NORTH CENTRAL COLLEGE 1861

<*CAR RENTAL CENTRAL*>

TERM PROJECT REPORT

Version <*1.7*> *by Alexander Rafacz, Usman Kaleel, Briggs Cecil*

<12/3/2023>

TABLE OF CONTENTS

1 Project Description

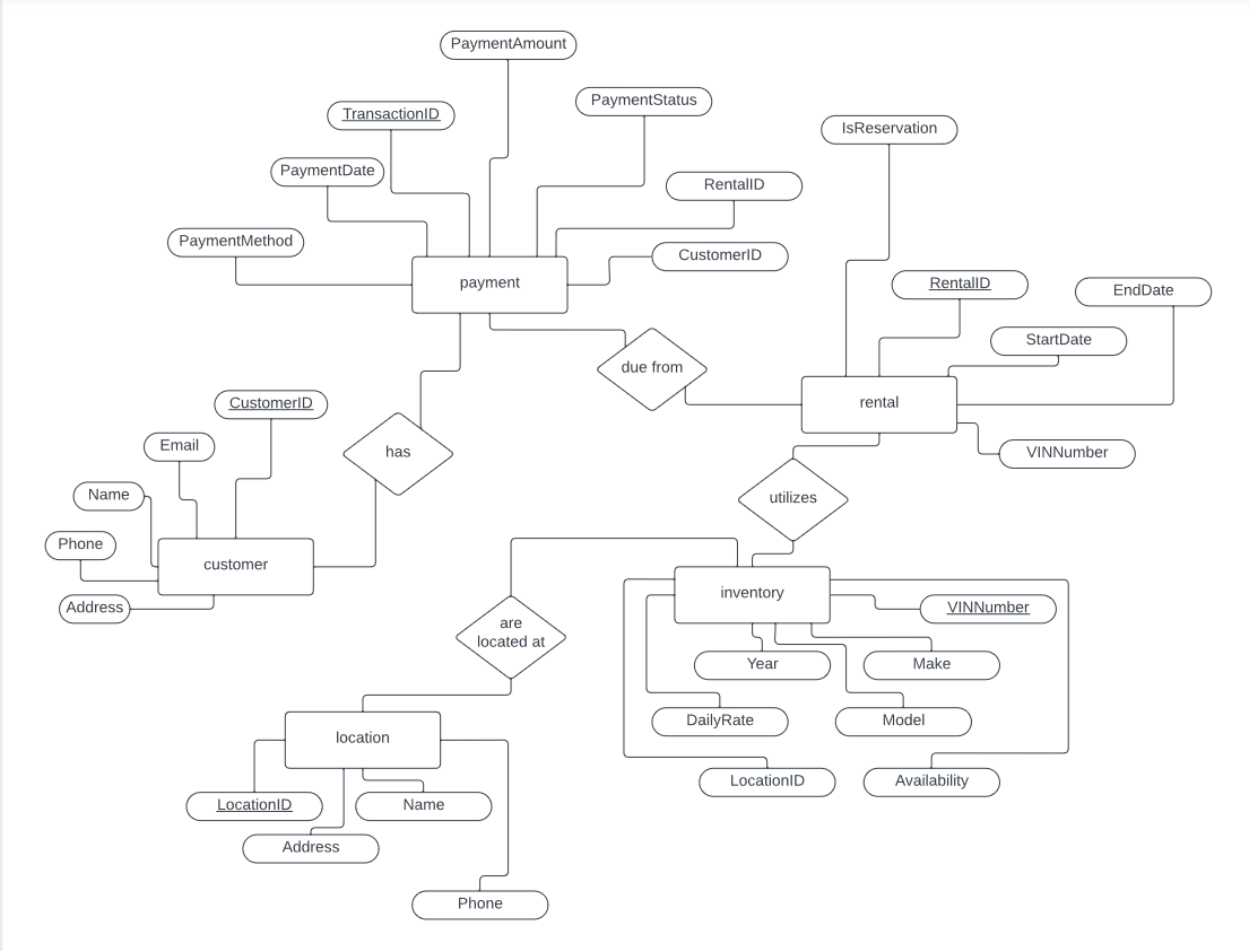
The goal of the project was to implement a database in Java using JDBC API. mimicking any industry domain. Selecting the car rental industry, our database simulates the records and tables needed to manage a car rental establishment. Though relatively simplified, the database reflects how a car would be entered, the details of the car, the rental information for the car, and the renter's details themselves.

2 Team Members' role

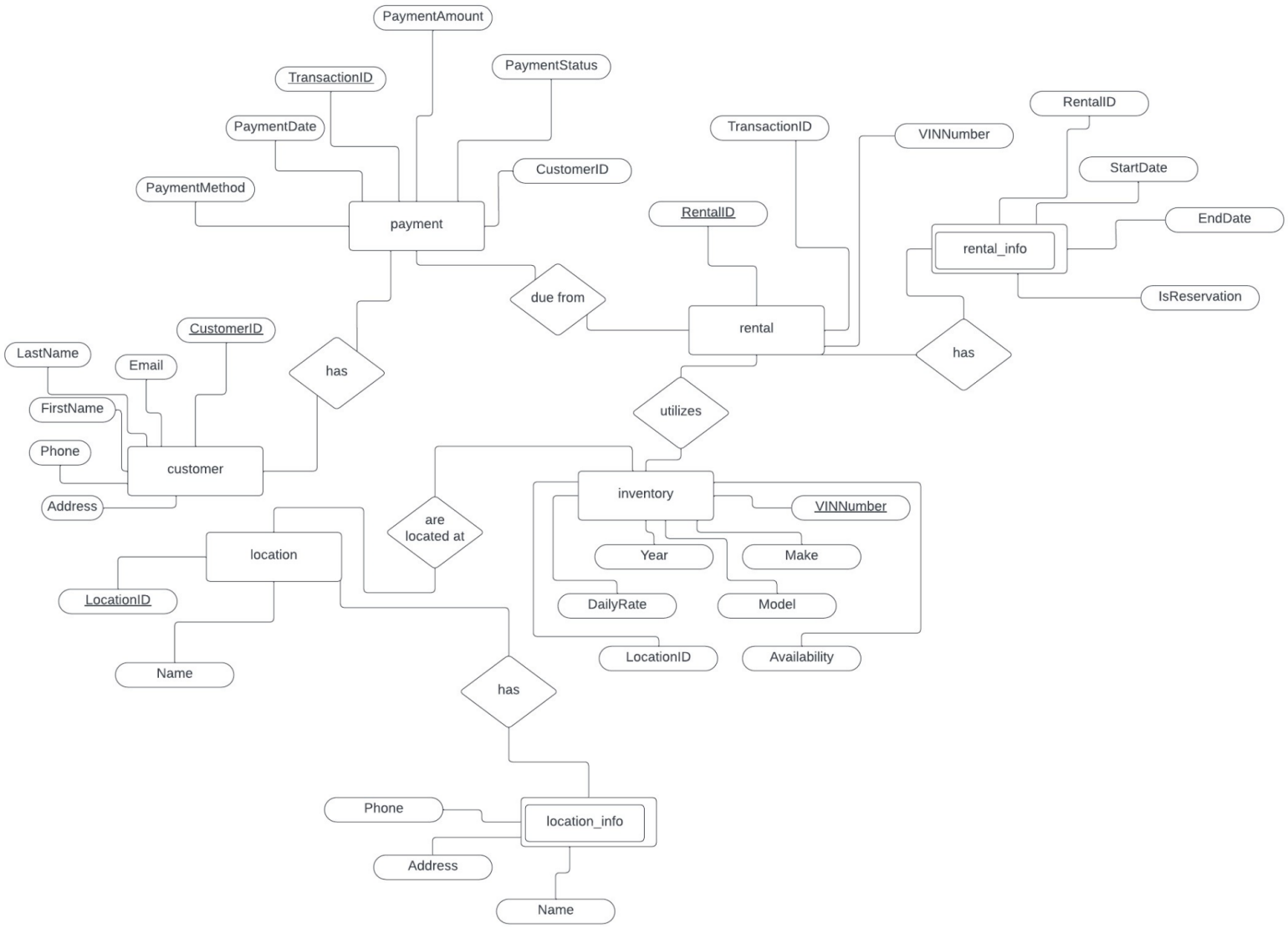
Each team member had to brainstorm ideas for the database initially, and work to develop tables and attributes to populate the database. Once done, each team member was assigned with creating data insertions in SQL and converting it to Java in order to maximize learning amongst all team members. Each team member was responsible for working on the report, and making edits as necessary. Lastly, team members needed to contribute evenly on the presentation, with approximately 6 and a half minutes worth of content prepared.

3 ER Diagram and relational schema

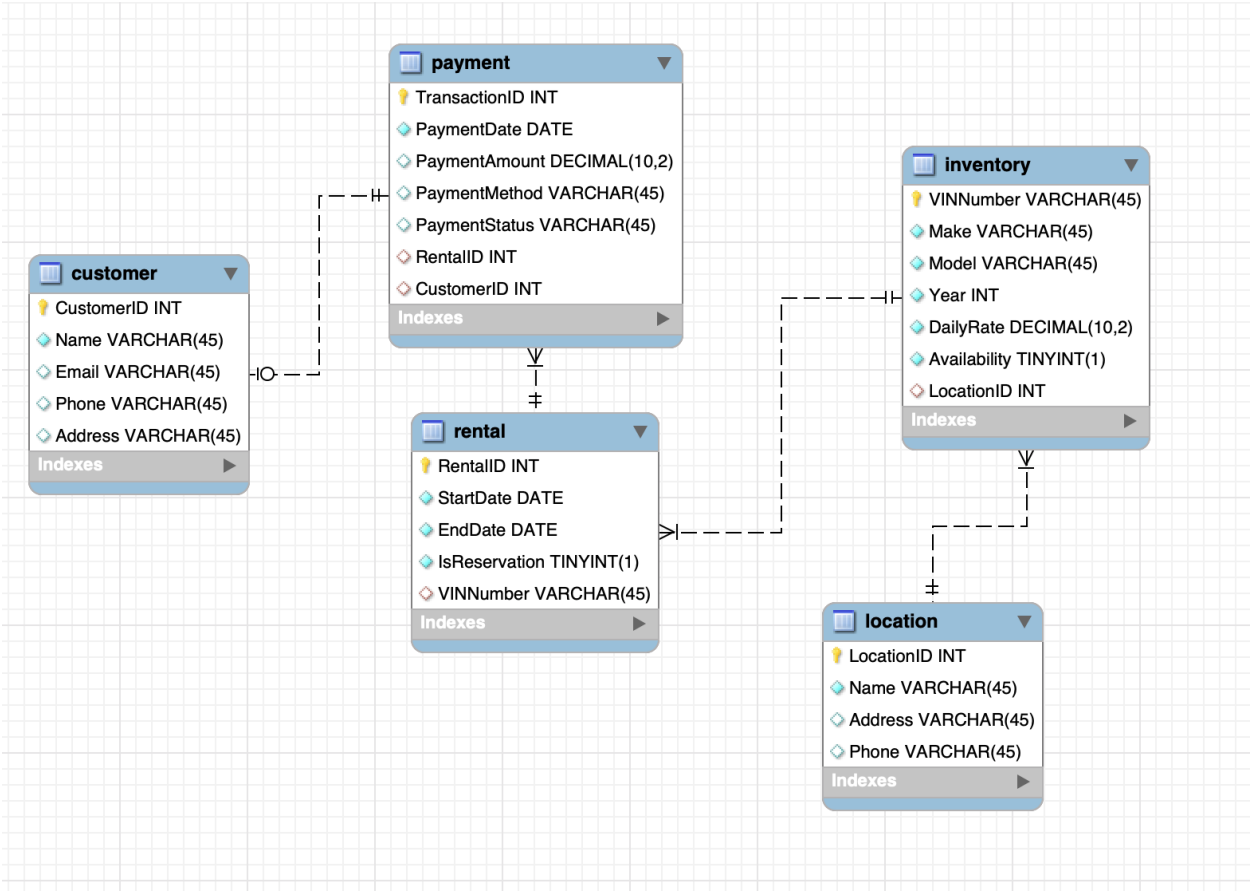
Our Relational Schema Before Normalization:



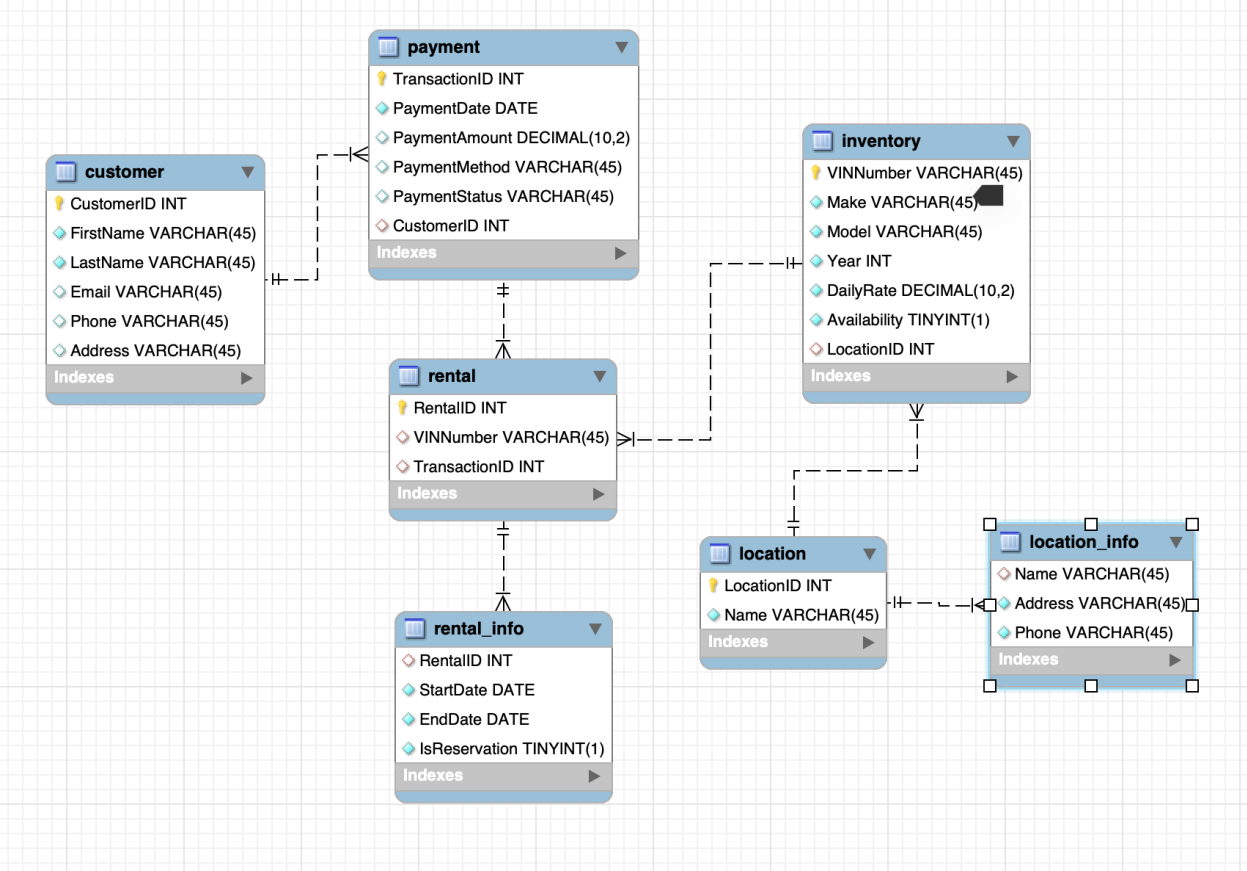
Our Relational Schema After 3NF Normalization:



Our Entity-Relation Diagram Before Normalization:



Our Entity-Relation Diagram After 3NF Normalization:



4 Assumption and Limitations of the Design

Our original plan was to split our five tables into eight tables after 3NF normalization. We wanted to have a customer table that contains the customer's ID, first name and last name, and a customer_details table that contained the customer's email, address, and phone number. However this was changed to seven tables, due to a limitation of the customer table, where we planned to have the first and last name be a primary key that related to the customer's email, address, and phone number. Since this could conflict with other customers if they have the same first name, we decided to structure the customer table such that all attributes were included. This resulted in our database having seven tables after normalization.

5 Cardinality Relationships

Each Payment can have only one Customer, and each Customer can have many Payments. A Rental can have many Payments and each Payment can only relate to one Rental. Rental_Info and Rental have a 1:1 cardinality relationship, since a Rental should only have information associated with that Rental. An Inventory item can have only one Rental, and each Rental can only be related to one Inventory item. One Inventory item is related to one Location, and one Location can have many items in their Inventory. Lastly, Location and Location_Info have a 1:1 cardinality relationship, since each Location should not have multiple names or addresses.

6 Data Dictionary

Table	Field Name	Data Type	Nullable	Field Size	Ref	Description
Customer	CustomerID	INT	No	N/A	N/A	Primary key for customer table, uniquely identifies customers
Customer	FirstName	VARCHAR	No	45	N/A	First name of customer
Customer	LastName	VARCHAR	No	45	N/A	Last name of customer
Customer	Email	VARCHAR	Yes	45	N/A	Email of customer
Customer	Phone	VARCHAR	Yes	45	N/A	Phone number of customer
Customer	Address	VARCHAR	Yes	45	N/A	Address of customer
Payment	TransactionID	INT	No	N/A	N/A	Primary key for payment, uniquely identifies transactions
Payment	PaymentDate	DATE	No	N/A	N/A	Date of payment
Payment	PaymentMethod	VARCHAR	Yes	45	N/A	Method customer used to pay
Payment	PaymentStatus	VARCHAR	Yes	45	N/A	Status of payment
Payment	CustomerID	INT	No	N/A	CustomerID from Customer	Foreign key from customer table
Rental	RentalID	INT	No	N/A	N/A	Primary key for rental table, uniquely identifies a rental
Rental_Info	RentalID	INT	No	N/A	RentalID from Rental	Foreign key for rental_info, weak entity
Rental_Info	StartDate	DATE	No	N/A	N/A	Start date for a rental
Rental_Info	EndDate	Date	No	N/A	N/A	End date for a rental
Rental_Info	IsReservation	TINYINT	No	1	N/A	Whether or not rental is a reservation
Rental	VINNumber	VARCHAR	No	45	VINNumber from Inventory	Foreign key from inventory table
Rental	TransactionId	INT	No	N/A	TransactionID from Payment	Foreign key from payment table
Inventory	VINNumber	VARCHAR	No	45	N/A	Primary key for inventory table, uniquely identifies a car
Inventory	Make	VARCHAR	No	45	N/A	Make of car
Inventory	Model	VARCHAR	No	45	N/A	Model of car
Inventory	Year	INT	No	N/A	N/A	Year of car
Inventory	DailyRate	DECIMAL	No	10, 2	N/A	Daily rate to rent car
Inventory	Availability	TINYINT	No	1	N/A	Whether or not car is available to rent
Inventory	LocationID	INT	No	N/A	LocationID from Location	Foreign key from location table

Table	Field Name	Data Type	Nullable	Field Size	Ref	Description
Location	LocationID	INT	No	N/A	N/A	Primary key for location table, uniquely identifies the rental dealership
Location	Name	VARCHAR	No	45	N/A	Name of the rental dealership
Location_Info	Address	VARCHAR	No	45	N/A	Address of the rental dealership
Location_Info	Phone	VARCHAR	No	45	N/A	Phone number of the rental dealership
Location_Info	Name	VARCHAR	No	45	Name from Location	Foreign key for location_info, weak entity

7 Sample Data

CUSTOMER DATA:

CustomerID	FirstName	LastName	Email	Phone	Address
1	John	Doe	john.doe@email.com	555-1234	123 Main St
2	Jane	Smith	jane.smith@email.com	555-5678	456 Oak St
3	Mike	Johnson	mike.johnson@email.com	555-9876	789 Pine St
4	Sarah	Williams	sarah.williams@email.com	555-4321	321 Elm St
5	David	Brown	david.brown@email.com	555-8765	654 Birch St

INVENTORY DATA:

VINNumber	Make	Model	Year	DailyRate	Availability	LocationID
10PQR78901J234567	Audi	Q7	2023	100.00	1	101
11STU98765K876543	Hyundai	Elantra	2022	45.00	1	102
12VWX12345L654321	Kia	Sorento	2021	65.00	1	110
13YZA45678M789012	Subaru	Outback	2022	55.00	1	111
14BCD98765N123456	Mazda	CX-5	2023	70.00	1	112

LOCATION DATA:

Name	Address	Phone
Downtown	123 City Blvd	555-1111
Suburb	456 Suburb Ln	555-2222
Airport	789 Terminal Ave	555-3333
Shopping Mall	101 Shopper St	555-4444
Beachfront	555 Beach Rd	555-5555
Mountain View	777 Summit Ln	555-6666
Business District	321 Corporate St	555-7777
Industrial Area	444 Factory Rd	555-8888
Residential Zone	789 Home Ave	555-9999
Historical Quarter	888 Heritage St	555-0000
Tech Park	123 Tech St	555-1122
Parkside	456 Park Ave	555-2233

LocationID	Name
10	Airport
12	Beachfront
21	Business District
120	Countryside
1	Downtown
112	Garden Heights
101	Historical Quarter
22	Industrial Area
20	Mountain View
110	Parkside

PAYMENT DATA:

TransactionID	PaymentDate	PaymentAmount	PaymentMethod	PaymentStatus	CustomerID
1	2023-01-10	150.00	Credit Card	Completed	1
10	2023-02-20	200.00	Cash	Completed	2
11	2023-03-30	180.00	Online	Pending	3
100	2023-04-05	250.00	Credit Card	Completed	4
101	2023-05-15	220.00	Cash	Completed	5

RENTAL DATA:

RentalID	VINNumber	TransactionID
1	1ABCD12345A112233	1
2	2DCBA54321B445566	10
3	3AABB66666C987654	11
4	4CCDD22222D001122	100
5	5EFGH99887E557799	101
6	6TUVW13579F654321	110
7	7XYZ12345G789012	111
8	8ABC98765H123456	1000
9	9LMN45678I789012	1001
10	10PQR78901J234567	1010
11	11STU98765K876543	1011
12	12VWX12345L654321	1100

RentalID	StartDate	EndDate	IsReservation
1	2023-01-01	2023-01-05	0
2	2023-02-10	2023-02-15	1
3	2023-03-20	2023-03-25	0
4	2023-04-01	2023-04-10	1
5	2023-05-10	2023-05-20	0
6	2023-06-20	2023-06-25	1
7	2023-07-05	2023-07-15	0
8	2023-08-15	2023-08-20	1
9	2023-09-25	2023-09-30	0
10	2023-10-10	2023-10-20	1
11	2023-11-05	2023-11-10	0
12	2023-12-15	2023-12-20	1

8 Java Code

8.1 Table creation Code

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
public class TableCreationProject {
	public static void main(String[] args) {
		Connection conn;
		try {
			// 1. driver loading and DB connection
			conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/project","root", "password");
			System.out.println("DB Connection Success!!");
			// 2. Write SQL queries to create DB table

			// customer: CustomerID, FirstName, LastName
			String sql1 = "CREATE TABLE customer" +
					"(CustomerID int AUTO_INCREMENT not null primary key," +
					"FirstName varchar(45) not null," +
					"LastName varchar(45) not null," +
					"Email varchar(45) null," +
					"Phone varchar(45) null," +
					"Address varchar(45) null)";

			// location: LocationID, Name; relates to location_info
			String sql2 = "CREATE TABLE location" +
					"(LocationID int AUTO_INCREMENT not null primary key," +
					"Name varchar(45) not null unique)";

			// location_info: Name, Address, Phone
			String sql7 = "CREATE TABLE location_info" +
					"(Name varchar(45)," +
					"foreign key (Name) references location(Name)," +
					"Address varchar(45) not null," +
					"Phone varchar(45) not null)";

			// inventory: VINNumber, Make, Model, Year, DailyRate, Availability, LocationID
			String sql3 = "CREATE TABLE inventory" +
				"(VINNumber varchar(45) not null primary key," +
				"Make varchar(45) not null," +
				"Model varchar(45) not null," +
				"Year int not null," +
				"DailyRate decimal(10,2) not null," +
				"Availability tinyint(1) not null," +
				"LocationID int," +
```

```
                                        "foreign key (LocationID) references
location(LocationID))";

                    // rental: RentalID, VINNumber, TransactionID; relates to rental_info
                    String sql5 = "CREATE TABLE rental" +
                            "(RentalID int AUTO_INCREMENT not null primary key," +
                            "VINNumber varchar(45)," +
                            "TransactionID int," +
                                        "foreign key (VINNumber) references
inventory(VINNumber)," +
                                    "foreign key (TransactionID) references
payment(TransactionID))";

                    // rental_info: RentalID, StartDate, EndDate, IsReservation
                    String sql6 = "CREATE TABLE rental_info" +
                            "(RentalID int," +
                                "foreign key (RentalID) references rental(RentalID)," +
                            "StartDate date not null," +
                            "EndDate date not null," +
                            "IsReservation tinyint(1) not null)";
                    // payment: TransactionID, PaymentDate, PaymentAmount,
PaymentMethod, PaymentStatus, CustomerID
                    String sql4 = "CREATE TABLE payment" +
                                "(TransactionID int AUTO_INCREMENT not null
primary key," +
                                "PaymentDate DATE not null," +
                                "PaymentAmount decimal(10,2) null," +
                                "PaymentMethod varchar(45) null," +
                                "PaymentStatus varchar(45) null," +
                                "CustomerID int," +
                                "foreign key (CustomerID) references
customer(CustomerID))";

                    // 3. Create the statement object to execute SQL queries.
                    Statement smt = conn.createStatement();
                    // 4. Execute SQL query using the execute method of the statement
object
                    boolean result1 = smt.execute(sql1);
                    boolean result2 = smt.execute(sql2);
                    boolean result3 = smt.execute(sql3);
                    boolean result4 = smt.execute(sql4);
                    boolean result5 = smt.execute(sql5);
                    boolean result6 = smt.execute(sql6);
                    boolean result7 = smt.execute(sql7);


                    System.out.println("Result: " + result1);
                    System.out.println("Result: " + result2);
                    System.out.println("Result: " + result3);
```

```
                        System.out.println("Result: " + result4);
                        System.out.println("Result: " + result5);
                        System.out.println("Result: " + result6);
                        System.out.println("Result: " + result7);
                        // 5. Close
                        if (smt != null)
                                smt.close();
                        if (conn != null)
                                conn.close();
                } catch (Exception e) {
                        System.out.println("Error: " + e);
                }
        }
}
```

8.2 Data Insertion Code

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
public class DataInsertionProject {
  public static void main(String[] args) {
     Connection conn;
     try {
        // 1. driver loading and DB connection
        conn = DriverManager.getConnection
              ("jdbc:mysql://localhost:3306/project", "root", "password");
        System.out.println("DB Connection Success!!");
        // 2. Write SQL queries to create DB table
        // CustomerID, FirstName, LastName, Email, Phone, Address
        String sql1 = "Insert INTO customer "
                        + "VALUE(1, 'John', 'Doe', 'john.doe@email.com', '555-1234', '123
Main St'),"
                        + "(2, 'Jane', 'Smith', 'jane.smith@email.com', '555-5678', '456 Oak
St'),"
                        + "(3, 'Mike', 'Johnson', 'mike.johnson@email.com', '555-9876', '789
Pine St'),"
                        + "(4, 'Sarah', 'Williams', 'sarah.williams@email.com', '555-4321',
'321 Elm St'),"
                        + "(5, 'David', 'Brown', 'david.brown@email.com', '555-8765', '654
Birch St'),"
                        + "(6, 'Emily', 'Davis', 'emily.davis@email.com', '555-2345', '234
Cedar St'),"
                        + "(7, 'Anna', 'Ziegler', 'anna.ziegler@email.com', '444-1234', '245
Somerset St'),"
                        + "(8, 'Chris', 'Taylor', 'chris.taylor@email.com', '444-5678', '678
Maple St'),"
                        + "(9, 'Laura', 'White', 'laura.white@email.com', '444-9876', '789
Oak St'),"
```

```
                    + "(10, 'Mark', 'Miller', 'mark.miller@email.com', '444-4321', '432
Pine St'),"
                    + "(11, 'Alex', 'Johnson', 'alex.johnson@email.com', '555-1111', '111
Main St'),"
                    + "(12, 'Sophia', 'Clark', 'sophia.clark@email.com', '555-2222', '222
Oak St'),"
                    + "(13, 'Ryan', 'Anderson', 'ryan.anderson@email.com', '555-3333',
'333 Pine St'),"
                    + "(14, 'Olivia', 'Moore', 'olivia.moore@email.com', '555-4444', '444
Birch St'),"
                    + "(15, 'Daniel', 'Hall', 'daniel.hall@email.com', '555-5555', '555
Cedar St')";

        //Insert into location_info
        // Name, Address, Phone
        String sql7 = "INSERT INTO location_info "
                + "VALUES ('Downtown', '123 City Blvd', '555-1111'),"
                + "('Suburb', '456 Suburb Ln', '555-2222'),"
                + "('Airport', '789 Terminal Ave', '555-3333'),"
                + "('Shopping Mall', '101 Shopper St', '555-4444'),"
                + "('Beachfront', '555 Beach Rd', '555-5555'),"
                + "('Mountain View', '777 Summit Ln', '555-6666'),"
                + "('Business District', '321 Corporate St', '555-7777'),"
                + "('Industrial Area', '444 Factory Rd', '555-8888'),"
                + "('Residential Zone', '789 Home Ave', '555-9999'),"
                + "('Historical Quarter', '888 Heritage St', '555-0000'),"
                + "('Tech Park', '123 Tech St', '555-1122'),"
                + "('Parkside', '456 Park Ave', '555-2233'),"
                + "('Riverfront', '789 River Rd', '555-3344'),"
                + "('Garden Heights', '101 Garden Blvd', '555-4455'),"
                + "('Countryside', '555 Country Ln', '555-5566')";

        // LocationID, Name
        String sql2 = "INSERT INTO location "
                + "VALUES (001, 'Downtown'),"
                + "(002, 'Suburb'),"
                + "(010, 'Airport'),"
                + "(011, 'Shopping Mall'),"
                + "(012, 'Beachfront'),"
                + "(020, 'Mountain View'),"
                + "(021, 'Business District'),"
                + "(022, 'Industrial Area'),"
                + "(100, 'Residential Zone'),"
                + "(101, 'Historical Quarter'),"
                + "(102, 'Tech Park'),"
                + "(110, 'Parkside'),"
                + "(111, 'Riverfront'),"
                + "(112, 'Garden Heights'),"
                + "(120, 'Countryside')";
```

```
        // VINNumber, Make, Model, Year, DailyRate, Availability, LocationID
        String sql3 = "INSERT INTO inventory "
                    + "VALUES ('1ABCD12345A112233', 'Toyota', 'Camry', 2020, 50.00, 1,
001),"
                + "('2DCBA54321B445566', 'Honda', 'Accord', 2019, 55.00, 1, 002),"
                + "('3AABB66666C987654', 'Ford', 'Mustang', 2021, 70.00, 0, 010),"
                + "('4CCDD22222D001122', 'Chevrolet', 'Malibu', 2022, 60.00, 1, 011),"
                + "('5EFGH99887E557799', 'Nissan', 'Altima', 2021, 55.00, 1, 012),"
                + "('6TUVW13579F654321', 'Jeep', 'Wrangler', 2020, 75.00, 0, 020),"
                + "('7XYZ12345G789012', 'Tesla', 'Model S', 2023, 120.00, 1, 021),"
                + "('8ABC98765H123456', 'BMW', 'X5', 2022, 90.00, 1, 022),"
                + "('9LMN45678I789012', 'Mercedes', 'C-Class', 2022, 80.00, 1, 100),"
                + "('10PQR78901J234567', 'Audi', 'Q7', 2023, 100.00, 1, 101),"
                + "('11STU98765K876543', 'Hyundai', 'Elantra', 2022, 45.00, 1, 102),"
                + "('12VWX12345L654321', 'Kia', 'Sorento', 2021, 65.00, 1, 110),"
                + "('13YZA45678M789012', 'Subaru', 'Outback', 2022, 55.00, 1, 111),"
                + "('14BCD98765N123456', 'Mazda', 'CX-5', 2023, 70.00, 1, 112),"
                + "('15CDE12345O654321', 'Volvo', 'XC90', 2022, 85.00, 1, 120)";

        // RentalID, VINNumber, TransactionID
        String sql5 = "INSERT INTO rental "
                + "VALUES (1,'1ABCD12345A112233', 0001),"
                + "(2,'2DCBA54321B445566', 0010),"
                + "(3,'3AABB66666C987654', 0011),"
                + "(4,'4CCDD22222D001122', 0100),"
                + "(5,'5EFGH99887E557799', 0101),"
                + "(6,'6TUVW13579F654321', 0110),"
                + "(7,'7XYZ12345G789012', 0111),"
                + "(8,'8ABC98765H123456', 1000),"
                + "(9,'9LMN45678I789012', 1001),"
                + "(10,'10PQR78901J234567', 1010),"
                + "(11,'11STU98765K876543', 1011),"
                + "(12,'12VWX12345L654321', 1100),"
                + "(13,'13YZA45678M789012', 1101),"
                + "(14,'14BCD98765N123456', 1110),"
                + "(15,'15CDE12345O654321', 1111)";

        // RentalID, StartDate, EndDate, IsReservation,
        String sql6 = "INSERT INTO rental_info "
                + "VALUES (1,'2023-01-01', '2023-01-05', 0),"
                + "(2, '2023-02-10', '2023-02-15', 1),"
                + "(3, '2023-03-20', '2023-03-25', 0),"
                + "(4, '2023-04-01', '2023-04-10', 1),"
                + "(5, '2023-05-10', '2023-05-20', 0),"
                + "(6, '2023-06-20', '2023-06-25', 1),"
                + "(7, '2023-07-05', '2023-07-15', 0),"
                + "(8, '2023-08-15', '2023-08-20', 1),"
                + "(9, '2023-09-25', '2023-09-30', 0),"
                + "(10, '2023-10-10', '2023-10-20', 1),"
```

```
                + "(11, '2023-11-05', '2023-11-10', 0),"
                + "(12, '2023-12-15', '2023-12-20', 1),"
                + "(13, '2024-01-25', '2024-01-30', 0),"
                + "(14, '2024-02-10', '2024-02-20', 1),"
                + "(15, '2024-03-05', '2024-03-15', 0)";
                        // TransactionID, PaymentDate, PaymentAmount, PaymentMethod,
PaymentStatus, CustomerID
        String sql4 = "INSERT INTO payment "
                + "VALUES (0001, '2023-01-10', 150.00, 'Credit Card', 'Completed', 1),"
                + "(0010, '2023-02-20', 200.00, 'Cash', 'Completed', 2),"
                + "(0011, '2023-03-30', 180.00, 'Online', 'Pending', 3),"
                + "(0100, '2023-04-05', 250.00, 'Credit Card', 'Completed', 4),"
                + "(0101, '2023-05-15', 220.00, 'Cash', 'Completed', 5),"
                + "(0110, '2023-06-25', 300.00, 'Online', 'Pending', 6),"
                + "(0111, '2023-07-10', 180.00, 'Credit Card', 'Pending', 7),"
                + "(1000, '2023-08-20', 250.00, 'Cash', 'Completed', 8),"
                + "(1001, '2023-09-30', 200.00, 'Online', 'Pending', 9),"
                + "(1010, '2023-10-15', 280.00, 'Credit Card', 'Completed', 10),"
                + "(1011, '2023-11-05', 150.00, 'Credit Card', 'Completed', 11),"
                + "(1100, '2023-12-20', 220.00, 'Cash', 'Pending', 12),"
                + "(1101, '2024-01-30', 190.00, 'Online', 'Completed', 13),"
                + "(1110, '2024-02-20', 270.00, 'Credit Card', 'Pending', 14),"
                + "(1111, '2024-03-15', 240.00, 'Cash', 'Completed', 15)";

        System.out.println("All Data Points Added");
        // 3. Create the statement object to execute SQL queries.
        Statement stmt = conn.createStatement();
        // 4. Execute SQL query using the execute method of the statement object
        boolean result1 = stmt.execute(sql1);
        boolean result2 = stmt.execute(sql2);
        boolean result3 = stmt.execute(sql3);
        boolean result4 = stmt.execute(sql4);
        boolean result5 = stmt.execute(sql5);
        boolean result6 = stmt.execute(sql6);
        boolean result7 = stmt.execute(sql7);
        System.out.println("Result: " + result1);
        System.out.println("Result: " + result2);
        System.out.println("Result: " + result3);
        System.out.println("Result: " + result4);
        System.out.println("Result: " + result5);
        System.out.println("Result: " + result6);
        System.out.println("Result: " + result7);
        // 5. Close
        if (stmt != null)
            stmt.close();
        if (conn != null)
            conn.close();
    } catch (Exception e) {
        System.out.println("Error: " + e);
```

```
    }
  }
}
```

8.3 Screenshot of DB tables and data instances

Customer Table:

CustomerID	FirstName	LastName	Email	Phone	Address
1	John	Doe	john.doe@email.com	555-1234	123 Main St
2	Jane	Smith	jane.smith@email.com	555-5678	456 Oak St
3	Mike	Johnson	mike.johnson@email.com	555-9876	789 Pine St
4	Sarah	Williams	sarah.williams@email.com	555-4321	321 Elm St
5	David	Brown	david.brown@email.com	555-8765	654 Birch St
6	Emily	Davis	emily.davis@email.com	555-2345	234 Cedar St
7	Anna	Ziegler	anna.ziegler@email.com	444-1234	245 Somerset St
8	Chris	Taylor	chris.taylor@email.com	444-5678	678 Maple St
9	Laura	White	laura.white@email.com	444-9876	789 Oak St
10	Mark	Miller	mark.miller@email.com	444-4321	432 Pine St
11	Alex	Johnson	alex.johnson@email.com	555-1111	111 Main St
12	Sophia	Clark	sophia.clark@email.com	555-2222	222 Oak St
13	Ryan	Anderson	ryan.anderson@email.com	555-3333	333 Pine St
14	Olivia	Moore	olivia.moore@email.com	555-4444	444 Birch St
15	Daniel	Hall	daniel.hall@email.com	555-5555	555 Cedar St
NULL	NULL	NULL	NULL	NULL	NULL

Inventory Table:

VINNumber	Make	Model	Year	DailyRate	Availability	LocationID
10PQR78901J234567	Audi	Q7	2023	100.00	1	101
11STU98765K876543	Hyundai	Elantra	2022	45.00	1	102
12VWX12345L654321	Kia	Sorento	2021	65.00	1	110
13YZA45678M789012	Subaru	Outback	2022	55.00	1	111
14BCD98765N123456	Mazda	CX-5	2023	70.00	1	112
15CDE12345O654321	Volvo	XC90	2022	85.00	1	120
1ABCD12345A112233	Toyota	Camry	2020	50.00	1	1
2DCBA54321B445566	Honda	Accord	2019	55.00	1	2
3AABB66666C987654	Ford	Mustang	2021	70.00	0	10
4CCDD22222D001122	Chevrolet	Malibu	2022	60.00	1	11
5EFGH99887E557799	Nissan	Altima	2021	55.00	1	12
6TUVW13579F654321	Jeep	Wrangler	2020	75.00	0	20
7XYZ12345G789012	Tesla	Model S	2023	120.00	1	21
8ABC98765H123456	BMW	X5	2022	90.00	1	22
9LMN45678I789012	Mercedes	C-Class	2022	80.00	1	100
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Location Table:

LocationID	Name
10	Airport
12	Beachfront
21	Business District
120	Countryside
1	Downtown
112	Garden Heights
101	Historical Quarter
22	Industrial Area
20	Mountain View
110	Parkside
100	Residential Zone
111	Riverfront
11	Shopping Mall
2	Suburb
102	Tech Park
NULL	NULL

Location_Info Table:

Name	Address	Phone
Downtown	123 City Blvd	555-1111
Suburb	456 Suburb Ln	555-2222
Airport	789 Terminal Ave	555-3333
Shopping Mall	101 Shopper St	555-4444
Beachfront	555 Beach Rd	555-5555
Mountain View	777 Summit Ln	555-6666
Business District	321 Corporate St	555-7777
Industrial Area	444 Factory Rd	555-8888
Residential Zone	789 Home Ave	555-9999
Historical Quarter	888 Heritage St	555-0000
Tech Park	123 Tech St	555-1122
Parkside	456 Park Ave	555-2233
Riverfront	789 River Rd	555-3344
Garden Heights	101 Garden Blvd	555-4455
Countryside	555 Country Ln	555-5566

Payment Table:

TransactionID	PaymentDate	PaymentAmount	PaymentMethod	PaymentStatus	CustomerID
1	2023-01-10	150.00	Credit Card	Completed	1
10	2023-02-20	200.00	Cash	Completed	2
11	2023-03-30	180.00	Online	Pending	3
100	2023-04-05	250.00	Credit Card	Completed	4
101	2023-05-15	220.00	Cash	Completed	5
110	2023-06-25	300.00	Online	Pending	6
111	2023-07-10	180.00	Credit Card	Pending	7
1000	2023-08-20	250.00	Cash	Completed	8
1001	2023-09-30	200.00	Online	Pending	9
1010	2023-10-15	280.00	Credit Card	Completed	10
1011	2023-11-05	150.00	Credit Card	Completed	11
1100	2023-12-20	220.00	Cash	Pending	12
1101	2024-01-30	190.00	Online	Completed	13
1110	2024-02-20	270.00	Credit Card	Pending	14
1111	2024-03-15	240.00	Cash	Completed	15
NULL	NULL	NULL	NULL	NULL	NULL

Rental Table:

RentalID	VINNumber	TransactionID
1	1ABCD12345A112233	1
2	2DCBA54321B445566	10
3	3AABB66666C987654	11
4	4CCDD22222D001122	100
5	5EFGH99887E557799	101
6	6TUVW13579F654321	110
7	7XYZ12345G789012	111
8	8ABC98765H123456	1000
9	9LMN45678I789012	1001
10	10PQR78901J234567	1010
11	11STU98765K876543	1011
12	12VWX12345L654321	1100
13	13YZA45678M789012	1101
14	14BCD98765N123456	1110
15	15CDE12345O654321	1111
NULL	NULL	NULL

Rental_Info Table:

RentalID	StartDate	EndDate	IsReservation
1	2023-01-01	2023-01-05	0
2	2023-02-10	2023-02-15	1
3	2023-03-20	2023-03-25	0
4	2023-04-01	2023-04-10	1
5	2023-05-10	2023-05-20	0
6	2023-06-20	2023-06-25	1
7	2023-07-05	2023-07-15	0
8	2023-08-15	2023-08-20	1
9	2023-09-25	2023-09-30	0
10	2023-10-10	2023-10-20	1
11	2023-11-05	2023-11-10	0
12	2023-12-15	2023-12-20	1
13	2024-01-25	2024-01-30	0
14	2024-02-10	2024-02-20	1
15	2024-03-05	2024-03-15	0

9 Own SQL queries in Java

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
public class SQLQueryProject {
	public static void main(String[] args) {
		Connection conn;

		try {
			// 1. driver loading and DB connection
			conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/project","root", "password");
			System.out.println("DB Connection Success!!");

			// 2. Write SQL queries to create DB table
			/*
			 Show all completed rental payments, who's rental end date are
prior to June,
			 and are currently available to rent (might need to make a more
practical query).
			 Display payment amount, status, date, and transaction id utilizing a
stored procedure,
			*/

			String u_sql1 = "CREATE PROCEDURE
Completed_Rentals_Before_June() "+
				"BEGIN " +
					"SELECT p.PaymentAmount, p.PaymentStatus,
p.PaymentDate, p.TransactionID " +
					"FROM Rental r " +
					"JOIN Payment p " +
					" ON r.TransactionID = p.TransactionID " +
					" JOIN Inventory i " +
					" ON r.VINNumber = i.VINNumber " +
					" JOIN rental_info ri " +
					" ON r.RentalID = ri.RentalID " +
					"WHERE ri.IsReservation = 0 AND " +
					" ri.EndDate < '2023-06-01' AND " +
					"i.Availability = 1; " +
				"END";

			/*
			Show all vehicles available that are located in addresses that
contain "St".
			Display the vehicle make, model, year, rate and VIN using a view.
			*/

			String u_sql2 = "CREATE OR REPLACE VIEW
```

```
Available_Vehicles_With_No_Payments AS " +
                    "SELECT i.Make, i.Model, i.Year, i.DailyRate, i.VINNumber,
li.Address AS Location_Address " +
                            "FROM Inventory i JOIN Location l on i.LocationID =
l.LocationID " +
                            "JOIN location_info li on l.Name = li.Name " +
                    "WHERE li.Address LIKE '%St%'";

                    /*
                    Select all cash transactions that have payment amount above 200
and a daily rate greater
                    than 50 that started prior to July. Display the payment amount, daily
rate, and start date.
                    */

                    String u_sql3 = "CREATE OR REPLACE VIEW
High_Value_Cash_Transactions AS " +
                    "SELECT p.PaymentAmount, i.DailyRate, ri.StartDate " +
                            "FROM Payment p JOIN Rental r ON p.TransactionID
= r.TransactionID " +
                      "JOIN Inventory i ON r.VINNumber = i.VINNumber " +
                      "JOIN rental_info ri ON r.RentalID = ri.RentalID " +
                            "WHERE p.PaymentMethod = 'Cash' AND " +
                      "p.PaymentAmount > 200 AND " +
                      "i.DailyRate > 50 AND " +
                      "ri.StartDate < '2023-07-01'";

                    // Displays the average length of rental by location
                    String b_sql1 = "SELECT location.Name, "
                            +       "round(AVG(DATEDIFF(rental_info.EndDate,
rental_info.StartDate)),1) AS AvgRentalDuration "
                            + "FROM location "
                            + "JOIN inventory ON location.LocationID =
inventory.LocationID "
                            + "JOIN rental ON inventory.VINNumber =
rental.VINNumber "
                            + "JOIN rental_info ON rental.RentalID =
rental_info.RentalID "
                            + "GROUP BY location.Name;";

                    // Displays the first 2 reservations along with customer data
                    String b_sql2 = "SELECT rental.RentalID, rental.VINNumber,
rental_info.StartDate, "
                            + "rental_info.EndDate, rental_info.IsReservation, "
                            + "       customer.FirstName, customer.LastName,
customer.Email, customer.Phone "
                            + "FROM customer "
                            + "LEFT JOIN rental ON customer.CustomerID =
rental.TransactionID "
```

```
                                + "LEFT JOIN rental_info ON rental.RentalID =
rental_info.RentalID AND "
                                + "rental_info.IsReservation = 1 "
                                + "ORDER BY rental.RentalID DESC,
customer.CustomerID DESC "
                                + "LIMIT 2;";

                    // Displays how many cars are at a given location given a certain
month (1-12)
                    String b_sql3 = "SELECT location.Name, COUNT(DISTINCT
inventory.VINNumber) AS CarsOnLocation "
                                + "FROM location "
                                + "LEFT JOIN inventory ON location.LocationID =
inventory.LocationID "
                                + "LEFT JOIN rental ON inventory.VINNumber =
rental.VINNumber "
                                + "LEFT JOIN rental_info ON rental.RentalID =
rental_info.RentalID "
                                +                                   "WHERE
MONTH(COALESCE(rental_info.StartDate, rental_info.EndDate)) = 2 "
                                + "OR (rental_info.StartDate IS NULL AND
rental_info.EndDate IS NULL) "
                                + "GROUP BY location.Name;";

                    // Retrieves the vehicles that are being reserved for rental by
customers at their respective locations
                    String a_sql1 = "CREATE PROCEDURE reserved_inventory() "
                                + "BEGIN "
                                + " SELECT C.FirstName, C.LastName, I.VINNumber,
"
                                + " CONCAT(LI.Name, \", \", LI.Address) AS Location,
StartDate, EndDate "
                                + "FROM customer C, payment P, rental R,
inventory I, location L, "
                                + "location_info LI, rental_info RI "
                                + "WHERE C.CustomerID = P.CustomerID "
                                + "AND P.TransactionID = R.TransactionID
"
                                + "AND R.VINNumber = I.VINNumber "
                                + "AND I.LocationID = L.LocationID "
                                + "AND L.Name = LI.Name "
                                + "AND R.RentalID = RI.RentalID "
                                + "AND IsReservation = 1; "
                                + "END";

                    String a_sql4 = "call project.reserved_inventory()";

                    // Retrieves all of the rented cars at every location, with the renter's
full name
```

```
                String a_sql2 = "CREATE OR REPLACE VIEW rented_inventory
AS "
                        + "SELECT CONCAT(LI.Name, \", \", LI.Address) AS
Location, "
                        + "  CONCAT(Year, \" \", Make, \" \", Model) AS
Vehicle, "
                        + "     CONCAT(C.LastName, \", \", C.FirstName) AS
\"Rented By\" "
                        + "FROM location L, inventory I, rental R, payment P,
customer C, location_info LI "
                        + "WHERE L.LocationID = I.LocationID "
                        + "     AND I.VINNumber = R.VINNumber "
                        + " AND R.TransactionID = P.TransactionID "
                        + "     AND P.CustomerID = C.CustomerID "
                        + " AND L.Name = LI.Name "
                        + "AND Availability = 0 "
                        + "ORDER BY L.LocationID;";

                String a_sql5 = "SELECT * FROM project.rented_inventory;";

                // Calculates the daily revenue based on availability of the vehicles
and their daily rates
                String a_sql3 = "CREATE FUNCTION potential_revenue
(availability_var int) RETURNS VARCHAR(45) "
                        + "BEGIN "
                        + "     DECLARE revenue DECIMAL(9, 2); "
                        + "   DECLARE total_rentals DECIMAL(9, 2); "
                        + "   DECLARE total_daily_rate DECIMAL(9, 2); "
                        + ""
                        + "   SELECT COUNT(RentalID), SUM(DailyRate) "
                        + "   INTO total_rentals, total_daily_rate "
                        + "   FROM rental R, inventory I "
                        + "     WHERE R.VINNumber = I.VINNumber AND
Availability = availability_var; "
                        + ""
                        + "       SET revenue = ROUND(total_rentals *
total_daily_rate, 2); "
                        + "RETURN CONCAT(\"$\", revenue); "
                        + "END";

                String a_sql6 = "SELECT project.potential_revenue(1)";
                String a_sql7 = "SELECT project.potential_revenue(0)";

                // 3. Create the statement object to execute SQL queries.
                Statement stmt = conn.createStatement();

                // 4. Execute SQL query using the execute method of the statement
object
                // SQL Query Creation
```

```
                    stmt.execute(u_sql1);       //                              creates
Completed_Rentals_Before_June()
                    stmt.execute(u_sql2);       //                              creates
available_vehicles_with_no_payments
                    stmt.execute(u_sql3);       // creates high_value_cash_transactions

                    stmt.execute(a_sql1);       // creates reserved_inventory()
                    stmt.execute(a_sql2);       // creates rented_inventory
                    stmt.execute(a_sql3);       // creates potential_revenue()

                    String u_sql4 = "call project.Completed_Rentals_Before_June()";
                    String      u_sql5      =      "SELECT      *      "      +      "FROM
project.available_vehicles_with_no_payments;";
                    String      u_sql6      =      "SELECT      *      "+"      FROM
project.high_value_cash_transactions;";

                    // Output for Completed_Rentals_Before_June()
                    ResultSet rs1 = stmt.executeQuery(u_sql4);
                    System.out.println("\n===Procedure                Call                for
Completed_Rentals_Before_June===");
                    while (rs1.next()) {
                            Double                    PaymentAmount                    =
rs1.getDouble("PaymentAmount");
                            String PaymentStatus = rs1.getString("PaymentStatus");
                            String PaymentDate = rs1.getString("PaymentDate");
                            int TransactionID = rs1.getInt("TransactionID");
                            System.out.println("PaymentAmount: " + PaymentAmount +
"\n" + "PaymentStatus: " + PaymentStatus + "\n" + "PaymentDate: " + PaymentDate +
"\n"
                                    + "TransactionID: " + TransactionID + "\n");
                    }
                    // Output for available_vehicles_with_no_payments
                    ResultSet rs2 = stmt.executeQuery(u_sql5);
                    System.out.println("===View                Call                for
available_vehicles_with_no_payments===");
                    while (rs2.next()) {
                            String Make = rs2.getString("Make");
                            String Model = rs2.getString("Model");
                            int Year = rs2.getInt("Year");
                            Double DailyRate = rs2.getDouble("DailyRate");
                            String VINNumber = rs2.getString("VINNumber");
                            String                    Location_Address                    =
rs2.getString("Location_Address");
                            System.out.println("Make:  "  +  Make  +  "\n"  +  "Model:  "  +
Model +
                                    "\n" + "Year: " + Year + "\n" +  "DailyRate: " +
DailyRate +
                                    "\n"  +  "VINNumber:  "  +  VINNumber  +  "\n"  +
"Location_Address: " +
```

```
                                        Location_Address + "\n");
                }

                // Output for high_value_cash_transactions
                ResultSet rs3 = stmt.executeQuery(u_sql6);

                System.out.println("===View Call for
high_value_cash_transactions===");
                while (rs3.next()) {
                        Double PaymentAmount =
rs3.getDouble("PaymentAmount");
                        Double DailyRate = rs3.getDouble("DailyRate");
                        String StartDate = rs3.getString("StartDate");
                        System.out.println("PaymentAmount: " + PaymentAmount +
"\n" +
                                        "DailyRate: " + DailyRate + "\n" + "StartDate: "
+ StartDate + "\n");
                }

                // Output for the average length of rental by location
                ResultSet rs4 = stmt.executeQuery(b_sql1);

                System.out.println("=== Average Length of Rental By Location
===");
                while(rs4.next()) {
                        String Name = rs4.getString("Name");
                        String AvgRentalDuration =
rs4.getString("AvgRentalDuration");

                        System.out.println("Name: " + Name + "\n" +
                                        "Average Rental Duration: " +
AvgRentalDuration + "\n");
                }

                // Output for the 2 current reservations along with customer data
                ResultSet rs5 = stmt.executeQuery(b_sql2);

                System.out.println("=== First 2 Current Reservations ===");
                while(rs5.next()) {
                        int RentalID = rs5.getInt("RentalID");
                        String VINNumber = rs5.getString("VINNumber");
                        String StartDate = rs5.getString("StartDate");
                        String EndDate = rs5.getString("EndDate");
                        int isReservation = rs5.getInt("IsReservation");
                        String FirstName = rs5.getString("FirstName");
                        String LastName = rs5.getString("LastName");
                        String Email = rs5.getString("Email");
                        String Phone = rs5.getString("Phone");
```

```
                        System.out.println("RentalID: " + RentalID + "\n" +
                                "VIN Number: " + VINNumber + "\n" +
                                "Start Date: " + StartDate + "\n" +
                                "End Date: " + EndDate + "\n" +
                                "Reservation Status: " + isReservation + "\n" +
                                "First Name: " + FirstName + "\n" +
                                "Last Name: " + LastName + "\n" +
                                "Email: " + Email + "\n" +
                                "Phone: " + Phone + "\n");
                }

                // Output for how many cars at a given location in a certain month
                ResultSet rs6 = stmt.executeQuery(b_sql3);

                System.out.println("=== Cars At a Location in a given Month ===");
                while(rs6.next()) {
                        String Name = rs6.getString("Name");
                        int CarsOnLocation = rs6.getInt("CarsOnLocation");

                        System.out.println("Location: " + Name + "\n" +
                                "# Cars on Location: " + CarsOnLocation +
"\n");
                }

                // Output for reserved_inventory()
                ResultSet a_rs1 = stmt.executeQuery(a_sql4);

                System.out.println("=== Procedure Call for reserved_inventory()
===");
                while(a_rs1.next()) {
                        String FirstName = a_rs1.getString("FirstName");
                        String LastName = a_rs1.getString("LastName");
                        String VINNumber = a_rs1.getString("VINNumber");
                        String Location = a_rs1.getString("Location");
                        String StartDate = a_rs1.getString("StartDate");
                        String EndDate = a_rs1.getString("EndDate");

                        System.out.println("First Name: " + FirstName + "\n" + "Last
Name: " + LastName + "\n"
                                + "VIN Number: "+ VINNumber + "\n" +
"Location: " + Location + "\n"
                                + "Start Date: " + StartDate + "\n" + "End Date:
"+ EndDate + "\n");
                }

                // Output for rented_inventory()
                ResultSet a_rs2 = stmt.executeQuery(a_sql5);

                System.out.println("\n=== View Call for rented_inventory ===");
```

```
                        while(a_rs2.next()) {
                                String Location = a_rs2.getString("Location");
                                String Vehicle = a_rs2.getString("Vehicle");
                                String Customer = a_rs2.getString("Rented By");

                                System.out.println("Location: " + Location + "\n"
                                                + "Vehicle: " + Vehicle + "\n"
                                                + "Rented By: " + Customer + "\n");
                        }

                        // Output for potential_revenue()
                        ResultSet a_rs3 = stmt.executeQuery(a_sql6);

                        System.out.println("\n=== Function Call for potential_revenue(1)
===");
                        while(a_rs3.next()) {
                                String                          Revenue                         =
a_rs3.getString("project.potential_revenue(1)");

                                System.out.println("Potential Daily Revenue for Available
Rentals: " + Revenue +  "\n");
                        }

                        ResultSet a_rs4 = stmt.executeQuery(a_sql7);

                        System.out.println("\n=== Function Call for potential_revenue(0)
===");
                        while(a_rs4.next()) {
                                String                          Revenue                         =
a_rs4.getString("project.potential_revenue(0)");

                                System.out.println("Daily Revenue of Active Rentals: " +
Revenue +  "\n");
                        }

                        // 5. Close
                        if (stmt != null)
                                stmt.close();
                        if (conn != null)
                                conn.close();
                } catch (Exception e) {
                        System.out.println("Error: " + e);
                }
        }
}
```

Query Output:

```
DB Connection Success!!

===Procedure Call for Completed_Rentals_Before_June===
PaymentAmount: 150.0
PaymentStatus: Completed
PaymentDate: 2023-01-10
TransactionID: 1

PaymentAmount: 220.0
PaymentStatus: Completed
PaymentDate: 2023-05-15
TransactionID: 101

===View Call for available_vehicles_with_no_payments===
Make: Chevrolet
Model: Malibu
Year: 2022
DailyRate: 60.0
VINNumber: 4CCDD22222D001122
Location_Address: 101 Shopper St

Make: Tesla
Model: Model S
Year: 2023
DailyRate: 120.0
VINNumber: 7XYZ12345G789012
Location_Address: 321 Corporate St

Make: Audi
Model: Q7
Year: 2023
DailyRate: 100.0
VINNumber: 10PQR78901J234567
Location_Address: 888 Heritage St
```

```
Make: Hyundai
Model: Elantra
Year: 2022
DailyRate: 45.0
VINNumber: 11STU98765K876543
Location_Address: 123 Tech St

===View Call for high_value_cash_transactions===
PaymentAmount: 220.0
DailyRate: 55.0
StartDate: 2023-05-10

=== Average Length of Rental By Location ===
Name: Airport
Average Rental Duration: 5.0

Name: Beachfront
Average Rental Duration: 10.0

Name: Business District
Average Rental Duration: 10.0

Name: Countryside
Average Rental Duration: 10.0

Name: Downtown
Average Rental Duration: 4.0

Name: Garden Heights
Average Rental Duration: 10.0

Name: Historical Quarter
Average Rental Duration: 10.0

Name: Industrial Area
Average Rental Duration: 5.0
```

```
Name: Mountain View
Average Rental Duration: 5.0

Name: Parkside
Average Rental Duration: 5.0

Name: Residential Zone
Average Rental Duration: 5.0

Name: Riverfront
Average Rental Duration: 5.0

Name: Shopping Mall
Average Rental Duration: 9.0

Name: Suburb
Average Rental Duration: 5.0

Name: Tech Park
Average Rental Duration: 5.0

=== First 2 Current Reservations ===
RentalID: 3
VIN Number: 3AABB66666C987654
Start Date: null
End Date: null
Reservation Status: 0
First Name: Alex
Last Name: Johnson
Email: alex.johnson@email.com
Phone: 555-1111
```

```
RentalID: 2
VIN Number: 2DCBA54321B445566
Start Date: 2023-02-10
End Date: 2023-02-15
Reservation Status: 1
First Name: Mark
Last Name: Miller
Email: mark.miller@email.com
Phone: 444-4321

=== Cars At a Location in a given Month ===
Location: Garden Heights
# Cars on Location: 1

Location: Suburb
# Cars on Location: 1

=== Procedure Call for reserved_inventory() ===
First Name: Jane
Last Name: Smith
VIN Number: 2DCBA54321B445566
Location: Suburb, 456 Suburb Ln
Start Date: 2023-02-10
End Date: 2023-02-15

First Name: Sarah
Last Name: Williams
VIN Number: 4CCDD22222D001122
Location: Shopping Mall, 101 Shopper St
Start Date: 2023-04-01
End Date: 2023-04-10
```

```
First Name: Emily
Last Name: Davis
VIN Number: 6TUVW13579F654321
Location: Mountain View, 777 Summit Ln
Start Date: 2023-06-20
End Date: 2023-06-25

First Name: Chris
Last Name: Taylor
VIN Number: 8ABC98765H123456
Location: Industrial Area, 444 Factory Rd
Start Date: 2023-08-15
End Date: 2023-08-20

First Name: Mark
Last Name: Miller
VIN Number: 10PQR78901J234567
Location: Historical Quarter, 888 Heritage St
Start Date: 2023-10-10
End Date: 2023-10-20

First Name: Sophia
Last Name: Clark
VIN Number: 12VWX12345L654321
Location: Parkside, 456 Park Ave
Start Date: 2023-12-15
End Date: 2023-12-20

First Name: Olivia
Last Name: Moore
VIN Number: 14BCD98765N123456
Location: Garden Heights, 101 Garden Blvd
Start Date: 2024-02-10
End Date: 2024-02-20
```

```
=== View Call for rented_inventory ===
Location: Airport, 789 Terminal Ave
Vehicle: 2021 Ford Mustang
Rented By: Johnson, Mike

Location: Mountain View, 777 Summit Ln
Vehicle: 2020 Jeep Wrangler
Rented By: Davis, Emily


=== Function Call for potential_revenue(1) ===
Potential Daily Revenue for Available Rentals: $12090.00


=== Function Call for potential_revenue(0) ===
Daily Revenue of Active Rentals: $290.00
```