David Arthasseril – dja35@njit.edu
Matthew Soto – mes3@njit.edu

CS 331 – 002 – Group 3

FINAL PROJECT

“RentACar”

Introduction

The objective of the RentACar database is to control all aspects of a car rental business's operations. This includes tracking car inventories, rental contracts, customer relations, and billing. We have devised an approach to creating a database that can fulfill these tasks.

The system's requirements and specifications:

➡ *Cars are assigned to locations and each location has one or more cars. Each car is uniquely identified by a vehicle ID number (VIN). The location to which the vehicle is assigned has an address and a location ID.*

➡ *A customer who wants to rent a car makes a reservation. The reservation is made for the pickup of a class of car at a location. The same customer may make more than one reservation.*

➡ *Normally, a reservation results in a rental agreement when the customer comes to the location to pick up the car. However, this is not always the case, since a reservation may be canceled or the customer with a reservation may not show up.*

➡ *A rental agreement is for a specific vehicle. At any point in time, a specific vehicle may have participated in zero, one, or more rental agreements.*

➡ *Car rental rates are determined by the class of the car. There are two rental rates for each class: daily and weekly.*

➡ *The car model includes a make (Ford, Honda, etc.), the year of the model, and the model name.*

➡ *The process of renting a car is as follows: Typically, a customer first makes a reservation with a location prior to arriving at the location to pick up the car. The CS331-RentACar service representative takes the customer's name and address, the class of a vehicle, and the period of rental (date and time in and out) that the customer desires.*

➡ *The customer is informed of the rental rate.*

➡ *When the customer arrives at the branch location to pick up the car, the service representative first checks for a reservation and, if a reservation exists, she draws up a rental agreement.*

➡ *At that time the service representative obtains other customer information, such as his operator's license number and the state that issued it, and the customer's credit card type and number, including the expiration month and year.*

➡ *If the customer has made a reservation, then the reservation information is used to assign a specific vehicle to the rental agreement. If the customer is a walk-in (no reservation), the service representative fills out the reservation information first as part of the process.*

➡ *All rentals must be associated with a reservation. The rental agreement has a contract number that uniquely identifies it, the VIN number of the vehicle that is being rented, the current date and time for the rental to start, and a current odometer reading.*

➡ *After use, the car is returned to the branch location. Information that will be filled in when the car is returned is the date and time at which the rental ends and the editing odometer reading. When the rental agreement is completed, the actual cost of the rental is computed using the class rental rate, and the cost is charged to the customer's credit card. No other form of payment is accepted.*

System Requirements

In order to properly implement this system, we first downloaded a DBMS software, or a software package/system that facilitates the creation and maintenance of a computerized database. We recommend using Oracle's *SQL Developer* to complete this task.

- Download: https://www.oracle.com/database/sqldeveloper/

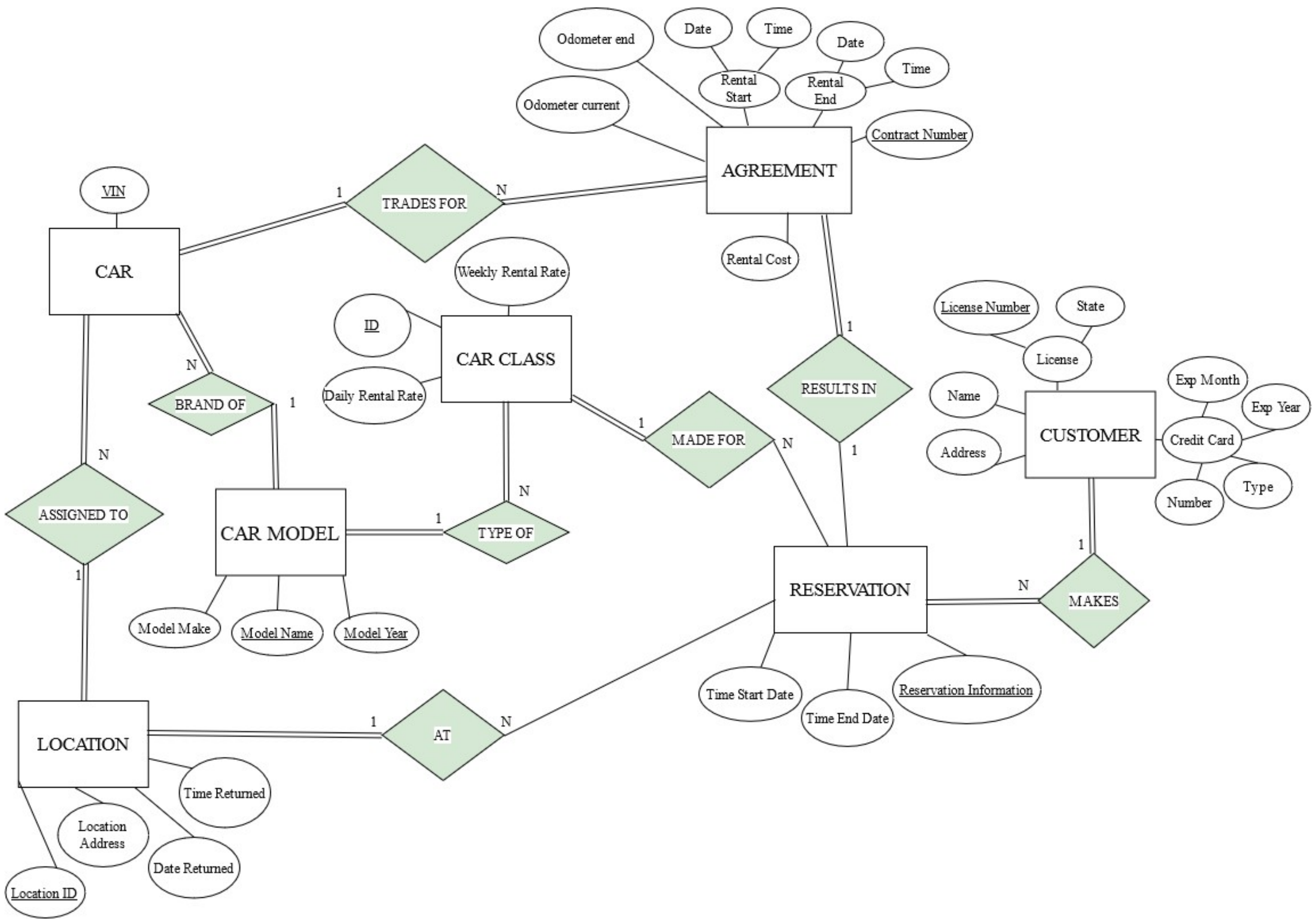
Once set up, we gathered the necessary information from the relevant parties according to the list of specifications detailed in the Introduction section.

We acquired information about the following entities:

- Customers
- Locations
- Reservations
- Agreements
- Cars
- Car Classes
- Car Models

After gathering the necessary information, we were able to proceed with creating an "entity-relationship diagram," or ER Diagram. An ER Diagram consists of all the aforementioned entities and how they relate and ultimately interact in a mini-world, or the localized space in which a database exists.

ER Diagram



(Figure 1. The ER Diagram for the RentACar database.)

The process of creating the ER diagram is as follows:

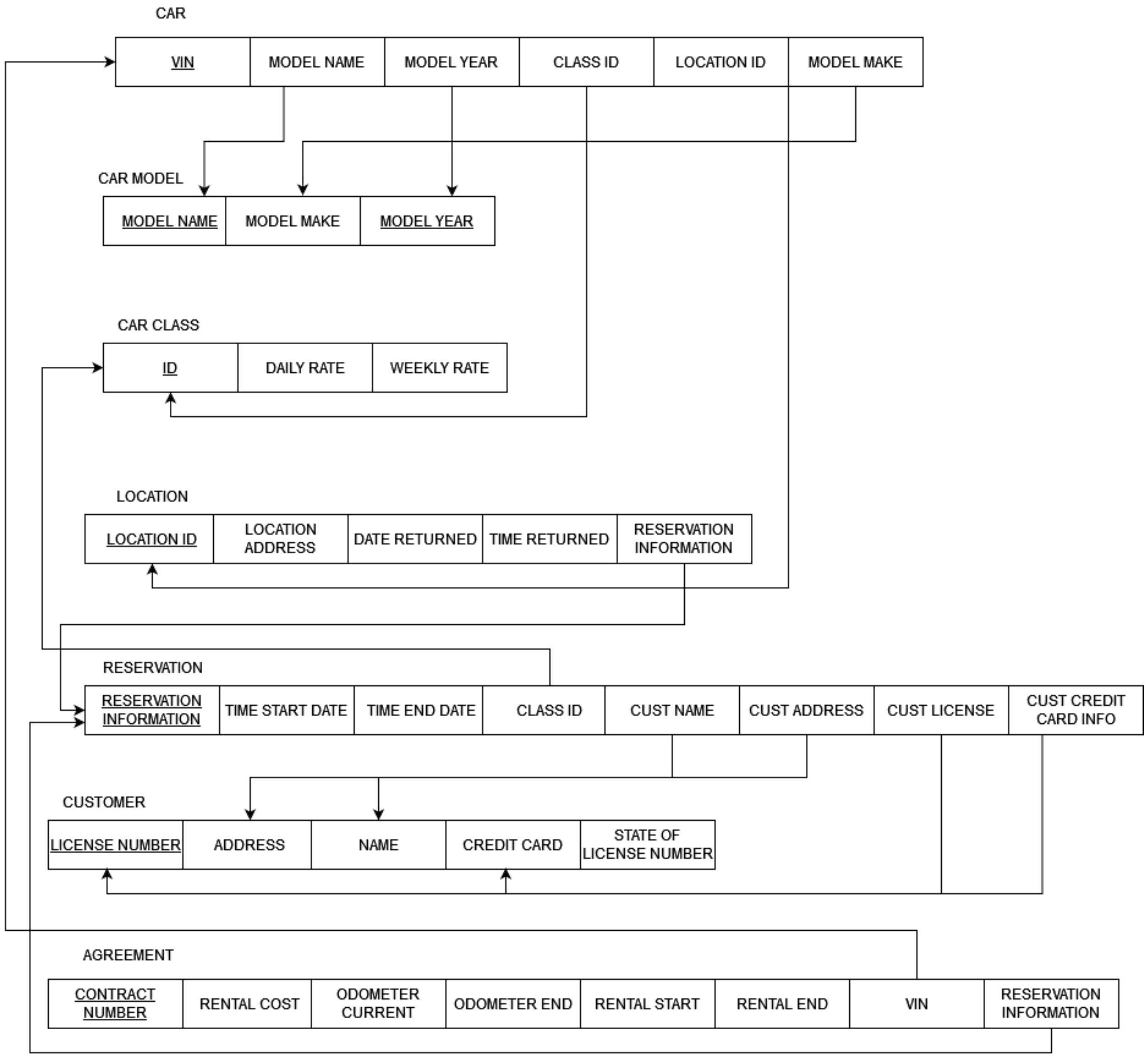
Each entity from the System Requirements section is placed on an empty page and is represented by a rectangle.

The associated attributes for each entity is connected to the respective entity via a line and is represented by an oval. The key attribute for each entity is the primary key/identifying attribute and thus has its descriptive text underlined to indicate this. Additionally, an attribute can either be simple, composite, or multi-valued. Simple attributes are primarily shown in the ER diagram and they possess only single values. Composite attributes have multiple sub-attributes that branch off of a main one and four are used in the ER diagram in *Figure 1*: Credit Card and License for CUSTOMER and Rental Start and Rental End for AGREEMENT. Multi-valued attributes are attributes that hold multiple values at once and are represented by double ovals; none of which are used here.

The relationships between each of the entities are created once attribute relations between entities have been identified and established. Each relationship is highlighted in green (to serve as a visual aid), connected by lines, and represented by a diamond. Each relationship has an associated pair of cardinality ratios and degrees of participation. Cardinality ratios indicate the maximum number of relationship instances that an entity can participate in. Degrees of participation indicate whether an entity's participation is deemed to be optional or mandatory.

Once the ER diagram has been created and finalized, the next step is to create a relational schema, or a set of tables that illustrates the relations between all the entities and their attributes. This can be done using the information given to us in the Introduction and the previously crafted ER diagram.

Relational Schema



(Figure 2. The relational schema diagram for the RentACar database.)

The relational schema above was created using our ER diagram as a reference point. This diagram exhibits how each of the entities and their attributes relate with each other. It brings us one step closer to implementing the database outline in a DBMS.

The relational schema is created using the following steps:

(1) For each regular (strong) entity type E in the ER schema, create a relation R that includes all the simple attributes of E.

(2) For each weak entity type W in the ER schema with owner E, create a relation R & include all simple attributes of W as attributes of R.

(3) Mapping Binary 1:1 Relation Types:

Identify the relations that correspond to the entity types participating in R.

1) Foreign Key (2 relations) option: Choose one of the relations, say S, and include a foreign key in S – the primary key of T. It is better to choose an entity type with total participation in the role of S.

2) Merged relation (1 relation) option: Merging 2 entity types and the relationship into a single relation. This may be appropriate when both participations are total.

3) Cross-reference or relationship relation (3 relations) option: Set up a third relation R for the purpose of cross-referencing the primary keys of the two relations S and T.

(4) Mapping Binary 1:N Relation Types:

For each regular 1:N relationship type R, identify the relation S that represents the N-side of the relation type.

Include, as a foreign key in S, the primary key of the relation T that represents the other entity.

Include any simple attributes of the 1:N relation as attributes of S.

(5) Mapping of Binary M:N Relation Types:

For each regular binary M:N relationship type R, create a new relation S to represent R. This is a relationship relation.

Include, as foreign key attributes in S, the primary keys of the relations that represent the participating entity types; their combination will form the primary key of S.

Also include any attributes of the M:N relation (or simple components of composite attributes) as attributes of S.

(6) Mapping of Multivalued Attributes:

For each multivalued attribute A, create a new relation R.

This relation will include an attribute corresponding to A, plus the primary key attribute K, as a foreign key in R, of the relation that represents the entity type of relationship type that has A as an attribute.

The primary key of R is the combination of A and K. If the multivalued attribute is composite, we include its simple components.

(7) Mapping of N-ary Relationship Types:

For each n-ary relationship type R, where n > 2, create a new relationship S to represent R.

Include, as foreign key attributes in S, the primary keys of the relations that represent the participating entity types.

Also include any simple attributes of the n-ary relationship type as attributes of S.

Following these steps, we were able to create a suitable relational schema to represent our ER diagram. What's unique about the relational schema is it shows how foreign keys in one table can relate to primary keys in another. This sort of unique diagramming is not used in the ER diagram; only in the relational schemas.

Design Decisions & Justifications

Some special decisions needed to be made along the way.

For example, in the ER diagram, there were issues with the CAR MODEL entity and its placement. However, after receiving counsel regarding our diagram format, we altered it. Initially, it was only connected to CAR, but we modified its relations and positioning in our diagram. Additionally, we made the following assumptions with our diagram:

- Rental Start/End are composite attributes because we chose to split each into their own date/time.
- Rental Cost is part of AGREEMENT because the cost to the credit card is calculated once the AGREEMENT finishes.
- CAR MODEL's "BRAND OF" constraint is mandatory because while you can have a car that's not officially branded, we're considering the most common instance.
- RESERVATION's outgoing constraints (minus the customer relationship) are optional because without a RESERVATION, all the other aspects can't be fulfilled.

For the relational schema, we made the following assumptions:

- **AGREEMENT**

  We added the "reservation information" value since the information for the reservation will result in an agreement, thus connecting the two.

- **LOCATION**

  It was a similar approach to AGREEMENT as the location gets assigned a car which is then traded in from an agreement and processed through a reservation. Thus, the reservation information is a necessary foreign key for the location.

- **CAR**

  We needed to establish a connection for the class and model of vehicle we were working with. There was also the fact that each car is assigned to a certain location, and so, we needed to add the location ID of the location where the car would be.

- **RESERVATION**

  Since reservations are made by the customer, we needed the customer's name, address, credit card info, and license number. We also included the type of car class that would be made with the reservation, and for that reason, we added the class ID of the class that was made from the reservation.

Intermediate Phase

Before heading into the following section about normalization, we've chosen to include every table creation statement, table alteration statement, and table insertion statement used up to this point. We have provided fictitious sample information for each insertion statement to assist in populating the tables. Below is every statement (in order of entry) to mimic our database:

Table Creation

```
CREATE TABLE CAR (

   VIN VARCHAR(17) NOT NULL,

   MODEL_NAME VARCHAR(50) NOT NULL,

   MODEL_YEAR INT NOT NULL,

   CLASS_ID INT NOT NULL,

   LOCATION_ID INT NOT NULL,
   PRIMARY KEY (VIN)
);
CREATE TABLE CAR_MODEL (
   MODEL_NAME VARCHAR(50) NOT NULL,
   MODEL_MAKE VARCHAR(50) NOT NULL,
   MODEL_YEAR INT NOT NULL,
   CONSTRAINT CAR_MODEL_NAME_YEAR PRIMARY KEY (MODEL_NAME, MODEL_YEAR)
);
CREATE TABLE CAR_CLASS (
   ID INT NOT NULL,
   DAILY_RATE DECIMAL(8,2) NOT NULL,
   WEEKLY_RATE DECIMAL(8,2) NOT NULL,
   PRIMARY KEY (ID)
);
CREATE TABLE LOCATION (
   LOCATION_ID INT NOT NULL,
   LOCATION_ADDRESS VARCHAR(100) NOT NULL,
   DATE_RETURNED DATE NOT NULL,
   TIME_RETURNED TIMESTAMP,
   RESERVATION_INFORMATION VARCHAR(50) NOT NULL,
   PRIMARY KEY (LOCATION_ID)
);
CREATE TABLE RESERVATION (
   RESERVATION_INFORMATION VARCHAR(50) NOT NULL,
   TIME_START_DATE TIMESTAMP NOT NULL,
   TIME_END_DATE TIMESTAMP NOT NULL,
   CLASS_ID INT NOT NULL,
   CUST_LICENSE_NUMBER VARCHAR(15) NOT NULL,
   PRIMARY KEY (RESERVATION_INFORMATION)
);
CREATE TABLE CUSTOMER (
   LICENSE_NUMBER VARCHAR(15) NOT NULL,
   ADDRESS VARCHAR(100) NOT NULL,
   NAME VARCHAR(50) NOT NULL,
   CREDIT_CARD VARCHAR(19) NOT NULL,
   STATE_OF_LICENSE_NUMBER VARCHAR(50),
   PRIMARY KEY (LICENSE_NUMBER)
);
CREATE TABLE AGREEMENT (
   CONTRACT_NUMBER INT NOT NULL,
   RENTAL_COST DECIMAL(6,2) NOT NULL,
   ODOMETER_CURRENT INT NOT NULL,
   ODOMETER_END INT,
   RENTAL_START TIMESTAMP NOT NULL,
   RENTAL_END TIMESTAMP,
   VIN VARCHAR(17) NOT NULL,
   RESERVATION_INFORMATION VARCHAR(50) NOT NULL,
   PRIMARY KEY (CONTRACT_NUMBER)
);
```

Table Alteration

```
ALTER TABLE CAR ADD CONSTRAINT FK_CAR_CLASS_ID_CAR FOREIGN KEY (CLASS_ID) REFERENCES
CAR_CLASS (ID) ON DELETE CASCADE;
ALTER TABLE CAR ADD CONSTRAINT FK_LOCATION_ID FOREIGN KEY (LOCATION_ID) REFERENCES
LOCATION (LOCATION_ID) ON DELETE CASCADE;
```

ALTER TABLE CAR ADD CONSTRAINT FK_CAR_MODEL FOREIGN KEY (MODEL_NAME, MODEL_YEAR) REFERENCES CAR_MODEL (MODEL_NAME, MODEL_YEAR) ON DELETE CASCADE;

ALTER TABLE LOCATION ADD CONSTRAINT FK_RESERVATION_INFO_LOC FOREIGN KEY (RESERVATION_INFORMATION) REFERENCES RESERVATION (RESERVATION_INFORMATION) ON DELETE CASCADE;

ALTER TABLE RESERVATION ADD CONSTRAINT FK_CAR_CLASS_ID_RES FOREIGN KEY (CLASS_ID) REFERENCES CAR_CLASS (ID) ON DELETE CASCADE;

ALTER TABLE RESERVATION ADD CONSTRAINT FK_CUSTOMER_LICENSE_NUM FOREIGN KEY (CUST_LICENSE_NUMBER) REFERENCES CUSTOMER (LICENSE_NUMBER) ON DELETE CASCADE;

ALTER TABLE AGREEMENT ADD CONSTRAINT FK_CAR_VIN FOREIGN KEY (VIN) REFERENCES CAR (VIN) ON DELETE CASCADE;

ALTER TABLE AGREEMENT ADD CONSTRAINT FK_RESERVATION_INFO_AGR FOREIGN KEY (RESERVATION_INFORMATION) REFERENCES RESERVATION (RESERVATION_INFORMATION) ON DELETE CASCADE;

Data Insertion - CAR_CLASS

INSERT INTO CAR_CLASS VALUES (1, 323.77, 857.56)

INSERT INTO CAR_CLASS VALUES (2, 235.93, 703.38)

INSERT INTO CAR_CLASS VALUES (3, 445.92, 1902.84)

INSERT INTO CAR_CLASS VALUES (4, 123.45, 678.90)

INSERT INTO CAR_CLASS VALUES (5, 848.33, 2304.87)

Data Insertion – CAR_MODEL

INSERT INTO CAR_MODEL VALUES (‘RAV4’, ‘Toyota’, 2004)

INSERT INTO CAR_MODEL VALUES (‘Highlander’, ‘Toyota’, 2018)

INSERT INTO CAR_MODEL VALUES (‘Focus’, ‘Ford’, 2007)

INSERT INTO CAR_MODEL VALUES (‘Q2’, ‘Audi’, 2015)

INSERT INTO CAR_MODEL VALUES (‘Ecosport’, ‘Ford’, 2017)

Data Insertion – CUSTOMER

```
INSERT INTO CUSTOMER VALUES ('VTCZHYARE7JWGB1', '87 Fake Place, Townville, 09813', 'John', '4571 5112 6623 9943', 'NJ')
INSERT INTO CUSTOMER VALUES ('T4U2QN5SWZZ6JR5', '43 Fake Avenue, Townplace, 09413', 'Amy', '4674 0687 1300 2560', 'NY')
INSERT INTO CUSTOMER VALUES ('IXQB71W0F2H44CD', '12 Fake Boulevard, Townscape, 39294', 'David', '4291 5849 2620 5207', 'MA')
INSERT INTO CUSTOMER VALUES ('73NU1RJIPMD2ZF6', '75 Fake Lane, Townarea, 61245', 'Sean', '4765 3028 0884 3889', 'MI')
INSERT INTO CUSTOMER VALUES ('TB6PJV6G0F3NZZ6', '97 Fake Spot, Townfun, 74126', 'Claire', '4132 4226 1942 1802', 'FL')
```

Data Insertion – RESERVATION

```
INSERT INTO RESERVATION VALUES ('R81HSHA9D8CYOI1', '24-MAY-23 12:35:20 AM', '25-MAY-23 12:45:05 AM', 2, 'T4U2QN5SWZZ6JR5')
INSERT INTO RESERVATION VALUES ('RBWIDO12OIO90HI', '25-MAY-23 09:12:00 PM', '26-MAY-23 09:22:05 PM', 3, 'IXQB71W0F2H44CD')
INSERT INTO RESERVATION VALUES ('RASDO902ANSNNS8', '26-MAY-23 11:10:05 PM', '27-MAY-23 11:20:05 PM', 1, 'VTCZHYARE7JWGB1')
INSERT INTO RESERVATION VALUES ('R01MAM90SDIO1SO', '27-MAY-23 10:13:27 PM', '28-MAY-23 10:23:05 PM', 4, '73NU1RJIPMD2ZF6')
INSERT INTO RESERVATION VALUES ('R099102IOASDIOH', '28-MAY-23 08:11:25 PM', '29-MAY-23 08:21:05 PM', 5, 'TB6PJV6G0F3NZZ6')
```

Data Insertion – LOCATION

```
INSERT INTO LOCATION VALUES (09123, '936 Addressplace, Townplace, NJ 09172', '25-MAY-2023', '25-MAY-2023 12:45:05 AM', 'RASDO902ANSNNS8')
```

INSERT INTO LOCATION VALUES (04572, '125 Addressave, Townave, NJ 09172', '26-MAY-2023', '26-MAY-2023 9:22:05 PM', 'R81HSHA9D8CYOI1')

INSERT INTO LOCATION VALUES (76625, '237 Addressblvd, Townblvd, NJ 09172', '27-MAY-2023', '27-MAY-2023 11:20:05 PM', 'RBWIDO12OIO90HI')

INSERT INTO LOCATION VALUES (89782, '612 Addresslane, Townlane, NJ 09172', '28-MAY-2023', '28-MAY-2023 10:23:05 PM', 'R01MAM90SDIO1SO')

INSERT INTO LOCATION VALUES (41480, '541 Addressspot, Townspot, NJ 09172', '29-MAY-2023', '29-MAY-2023 8:21:05 PM', 'R099102IOASDIOH')

Data Insertion – CAR

INSERT INTO CAR VALUES ('V01902KNASD09', 'RAV4', 2004, 1, 09123)

INSERT INTO CAR VALUES ('V12OAOSND81AS', 'Highlander', 2018, 2, 04572)

INSERT INTO CAR VALUES ('VP12UE90NUAS9', 'Focus', 2007, 3, 76625)

INSERT INTO CAR VALUES ('VADUMU0M91UAK', 'Q2', 2015, 4, 89782)

INSERT INTO CAR VALUES ('V0912SAM091MA', 'Ecosport', 2017, 5, 41480)

Data Insertion – AGREEMENT

INSERT INTO AGREEMENT VALUES(9139109394, 2395.37, 12378, NULL, '24-MAY-2023 12:35:20 AM', NULL, 'V01902KNASD09', 'RASDO902ANSNNS8')

INSERT INTO AGREEMENT VALUES(7723669579, 1812.79, 32451, NULL, '25-MAY-2023 09:12:00 PM', NULL, 'V12OAOSND81AS', 'R81HSHA9D8CYOI1')

INSERT INTO AGREEMENT VALUES(8123987609, 3889.01, 63334, NULL, '26-MAY-2023 11:10:05 PM', NULL, 'VP12UE90NUAS9', 'RBWIDO12OIO90HI')

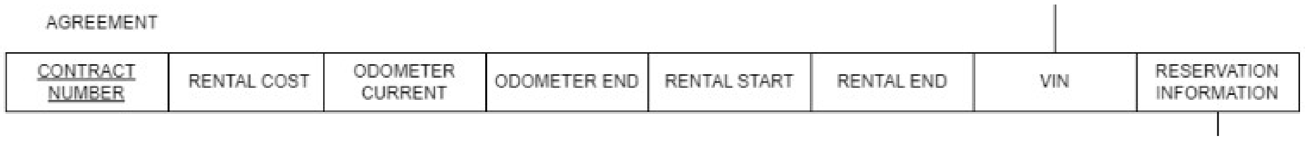
INSERT INTO AGREEMENT VALUES(0217378900, 4395.37, 23562, NULL, '27-MAY-2023 10:13:27 PM', NULL, 'VADUMU0M91UAK', 'R01MAM90SDIO1SO')

INSERT INTO AGREEMENT VALUES(0187576709, 2682.74, 73562, NULL, '28-MAY-2023 08:11:25 PM', NULL, 'V0912SAM091MA', 'R099102IOASDIOH')

Normalization

AGREEMENT

A)

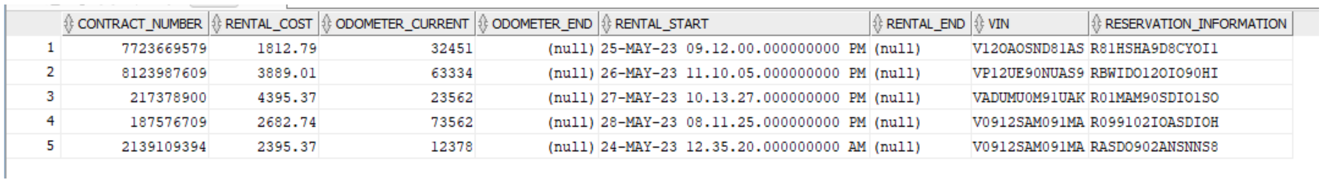


(Figure 3. Relational Schema – AGREEMENT Table)

Primary Key - Contract Number

Foreign Keys - VIN and RESERVATION INFORMATION.

B)



	CONTRACT_NUMBER	RENTAL_COST	ODOMETER_CURRENT	ODOMETER_END	RENTAL_START	RENTAL_END	VIN	RESERVATION_INFORMATION
1	7723669579	1812.79	32451	(null)	25-MAY-23 09.12.00.000000000 PM	(null)	V12OAOSND81AS	R81HSHA9D8CYOI1
2	8123987609	3889.01	63334	(null)	26-MAY-23 11.10.05.000000000 PM	(null)	VP12UE90NUAS9	RBWIDO12OIO90HI
3	217378900	4395.37	23562	(null)	27-MAY-23 10.13.27.000000000 PM	(null)	VADUMU0M91UAK	R01MAM90SDIO1SO
4	187576709	2682.74	73562	(null)	28-MAY-23 08.11.25.000000000 PM	(null)	V0912SAM091MA	R099102IOASDIOH
5	2139109394	2395.37	12378	(null)	24-MAY-23 12.35.20.000000000 AM	(null)	V0912SAM091MA	RASDO902ANSNNS8

(Figure 4. Database – AGREEMENT + Data)

C)

Contract Number -> RENTAL COST, ODOMETER END, ODOMETER END, RENTAL START, VIN, RESERVATION INFORMATION, RESERVATION INFORMATION-> RENTAL COST, RENTAL START, RENTAL END.

D)

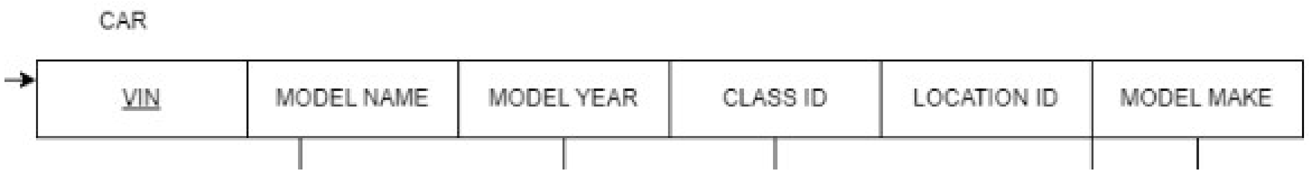
1NF = Yes, because there are no multi-values, composite attributes, or any nested relations.

2NF = Yes, because all the keys are fully dependent on the primary key.

3NF = Yes, since there is no transitivity on the table and changing a non-key value will not affect the value of other values.

CAR

A)

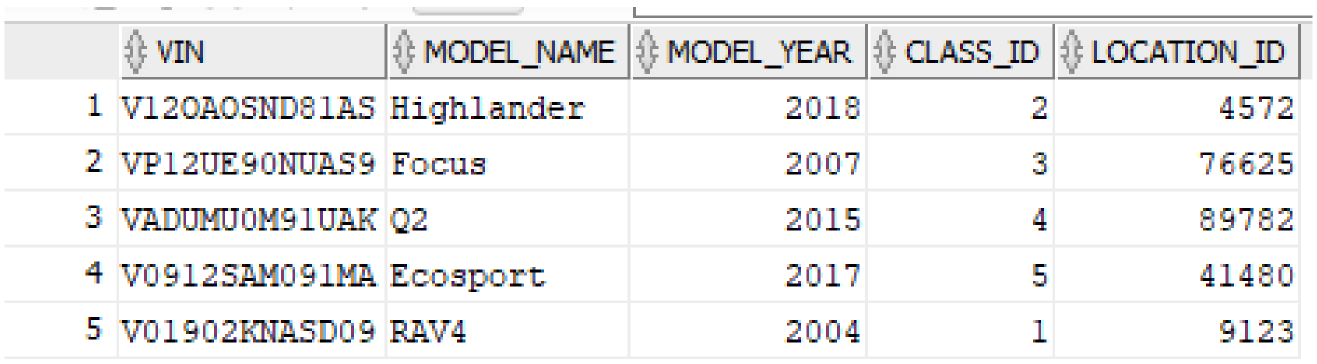


(Figure 5. Relational Schema – CAR Table)

Primary Key - VIN

Foreign Keys – CLASS ID, MODEL MAKE, MODEL NAME, MODEL YEAR, LOCATION ID

B)



	VIN	MODEL_NAME	MODEL_YEAR	CLASS_ID	LOCATION_ID
1	V12OAOSND81AS	Highlander	2018	2	4572
2	VP12UE90NUAS9	Focus	2007	3	76625
3	VADUMU0M91UAK	Q2	2015	4	89782
4	V0912SAM091MA	Ecosport	2017	5	41480
5	V01902KNASD09	RAV4	2004	1	9123

(Figure 6. Database – CAR + Data)

C) F1 - VIN -> MODEL_MAKE, MODEL_YEAR, MODEL_NAME, CLASS_ID, LOCATION_ID, MODEL_MAKE

F2 - MODEL_NAME, MODEL_YEAR -> MODEL_MAKE

D)

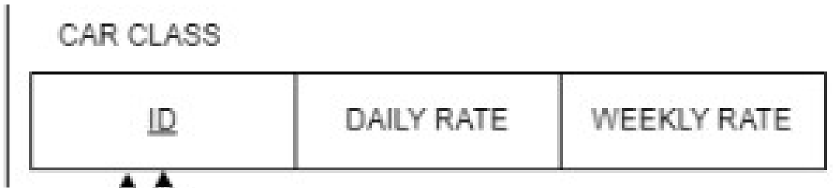
1NF – Yes, all cells hold one value, there is a primary key, no duplicated rows or columns, and each column has one value for each row in the table.

2NF – Yes, since it is both in 1NF and it has no partial dependencies since all non-key values depend on the primary key.

3NF – Yes, since it is both in 2NF and even though there is a transitive dependency in F2 as stated, it still doesn't violate the 3NF rule since everything in F2 is still in relation and depend on the primary key as seen in F1 and so changing a non-key value will not cause another value in its row to change.

CAR_CLASS

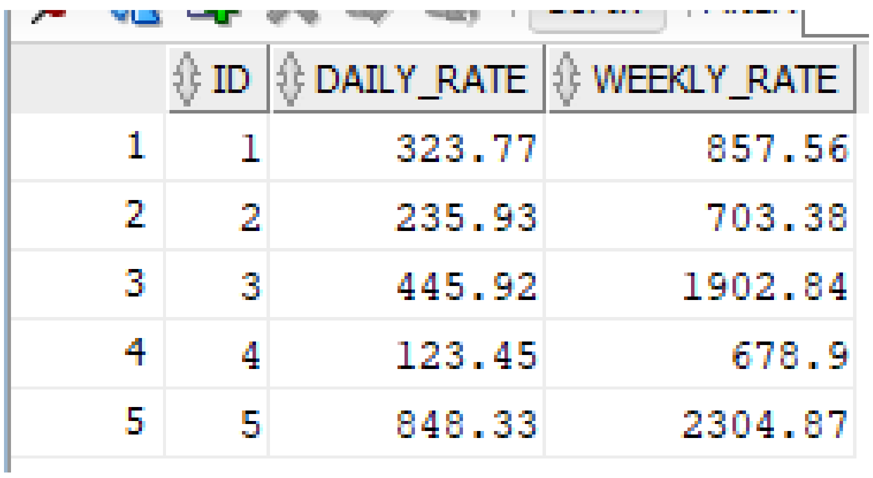
A)



(Figure 7. Relational Schema – CAR CLASS Table)

Primary Key - ID

B)



	ID	DAILY_RATE	WEEKLY_RATE
1	1	323.77	857.56
2	2	235.93	703.38
3	3	445.92	1902.84
4	4	123.45	678.9
5	5	848.33	2304.87

(Figure 8. Database – CAR_CLASS + Data)

C) FD1 - ID -> DAILY RATE, WEEKLY RATE

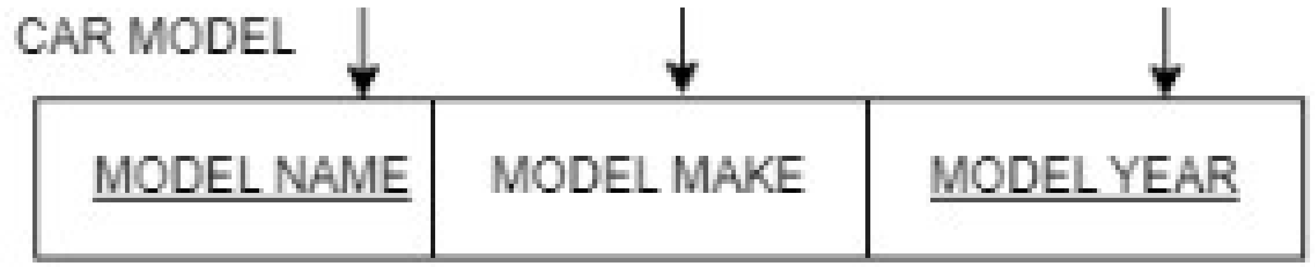
D) 1NF – Yes, since all cells hold one value, there is a primary key, no duplicated row or columns, and each column has one value for each row in the table.

2NF – Yes, since it is in 1NF and there are no partial dependencies since they all rely on the ID key

3NF – Yes, since there is a 2NF and there are no transitive properties.

CAR_MODEL

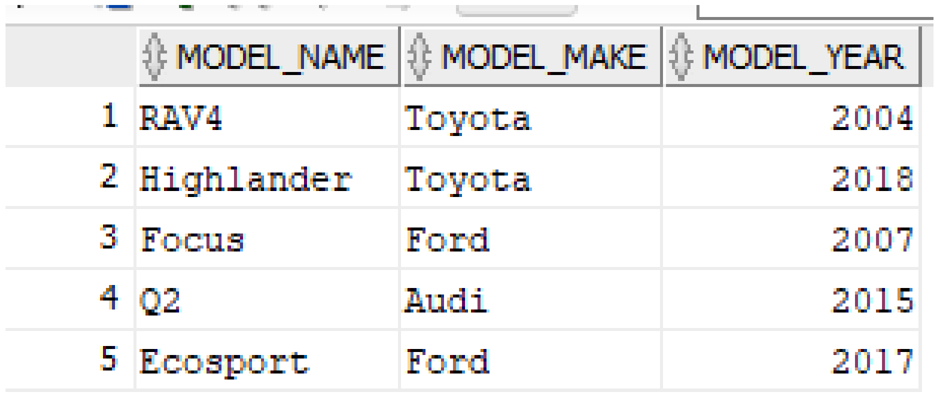
A)



(Figure 9. Relational Schema – CAR MODEL Table)

Primary Key - MODEL NAME, MODEL YEAR

B)



	MODEL_NAME	MODEL_MAKE	MODEL_YEAR
1	RAV4	Toyota	2004
2	Highlander	Toyota	2018
3	Focus	Ford	2007
4	Q2	Audi	2015
5	Ecosport	Ford	2017

(Figure 10. Database – CAR_MODEL + Data)

C)

F1 - MODEL NAME, MODEL YEAR -> MODEL MAKE

D)

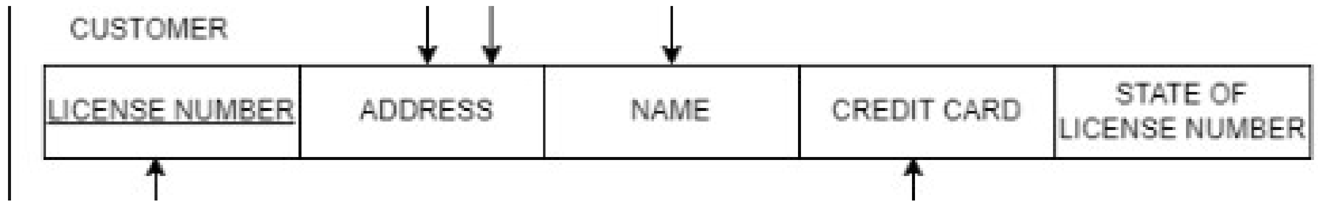
1NF – Yes, since it has a primary key, no single attribute has multiple values, and no columns contain duplicate values.

2NF – Yes, since it is in 1NF and there are no partial dependencies since they all depend on our table's primary keys.

3NF – Yes, since it is in 2NF and there are no transitive dependencies that violate the 3NF rule.

CUSTOMER

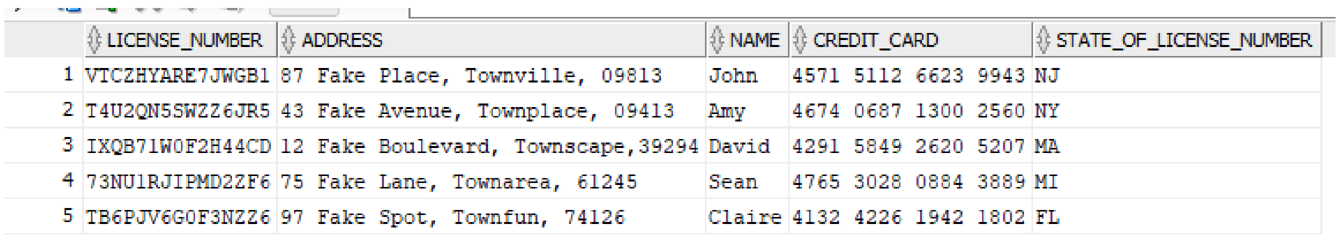
A)



(Figure 11. Relational Schema – CUSTOMER Table)

Primary Key - LICENSE NUMBER

B)



	LICENSE_NUMBER	ADDRESS	NAME	CREDIT_CARD	STATE_OF_LICENSE_NUMBER
1	VTCZHYARE7JWGB1	87 Fake Place, Townville, 09813	John	4571 5112 6623 9943	NJ
2	T4U2QN5SWZZ6JR5	43 Fake Avenue, Townplace, 09413	Amy	4674 0687 1300 2560	NY
3	IXQB71W0F2H44CD	12 Fake Boulevard, Townscape,39294	David	4291 5849 2620 5207	MA
4	73NU1RJIPMD2ZF6	75 Fake Lane, Townarea, 61245	Sean	4765 3028 0884 3889	MI
5	TB6PJV6G0F3NZZ6	97 Fake Spot, Townfun, 74126	Claire	4132 4226 1942 1802	FL

(Figure 12. Database – CUSTOMER + Data)

C)

F1 - LICENSE NUMBER -> ADDRESS, NAME, CREDIT CARD, STATE OF LICENSE NUMBER

D)

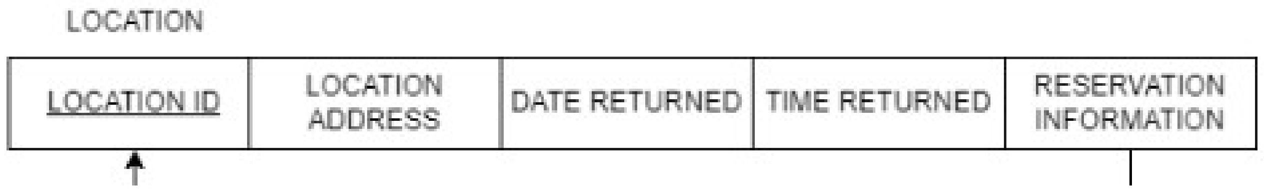
1NF - Yes, since it has a primary key, no single attribute has multiple values, and no columns contain duplicate values.

2NF – Yes, since it is in 1NF and there are no partial dependencies since they all depend on our table's primary keys

3NF – Yes, since it is in 2NF and there are no transitive dependencies from any non-key column that may change the data.

LOCATION

A)

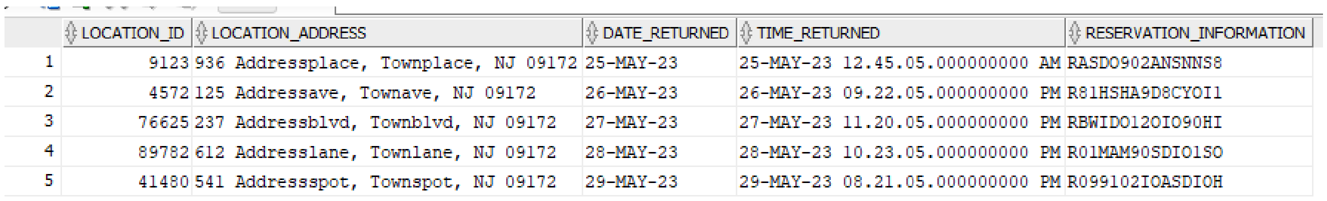


(Figure 13. Relational Schema – LOCATION Table)

Primary Key - LOCATION ID

Foreign Key - RESERVATION INFORMATION

B)



	LOCATION_ID	LOCATION_ADDRESS	DATE_RETURNED	TIME_RETURNED	RESERVATION_INFORMATION
1	9123	936 Addressplace, Townplace, NJ 09172	25-MAY-23	25-MAY-23 12.45.05.000000000 AM	RASDO902ANSNNS8
2	4572	125 Addressave, Townave, NJ 09172	26-MAY-23	26-MAY-23 09.22.05.000000000 PM	R81HSHA9D8CYOI1
3	76625	237 Addressblvd, Townblvd, NJ 09172	27-MAY-23	27-MAY-23 11.20.05.000000000 PM	RBWIDO12OIO90HI
4	89782	612 Addresslane, Townlane, NJ 09172	28-MAY-23	28-MAY-23 10.23.05.000000000 PM	R01MAM90SDIO1SO
5	41480	541 Addressspot, Townspot, NJ 09172	29-MAY-23	29-MAY-23 08.21.05.000000000 PM	R099102IOASDIOH

(Figure 14. Database – LOCATION + Data)

C)

FD1 - LOCATION ID -> LOCATION ADDRESS, DATE RETURNED, TIME RETURNED, RESERVATION INFORMATION

D)

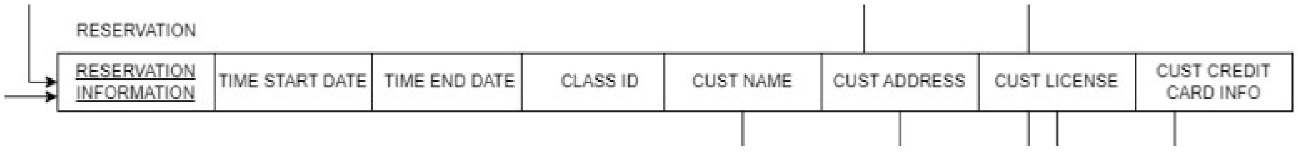
1NF – Yes, since it has a primary key, no single attribute has multiple values, and no columns contain duplicate values.

2NF – Yes, since it is in 1NF and there are no partial dependencies since they all depend on our table's primary keys.

3NF – Yes, since it is in 2NF and there are no transitive dependencies.

RESERVATION

A)

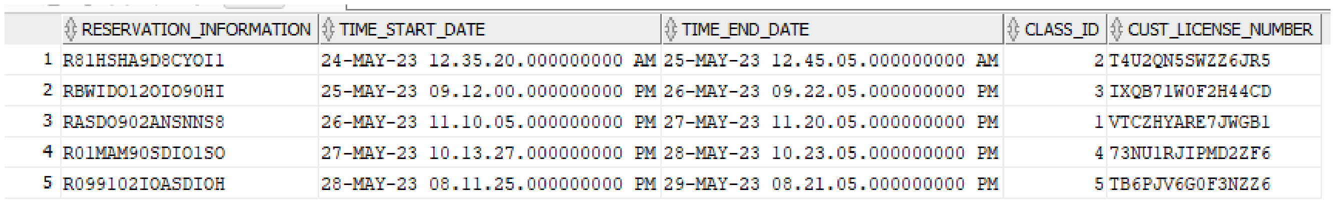


(Figure 15. Relational Schema – RESERVATION Table)

Primary Key - RESERVATION INFORMATION

Foreign Keys - CUST NAME, CUST ADDRESS, CUST CREDIT CARD INFO, CLASS ID

B)



	RESERVATION_INFORMATION	TIME_START_DATE	TIME_END_DATE	CLASS_ID	CUST_LICENSE_NUMBER
1	R81HSHA9D8CYOI1	24-MAY-23 12.35.20.000000000 AM	25-MAY-23 12.45.05.000000000 AM	2	T4U2QN5SWZZ6JR5
2	RBWIDO12OI090HI	25-MAY-23 09.12.00.000000000 PM	26-MAY-23 09.22.05.000000000 PM	3	IXQB71W0F2H44CD
3	RASDO902ANSNNS8	26-MAY-23 11.10.05.000000000 PM	27-MAY-23 11.20.05.000000000 PM	1	VTCZHYARE7JWGB1
4	R01MAM90SDIO1SO	27-MAY-23 10.13.27.000000000 PM	28-MAY-23 10.23.05.000000000 PM	4	73NU1RJIPMD2ZF6
5	R099102IOASDIOH	28-MAY-23 08.11.25.000000000 PM	29-MAY-23 08.21.05.000000000 PM	5	TB6PJV6G0F3NZZ6

(Figure 16. Database – RESERVATION + Data)

C)

F1 - RESERVATION INFORMATION -> TIME START DATE, TIME END DATE, CLASS ID, CUST NAME, CUST ADDRESS, CUST LICENSE, CUST CREDIT CARD INFO

F2 - CUST LICENSE -> CUST CREDIT CARD INFO, CUST NAME, CUST ADDRESS

D)

1NF – Yes, since it has a primary key, no single attribute has multiple values, and no columns contain duplicate values.

2NF – Yes, since it is in 1NF and there are no partial dependencies since they all depend on our table's primary keys.

3NF – There is a transitive relation as seen in F2. However, CUST LICENSE is still a primary key in itself and so, it does not violate the 3NF rule and is still in 3NF form.

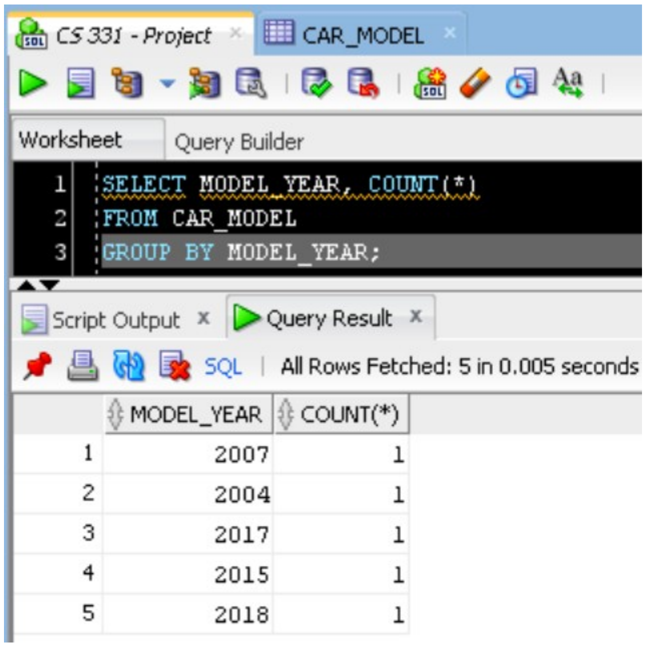
Queries & SQL

Now that we've thoroughly analyzed the normalization of our tables, we proceed to use SQL to manipulate and retrieve data from them. Here, we're using retrieval queries to retrieve data from our tables specifically using the following keywords:

- GROUP BY
- GROUP BY and HAVING
- ALL {nested query}
- IN {nested query}

GROUP BY:

Select the model years from CAR_MODEL and group the table by the year.



```
SELECT MODEL_YEAR, COUNT(*)
FROM CAR_MODEL
GROUP BY MODEL_YEAR;
```

All Rows Fetched: 5 in 0.005 seconds

	MODEL_YEAR	COUNT(*)
1	2007	1
2	2004	1
3	2017	1
4	2015	1
5	2018	1

(Figure 17. Results of using GROUP BY on CAR_MODEL.)

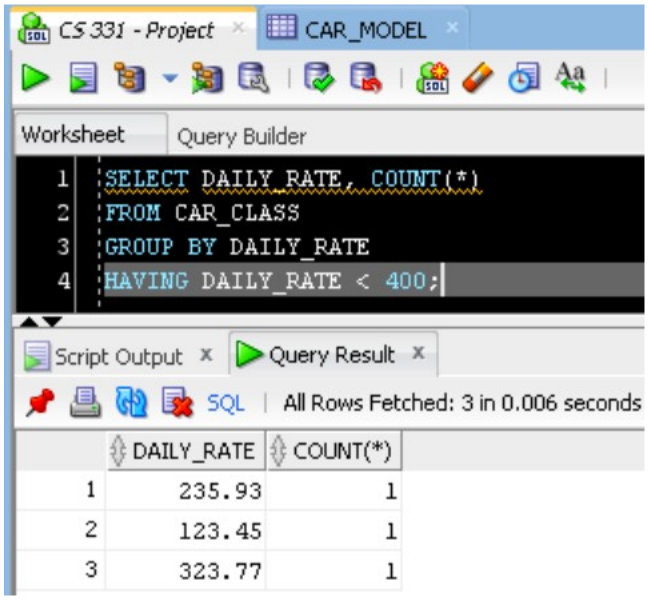
SELECT MODEL_YEAR, COUNT(*)

FROM CAR_MODEL

GROUP BY MODEL_YEAR;

GROUP BY & HAVING:

Select the daily rates from CAR_CLASS and group them by these daily rates, but only display the rates that are less than 400.



```
SELECT DAILY_RATE, COUNT(*)
FROM CAR_CLASS
GROUP BY DAILY_RATE
HAVING DAILY_RATE < 400;
```

All Rows Fetched: 3 in 0.006 seconds

	DAILY_RATE	COUNT(*)
1	235.93	1
2	123.45	1
3	323.77	1

(Figure 18. Results of using GROUP BY and HAVING on CAR_MODEL.)

SELECT DAILY_RATE, COUNT(*)

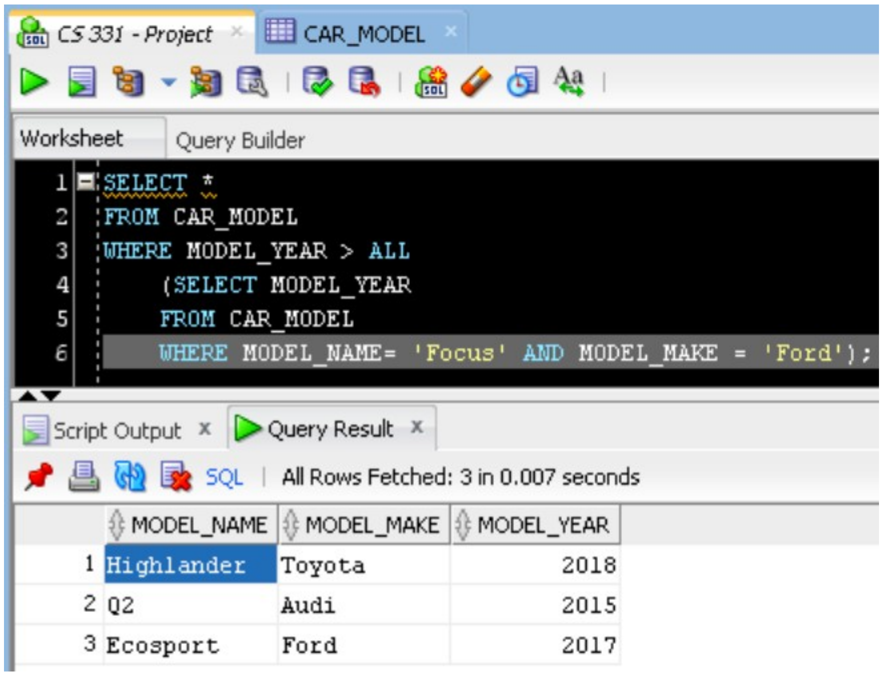
FROM CAR_CLASS

GROUP BY DAILY_RATE

HAVING DAILY_RATE < 400;

ALL:

Select from CAR_MODEL cars that are <u>*younger*</u> *than the Ford Focus from the year 2007 (year number greater than 2007).*



```
SELECT *
FROM CAR_MODEL
WHERE MODEL_YEAR > ALL
    (SELECT MODEL_YEAR
    FROM CAR_MODEL
    WHERE MODEL_NAME= 'Focus' AND MODEL_MAKE = 'Ford');
```

All Rows Fetched: 3 in 0.007 seconds

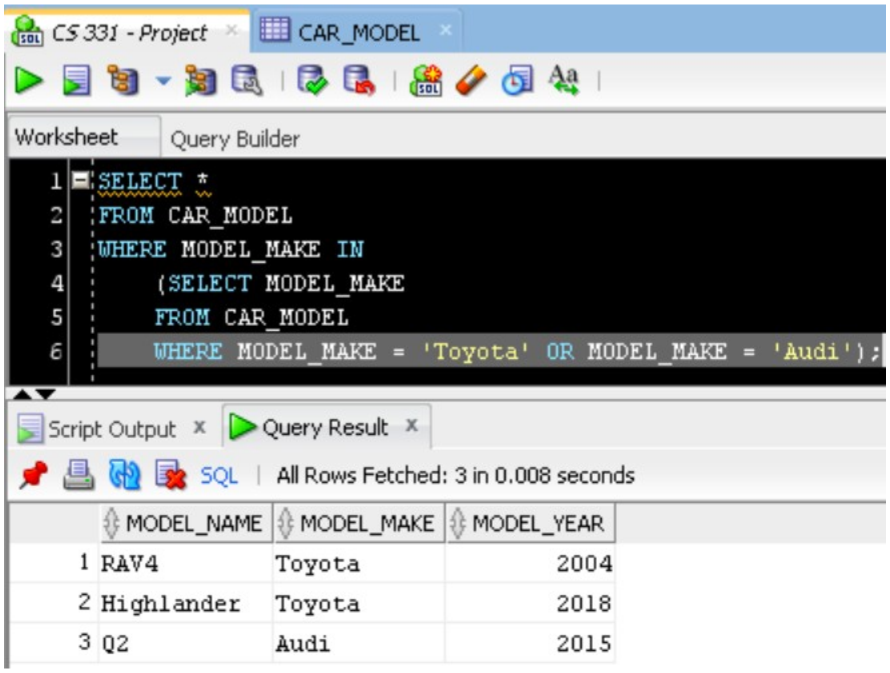
	MODEL_NAME	MODEL_MAKE	MODEL_YEAR
1	Highlander	Toyota	2018
2	Q2	Audi	2015
3	Ecosport	Ford	2017

(Figure 19. Results of using ALL on CAR_MODEL.)

```
SELECT *
FROM CAR_MODEL
WHERE MODEL_YEAR > ALL
        (SELECT MODEL_YEAR
        FROM CAR_MODEL
        WHERE MODEL_NAME = ’Focus’ AND MODEL_MAKE = ’Ford’);
```

IN:

Select from CAR_MODEL all the models that are made by Toyota or Audi.



```
SELECT *
FROM CAR_MODEL
WHERE MODEL_MAKE IN
    (SELECT MODEL_MAKE
    FROM CAR_MODEL
    WHERE MODEL_MAKE = 'Toyota' OR MODEL_MAKE = 'Audi');
```

All Rows Fetched: 3 in 0.008 seconds

	MODEL_NAME	MODEL_MAKE	MODEL_YEAR
1	RAV4	Toyota	2004
2	Highlander	Toyota	2018
3	Q2	Audi	2015

(Figure 20. Results of using IN on CAR_MODEL.)

```
SELECT *

FROM CAR_MODEL

WHERE MODEL_MAKE IN

        (SELECT MODEL_MAKE

        FROM CAR_MODEL

        WHERE MODEL_MAKE = 'Toyota' OR MODEL_MAKE = 'Audi');
```

Conclusion

The overall experience of creating this database was enlightening and transformative.

It was fascinating witnessing theoretical diagrams and schemas come to life in the form of actual databases on one’s computer. Furthermore, it’s impressive that one can run an entire database system from their own personal computer to begin with.

The process of creating a database prototype from scratch showed how much care goes into one. We learned proper practices for SQL and database management – including how data is properly organized, stored, and managed.

Additionally, we developed our teamwork skills. We were able to efficiently collaborate by collectively pooling any created documents and imagery into our private messaging service. We kept track of anything important, sent each other necessary files for progression, and made sure to have the most updated content at our disposal for each forthcoming phase of the project.

The hardest part of the project was definitely making the database. We ran into issues with insertion anomalies because it meant that we had to insert data for our tables in a particular order lest they give us errors in the console.

The easiest part of the project, however, was the ER diagram. Following the step-by-step process detailed in the ER Diagram section, we were able to assemble each piece of the puzzle separately and simply combine them all at the end. We had some trouble understanding the instructions (and we believe they should have been written more clearly, concisely, and in bullet point format), but otherwise, it was easy and even fun making the ER diagram.

If we had to do it all over again, we would’ve checked over our schema more. There were some slight errors that we looked over which have been fixed for this submission.