Rent-A-Car Term Project

CS 331-H01 | Group 9

Dennis Gannon & Michael Peluso

dmg52@njit.edu | mp272@njit.edu

Business Requirements

For this project, the RentACar business wants to develop a database system to track car inventories, rental contracts, and billing. The business requirements for this database include eight entities each with multiple attributes: Class, Credit Card, Customer, Location, Model, Rental Agreement, Reservation, and Vehicle. These entities are connected via eight different relationships: AT, IS, HAS, CHOSEN, BECOMES, RENTS, AGREES, and MAKES.

Additional assumptions were made based on the original requirements to solidify the database such as giving the Customer, Model, Class, and Reservation entities unique ID attributes (CID, MID, CLID, and RID respectively) to distinguish between instances. Furthermore, the date and time recorded by the Reservation entity do not have to be equal to that of the corresponding Rental_Agreement, every vehicle is given a Color attribute, location phone numbers are tracked, and a customer can only have one concurrent rental agreement at a time. Moreover, the Credit Card attribute of the rental agreement becomes its own entity that is taken from the customer when the agreement is made during the AGREES ternary relationship. This is due to the assumption that a credit card is only provided once the customer agrees to a rental agreement.

Entity-Relationship Design

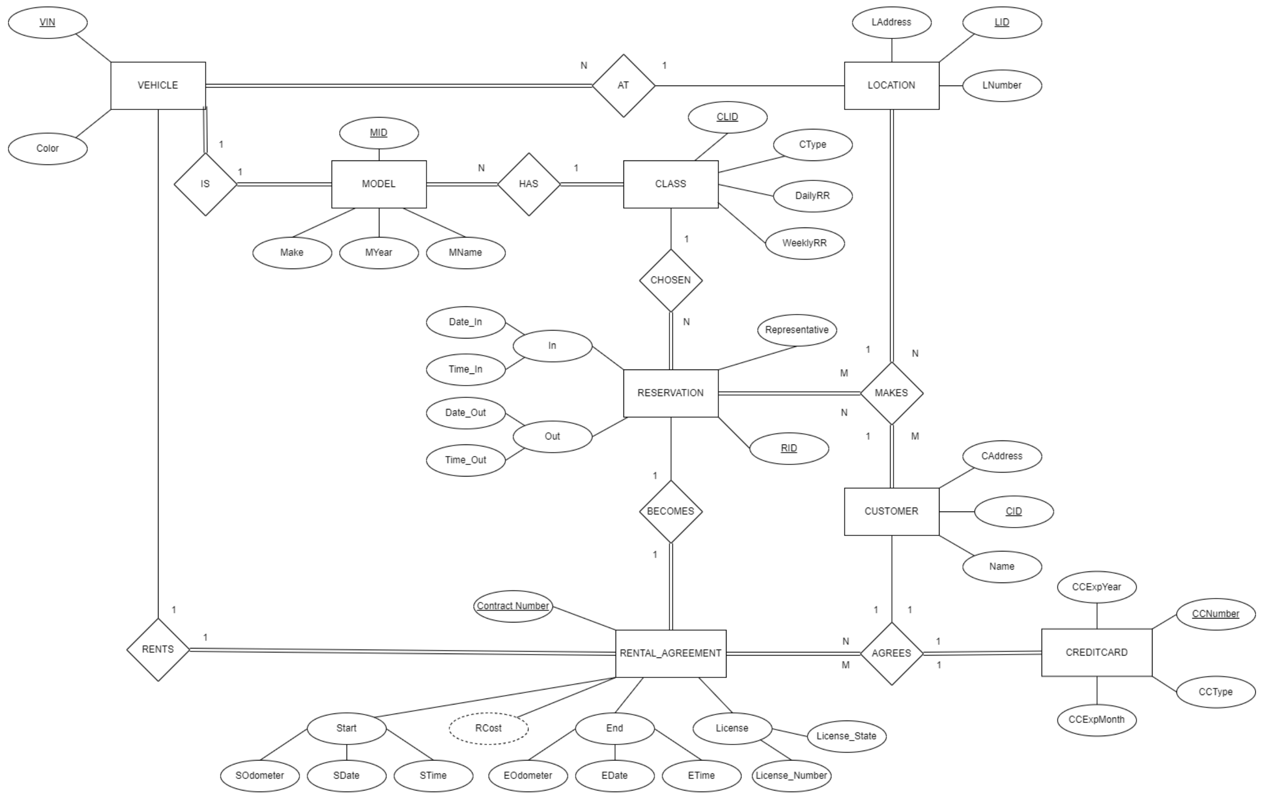


Figure 1

While developing the entity-relationship diagram, there were several major design decisions made that determined the functionality of the database. One such obstacle was finding the primary key for the Reservation, Customer, Class, and Model entities. Initially, the goal was to not add any new attributes; however, the design became too complex and counterintuitive without the addition of their primary keys.

Another challenge was deciding how the Credit Card attribute was to fit in the design as it is technically linked with the customer. A decision was made to make the Credit Card a separate entity. This allows the credit card to properly be traced to both its associated rental agreement and the customer via the AGREES relationship. This also keeps the schema in the third normal form.

Lastly, there was initial difficulty when building a relationship between the Vehicle and Reservation entities to get the Vehicle rental rates (DailyRR and WeeklyRR). To resolve this problem, the Model and Class entities were created, whose relationships chained together allowed for the seamless transfer of attributes under the respective, reserved-vehicle entity.

Logical Database Design

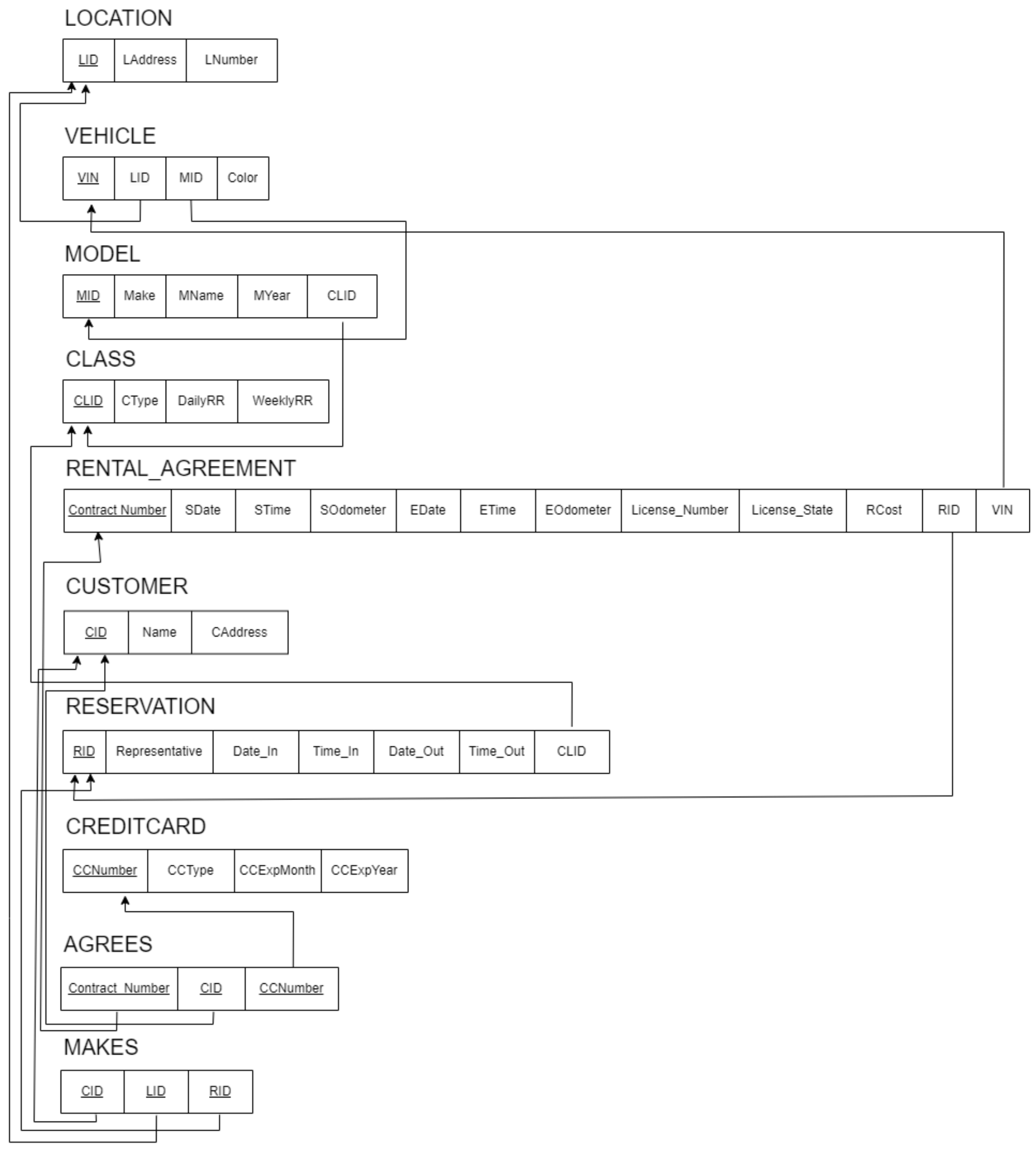

LOCATION
LID LAddress LNumber
VEHICLE
VIN LID MID Color
MODEL
MID Make MName MYear CLID
CLASS
CLID CType DailyRR WeeklyRR
RENTAL_AGREEMENT
Contract Number SDate STime SOdometer EDate ETime EOdometer License_Number License_State RCost RID VIN
CUSTOMER
CID Name CAddress
RESERVATION
RID Representative Date_In Time_In Date_Out Time_Out CLID
CREDITCARD
CCNumber CCType CCExpMonth CCExpYear
AGREES
Contract_Number CID CCNumber
MAKES
CID LID RID

Figure 2

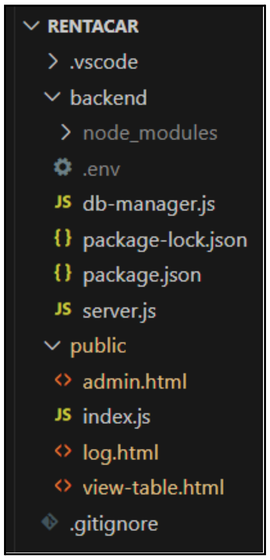
Mapping the relational database was fairly straightforward with each entity becoming its own table along with the ternary relationships AGREES and MAKES. These two relationship tables allow for further normalization, organization, and simplification of the data. The foreign keys are the foundation of the database and are determined by the relationships that the entities belong to. Using the relational map schema in Figure 2, the database was coded into SQL Developer, and the foreign key references were specified.

Decisions needed to be made when creating these foreign keys for "on delete" actions. The majority of the foreign key references are set to NULL values ("on delete set NULL") upon deletion because the referenced entity keys typically should not be deleted as well. The MAKES relationship, however, uses "on delete cascade" for its foreign keys CID, LID, and RID because this table solely depends on foreign key attributes. They make up the table's primary key and should not exist if the customer, location, or reservation that it refers to no longer exists.

It was also decided that the credit card attributes from the rental agreement need to be in its own table to match the third normal form conditions. As a result, the AGREES relationship becomes ternary, similar to the MAKES relationship relation. Likewise, the AGREES relationship uses three foreign for its primary key that cascade on deletion: Contract_Number, CID, and CCNumber.

Application Program Design

It was decided that the application was to be programmed in HTML and JavaScript, with the addition of the Node and Bootstrap frameworks. Several other npm packages and libraries were installed for our workflow as well. Among these, the "**`oracledb`**" npm package allowed for a successful connection to the Oracle database. Due to the nature of these connections, a backend server connection was required and, therefore, was run and managed locally for simplicity.

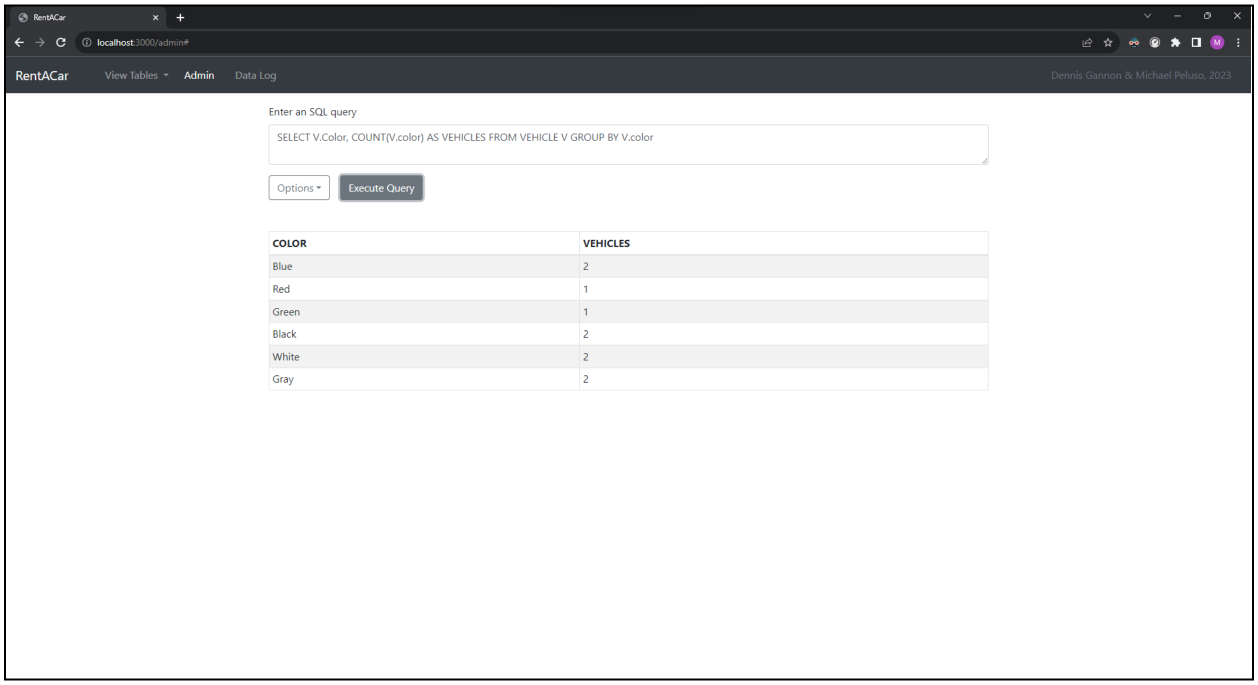


```
∨ RENTACAR
  > .vscode
  ∨ backend
    > node_modules
    .env
    db-manager.js
    package-lock.json
    package.json
    server.js
  ∨ public
    admin.html
    index.js
    log.html
    view-table.html
  .gitignore
```

Figure 3.1

The project directory mainly consisted of a frontend and backend folder, as can be seen in Figure 3.1. These files are available at "https://github.com/michaelpeluso/Rent-A-Car-TermProject".

The frontend folder contains all of the files that manage the front end of the website, relating to its text, visuals, and interactive navigation. Of these, "admin.html" includes the code required to display a page that allows the user to enter their own SQL query or select from the queries listed below in the *Queries in SQL* section of this document. As seen in Figure 3.2, these queries will return a table that can be seen by the user. This page can be accessed by selecting "Admin" at the top left of the page, inside the navigation bar.

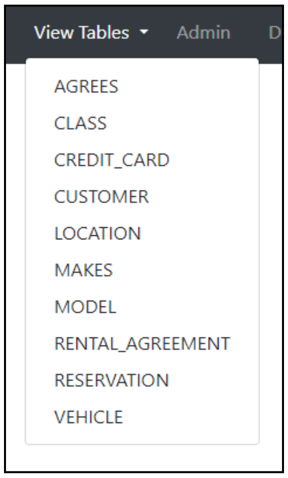


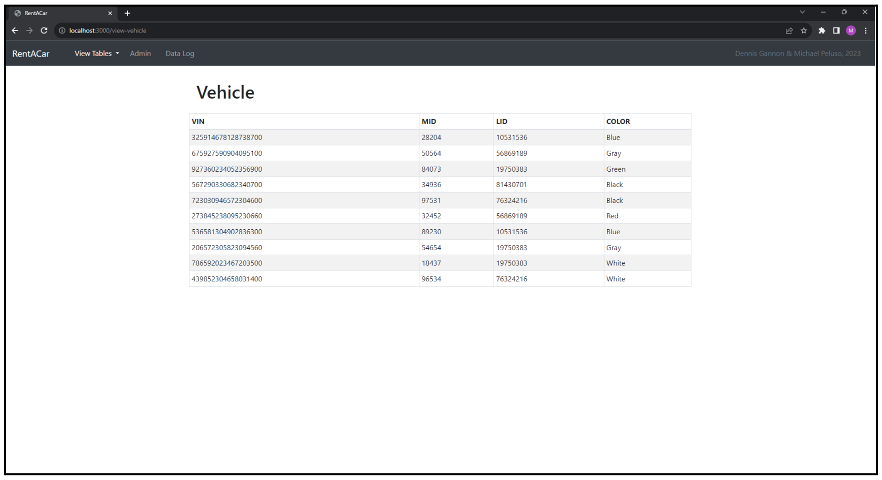
```
SELECT V.Color, COUNT(V.color) AS VEHICLES FROM VEHICLE V GROUP BY V.color
```

COLOR	VEHICLES
Blue	2
Red	1
Green	1
Black	2
White	2
Gray	2

Figure 3.2

The “**`view-table.html`**” file is returned when the user selects any of the options listed in the “View Tables” dropdown menu of Figure 3.3.1. These options consist of each table located in the Oracle database. The table that will be displayed is dependent on the dropdown selection. Figure 3.3.2 shows “**`view-table.html`**” after “VEHICLE” is selected from the “View Tables” dropdown.

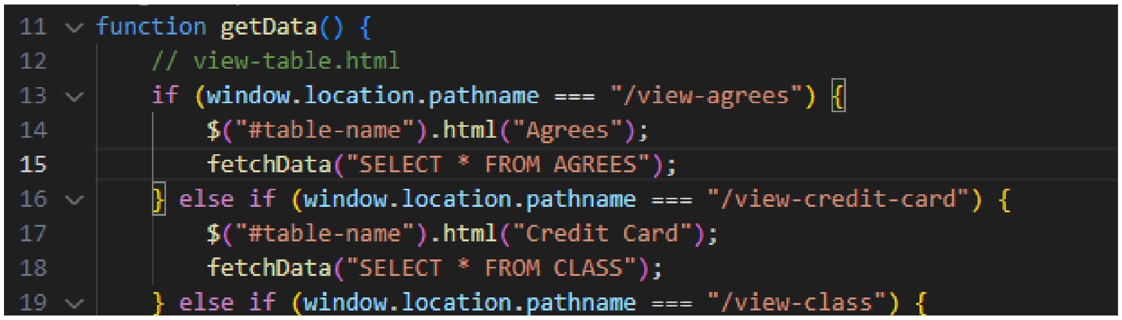



RentACar
localhost:3000/view-vehicle
RentACar
View Tables
Admin
Data Log
Dennis Gannon & Michael Peluso, 2023
Vehicle

VIN	MID	LID	COLOR
325914678128738700	28204	10531536	Blue
675927590904095100	50564	56869189	Gray
927360234052356900	84073	19750383	Green
567290330682340700	34936	81430701	Black
723030946572304600	97531	76324216	Black
273845238095230660	32452	56869189	Red
536581304902836300	89230	10531536	Blue
206572305823094560	54654	19750383	Gray
786592023467203500	18437	19750383	White
439852304658031400	96534	76324216	White

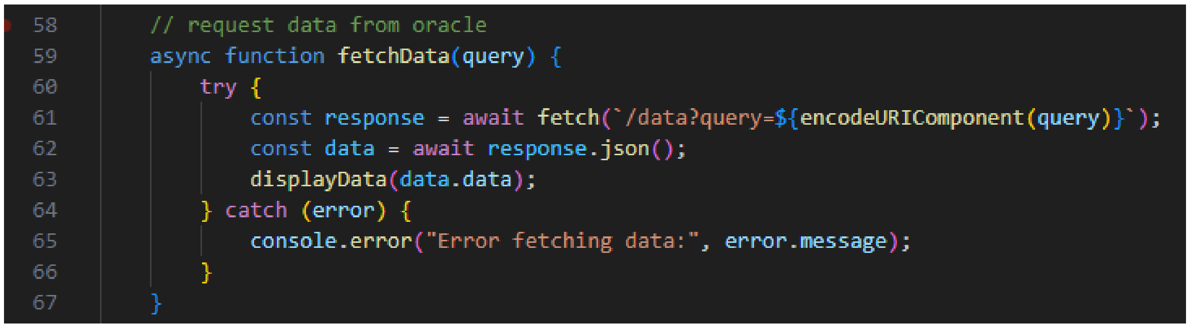
Figure 3.3.1 (left) & Figure 3.3.2 (right)

Accessing “**admin.html**” or “**view-table.html**” will forward a query request from “**index.js**”, which includes the specific SQL query that will return the given data. Two main JavaScript functions execute this request: “**getData()**” and “**fetchData()**” as seen in Figure 3.4.1 and Figure 3.4.2 respectively. A specific SQL query is selected in “**getData()**” and sent to “**fetchData()**” when a given page loads. The function “**fetchData()**” will then take that query and fetch a **JSON** object (line 62) including the data from the Oracle database. The returned data is then passed into a new function, where the query response is parsed and printed into a table on the given HTML file.



```
11  function getData() {
12      // view-table.html
13      if (window.location.pathname === "/view-agrees") {
14          $("#table-name").html("Agrees");
15          fetchData("SELECT * FROM AGREES");
16      } else if (window.location.pathname === "/view-credit-card") {
17          $("#table-name").html("Credit Card");
18          fetchData("SELECT * FROM CLASS");
19      } else if (window.location.pathname === "/view-class") {
```

Figure 3.4.1



```
58      // request data from oracle
59      async function fetchData(query) {
60          try {
61              const response = await fetch(`/data?query=${encodeURIComponent(query)}`);
62              const data = await response.json();
63              displayData(data.data);
64          } catch (error) {
65              console.error("Error fetching data:", error.message);
66          }
67      }
```

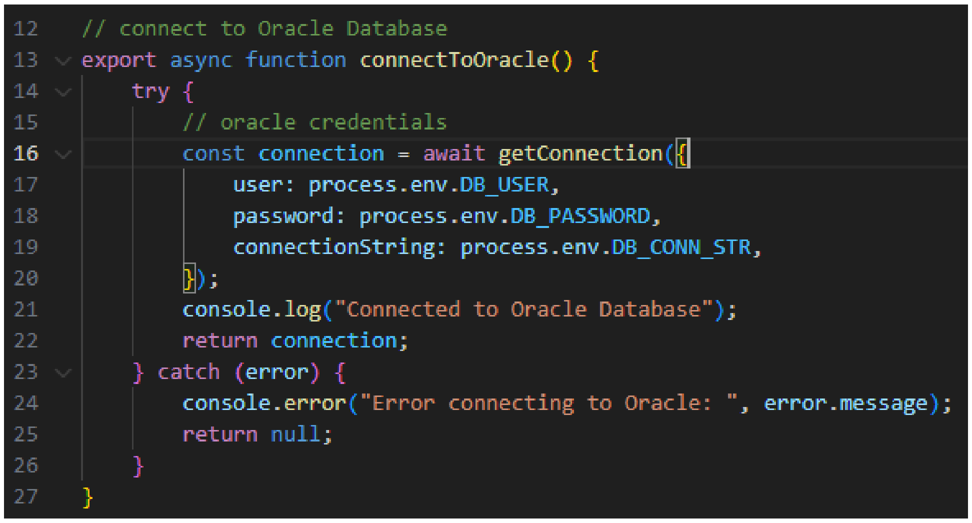
Figure 3.4.2

“**log.html**” simply displays all recent SQL queries made by the server by intercepting these fetch calls, collecting the data returned, and printing it in the HTML file.

Regarding the back end of the website, the routing is done by the "**`server.js`**" file. It manages the fetch calls executed above in "**`index.js`**". Simply put, the server first attempts to connect to the Oracle database, request data given a SQL query and then close the database connection.

The "**`db-manager.js`**" holds all of the code that manages the interactions between the server and the Oracle database. Its functions "**`connectToOracle()`**", "**`getData()`**", and "**`closeConnection()`**" are held here and exported to "**`server.js`**" for practical use. They are also executed sequentially in their respective order each time a new data request is made.

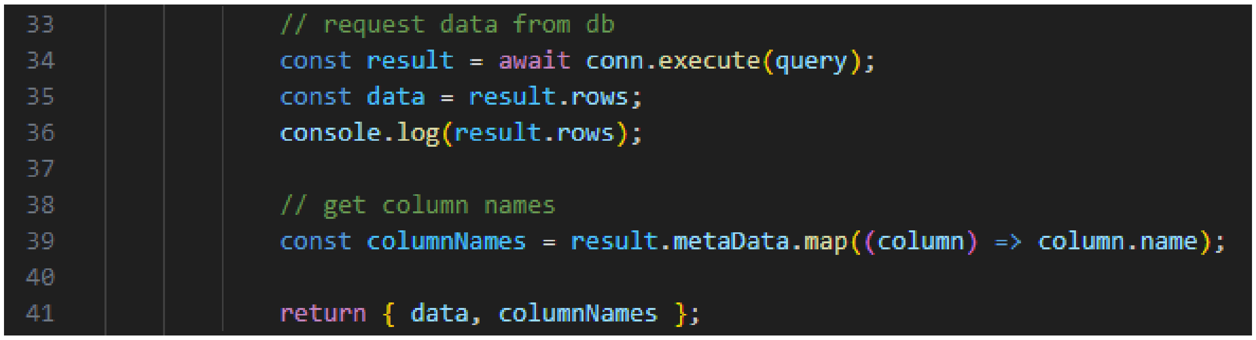
The code connecting to the database can be seen in Figure 3.5.1. It consists of an asynchronous call to the database and passes the required parameters - a username, password, and database connection string. The connection string consists of several components of which the notable values include the *host address*, *port number*, and *SID*. Using the values given in the beginning of the semester, the connection string is "**`(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=prophet.njit.edu)(PORT=1521))(CONNECT_DATA=(SERVER=DEDICATED)(SID=course)))`**". These three connection values, which are stored in a private "**`.env`**" file, are passed as a JavaScript object into a function included in the "**`oracledb`**" package, as seen on lines 16 through 20.



```
// connect to Oracle Database
export async function connectToOracle() {
    try {
        // oracle credentials
        const connection = await getConnection({
            user: process.env.DB_USER,
            password: process.env.DB_PASSWORD,
            connectionString: process.env.DB_CONN_STR,
        });
        console.log("Connected to Oracle Database");
        return connection;
    } catch (error) {
        console.error("Error connecting to Oracle: ", error.message);
        return null;
    }
}
```

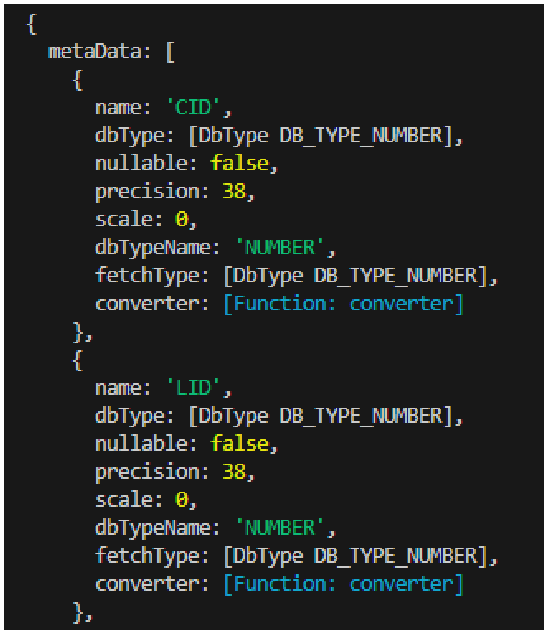
Figure 3.5.1

The "**`getData()`**" function simply confirms a valid query and database connection. It uses this connection to pass the given query and expects an object in return (see Figure 3.6.1). This object consists of some metadata regarding the data table as well as the rows containing the actual data values in the table. The response for the query "**`SELECT * FROM MAKES`**" can be seen below in Figures 3.6.2 and 3.6.3.

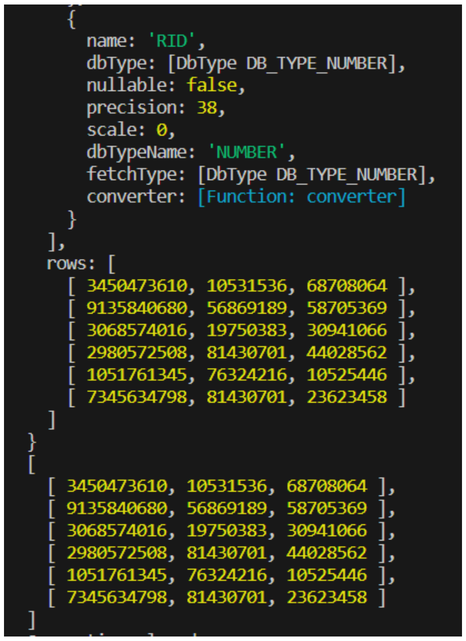


```
            // request data from db
            const result = await conn.execute(query);
            const data = result.rows;
            console.log(result.rows);

            // get column names
            const columnNames = result.metaData.map((column) => column.name);

            return { data, columnNames };
```

Figure 3.6.1



```
{
  metaData: [
    {
      name: 'CID',
      dbType: [DbType DB_TYPE_NUMBER],
      nullable: false,
      precision: 38,
      scale: 0,
      dbTypeName: 'NUMBER',
      fetchType: [DbType DB_TYPE_NUMBER],
      converter: [Function: converter]
    },
    {
      name: 'LID',
      dbType: [DbType DB_TYPE_NUMBER],
      nullable: false,
      precision: 38,
      scale: 0,
      dbTypeName: 'NUMBER',
      fetchType: [DbType DB_TYPE_NUMBER],
      converter: [Function: converter]
    },
```



```
    {
      name: 'RID',
      dbType: [DbType DB_TYPE_NUMBER],
      nullable: false,
      precision: 38,
      scale: 0,
      dbTypeName: 'NUMBER',
      fetchType: [DbType DB_TYPE_NUMBER],
      converter: [Function: converter]
    }
  ],
  rows: [
    [ 3450473610, 10531536, 68708064 ],
    [ 9135840680, 56869189, 58705369 ],
    [ 3068574016, 19750383, 30941066 ],
    [ 2980572508, 81430701, 44028562 ],
    [ 1051761345, 76324216, 10525446 ],
    [ 7345634798, 81430701, 23623458 ]
  ]
}
[
  [ 3450473610, 10531536, 68708064 ],
  [ 9135840680, 56869189, 58705369 ],
  [ 3068574016, 19750383, 30941066 ],
  [ 2980572508, 81430701, 44028562 ],
  [ 1051761345, 76324216, 10525446 ],
  [ 7345634798, 81430701, 23623458 ]
]
```

Figure 3.6.2 (left) and Figure 3.6.1 (right)

This long line of function calls between several files allows the data from Oracle to be viewed on a webpage. In other words, the SQL query is built from user input. Then a connection is made to the database, the query is executed, and that data is sent back to the user as output.

Normalize the Relations

Using the schema in Figure 2, the relations can be written as follows:

LOCATION (LID, LAddress, LNumber)

VEHICLE (VIN, Color, *LID*, *MID*)

MODEL (MID, Make, MName, MYear, *CLID*)

CLASS (CLID, CType, DailyRR, WeeklyRR)

RENTAL_AGREEMENT (Contract_Number, SDate, Stime, SOdometer, EDate, ETime, EOdometer, License_Number, License_State, RCost, *RID*, *VIN*)

CUSTOMER (CID, Name, CAddress)

RESERVATION (RID, Representative, Date_In, Time_In, Date_Out, Time_Out, *CLID*)

CREDITCARD (CCNumber, CCType, CCExpMonth, CCExpYear)

MAKES (*CID*, *LID*, *RID*)

AGREES (*Contract_Number*, *CID*, *CCNumber*)

† *Underlined attributes indicate keys and italicized attributes indicate foreign keys.*

The sample data to be used in their respective tables (in SQL):

```
//LOCATION
INSERT INTO LOCATION VALUES (10531536, '123 Main St', 3908810909);
INSERT INTO LOCATION VALUES (56869189, '409 Ridgedale Ave', 1234567890);
INSERT INTO LOCATION VALUES (19750383, '70 Pine Dr', 3210984567);
INSERT INTO LOCATION VALUES (81430701, '101 Elm St', 9001000000);
INSERT INTO LOCATION VALUES (76324216, '20 Maple St', 1112223333);

//VEHICLE
INSERT INTO VEHICLE VALUES (325914678128738695, 28204, 10531536, 'Blue');
INSERT INTO VEHICLE VALUES (675927590904095109, 50564, 56869189, 'Gray');
INSERT INTO VEHICLE VALUES (927360234052356823, 84073, 19750383, 'Green');
INSERT INTO VEHICLE VALUES (567290330682340697, 34936, 81430701, 'Black');
INSERT INTO VEHICLE VALUES (723030946572304659, 97531, 76324216, 'Black');

//MODEL
INSERT INTO MODEL VALUES (28204, 15, 'Toyota', 'Camry', 2021);
INSERT INTO MODEL VALUES (50564, 19, 'Honda', 'CR-V', 2017);
INSERT INTO MODEL VALUES (84073, 28, 'Ford', 'F-150', 2019);
INSERT INTO MODEL VALUES (34936, 30, 'Chevrolet', 'Camaro', 2022);
INSERT INTO MODEL VALUES (97531, 43, 'Nissan', 'Quest', 2020);

//CLASS
INSERT INTO CLASS VALUES (15, 'Sedan', 29.99, 199.99);
INSERT INTO CLASS VALUES (19, 'SUV', 39.99, 249.99);
INSERT INTO CLASS VALUES (28, 'Truck', 54.99, 299.99);
INSERT INTO CLASS VALUES (30, 'Coupe', 59.99, 349.99);
INSERT INTO CLASS VALUES (43, 'Van', 39.99, 219.99);

//RENTAL_AGREEMENT
```

```
INSERT INTO RENTAL_AGREEMENT (ContractNumber, RID, VIN, SDate, STime,
SOdometer, EDate, ETime, EOdometer, LicenseNumber, LicenseState)
VALUES (8120934680987530 1715, 68708064, 325914678128738695, '2023-01-13', '12:00',
95672, '2023-01-16', '15:00', 98056, 'DL123', 'CA');
INSERT INTO RENTAL_AGREEMENT (ContractNumber, RID, VIN, SDate, STime,
SOdometer, EDate, ETime, EOdometer, LicenseNumber, LicenseState)
VALUES (23740958723453128953, 58705369, 675927590904095109, '2023-02-05', '14:30',
52087, '2023-02-10', '10:00', 59318, 'DL456', 'NY');
INSERT INTO RENTAL_AGREEMENT (ContractNumber, RID, VIN, SDate, STime,
SOdometer, EDate, ETime, EOdometer, LicenseNumber, LicenseState)
VALUES (54659783665120063319, 30941066, 927360234052356823, '2023-06-10', '10:00',
32798, '2023-06-12', '16:00', 33034, 'DL789', 'TX');
INSERT INTO RENTAL_AGREEMENT (ContractNumber, RID, VIN, SDate, STime,
SOdometer, EDate, ETime, EOdometer, LicenseNumber, LicenseState)
VALUES (84405840680120659817, 44028562, 567290330682340697, '2023-07-15', '09:00',
16864, '2023-07-29', '12:30', 19073, 'DL012', 'FL');
INSERT INTO RENTAL_AGREEMENT (ContractNumber, RID, VIN, SDate, STime,
SOdometer, EDate, ETime, EOdometer, LicenseNumber, LicenseState)
VALUES (75092134509172058434, 10525446, 723030946572304659, '2023-10-20', '08:00',
10156, '2023-10-25', '14:00', 11608, 'DL345', 'CA');

//CUSTOMER
INSERT INTO CUSTOMER VALUES (3450473610, 'John Doe', '36 Poplar Dr');
INSERT INTO CUSTOMER VALUES (9135840680, 'Jane Smith', '12 Grove Pl');
INSERT INTO CUSTOMER VALUES (3068574016, 'Bob Johnson', '29 Parkshire Ln');
INSERT INTO CUSTOMER VALUES (2980572508, 'Alice Brown', '94 Summerset Ln');
INSERT INTO CUSTOMER VALUES (1051761345, 'Charlie Wilson', '72 Washington Rd');

//RESERVATION
INSERT INTO RESERVATION VALUES (68708064, 'Danielle Cannon', '2023-01-01', '12:00',
'2023-01-03', '15:00', 15);
INSERT INTO RESERVATION VALUES (58705369, 'Joe Dustin', '2023-02-05', '14:30',
'2023-02-10', '10:00', 19);
INSERT INTO RESERVATION VALUES (30941066, 'Alex Lennon', '2023-05-21', '10:00',
'2023-05-28', '16:00', 28);
INSERT INTO RESERVATION VALUES (44028562, 'Sally Rose', '2023-04-15', '09:00',
'2023-04-20', '12:30', 30);
INSERT INTO RESERVATION VALUES (10525446, 'Charlie Wilson', '2023-05-20', '08:00',
'2023-05-25', '14:00', 43);
```

```
//CREDITCARD
INSERT INTO CREDITCARD VALUES (7223340130693475, 'Visa', 12, 2024);
INSERT INTO CREDITCARD VALUES (2612569825903560, 'Mastercard', 10, 2023);
INSERT INTO CREDITCARD VALUES (4064014054106690, 'Amex', 3, 2025);
INSERT INTO CREDITCARD VALUES (2087602375902761, 'Discover', 9, 2023);
INSERT INTO CREDITCARD VALUES (4653206802345978, 'Visa', 8, 2024);

//MAKES
INSERT INTO MAKES VALUES (3450473610, 10531536, 68708064);
INSERT INTO MAKES VALUES (9135840680, 56869189, 58705369);
INSERT INTO MAKES VALUES (3068574016, 19750383, 30941066);
INSERT INTO MAKES VALUES (2980572508, 81430701, 44028562);
INSERT INTO MAKES VALUES (1051761345, 76324216, 10525446);

//AGREES
INSERT INTO AGREES VALUES (81209346809875301715, 3450473610,
7223340130693475);
INSERT INTO AGREES VALUES (23740958723453128953, 9135840680,
2612569825903560);
INSERT INTO AGREES VALUES (54659783665120063319, 3068574016,
4064014054106690);
INSERT INTO AGREES VALUES (84405840680120659817, 2980572508,
2087602375902761);
INSERT INTO AGREES VALUES (75092134509172058434, 1051761345,
4653206802345978);
```

In relation LOCATION, the only candidate key is LID and, therefore, must be the primary key. The attributes LAddress and LNumber are fully functional dependencies on the key LID.

In relation VEHICLE, the primary key is the VIN attribute and the attribute Color is fully functionally dependent on VIN. This relation also has two foreign keys referencing two different relations, LID and MID.

Relation MODEL has the primary key MID and the fully functionally dependent attributes Make, MName, and MYear. CLID is a foreign key referencing the CLASS relation.

Relation CLASS has the primary key CLID, in which the attributes CType, DailyRR, and WeeklyRR are fully functional dependencies.

The relation RENTAL_AGREEMENT has the primary key Cotnract_Number. This relation has many attributes that are all fully functional dependencies upon the primary key: SDate, Stime, SOdometer, EDate, ETime, EOdometer, License_Number, License_State, and RCost. This relation also has the foreign keys RID and VIN.

In relation CUSTOMER, the primary key is CID, and the attributes Name and CAddress are attributes that are fully functional dependencies. This relation does not have any foreign keys but is associated with both relationship relations, AGREES and MAKES.

Relation RESERVATION has the primary key of RID. The attributes Representative, Date_In, Time_In, Date_Out, and Time_Out all have fully functional dependencies on the primary key, RID. CLID is a foreign key that references the CLASS relation.

The relation CREDITCARD takes the primary key of CCNumber. Its attributes CCType, ExpMonth, and ExpYear have fully functional dependencies with the primary key. This relation has no foreign keys but is referenced for the rental agreement.

The relation MAKES is a relationship relation that has the superkey CID, LID, and RID which are all foreign keys referencing CUSTOMER, LOCATION, and RESERVATION respectively. There are no other dependencies as there are no other attributes in the relation.

Similarly, the relationship relation AGREES has a superkey made up of three foreign keys: Contract_Number, CID, and CCNumber. These are the only columns in this relation so that it can link the relations together.

Based on the above information, all of the database relations are in the first normal form because all the values in each relation are atomic. Thus, only one value can be in each attribute for an instance of the entity. Additionally, all the attributes have unique names and do not collide between multiple entities.

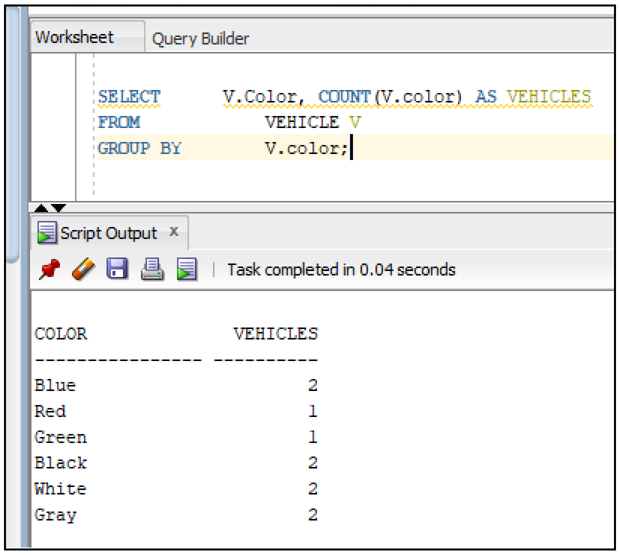
As aforementioned, all the attributes in every relation that are not the primary keys, are fully functionally dependent on the primary key. There are no partial dependencies of non-prime attributes. Combined with the fact that the relations are already in the first normal form (1NF), the relations are, therefore, in the second normal form (2NF).

To check for the third normal form (3NF), the relations must be in 2NF and there can not be any transitive dependencies. These dependencies were removed from the schema by decomposing the RENTAL_AGREEMENT relation to take out the credit card attributes that were transitive to the CCNumber. With the AGREES relationship relation and the CREDITCARD relation, the entire database schema is now in 3NF.

Queries in SQL

1) Select the number of vehicles of each color that are sold across all locations.

```
SELECT          V.Color, COUNT(V.color) AS VEHICLES
FROM            VEHICLE V
GROUP BY        V.color;
```



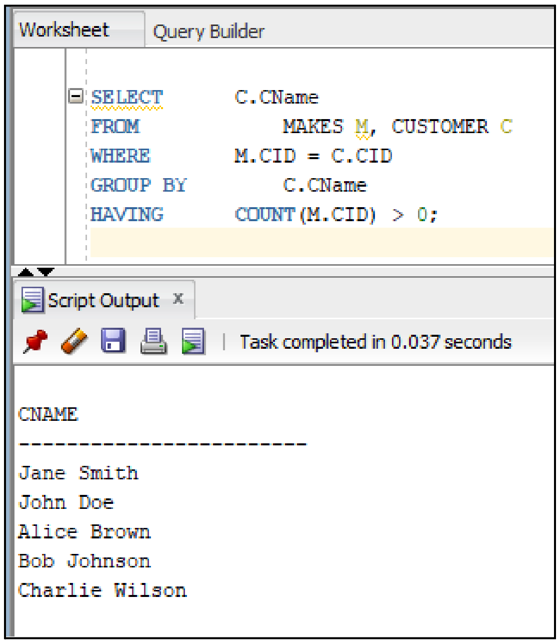
```
SELECT      V.Color, COUNT(V.color) AS VEHICLES
FROM              VEHICLE V
GROUP BY          V.color;
```

```
COLOR              VEHICLES
---------------- ----------
Blue                      2
Red                       1
Green                     1
Black                     2
White                     2
Gray                      2
```

Figure 4.1

2) Select all the customers that have made a reservation at any location.

```
SELECT          C.CName
FROM                MAKES M, CUSTOMER C
WHERE           M.CID = C.CID
GROUP BY        C.CName
HAVING          COUNT(M.CID) > 0;
```



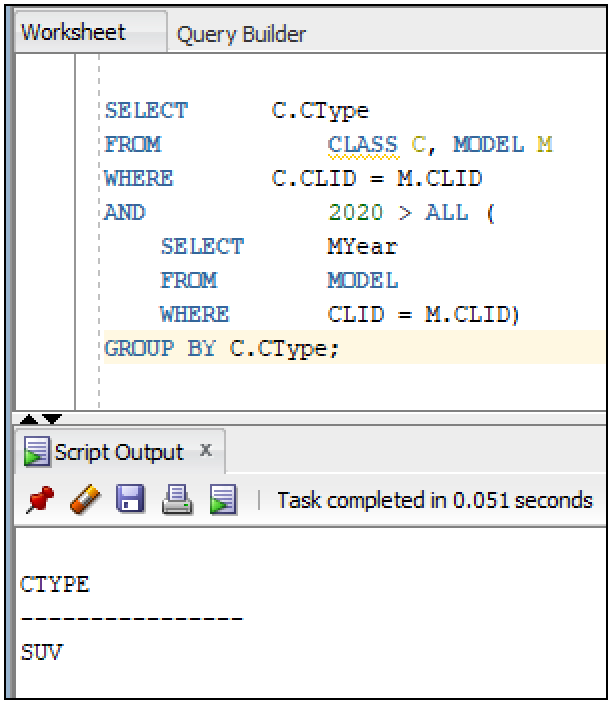
```
SELECT       C.CName
FROM             MAKES M, CUSTOMER C
WHERE        M.CID = C.CID
GROUP BY         C.CName
HAVING       COUNT(M.CID) > 0;
```

```
CNAME
------------------------
Jane Smith
John Doe
Alice Brown
Bob Johnson
Charlie Wilson
```

Figure 4.2

3) Select the classes where all of its models are older than 2020.

```
SELECT          C.CType
FROM                CLASS C, MODEL M
WHERE           C.CLID = M.CLID
AND             2020 > ALL (
   SELECT       MYear
   FROM         MODEL
   WHERE        CLID = M.CLID)
GROUP BY C.CType;
```



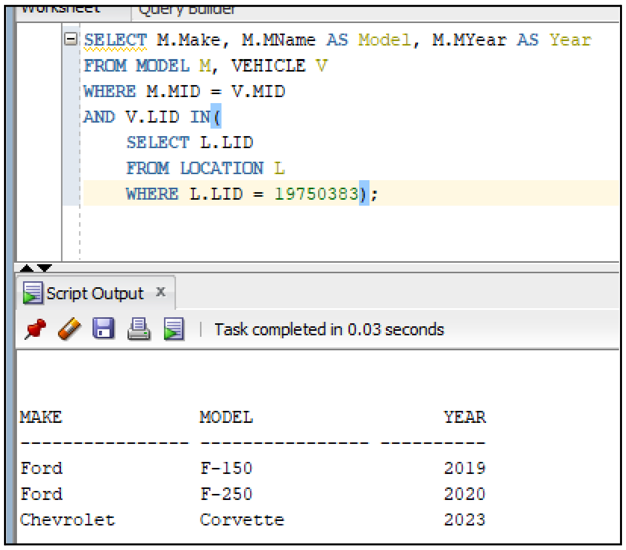
```
SELECT      C.CType
FROM            CLASS C, MODEL M
WHERE       C.CLID = M.CLID
AND             2020 > ALL (
    SELECT      MYear
    FROM        MODEL
    WHERE       CLID = M.CLID)
GROUP BY C.CType;
```

```
CTYPE
----------------
SUV
```

Figure 4.3

4) For each vehicle model at Location with an ID of 19750383, select the make, model, and year.

SELECT M.Make, M.MName AS Model, M.MYear AS Year
FROM MODEL M, VEHICLE V
WHERE M.MID = V.MID
AND V.LID IN(
   SELECT L.LID
   FROM LOCATION L
   WHERE L.LID = 19750383);



```
SELECT M.Make, M.MName AS Model, M.MYear AS Year
FROM MODEL M, VEHICLE V
WHERE M.MID = V.MID
AND V.LID IN(
    SELECT L.LID
    FROM LOCATION L
    WHERE L.LID = 19750383);
```

```
MAKE             MODEL                  YEAR
---------------- ---------------- ----------
Ford             F-150                  2019
Ford             F-250                  2020
Chevrolet        Corvette               2023
```

Figure 4.4

Conclusion

Upon reflection of the three phases of this project, the most difficult was most likely the first, where the task was to develop the entity-relationship diagram of the selected project. In this portion of the assignment, the project descriptions were picked apart and analyzed thoroughly to understand the requirements that build the diagram and schema. If one relationship was incorrect or did not effectively demonstrate the database's purpose or objective, then the diagram needed to be reworked. Multiple different versions of the diagram and schema were developed and iterated upon to reach the final database plan. However, the final phase of the project provided its challenges as it required the use of unfamiliar techniques to create the application program.

Meanwhile, the easiest phase for the group was the second phase to develop the Oracle database on SQL developer. After learning the process in class, coding was straightforward and did not pose many problems. It took some time but was nothing too difficult.

If we were to do this project again with what we know now, all of the phases would have been completed much more efficiently. There were several mistakes we made in the first phase that had to be adjusted during the second phase and could have saved us time. Furthermore, the application program would have changed slightly as we have more experience with creating the connection with the database.

Overall, this project was a success because we used an accumulation of lessons learned throughout the semester and could apply them in practical, sample situations. We do believe it would have been beneficial if we learned a little more about setting up the connection to the database with JavaScript and Node before the final phase of creating the web pages.

Ultimately, developing a database requires methodical planning and a thorough understanding of the process. From dissecting business requirements to developing a schema and creating a database to be accessed from web pages, there are many steps involved. We believe that we have effectively demonstrated this process and have simultaneously learned more about its application.

Appendix

Complete SQL database code:

```
// Create Tables
ALTER SESSION SET nls_date_format='yyyy-mm-dd';
ALTER SESSION SET nls_timestamp_format='hh24:mi';

CREATE TABLE VEHICLE (
   VIN INT NOT NULL,
   MID INT,
   LID INT,
   Color VARCHAR(16),
   PRIMARY KEY (VIN)
);
CREATE TABLE LOCATION (
   LID INT NOT NULL,
   LAddress VARCHAR(48),
   LNumber INT,
   PRIMARY KEY (LID)
);
CREATE TABLE MODEL (
   MID INT NOT NULL,
   CLID INT,
   Make VARCHAR(16),
   MName VARCHAR(16),
   MYear INT,
   PRIMARY KEY (MID)
);
CREATE TABLE CLASS (
   CLID INT NOT NULL,
   CType VARCHAR(16),
   DailyRR FLOAT,
   WeeklyRR FLOAT,
   PRIMARY KEY (CLID)
);
CREATE TABLE RENTAL_AGREEMENT (
   ContractNumber INT NOT NULL,
   RID INT,
   VIN INT,
   SDate DATE,
   STime TIMESTAMP,
   SOdometer INT,
   EDate DATE,
   ETime TIMESTAMP,
   EOdometer INT,
   LicenseNumber VARCHAR(16),
   LicenseState VARCHAR(12),
   RCost FLOAT,
   PRIMARY KEY (ContractNumber),
   FOREIGN KEY (VIN) REFERENCES VEHICLE(VIN) ON DELETE SET NULL
);
CREATE TABLE CUSTOMER (
   CID INT NOT NULL,
   CName VARCHAR(24) NOT NULL,
   CAddress VARCHAR(48),
   PRIMARY KEY (CID)
);
CREATE TABLE RESERVATION (
   RID INT NOT NULL,
   Representative VARCHAR(24),
   Date_In DATE,
   Time_In TIMESTAMP,
   Date_Out DATE,
   Time_Out TIMESTAMP,
   CLID INT,
   PRIMARY KEY (RID),
   FOREIGN KEY (CLID) REFERENCES CLASS(CLID) ON DELETE SET NULL
);
CREATE TABLE CREDITCARD(
   CCNumber INT NOT NULL,
   CCType VARCHAR(16) NOT NULL,
   CCExpMonth INT NOT NULL,
   CCExpYear INT NOT NULL,
   PRIMARY KEY (CCNumber)
);
CREATE TABLE MAKES(
   CID INT,
   LID INT,
   RID INT,
   PRIMARY KEY (CID, LID, RID),
   FOREIGN KEY (CID) REFERENCES CUSTOMER(CID) ON DELETE CASCADE,
   FOREIGN KEY (LID) REFERENCES LOCATION(LID) ON DELETE CASCADE,
   FOREIGN KEY (RID) REFERENCES RESERVATION(RID) ON DELETE CASCADE
);
CREATE TABLE AGREES(
   ContractNumber INT,
   CID INT,
   CCNumber INT,
   PRIMARY KEY (ContractNumber, CID, CCNumber),
   FOREIGN KEY (ContractNumber) REFERENCES
RENTAL_AGREEMENT(ContractNumber) ON DELETE CASCADE,
   FOREIGN KEY (CID) REFERENCES CUSTOMER(CID) ON DELETE CASCADE,
   FOREIGN KEY (CCNumber) REFERENCES CREDITCARD(CCNumber) ON DELETE
CASCADE
);
ALTER TABLE VEHICLE ADD FOREIGN KEY (MID) REFERENCES MODEL(MID) ON
DELETE SET NULL;
ALTER TABLE VEHICLE ADD FOREIGN KEY (LID) REFERENCES LOCATION(LID) ON
DELETE SET NULL;
ALTER TABLE MODEL ADD FOREIGN KEY (CLID) REFERENCES CLASS(CLID);
ALTER TABLE RENTAL_AGREEMENT ADD FOREIGN KEY (RID) REFERENCES
RESERVATION(RID) ON DELETE SET NULL;

/
//Create Trigger for RCost
CREATE OR REPLACE TRIGGER calculate_cost BEFORE INSERT OR UPDATE ON
RENTAL_AGREEMENT
FOR EACH ROW
DECLARE
   rental_days NUMBER;
   rental_rate NUMBER;
BEGIN
   rental_days := NVL(:NEW.EDate - :NEW.SDate, 0);

   SELECT
      CASE
         WHEN rental_days < 7 THEN c.DailyRR
         ELSE c.WeeklyRR
      END
   INTO rental_rate
```

```
    FROM RESERVATION r
    JOIN CLASS c ON r.CLID = c.CLID
    WHERE r.RID = :NEW.RID;

    :NEW.RCost := rental_days * rental_rate;
END;

/
// INSERT data
ALTER SESSION SET nls_date_format='yyyy-mm-dd';
ALTER SESSION SET nls_timestamp_format='hh24:mi';

INSERT INTO LOCATION VALUES (10531536, '123 Main St', 3908810909);
INSERT INTO LOCATION VALUES (56869189, '409 Ridgedale Ave', 1234567890);
INSERT INTO LOCATION VALUES (19750383, '70 Pine Dr', 3210984567);
INSERT INTO LOCATION VALUES (81430701, '101 Elm St', 9001000000);
INSERT INTO LOCATION VALUES (76324216, '20 Maple St', 1112223333);
INSERT INTO LOCATION VALUES (24562345, '204 Bernard Rd', 8008889000);
INSERT INTO LOCATION VALUES (34674326, '82 Salem Dr', 4215553333);

INSERT INTO CLASS VALUES (15, 'Sedan', 29.99, 199.99);
INSERT INTO CLASS VALUES (19, 'SUV', 39.99, 249.99);
INSERT INTO CLASS VALUES (28, 'Truck', 54.99, 299.99);
INSERT INTO CLASS VALUES (30, 'Coupe', 59.99, 349.99);
INSERT INTO CLASS VALUES (43, 'Van', 39.99, 219.99);

INSERT INTO MODEL VALUES (28204, 15, 'Toyota', 'Camry', 2021);
INSERT INTO MODEL VALUES (50564, 19, 'Honda', 'CR-V', 2017);
INSERT INTO MODEL VALUES (84073, 28, 'Ford', 'F-150', 2019);
INSERT INTO MODEL VALUES (34936, 30, 'Chevrolet', 'Camaro', 2022);
INSERT INTO MODEL VALUES (97531, 43, 'Nissan', 'Quest', 2020);
INSERT INTO MODEL VALUES (32452, 15, 'Toyota', 'Corolla', 2022);
INSERT INTO MODEL VALUES (89230, 19, 'Honda', 'Pilot', 2018);
INSERT INTO MODEL VALUES (54654, 28, 'Ford', 'F-250', 2020);
INSERT INTO MODEL VALUES (18437, 30, 'Chevrolet', 'Corvette', 2023);
INSERT INTO MODEL VALUES (96534, 43, 'Nissan', 'NV200', 2021);

INSERT INTO RESERVATION VALUES (68708064, 'Danielle Cannon', '2023-01-01', '12:00',
'2023-01-03', '15:00', 15);
```

```
INSERT INTO RESERVATION VALUES (58705369, 'Joe Dustin', '2023-02-05', '14:30',
'2023-02-10', '10:00', 19);
INSERT INTO RESERVATION VALUES (30941066, 'Alex Lennon', '2023-05-21', '10:00',
'2023-05-28', '16:00', 28);
INSERT INTO RESERVATION VALUES (44028562, 'Sally Rose', '2023-04-15', '09:00',
'2023-04-20', '12:30', 30);
INSERT INTO RESERVATION VALUES (10525446, 'Charlie Wilson', '2023-05-20', '08:00',
'2023-05-25', '14:00', 43);
INSERT INTO RESERVATION VALUES (23623458, 'James Madison', '2023-11-01', '12:00',
'2023-11-03', '15:00', 30);

INSERT INTO VEHICLE VALUES (325914678128738695, 28204, 10531536, 'Blue');
INSERT INTO VEHICLE VALUES (675927590904095109, 50564, 56869189, 'Gray');
INSERT INTO VEHICLE VALUES (927360234052356823, 84073, 19750383, 'Green');
INSERT INTO VEHICLE VALUES (567290330682340697, 34936, 81430701, 'Black');
INSERT INTO VEHICLE VALUES (723030946572304659, 97531, 76324216, 'Black');
INSERT INTO VEHICLE VALUES (273845238095230649, 32452, 56869189, 'Red');
INSERT INTO VEHICLE VALUES (536581304902836289, 89230, 10531536, 'Blue');
INSERT INTO VEHICLE VALUES (206572305823094573, 54654, 19750383, 'Gray');
INSERT INTO VEHICLE VALUES (786592023467203456, 18437, 19750383, 'White');
INSERT INTO VEHICLE VALUES (439852304658031434, 96534, 76324216, 'White');

INSERT INTO RENTAL_AGREEMENT (ContractNumber, RID, VIN, SDate, STime,
SOdometer, EDate, ETime, EOdometer, LicenseNumber, LicenseState)
VALUES (81209346809875301715, 68708064, 325914678128738695, '2023-01-13', '12:00',
95672, '2023-01-16', '15:00', 98056, 'DL123', 'CA');
INSERT INTO RENTAL_AGREEMENT (ContractNumber, RID, VIN, SDate, STime,
SOdometer, EDate, ETime, EOdometer, LicenseNumber, LicenseState)
VALUES (23740958723453128953, 58705369, 675927590904095109, '2023-02-05', '14:30',
52087, '2023-02-10', '10:00', 59318, 'DL456', 'NY');
INSERT INTO RENTAL_AGREEMENT (ContractNumber, RID, VIN, SDate, STime,
SOdometer, EDate, ETime, EOdometer, LicenseNumber, LicenseState)
VALUES (54659783665120063319, 30941066, 927360234052356823, '2023-06-10', '10:00',
32798, '2023-06-12', '16:00', 33034, 'DL789', 'TX');
INSERT INTO RENTAL_AGREEMENT (ContractNumber, RID, VIN, SDate, STime,
SOdometer, EDate, ETime, EOdometer, LicenseNumber, LicenseState)
VALUES (84405840680120659817, 44028562, 567290330682340697, '2023-07-15', '09:00',
16864, '2023-07-29', '12:30', 19073, 'DL012', 'FL');
INSERT INTO RENTAL_AGREEMENT (ContractNumber, RID, VIN, SDate, STime,
SOdometer, EDate, ETime, EOdometer, LicenseNumber, LicenseState)
```

```
VALUES (75092134509172058434, 10525446, 723030946572304659, '2023-10-20', '08:00',
10156, '2023-10-25', '14:00', 11608, 'DL345', 'CA');

INSERT INTO CUSTOMER VALUES (3450473610, 'John Doe', '36 Poplar Dr');
INSERT INTO CUSTOMER VALUES (9135840680, 'Jane Smith', '12 Grove Pl');
INSERT INTO CUSTOMER VALUES (3068574016, 'Bob Johnson', '29 Parkshire Ln');
INSERT INTO CUSTOMER VALUES (2980572508, 'Alice Brown', '94 Summerset Ln');
INSERT INTO CUSTOMER VALUES (1051761345, 'Charlie Wilson', '72 Washington Rd');
INSERT INTO CUSTOMER VALUES (7345634798, 'John Doe', '36 Poplar Dr');

INSERT INTO CREDITCARD VALUES (7223340130693475, 'Visa', 12, 2024);
INSERT INTO CREDITCARD VALUES (2612569825903560, 'Mastercard', 10, 2023);
INSERT INTO CREDITCARD VALUES (4064014054106690, 'Amex', 3, 2025);
INSERT INTO CREDITCARD VALUES (2087602375902761, 'Discover', 9, 2023);
INSERT INTO CREDITCARD VALUES (4653206802345978, 'Visa', 8, 2024);

INSERT INTO MAKES VALUES (3450473610, 10531536, 68708064);
INSERT INTO MAKES VALUES (9135840680, 56869189, 58705369);
INSERT INTO MAKES VALUES (3068574016, 19750383, 30941066);
INSERT INTO MAKES VALUES (2980572508, 81430701, 44028562);
INSERT INTO MAKES VALUES (1051761345, 76324216, 10525446);
INSERT INTO MAKES VALUES (7345634798, 81430701, 23623458);

INSERT INTO AGREES VALUES (81209346809875301715, 3450473610,
7223340130693475);
INSERT INTO AGREES VALUES (23740958723453128953, 9135840680,
2612569825903560);
INSERT INTO AGREES VALUES (54659783665120063319, 3068574016,
4064014054106690);
INSERT INTO AGREES VALUES (84405840680120659817, 2980572508,
2087602375902761);
INSERT INTO AGREES VALUES (75092134509172058434, 1051761345,
4653206802345978);
/

// COMMIT
COMMIT;
/

/
```

```
// SELECT data
SELECT * FROM VEHICLE;
SELECT * FROM LOCATION;
SELECT * FROM CUSTOMER;
SELECT * FROM MODEL;
SELECT * FROM CLASS;
SELECT * FROM RESERVATION;
SELECT * FROM RENTAL_AGREEMENT;
SELECT * FROM CREDITCARD
SELECT * FROM MAKES;
SELECT * FROM AGREES;

/
// UPDATE data
UPDATE VEHICLE
SET Color = 'Gray'
WHERE Color = 'Black';

UPDATE LOCATION
SET LNUMBER = (LNUMBER + 1);

UPDATE CUSTOMER
SET CAddress = '99 Forest Rd'
WHERE CNAME = 'Jane Smith';

UPDATE MODEL
SET MYear = 2002
WHERE MNAME = 'Camry' OR MNAME = 'CR-V';

UPDATE CLASS
SET DailyRR = (DailyRR + 1.50);

UPDATE RESERVATION
SET REPRESENTATIVE = 'John Doe';

UPDATE RENTAL_AGREEMENT
SET SOdometer = (SOdometer + 1000);

UPDATE MAKES
SET LID = 2
```

```
WHERE CID = 1;

/
// DELETE data
DELETE FROM VEHICLE WHERE VIN = 321123;
DELETE FROM LOCATION WHERE LID = 4;
DELETE FROM CUSTOMER WHERE CADDRESS = '101 Elm St';
DELETE FROM MODEL WHERE MAKE = 'Toyota';
DELETE FROM CLASS WHERE WeeklyRR = 300;
DELETE FROM RESERVATION WHERE RID = 4;
DELETE FROM RENTAL_AGREEMENT WHERE ContractNumber = 105;
DELETE FROM MAKES WHERE CID = '1';

/
// DROP tables
DROP TABLE MAKES CASCADE CONSTRAINTS;
DROP TABLE AGREES CASCADE CONSTRAINTS;
DROP TABLE RENTAL_AGREEMENT CASCADE CONSTRAINTS;
DROP TABLE RESERVATION CASCADE CONSTRAINTS;
DROP TABLE CUSTOMER CASCADE CONSTRAINTS;
DROP TABLE CREDITCARD CASCADE CONSTRAINTS;
DROP TABLE CLASS CASCADE CONSTRAINTS;
DROP TABLE MODEL CASCADE CONSTRAINTS;
DROP TABLE VEHICLE CASCADE CONSTRAINTS;
DROP TABLE LOCATION CASCADE CONSTRAINTS;

/
```