1.

```
public void addNewClient(Client client){
        Connection connect;
        Statement statement;
        String query;
        try {
            connect = DB_Connection.getConnection();
            statement = connect.createStatement();
            query = "INSERT INTO ClientRecordings (Firstname, Lastname, Address, Birthdate,
DriverLicenseNumber " +
                    "/*CreditCardNumber, CreditCardExpirationDate, Cvv*/) VALUES ('" +
                    client.getFirstname() + "', '" +
                    client.getLastname() + "','" +
                    client.getAddress() + "', '" +
                    client.getBirthdate() + "', '" +
                    client.getDriverLicenseNumber() /*+  "', '" +
                    client.getCreditCardNumber() + "', '" +
                    client.getCreditCardExpirationDate() + "', '" +
                    client.getCvv() */+"')";
            System.out.println(query);
            statement.execute(query);
            statement.close();
            System.out.println("Client added successfully");
            connect.close();//maybe problematic
        }
        catch (SQLException ex){
            ex.printStackTrace(System.err);
        }
    }
```

2.

```
public void addNewCar(Car newCar){
        Connection connect;
        Statement statement;
        try {
            connect = DB_Connection.getConnection();
            statement = connect.createStatement();
            String query = "INSERT INTO CarRecordings (Brand, Model, Color, DistanceRange, RentPrice,
InsurancePrice, CarType, Passengers, LicenseNumber) VALUES ('" +
                    newCar.getBrand() + "', '" +
                    newCar.getModel() + "', '" +
                    newCar.getColor() + "', " +
                    newCar.getRange() + ", " +
                    newCar.getRentPrice() + ", " +
                    newCar.getInsurancePrice() + ", '" +
                    newCar.getCarType() + "', " +
                    newCar.getPassengers() +",'"+
                    newCar.getLicenseNumber() +"')";
            statement.execute(query);
            statement.close();
            connect.close();
        }
        catch (SQLException ex){
            ex.printStackTrace(System.err);
        }
    }

public void addNewEBikeRecording(eBike newBike) {
        Connection connect = null;
        Statement statement = null;
        String query = null;
        try {
```

```
            connect = DB_Connection.getConnection();
            statement = connect.createStatement();
            query = "INSERT INTO eBikeRecordings (Brand, Model, Color, DistanceRange, " +
                    "RentPrice, InsurancePrice) VALUES ('" +
                    newBike.getBrand() + "', '" +
                    newBike.getModel() + "', '" +
                    newBike.getColor() + "', " +
                    newBike.getRange() + ", " +
                    newBike.getRentPrice() + ", " +
                    newBike.getInsurancePrice() + ")";
            statement.execute(query);
            statement.close();
            connect.close();//maybe problematic
        }
        catch (SQLException ex){
            ex.printStackTrace(System.err);
        }
    }

public void addNewMotorcycle(Motorcycle newMoto){
        Connection connect = null;
        Statement statement = null;
        String query = null;
        try {
            connect = DB_Connection.getConnection();
            statement = connect.createStatement();
            query = "INSERT INTO MotorcycleRecordings (Brand, Model, Color, DistanceRange," +
                    " RentPrice, InsurancePrice, LicenseNumber) VALUES ('" +
                    newMoto.getBrand() + "','" +
                    newMoto.getModel() + "','" +
                    newMoto.getColor() + "','" +
```

```
                newMoto.getRange() + "','" +
                newMoto.getRentPrice() + "','" +
                newMoto.getInsurancePrice() +"','"+
                newMoto.getLicenseNumber() +")";
            statement.execute(query);
            statement.close();
            connect.close();//maybe problematic
        }
        catch (SQLException ex){
            ex.printStackTrace(System.err);
        }
    }

public void addNewScooter(Scooter newScooter){
        Connection connect = null;
        Statement statement = null;
        String query = null;
        try {
            connect = DB_Connection.getConnection();
            statement = connect.createStatement();
            query = "INSERT INTO ScooterRecordings (Brand, Model, Color, DistanceRange, RentPrice, " +
                "InsurancePrice) VALUES ('" +
                newScooter.getBrand() + "', '" +
                newScooter.getModel() + "', '" +
                newScooter.getColor() + "', " +
                newScooter.getRange() + ", " +
                newScooter.getRentPrice() + ", " +
                newScooter.getInsurancePrice() + ")";
            statement.execute(query);
            statement.close();
            connect.close();//maybe problematic
```

```
        }
        catch (SQLException ex){
            ex.printStackTrace(System.err);
        }
    }
```

3.

```
public ArrayList<Vehicle> getAvailableVehicles() {
        Connection con = DB_Connection.getConnection();
        Statement stmt;
        String query1 = "SELECT * FROM CarRecordings WHERE NOT EXISTS (SELECT * FROM
RentalRecordings WHERE CarRecordings.LicenseNumber = RentalRecordings.VehicleID) AND ";
        String query2 = "SELECT * FROM eBikeRecordings WHERE NOT EXISTS (SELECT * FROM
RentalRecordings WHERE eBikeRecordings.BikeID = RentalRecordings.VehicleID) AND";
        String query3 = "SELECT * FROM MotorcycleRecordings WHERE NOT EXISTS (SELECT * FROM
RentalRecordings WHERE MotorcycleRecordings.LicenseNumber = RentalRecordings.VehicleID)
AND";
        String query4 = "SELECT * FROM ScooterRecordings WHERE NOT EXISTS (SELECT * FROM
RentalRecordings WHERE ScooterRecordings.ScooterID = RentalRecordings.VehicleID)";
        ArrayList<Vehicle> allAvailableVehicles = new ArrayList<>();
        ResultSet rs;
        try {
            stmt = con.createStatement();
            rs = stmt.executeQuery(query1+query2+query3+query4);
            while (rs.next()) {
                String json = DB_Connection.getResultsToJSON(rs);
                Gson gson = new Gson();
                Vehicle veh = gson.fromJson(json, Vehicle.class);
                allAvailableVehicles.add(veh);
            }
        }
            catch(SQLException e){
```

```
            e.printStackTrace(System.err);
        }
        if (!allAvailableVehicles.isEmpty()) {
            return allAvailableVehicles;
        } else {
            System.out.println("There are no available vehicles");
        }
        return null;
    }
```

4.

```
public void addNewRental(Rental rental){
        Connection connect;
        PreparedStatement statement;
        Statement stmt;
        String query = null;
        ResultSet rs=null;
        try {
            connect = DB_Connection.getConnection();
            query = "INSERT INTO RentalRecordings (CustomerName, RentalDate, ReturnDate,
VechicleType, PaymentAmount, RenterID, VehicleID, assignedDriverID) VALUES ('" +
                    rental.getCustomerName() + "', '" +
                    rental.getRentalDate() + "', '" +
                    rental.getReturnDate() + "', '" +
                    rental.getPaymentAmount() + "', '" +
                    rental.getVehicleType() + "', '" +
                    rental.getRenterID() + "', '" +
                    rental.getVehicleID() + "', '" +
                    rental.getAssignedDriverID() + "')";
            System.out.println(query);
```

```
        statement = connect.prepareStatement(query, Statement.RETURN_GENERATED_KEYS);
        statement.execute();
        rs = statement.getGeneratedKeys();
        if (rs.next()) {
            rental.setRentalID(rs.getInt(1));
        }
        statement.close();
        if(rental.getVehicleType().equals("Car"))
            query= "UPDATE carrecordings SET TimesRented = TimesRented + 1 WHERE
LicenseNumber= " + rental.getVehicleID();
        else if(rental.getVehicleType().equals("Motorcycle"))
            query= "UPDATE MotorcycleRecordings SET TimesRented = TimesRented + 1 WHERE
LicenseNumber= " + rental.getVehicleID();
        else if(rental.getVehicleType().equals("eBike"))
            query= "UPDATE eBikeRecordings SET TimesRented = TimesRented + 1 WHERE BikeID= " +
rental.getVehicleID();
        else if(rental.getVehicleType().equals("Scooter"))
            query= "UPDATE ScooterRecordings SET TimesRented = TimesRented + 1 WHERE
ScooterID= " + rental.getVehicleID();
        stmt= connect.createStatement();
        stmt.execute(query);
        System.out.println("Rental added successfully");
        connect.close();//maybe problematic
      }
      catch (Exception ex){
        ex.printStackTrace(System.err);
      }
   }
```

5.

```
public int returnVehicle(Date actualReturnDate, Rental rental) {
```

```
    Connection connect;
    Statement statement;
    int diffHours=0;
    String query = null;
    try {
       connect = DB_Connection.getConnection();
       statement = connect.createStatement();
       query = "DELETE FROM RentalRecordings WHERE RentalID="+rental.getRentalID();
       statement.execute(query);
       statement.close();
       if(rental.getReturnDate().getTime()<actualReturnDate.getTime()){
          System.out.println("You have exceeded the rental duration");
          Calendar startCalendar = new GregorianCalendar();
          startCalendar.setTime(rental.getReturnDate());
          Calendar endCalendar = new GregorianCalendar();
          endCalendar.setTime(actualReturnDate);
          long diffMillis = endCalendar.getTimeInMillis() - startCalendar.getTimeInMillis();
          diffHours = (int) diffMillis / (60 * 60 * 1000);
       }
       System.out.println("Rental removed successfully");
       connect.close();
    } catch (SQLException e) {
       e.printStackTrace(System.err);
    }
    return diffHours;
  }
```

6.

```
public void addNewDamagedVehicle(int vehicleID, Class<?> VehicleType){
    Connection connect;
```

```
        Statement statement;
        String query;
        try {
            connect = DB_Connection.getConnection();
            statement = connect.createStatement();
            query = "INSERT INTO DamagedVehicles (VehicleID) VALUES ('" + vehicleID + "')";
            statement.execute(query);
            query ="DELETE FROM RentalRecordings WHERE VehicleID="+vehicleID;
            statement.execute(query);
            if (VehicleType.equals(Car.class) || VehicleType.equals(Motorcycle.class)){
                query = "DELETE FROM CarRecordings WHERE LicenseNumber=" + vehicleID;
            }
            else if(VehicleType.equals(eBike.class)){
                query = "DELETE FROM eBikeRecordings WHERE BikeID=" + vehicleID;
            }
            else if(VehicleType.equals(Scooter.class)){
                query = "DELETE FROM ScooterRecordings WHERE ScooterID=" + vehicleID;
            }
            statement.close();
            System.out.println("Damaged Vehicle added successfully");
            connect.close();//maybe problematic
        }
        catch (Exception ex){
            ex.printStackTrace(System.err);
        }
    }
```

7.

```
public void ReportAccident(int vehicleID, int renterID){
        Connection connect;
```

```
        Statement statement;
        String query;
        try {
            connect = DB_Connection.getConnection();
            statement = connect.createStatement();
            query = "INSERT INTO AccidentRecordings (VehicleID, RenterID) VALUES ('" + vehicleID + "', '" + renterID + "')";
            statement.execute(query);
            statement.execute(query);
            statement.close();
            System.out.println("Accident added successfully");
            connect.close();//maybe problematic
        }
        catch (Exception ex){
            ex.printStackTrace(System.err);
        }
    }
```

8.

```
public void addNewMaintenance(int vehicleID, Date MaintenanceDay, boolean isRegularMaintenance){
        Connection connect;
        Statement statement;
        String query;
        Calendar cal = Calendar.getInstance();
        Date AvailableDay;
        cal.setTime(MaintenanceDay);
        try {
            connect = DB_Connection.getConnection();
            statement = connect.createStatement();
            if(isRegularMaintenance) {
```

```
            cal.add(Calendar.DAY_OF_MONTH, 1);
        }
        else{
            cal.add(Calendar.DAY_OF_MONTH, 3);
        }
        AvailableDay=cal.getTime();
        query = "INSERT INTO MaintenanceRecordings (VehicleID,MaintenanceDay,
MaintenanceUntil) VALUES ('" + vehicleID + "','"+MaintenanceDay+"', '" + AvailableDay + "')";
        statement.execute(query);
        statement.close();
        System.out.println("Maintenance added successfully");
        connect.close();//maybe problematic
      }
      catch (Exception ex){
        ex.printStackTrace(System.err);
      }
   }
```

9.

```
public ArrayList<ArrayList<?>> getAvailableOrRentedVehicles(){
   Connection con = DB_Connection.getConnection();
   Statement stmt;
   String query1 = "SELECT * FROM CarRecordings WHERE NOT EXISTS (SELECT * FROM
RentalRecordings WHERE CarRecordings.LicenseNumber = RentalRecordings.VehicleID)";
   String query2 = "SELECT * FROM eBikeRecordings WHERE NOT EXISTS (SELECT * FROM
RentalRecordings WHERE eBikeRecordings.BikeID = RentalRecordings.VehicleID)";
   String query3 = "SELECT * FROM MotorcycleRecordings WHERE NOT EXISTS (SELECT * FROM
RentalRecordings WHERE MotorcycleRecordings.LicenseNumber = RentalRecordings.VehicleID)";
   String query4 = "SELECT * FROM ScooterRecordings WHERE NOT EXISTS (SELECT * FROM
RentalRecordings WHERE ScooterRecordings.ScooterID = RentalRecordings.VehicleID)";
   String query5= "SELECT * FROM RentalRecordings GROUP BY rentalrecordings.VehicleType";
   ArrayList<Car> allAvailableCars = new ArrayList<>();
```

```
ArrayList<Motorcycle> allAvailableMotorcycles = new ArrayList<>();
ArrayList<eBike> allAvailableEBikes = new ArrayList<>();
ArrayList<Scooter> allAvailableScooters = new ArrayList<>();
ArrayList<Vehicle> allRentedVehicles = new ArrayList<>();
ArrayList<ArrayList<?>> allAvailableVehicles = new ArrayList<>();
ResultSet rs;
try {
   stmt = con.createStatement();
   rs = stmt.executeQuery(query1);
   while (rs.next()) {
      String json = DB_Connection.getResultsToJSON(rs);
      Gson gson = new Gson();
      Car veh = gson.fromJson(json, Car.class);
      allAvailableCars.add(veh);
   }
   rs = stmt.executeQuery(query2);
   while (rs.next()) {
      String json = DB_Connection.getResultsToJSON(rs);
      Gson gson = new Gson();
      Motorcycle veh = gson.fromJson(json, Motorcycle.class);
      allAvailableMotorcycles.add(veh);
   }
   rs = stmt.executeQuery(query3);
   while (rs.next()) {
      String json = DB_Connection.getResultsToJSON(rs);
      Gson gson = new Gson();
      eBike veh = gson.fromJson(json, eBike.class);
      allAvailableEBikes.add(veh);
   }
   rs = stmt.executeQuery(query4);
   while (rs.next()) {
```

```
            String json = DB_Connection.getResultsToJSON(rs);
            Gson gson = new Gson();
            Scooter veh = gson.fromJson(json, Scooter.class);
            allAvailableScooters.add(veh);
        }
        rs = stmt.executeQuery(query5);
        while (rs.next()) {
            String json = DB_Connection.getResultsToJSON(rs);
            Gson gson = new Gson();
            Vehicle veh = gson.fromJson(json, Vehicle.class);
            allRentedVehicles.add(veh);
        }
        allAvailableVehicles.add(allAvailableCars);
        allAvailableVehicles.add(allAvailableMotorcycles);
        allAvailableVehicles.add(allAvailableEBikes);
        allAvailableVehicles.add(allAvailableScooters);
        allAvailableVehicles.add(allRentedVehicles);
    }
    catch(SQLException e){
        e.printStackTrace(System.err);
    }
    return allAvailableVehicles;
}
/**Επιστρέφει τη κατάσταση των ενοικιάσεων ανά χρονική περίοδο*/
public ArrayList<Rental> RentalStatus(Date From, Date To){
    Connection con = DB_Connection.getConnection();
    Statement stmt;
    String query = "SELECT * FROM RentalRecordings WHERE RentalDate >= '" + From + "' AND
ReturnDate <='" + To + "'";
    ResultSet rs;
    ArrayList<Rental> rentStatus = new ArrayList<>();
```

```
    try {
        stmt = con.createStatement();
        rs = stmt.executeQuery(query);
        while (rs.next()) {
            String json = DB_Connection.getResultsToJSON(rs);
            Gson gson = new Gson();
            Rental rentStatusJSON = gson.fromJson(json, Rental.class);
            rentStatus.add(rentStatusJSON);
            System.out.println(json);
        }
    }
    catch(SQLException e){
        e.printStackTrace(System.err);
    }
    return rentStatus;
}
public HashMap<String,ArrayList<Integer>> RentalAverages(){
        Connection con = DB_Connection.getConnection();
        ResultSet rs;
        Statement stmt;
        HashMap<String, ArrayList<Integer>> averagesMap = new HashMap<>();
        ArrayList<Integer> statsArray = new ArrayList<>();
        try {
            stmt = con.createStatement();
            String query = "SELECT DISTINCT MAX(DATEDIFF(ReturnDate,RentalDate)),
MIN(DATEDIFF(ReturnDate,RentalDate)),AVG(DATEDIFF(ReturnDate,RentalDate)) FROM
RentalRecordings WHERE VehicleType = 'Car'";
            rs = stmt.executeQuery(query);
            while (rs.next()) {
                statsArray.add(rs.getInt(1));//max
                statsArray.add(rs.getInt(2));//min
                statsArray.add(rs.getInt(3));//avg
```

```
        }
        averagesMap.put("Car", statsArray);
        statsArray.clear();
        query = "SELECT DISTINCT MAX(DATEDIFF(ReturnDate,RentalDate)), MIN(DATEDIFF(ReturnDate,RentalDate)),AVG(DATEDIFF(ReturnDate,RentalDate)) FROM RentalRecordings WHERE VehicleType = 'Motorcycle'";
        rs = stmt.executeQuery(query);
        while (rs.next()) {
           statsArray.add(rs.getInt(1));//max
           statsArray.add(rs.getInt(2));//min
           statsArray.add(rs.getInt(3));//avg
        }
        averagesMap.put("Motorcycle", statsArray);
        statsArray.clear();
        query = "SELECT DISTINCT MAX(DATEDIFF(ReturnDate,RentalDate)), MIN(DATEDIFF(ReturnDate,RentalDate)),AVG(DATEDIFF(ReturnDate,RentalDate)) FROM RentalRecordings WHERE VehicleType = 'eBike'";
        rs = stmt.executeQuery(query);
        while (rs.next()) {
           statsArray.add(rs.getInt(1));//max
           statsArray.add(rs.getInt(2));//min
           statsArray.add(rs.getInt(3));//avg
        }
        averagesMap.put("eBike", statsArray);
        statsArray.clear();
        query = "SELECT DISTINCT MAX(DATEDIFF(ReturnDate,RentalDate)), MIN(DATEDIFF(ReturnDate,RentalDate)),AVG(DATEDIFF(ReturnDate,RentalDate)) FROM RentalRecordings WHERE VehicleType = 'Scooter'";
        rs = stmt.executeQuery(query);
        while (rs.next()) {
           statsArray.add(rs.getInt(1));//max
           statsArray.add(rs.getInt(2));//min
           statsArray.add(rs.getInt(3));//avg
```

```
            }
            averagesMap.put("Scooter", statsArray);
            statsArray.clear();
        } catch (SQLException e) {
            e.printStackTrace(System.err);
        }
        return averagesMap;
    }
    public HashMap<String,ArrayList<Integer>> RentalTotals(Date From, Date To){
        Connection con = DB_Connection.getConnection();
        ResultSet rs;
        Statement stmt;
        HashMap<String, ArrayList<Integer>> totalsMap = new HashMap<>();
        ArrayList<Integer> statsArray = new ArrayList<>();
        try {
            stmt = con.createStatement();
            String query = "SELECT SUM(PaymentAmount) FROM RentalRecordings WHERE VehicleType =
'Car' AND RentalDate >= '" + From + "' AND ReturnDate <='" + To + "'";
            rs = stmt.executeQuery(query);
            while (rs.next()) {
                statsArray.add(rs.getInt(1));//max
            }
            totalsMap.put("Car", statsArray);
            statsArray.clear();
            query = "SELECT SUM(PaymentAmount) FROM RentalRecordings WHERE VehicleType =
'Motorcycle' AND RentalDate >= '" + From + "' AND ReturnDate <='" + To + "'";
            rs = stmt.executeQuery(query);
            while (rs.next()) {
                statsArray.add(rs.getInt(1));//max
            }
            totalsMap.put("Motorcycle", statsArray);
            statsArray.clear();
```

```
            query = "SELECT SUM(PaymentAmount) FROM RentalRecordings WHERE VehicleType =
'eBike' AND RentalDate >= '" + From + "' AND ReturnDate <='" + To + "'";
            rs = stmt.executeQuery(query);
            while (rs.next()) {
                statsArray.add(rs.getInt(1));//max
            }
            totalsMap.put("eBike", statsArray);
            statsArray.clear();
            query = "SELECT SUM(PaymentAmount) FROM RentalRecordings WHERE VehicleType =
'Scooter' AND RentalDate >= '" + From + "' AND ReturnDate <='" + To + "'";
            rs = stmt.executeQuery(query);
            while (rs.next()) {
                statsArray.add(rs.getInt(1));//max
            }
            totalsMap.put("Scooter", statsArray);
            statsArray.clear();
        } catch (SQLException e) {
            e.printStackTrace(System.err);
        }
        return totalsMap;
    }
    public ArrayList<String> popularVehicles(){
        Connection con = DB_Connection.getConnection();
        ResultSet rs;
        Statement stmt;
        ArrayList<String> popularVehicles = new ArrayList<>();
        try {
            stmt = con.createStatement();
            String query = "SELECT Brand,Model FROM RentalRecordings,carrecordings WHERE
RentalRecordings.VehicleID = carrecordings.LicenseNumber AND (SELECT MAX(TimesRented) FROM
carrecordings)";
            rs = stmt.executeQuery(query);
```

```
            while (rs.next()) {
                popularVehicles.add(rs.getString(1)+" "+rs.getString(2));
            }
            query = "SELECT Brand,Model FROM RentalRecordings,motorcyclerecordings WHERE RentalRecordings.VehicleID = motorcyclerecordings.LicenseNumber AND (SELECT MAX(TimesRented) FROM evol_db.motorcyclerecordings)";
            rs = stmt.executeQuery(query);
            while (rs.next()) {
                popularVehicles.add(rs.getString(1)+" "+rs.getString(2));
            }
            query = "SELECT Brand,Model FROM RentalRecordings,ebikerecordings WHERE RentalRecordings.VehicleID = ebikerecordings.BikeID AND (SELECT MAX(TimesRented) FROM ebikerecordings)";
            rs = stmt.executeQuery(query);
            while (rs.next()) {
                popularVehicles.add(rs.getString(1)+" "+rs.getString(2));
            }
            query = "SELECT Brand,Model FROM RentalRecordings,scooterrecordings WHERE RentalRecordings.VehicleID =scooterrecordings.ScooterID AND (SELECT MAX(TimesRented) FROM scooterrecordings)";
            rs = stmt.executeQuery(query);
            while (rs.next()) {
                popularVehicles.add(rs.getString(1)+" "+rs.getString(2));
            }
        } catch (SQLException e) {
            e.printStackTrace(System.err);
        }
        return popularVehicles;
    }
```