prints odd numbers.

```
package company;
public class OddNumberChecker {
// Method to check and print odd numbers
public static void checkOddNumbers(int[] numbers) {
System.out.println("Odd Numbers:");
for (int num : numbers) {
if (num % 2 != 0) {
System.out.println(num);
}
}
}
public static void main(String[] args) {
// Example usage of the method
int[] array = {1, 2, 3, 4, 5, 6, 7, 8, 9};
checkOddNumbers(array);
}
}
```

Output:

```
Problems  Javadoc  Declaration  Console ×
<terminated> OddNumberChecker [Java Application] C:\Program Files\Java\jdk-18.0.2.1\bin\java
Odd Numbers:
1
3
5
7
9
```

4 . Create a table pesonal_details having attributes (id, name, age, location, address etc)

1. insert at least 3 rows,

```
mysql> INSERT INTO personal_details (name, age, location, address)
    -> VALUES('sakibsadri', 20,'bihar','fazilpur'),
    -> ('sadri', 21,'bihar','shahpur'),
    -> ('ahil', 25,'Patna','Hisar');
Query OK, 3 rows affected (0.23 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> select *from personal_details;
+----+------------+------+----------+----------+
| id | name       | age  | location | address  |
+----+------------+------+----------+----------+
|  1 | sakibsadri |   20 | bihar    | fazilpur |
|  2 | sadri      |   21 | bihar    | shahpur  |
|  3 | ahil       |   25 | Patna    | Hisar    |
+----+------------+------+----------+----------+
3 rows in set (0.03 sec)
```

2. update a row,

```
mysql> UPDATE personal_details
    -> SET age = 31
    -> WHERE name = 'sadri';
Query OK, 1 row affected (0.06 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select *from personal_details;
+----+------------+------+----------+----------+
| id | name       | age  | location | address  |
+----+------------+------+----------+----------+
|  1 | sakibsadri |   20 | bihar    | fazilpur |
|  2 | sadri      |   31 | bihar    | shahpur  |
|  3 | ahil       |   25 | Patna    | Hisar    |
+----+------------+------+----------+----------+
3 rows in set (0.00 sec)
```

3. delete a row

```
mysql> DELETE FROM personal_details
    -> WHERE name = 'ahil';
Query OK, 1 row affected (0.07 sec)

mysql> DELETE FROM personal_details
    -> select *from personal_details;
ERROR 1064 (42000): You have an error in your SQL syntax;
 for the right syntax to use near 'select *from personal_
mysql> select *from personal_details;
+----+------------+------+----------+----------+
| id | name       | age  | location | address  |
+----+------------+------+----------+----------+
|  1 | sakibsadri |   20 | bihar    | fazilpur |
|  2 | sadri      |   31 | bihar    | shahpur  |
+----+------------+------+----------+----------+
2 rows in set (0.00 sec)
```

SET 2

1. Write a Java program to create a class known as "BankAccount" with methods called deposit() and withdraw(). Create a subclass called SavingsAccount that overrides the withdraw() method to prevent withdrawals if the account balance falls below one hundred.

```
class BankAccount {
    private double balance;
    public BankAccount(double initialBalance) {
        balance = initialBalance;
    }
    public void deposit(double amount) {
        if (amount > 0) {
            balance += amount;
```

```
            System.out.println("Deposited $" + amount);
        } else {
            System.out.println("Invalid deposit amount");
        }
    }
    public void withdraw(double amount) {
        if (amount > 0 && balance >= amount) {
            balance -= amount;
            System.out.println("Withdrawn $" + amount);
        } else {
            System.out.println("Insufficient balance or invalid withdrawal amount");
        }
    }
    public double getBalance() {
        return balance;
    }
}
class SavingsAccount extends BankAccount {
    public SavingsAccount(double initialBalance) {
        super(initialBalance);
    }
    @Override
    public void withdraw(double amount) {
        if (getBalance() >= 100 && amount > 0 && getBalance() - amount >= 100) {
            super.withdraw(amount);
        } else {
```

```
            System.out.println("Withdrawal not allowed. Minimum balance of $100 must be maintained.");
        }
    }
}
public class Main {
    public static void main(String[] args) {
        BankAccount account1 = new BankAccount(500.0);
        SavingsAccount account2 = new SavingsAccount(200.0);
        // Test BankAccount
        System.out.println("Bank Account Balance: $" + account1.getBalance());
        account1.deposit(100.0);
        account1.withdraw(50.0);
        account1.withdraw(600.0);
        // Test SavingsAccount
        System.out.println("\nSavings Account Balance: $" + account2.getBalance());
        account2.deposit(50.0);
        account2.withdraw(75.0);
        account2.withdraw(150.0);
    }
}
```

OUTPUT:

Result

CPU Time: 0.16 sec(s), Memory: 33732 kilobyte(s)

```
Bank Account Balance: $500.0
Deposited $100.0
Withdrawn $50.0
Insufficient balance or invalid withdrawal amount

Savings Account Balance: $200.0
Deposited $50.0
Withdrawn $75.0
Withdrawal not allowed. Minimum balance of $100 must be maintained.
```

2 .Write a Java program to create an interface Shape with the getArea() method. Create three classes Rectangle, Circle, and Triangle that implement the Shape interface. Implement the getArea() method for each of the three classes.

```
// Define the Shape interface
interface Shape {
    double getArea();
}
// Implement the Rectangle class
class Rectangle implements Shape {
    private double width;
    private double height;

    public Rectangle(double width, double height) {
        this.width = width;
        this.height = height;
    }
    @Override
    public double getArea() {
        return width * height;
    }
}
// Implement the Circle class
```

```
class Circle implements Shape {
    private double radius;
    public Circle(double radius) {
        this.radius = radius;
    }
    @Override
    public double getArea() {
        return Math.PI * radius * radius;
    }
}
// Implement the Triangle class
class Triangle implements Shape {
    private double base;
    private double height;
    public Triangle(double base, double height) {
        this.base = base;
        this.height = height;
    }
    @Override
    public double getArea() {
        return 0.5 * base * height;
    }
}
public class Main {
    public static void main(String[] args) {
        // Create instances of different shapes and calculate their areas
        Shape rectangle = new Rectangle(5.0, 4.0);
        Shape circle = new Circle(3.0);
        Shape triangle = new Triangle(6.0, 8.0);
```

```
        System.out.println("Rectangle Area: " + rectangle.getArea());
        System.out.println("Circle Area: " + circle.getArea());
        System.out.println("Triangle Area: " + triangle.getArea());
    }
}
```

OUTPUT:

Result

CPU Time: 0.10 sec(s), Memory: 33700 kilobyte(s)

```
Rectangle Area: 20.0
Circle Area: 28.274333882308138
Triangle Area: 24.0
```

3. Write a Java program to create a class Vehicle with a method called speedUp(). Create two subclasses Car and Bicycle. Override the speedUp() method in each subclass to increase the vehicle's speed differently.

```
class Vehicle {
    private double speed;

    public Vehicle(double initialSpeed) {
        this.speed = initialSpeed;
    }

    public void speedUp() {
        System.out.println("Vehicle is accelerating.");
```

```
        speed += 10; // Increase speed by 10 units (generic for all vehicles)
    }

    public double getSpeed() {
        return speed;
    }
}

class Car extends Vehicle {
    public Car(double initialSpeed) {
        super(initialSpeed);
    }

    @Override
    public void speedUp() {
        System.out.println("Car is accelerating.");
        super.speedUp(); // Increase car speed by 10 units
    }
}

class Bicycle extends Vehicle {
    public Bicycle(double initialSpeed) {
        super(initialSpeed);
    }

    @Override
    public void speedUp() {
        System.out.println("Bicycle is pedaling faster.");
        super.speedUp(); // Increase bicycle speed by 10 units
```

```
    }
}

public class Main {
    public static void main(String[] args) {
        Vehicle vehicle = new Vehicle(0);
        Car car = new Car(0);
        Bicycle bicycle = new Bicycle(0);
        System.out.println("Initial speeds:");
        System.out.println("Vehicle speed: " + vehicle.getSpeed());
        System.out.println("Car speed: " + car.getSpeed());
        System.out.println("Bicycle speed: " + bicycle.getSpeed());
        // Accelerate the vehicles
        vehicle.speedUp();
        car.speedUp();
        bicycle.speedUp();
        System.out.println("\nSpeed after acceleration:");
        System.out.println("Vehicle speed: " + vehicle.getSpeed());
        System.out.println("Car speed: " + car.getSpeed());
        System.out.println("Bicycle speed: " + bicycle.getSpeed());
    }
}
```

OUTPUT:

CPU Time: 0.14 sec(s), Memory: 33900 kilobyte(s)

```
Initial speeds:
Vehicle speed: 0.0
Car speed: 0.0
Bicycle speed: 0.0
Vehicle is accelerating.
Car is accelerating.
Vehicle is accelerating.
Bicycle is pedaling faster.
Vehicle is accelerating.

Speed after acceleration:
Vehicle speed: 10.0
Car speed: 10.0
Bicycle speed: 10.0
```

4.**Employee(EmpID,EmpFname,EmpLname,Department,Project,Address,DOB,Gender)**

```
mysql> CREATE TABLE Employee (
    ->     EmpID INT AUTO_INCREMENT PRIMARY KEY,
    ->     EmpFname VARCHAR(50),
    ->     EmpLname VARCHAR(50),
    ->     Department VARCHAR(50),
    ->     Project VARCHAR(50),
    ->     Address VARCHAR(100),
    ->     DOB DATE,
    ->     Gender VARCHAR(10)
    -> );
Query OK, 0 rows affected (0.30 sec)

mysql> -- Insert data into the Employee table
mysql> INSERT INTO Employee (EmpFname, EmpLname, Department, Project, Address, DOB, Gender)
    -> VALUES
    ->     ('sakib', 'sadri', 'HR', 'Project A', 'bihar', '2002-10-15', 'Male'),
    ->     ('Ahil', 'sadri', 'HR', 'Project B', 'Delhi', '2003-10-15', 'Male'),
    -> ('Alice', 'Sadri', 'HR', 'Project C', 'Bihar', '2000-12-03', 'Female'),
    ->     ('Sabir', 'sadri', 'HR', 'Project D', 'Delhi', '2003-10-15', 'Male');
Query OK, 4 rows affected (0.41 sec)
Records: 4  Duplicates: 0  Warnings: 0
```

```
mysql> select *from Employee;
+-------+----------+----------+------------+-----------+---------+------------+--------+
| EmpID | EmpFname | EmpLname | Department | Project   | Address | DOB        | Gender |
+-------+----------+----------+------------+-----------+---------+------------+--------+
|     1 | sakib    | sadri    | HR         | Project A | bihar   | 2002-10-15 | Male   |
|     2 | Ahil     | sadri    | HR         | Project B | Delhi   | 2003-10-15 | Male   |
|     3 | Alice    | Sadri    | HR         | Project C | Bihar   | 2000-12-03 | Female |
|     4 | Sabir    | sadri    | HR         | Project D | Delhi   | 2003-10-15 | Male   |
+-------+----------+----------+------------+-----------+---------+------------+--------+
4 rows in set (0.00 sec)
```

* **EmployeePosition(EmpID,EmpPosition,DateOfJoining,Salary)**

```
mysql> CREATE TABLE EmployeePosition (
    ->     EmpID INT PRIMARY KEY,
    ->     EmpPosition VARCHAR(50),
    ->     DateOfJoining DATE,
    ->     Salary DECIMAL(10, 2),
    ->     FOREIGN KEY (EmpID) REFERENCES Employee(EmpID)
    -> );
Query OK, 0 rows affected (0.55 sec)

mysql> INSERT INTO EmployeePosition (EmpID, EmpPosition, DateOfJoining, Salary)
    -> VALUES
    ->     (1, 'Manager', '2010-02-15', 75000.00),
    ->     (2, 'Developer', '2015-06-10', 60000.00),
    ->     (3, 'Accountant', '2013-09-05', 55000.00),
    ->     (4, 'Manager', '2011-04-20', 72000.00);
Query OK, 4 rows affected (0.20 sec)
Records: 4  Duplicates: 0  Warnings: 0
```

```
mysql> select *from EmployeePosition;
+-------+-------------+---------------+----------+
| EmpID | EmpPosition | DateOfJoining | Salary   |
+-------+-------------+---------------+----------+
|     1 | Manager     | 2010-02-15    | 75000.00 |
|     2 | Developer   | 2015-06-10    | 60000.00 |
|     3 | Accountant  | 2013-09-05    | 55000.00 |
|     4 | Manager     | 2011-04-20    | 72000.00 |
+-------+-------------+---------------+----------+
4 rows in set (0.00 sec)
```

4.1 Write a query to fetch the EmpFname from the EmployeeInfo table in upper case and use the ALIAS name as EmpName.

```
mysql> SELECT UPPER(EmpFname) AS EmpName
    -> FROM Employee;
+---------+
| EmpName |
+---------+
| SAKIB   |
| AHIL    |
| ALICE   |
| SABIR   |
+---------+
4 rows in set (0.07 sec)
```

4 .2.Write a query to fetch the number of employees working in the department 'HR'.

```
mysql> SELECT COUNT(*) AS NumberOfEmployeesInHR
    -> FROM Employee
    -> WHERE Department = 'HR';
+----------------------+
| NumberOfEmployeesInHR |
+----------------------+
|                    4 |
+----------------------+
1 row in set (0.04 sec)
```

4.3. Write a query to fetch all employees who also hold the managerial position.

```
mysql> SELECT E.*
    -> FROM Employee E
    -> INNER JOIN EmployeePosition EP ON E.EmpID = EP.EmpID
    -> WHERE EP.EmpPosition = 'Manager';
+-------+----------+----------+------------+-----------+---------+------------+--------+
| EmpID | EmpFname | EmpLname | Department | Project   | Address | DOB        | Gender |
+-------+----------+----------+------------+-----------+---------+------------+--------+
|     1 | sakib    | sadri    | HR         | Project A | bihar   | 2002-10-15 | Male   |
|     4 | Sabir    | sadri    | HR         | Project D | Delhi   | 2003-10-15 | Male   |
+-------+----------+----------+------------+-----------+---------+------------+--------+
2 rows in set (0.06 sec)
```

SET – 3

1. Create a list of integers, iterrate over the list and find the maximum number in it.

```
import java.util.ArrayList;

import java.util.List;

public class FindMaximumNumber {

    public static void main(String[] args) {

        // Create a list of integers

        List<Integer> numbers = new ArrayList<>();

        numbers.add(10);

        numbers.add(5);

        numbers.add(17);

        numbers.add(8);

        numbers.add(25);


        // Initialize the maximum number with the first element in the list

        int maxNumber = numbers.get(0);
```

```
        // Iterate over the list to find the maximum number
        for (int i = 1; i < numbers.size(); i++) {
            int currentNumber = numbers.get(i);
            if (currentNumber > maxNumber) {
                maxNumber = currentNumber;
            }
        }

        // Print the maximum number
        System.out.println("The maximum number is: " + maxNumber);
    }
}
```

Output:

Result

CPU Time: 0.07 sec(s), Memory: 33396 kilobyte(s)

```
The maximum number is: 25
```

2. Create class Employee{name,age,salary,department}, store 10 employees in a list, print the details of each employee

```
import java.util.ArrayList;
import java.util.List;

class Employee {
    private String name;
```

```
private int age;
private double salary;
private String department;

public Employee(String name, int age, double salary, String department) {
    this.name = name;
    this.age = age;
    this.salary = salary;
    this.department = department;
}

public String getName() {
    return name;
}

public int getAge() {
    return age;
}

public double getSalary() {
    return salary;
}

public String getDepartment() {
    return department;
}
```

```
    @Override
    public String toString() {
        return "Employee [name=" + name + ", age=" + age + ", salary=" + salary + ", department="
+ department + "]";
    }
}

public class Main {
    public static void main(String[] args) {
        // Create a list to store employees
        List<Employee> employeeList = new ArrayList<>();

        // Add 10 employees to the list
        employeeList.add(new Employee("John", 30, 50000.0, "HR"));
        employeeList.add(new Employee("Alice", 28, 55000.0, "Engineering"));
        employeeList.add(new Employee("Bob", 35, 60000.0, "Finance"));
        employeeList.add(new Employee("Mary", 32, 52000.0, "HR"));
        employeeList.add(new Employee("David", 27, 58000.0, "Engineering"));
        employeeList.add(new Employee("Sarah", 33, 62000.0, "Finance"));
        employeeList.add(new Employee("James", 29, 53000.0, "HR"));
        employeeList.add(new Employee("Linda", 31, 59000.0, "Engineering"));
        employeeList.add(new Employee("Michael", 34, 61000.0, "Finance"));
        employeeList.add(new Employee("Emily", 26, 54000.0, "HR"));
        // Print the details of each employee
        for (Employee employee : employeeList) {
            System.out.println(employee);
        }
```

```
    }
}
```

OUTPUT:

Result

CPU Time: 0.19 sec(s), Memory: 35912 kilobyte(s)

```
Employee [name=John, age=30, salary=50000.0, department=HR]
Employee [name=Alice, age=28, salary=55000.0, department=Engineering]
Employee [name=Bob, age=35, salary=60000.0, department=Finance]
Employee [name=Mary, age=32, salary=52000.0, department=HR]
Employee [name=David, age=27, salary=58000.0, department=Engineering]
Employee [name=Sarah, age=33, salary=62000.0, department=Finance]
Employee [name=James, age=29, salary=53000.0, department=HR]
Employee [name=Linda, age=31, salary=59000.0, department=Engineering]
Employee [name=Michael, age=34, salary=61000.0, department=Finance]
Employee [name=Emily, age=26, salary=54000.0, department=HR]
```

3 .Find the employee with maximum salary

```
import java.util.ArrayList;

import java.util.List;


class Employee {

    private String name;

    private int age;

    private double salary;

    private String department;


    public Employee(String name, int age, double salary, String department) {

        this.name = name;

        this.age = age;

        this.salary = salary;

        this.department = department;

    }
```

```
    // Getter methods for attributes (name, age, salary, department)
    public String getName() {
        return name;
    }

    public int getAge() {
        return age;
    }

    public double getSalary() {
        return salary;
    }

    public String getDepartment() {
        return department;
    }

    @Override
    public String toString() {
        return "Employee [name=" + name + ", age=" + age + ", salary=" + salary + ", department=" +
department + "]";
    }
}

public class EmployeeMain {
    public static void main(String[] args) {
        // Create a list to store employees
        List<Employee> employees = new ArrayList<>();
```

```
    // Add 10 employees to the list
    employees.add(new Employee("John", 30, 60000.0, "HR"));
    employees.add(new Employee("Alice", 28, 55000.0, "Engineering"));
    employees.add(new Employee("Bob", 35, 70000.0, "Finance"));
    employees.add(new Employee("Mary", 32, 62000.0, "HR"));
    employees.add(new Employee("David", 27, 58000.0, "Engineering"));
    employees.add(new Employee("Sarah", 33, 75000.0, "Finance"));
    employees.add(new Employee("James", 29, 64000.0, "HR"));
    employees.add(new Employee("Linda", 31, 71000.0, "Engineering"));
    employees.add(new Employee("Michael", 34, 72000.0, "Finance"));
    employees.add(new Employee("Emily", 26, 59000.0, "HR"));

    // Print the details of each employee
    System.out.println("Details of each employee:");
    for (Employee employee : employees) {
        System.out.println(employee);
    }

    // Find the employee with the maximum salary
    Employee maxSalaryEmployee = findEmployeeWithMaxSalary(employees);
    if (maxSalaryEmployee != null) {
        System.out.println("\nEmployee with Maximum Salary:");
        System.out.println(maxSalaryEmployee);
    } else {
        System.out.println("\nNo employees found.");
    }
}
```

```
    // Function to find the employee with the maximum salary
    public static Employee findEmployeeWithMaxSalary(List<Employee> employees) {
        if (employees.isEmpty()) {
            return null;
        }

        Employee maxSalaryEmployee = employees.get(0);
        double maxSalary = maxSalaryEmployee.getSalary();

        for (Employee employee : employees) {
            double salary = employee.getSalary();
            if (salary > maxSalary) {
                maxSalary = salary;
                maxSalaryEmployee = employee;
            }
        }

        return maxSalaryEmployee;
    }
}
```

Output :

Result

CPU Time: 0.12 sec(s), Memory: 35476 kilobyte(s)

```
Details of each employee:
Employee [name=John, age=30, salary=60000.0, department=HR]
Employee [name=Alice, age=28, salary=55000.0, department=Engineering]
Employee [name=Bob, age=35, salary=70000.0, department=Finance]
Employee [name=Mary, age=32, salary=62000.0, department=HR]
Employee [name=David, age=27, salary=58000.0, department=Engineering]
Employee [name=Sarah, age=33, salary=75000.0, department=Finance]
Employee [name=James, age=29, salary=64000.0, department=HR]
Employee [name=Linda, age=31, salary=71000.0, department=Engineering]
Employee [name=Michael, age=34, salary=72000.0, department=Finance]
Employee [name=Emily, age=26, salary=59000.0, department=HR]

Employee with Maximum Salary:
Employee [name=Sarah, age=33, salary=75000.0, department=Finance]
```

SQL: Table Employee{name,age,salary,department}

```
mysql> CREATE TABLE Employee (
    ->     name VARCHAR(255),
    ->     age INT,
    ->     salary DECIMAL(10, 2),
    ->     department VARCHAR(255)
    -> );
Query OK, 0 rows affected (0.58 sec)

mysql>
mysql> INSERT INTO Employee (name, age, salary, department)
    -> VALUES ('John Doe', 30, 60000.00, 'HR');
Query OK, 1 row affected (0.22 sec)

mysql>  INSERT INTO Employee (name, age, salary, department) VALUES ('Sakib', 20, 80000.00, 'HR');
Query OK, 1 row affected (0.15 sec)

mysql>  INSERT INTO Employee (name, age, salary, department) VALUES ('Sadri', 24, 50000.00, 'Engineering');
Query OK, 1 row affected (0.08 sec)

mysql>  INSERT INTO Employee (name, age, salary, department) VALUES ('Ahil', 44,900000.00, 'IT');
Query OK, 1 row affected (0.08 sec)
```

```
mysql> select * from Employee;
+----------+------+-----------+-------------+
| name     | age  | salary    | department  |
+----------+------+-----------+-------------+
| John Doe |   30 |  60000.00 | HR          |
| Sakib    |   20 |  80000.00 | HR          |
| Sadri    |   24 |  50000.00 | Engineering |
| Ahil     |   44 | 900000.00 | IT          |
+----------+------+-----------+-------------+
4 rows in set (0.00 sec)
```

1. Find the employee with maximum salary

```
mysql> SELECT *
    -> FROM Employee
    -> WHERE salary = (SELECT MAX(salary) FROM Employee);
+------+------+-----------+------------+
| name | age  | salary    | department |
+------+------+-----------+------------+
| Ahil |   44 | 900000.00 | IT         |
+------+------+-----------+------------+
1 row in set (0.12 sec)
```

2. Calculate the average age of employees in each department:

```
mysql> SELECT department, AVG(age) AS average_age
    -> FROM Employee
    -> GROUP BY department;
+-------------+-------------+
| department  | average_age |
+-------------+-------------+
| HR          |     25.0000 |
| Engineering |     24.0000 |
| IT          |     44.0000 |
+-------------+-------------+
3 rows in set (0.10 sec)
```

SET – 4

1. Creat a hashmap to store data of roll number and marks <Inteeger, Double>, add details of 10 students and print them.

```
import java.util.HashMap;
import java.util.Map;

public class StudentDetails {
    public static void main(String[] args) {
        // Create a HashMap to store student details (Roll Number -> Marks)
        Map<Integer, Double> studentMap = new HashMap<>();

        // Add details of 10 students
        studentMap.put(101, 85.5);
        studentMap.put(102, 78.0);
        studentMap.put(103, 92.5);
        studentMap.put(104, 67.5);
        studentMap.put(105, 89.0);
        studentMap.put(106, 76.5);
        studentMap.put(107, 94.5);
        studentMap.put(108, 71.0);
        studentMap.put(109, 88.5);
        studentMap.put(110, 79.5);

        // Print the student details
        System.out.println("Student Details:");
        for (Map.Entry<Integer, Double> entry : studentMap.entrySet()) {
            int rollNumber = entry.getKey();
            double marks = entry.getValue();
            System.out.println("Roll Number: " + rollNumber + ", Marks: " + marks);
```

```
        }
    }
}
```

Output :

Result

CPU Time: 0.12 sec(s), Memory: 34432 kilobyte(s)

```
Student Details:
Roll Number: 101, Marks: 85.5
Roll Number: 102, Marks: 78.0
Roll Number: 103, Marks: 92.5
Roll Number: 104, Marks: 67.5
Roll Number: 105, Marks: 89.0
Roll Number: 106, Marks: 76.5
Roll Number: 107, Marks: 94.5
Roll Number: 108, Marks: 71.0
Roll Number: 109, Marks: 88.5
Roll Number: 110, Marks: 79.5
```

2 . Create a map to store details of employees {empId, empName, salary, address},add details of 10 employees and print them.

```
import java.util.HashMap;

import java.util.Map;

public class EmployeeDetails {

    public static void main(String[] args) {

        // Create a HashMap to store employee details

        Map<Integer, Employee> employeeMap = new HashMap<>();

        // Add details of 10 employees

        employeeMap.put(101, new Employee(101, "Sakib", 50000.0, "Bihar"));

        employeeMap.put(102, new Employee(102, "Ahil", 60000.0, "Delhi"));

        employeeMap.put(103, new Employee(103, "Sahil", 55000.0, "Sam"));

        employeeMap.put(104, new Employee(104, "Nishant", 52000.0, "USE"));

        employeeMap.put(105, new Employee(105, "Abhinay", 62000.0, "Goa"));

        employeeMap.put(106, new Employee(106, "Ankit", 53000.0, "Fazilpur"));

        employeeMap.put(107, new Employee(107, "abhi", 48000.0, "Ammk"));
```

```
        employeeMap.put(108, new Employee(108, "Sohan", 54000.0, "Ramki"));
        employeeMap.put(109, new Employee(109, "Nitn", 58000.0, "dkskfr"));
        employeeMap.put(110, new Employee(110, "payal", 51000.0, "sksl"));
        // Print the employee details
        System.out.println("Employee Details:");
        for (Map.Entry<Integer, Employee> entry : employeeMap.entrySet()) {
            int empId = entry.getKey();
            Employee employee = entry.getValue();
            System.out.println("Employee ID: " + empId);
            System.out.println("Employee Name: " + employee.getEmpName());
            System.out.println("Salary: " + employee.getSalary());
            System.out.println("Address: " + employee.getAddress());
            System.out.println();
        }
    }
}
class Employee {
    private int empId;
    private String empName;
    private double salary;
    private String address;
    public Employee(int empId, String empName, double salary, String address) {
        this.empId = empId;
        this.empName = empName;
        this.salary = salary;
        this.address = address;
    }

    public int getEmpId() {
```

```
        return empId;
    }
    public String getEmpName() {
        return empName;
    }
    public double getSalary() {
        return salary;
    }
    public String getAddress() {
        return address;
    }
}
```

OUTPUT :

CPU Time: 0.16 sec(s), Memory: 35212 kilobyte(s)

```
Employee Details:
Employee ID: 101
Employee Name: Sakib
Salary: 50000.0
Address: Bihar

Employee ID: 102
Employee Name: Ahil
Salary: 60000.0
Address: Delhi

Employee ID: 103
Employee Name: Sahil
Salary: 55000.0
Address: Sam

Employee ID: 104
Employee Name: Nishant
Salary: 52000.0
Address: USE

Employee ID: 105
Employee Name: Abhinay
Salary: 62000.0
Address: Goa

Employee ID: 106
Employee Name: Ankit
Salary: 53000.0
Address: Fazilpur

Employee ID: 107
Employee Name: abhi
Salary: 48000.0
Address: Ammk

Employee ID: 108
Employee Name: Sohan
Salary: 54000.0
Address: Ramki

Employee ID: 109
Employee Name: Nitn
Salary: 58000.0
Address: dkskfr

Employee ID: 110
Employee Name: payal
Salary: 51000.0
Address: sksl
```

3 . Write a function that takes a list of integers as input and returns the sum, average, maximum, and minimum values of the list.

```
import java.util.List;

import java.util.Collections;


public class ListStats {

   public static void main(String[] args) {

      // Example list of integers

      List<Integer> numbers = List.of(5, 10, 15, 20, 25);


      // Calculate and print the statistics
```

```
        calculateStatistics(numbers);
    }
    public static void calculateStatistics(List<Integer> numbers) {
        if (numbers.isEmpty()) {
            System.out.println("The list is empty.");
            return;
        }
        // Calculate the sum
        int sum = 0;
        for (int number : numbers) {
            sum += number;
        }
        // Calculate the average
        double average = (double) sum / numbers.size();
        // Find the maximum and minimum values
        int max = Collections.max(numbers);
        int min = Collections.min(numbers);
        // Print the results
        System.out.println("Sum: " + sum);
        System.out.println("Average: " + average);
        System.out.println("Maximum: " + max);
        System.out.println("Minimum: " + min);
    }
```

Output:

Result

CPU Time: 0.16 sec(s), Memory: 35016 kilobyte(s)

```
Sum: 75
Average: 15.0
Maximum: 25
Minimum: 5
```

SQL:

1. Retrieve the names of all employees along with the names of their respective departments from the employees and departments tables using an inner join.

Create Department table:

```
mysql> CREATE TABLE departments (
    ->     department_id INT PRIMARY KEY,
    ->     department_name VARCHAR(255)
    -> );
Query OK, 0 rows affected (0.30 sec)

mysql> INSERT INTO departments (department_id, department_name)
    -> VALUES
    ->     (1, 'HR'),
    ->     (2, 'Finance'),
    ->     (3, 'IT'),
    ->     (4, 'Marketing');
Query OK, 4 rows affected (0.13 sec)
Records: 4  Duplicates: 0  Warnings: 0
```

Create Employees table:

```
mysql> CREATE TABLE employees3 (
    ->     employee_id INT PRIMARY KEY,
    ->     employee_name VARCHAR(255),
    ->     department_id INT,
    ->     FOREIGN KEY (department_id) REFERENCES departments(department_id)
    -> );
Query OK, 0 rows affected (0.92 sec)

mysql> INSERT INTO employees3 (employee_id, employee_name, department_id)
    -> VALUES
    ->     (101, 'Sakib', 1),
    ->     (102, 'Sadri', 2),
    ->     (103, 'Sahil', 3),
    ->     (104, 'Sakshi', 1),
    ->     (105, 'Garima', 4);
Query OK, 5 rows affected (0.12 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

Perform the Inner join operation

```
mysql> SELECT e.employee_name, d.department_name
    -> FROM employees3 e
    -> INNER JOIN departments d ON e.department_id = d.department_id;
+---------------+-----------------+
| employee_name | department_name |
+---------------+-----------------+
| Sakib         | HR              |
| Sakshi        | HR              |
| Sadri         | Finance         |
| Sahil         | IT              |
| Garima        | Marketing       |
+---------------+-----------------+
5 rows in set (0.02 sec)

mysql> _
```

2 Retrieve a list of all customers and their corresponding orders, including customers who have not placed any orders. Use a left join between the customers and orders tables.

Create Customer Table:

```
mysql> CREATE TABLE customers (
    ->     customer_id INT PRIMARY KEY,
    ->     customer_name VARCHAR(255)
    -> );
Query OK, 0 rows affected (0.26 sec)
```

```
mysql> INSERT INTO customers (customer_id, customer_name)
    -> VALUES
    ->     (1, 'Sakib'),
    ->     (2, 'Sadri'),
    ->     (3, 'Sahil');
Query OK, 3 rows affected (0.17 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

Create Orders table:

```
mysql> CREATE TABLE orders (
    ->     order_id INT PRIMARY KEY,
    ->     customer_id INT,
    ->     order_date DATE,
    ->     FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
    -> );
Query OK, 0 rows affected (0.30 sec)
```

```
mysql> INSERT INTO orders (order_id, customer_id, order_date)
    -> VALUES
    ->     (101, 1, '2023-09-01'),
    ->     (102, 1, '2023-09-05'),
    ->     (103, 2, '2023-09-02');
Query OK, 3 rows affected (0.10 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

Perform the Left join operation:

```
mysql> SELECT c.customer_id, c.customer_name, o.order_id, o.order_date
    -> FROM customers c
    -> LEFT JOIN orders o ON c.customer_id = o.customer_id;
+-------------+---------------+----------+------------+
| customer_id | customer_name | order_id | order_date |
+-------------+---------------+----------+------------+
|           1 | Sakib         |      101 | 2023-09-01 |
|           1 | Sakib         |      102 | 2023-09-05 |
|           2 | Sadri         |      103 | 2023-09-02 |
|           3 | Sahil         |     NULL | NULL       |
+-------------+---------------+----------+------------+
4 rows in set (0.06 sec)
```

3 Retrieve the names of all employees along with the names of their managers. Use a self join on the employees table to achieve this.

```
mysql> CREATE TABLE employees (
    ->     employee_id INT PRIMARY KEY,
    ->     employee_name VARCHAR(255),
    ->     manager_id INT,
    ->     FOREIGN KEY (manager_id) REFERENCES employees(employee_id)
    -> );
Query OK, 0 rows affected (0.32 sec)

mysql> INSERT INTO employees (employee_id, employee_name, manager_id)
    -> VALUES
    ->     (1, 'Sakib', NULL),
    ->     (2, 'Abhi', 1),
    ->     (3, 'Subham', 1),
    ->     (4, 'Sahil', 2),
    ->     (5, 'Kunal', NULL);
Query OK, 5 rows affected (0.14 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

```
mysql> SELECT e.employee_name AS EmployeeName, m.employee_name AS ManagerNam
    -> FROM employees e
    -> LEFT JOIN employees m ON e.manager_id = m.employee_id;
+--------------+-------------+
| EmployeeName | ManagerName |
+--------------+-------------+
| Sakib        | NULL        |
| Abhi         | Sakib       |
| Subham       | Sakib       |
| Sahil        | Abhi        |
| Kunal        | NULL        |
+--------------+-------------+
5 rows in set (0.03 sec)
```