DATABASE MANAGEMENT SYSTEM

REPORT

ON

“SMARTPARK”

****

BY:

DHIRAJ LULLA

RONAK LAUNGANI

SANKALP REDGAONKAR

RADHIKA GOSWAMI

AISHWARYA CHODAPANEEDI

MAYURI KHADE

PROFESSOR: Prof. Namchul Shin

**Project Overview:**

SmartPark is a new venture initiated by the Smart team, which focuses on a very common problem that New Yorkers suffer from – parking. It realizes that many car owners in NYC either spend a fortune for a parking lot or deal with moving their car most of the time. The Smart team thinks that many New Yorkers will sign up for this service and turn out high ROI. This venture will be eventually carried forward to other metropolitan cities upon successful execution of its first phase in NYC.

**Business Objectives:-**

To allow the customers to store all their information like email, mailing address and their car model in their account;

To allow the customers to search for a nearby available parking spot location;

Easy to use application that is user friendly;

To allow the customers to setup their payment online;

To allow customers to discuss parking spots;

To allow customers to be notified with the parking spot regulations.

**Application Working: -**

When New Yorkers are looking for a parking spot, they can request the application to find one for them. If there is a spot available, the application will return an exact location for the spot (which street/avenue, which side of the street, how far down the block).  If there isn't one available, they can request that the attempt be tried every 2 minutes to see if a spot has become available. If they move or if they get a new car, they will need to update their profile.

**Normalization:**

* **Definition:**

Normalization is the process of reorganizing data in a database so that it meets two basic requirements: (1) There is no redundancy of data, and (2) data dependencies are logical. Normalization is important for many reasons, but chiefly because it allows databases to take up as little disk space as possible, resulting in increased performance.

* **Advantages:**

1. **Increased consistency**: Information is stored in one place and one place only, reducing the possibility of inconsistent data.
2. **Easier object-to-data mapping**: Highly-normalized data schemas in general are closer conceptually to object-oriented schemas because the object-oriented goals of promoting high cohesion and loose coupling between classes results in similar solutions (at least from a data point of view).

|  |  |
| --- | --- |
| LEVEL | RULE |
| First Normal Form(1NF) | An entity type is in 1NF when it contains no repeating groups of data. |
| Second Normal Form(2NF) | An entity type is in 2NF when it is in 1NF and when all of its non-key attributes are fully dependent on its [primary key](http://www.agiledata.org/essays/keys.html). |
| Third Normal Form(3NF) | An entity type is in 3NF when it is in 2NF and when all its attributes are directly dependent on the [primary key](http://www.agiledata.org/essays/keys.html). |
| BCNF | Data in table should be in the third normal form and for any dependency A-> B, A should be super key. |

**ENTITY RELATIONSHIP MODEL**

To following entities and attributes are required:

**Users** – All the details associated with the user are stored in this table. For example name, address, password, phone number, email, etc.

**Car** – All the cars that are parked using the SmartPark app are stored here. These are associated with the license number, the model and the user who owns it.

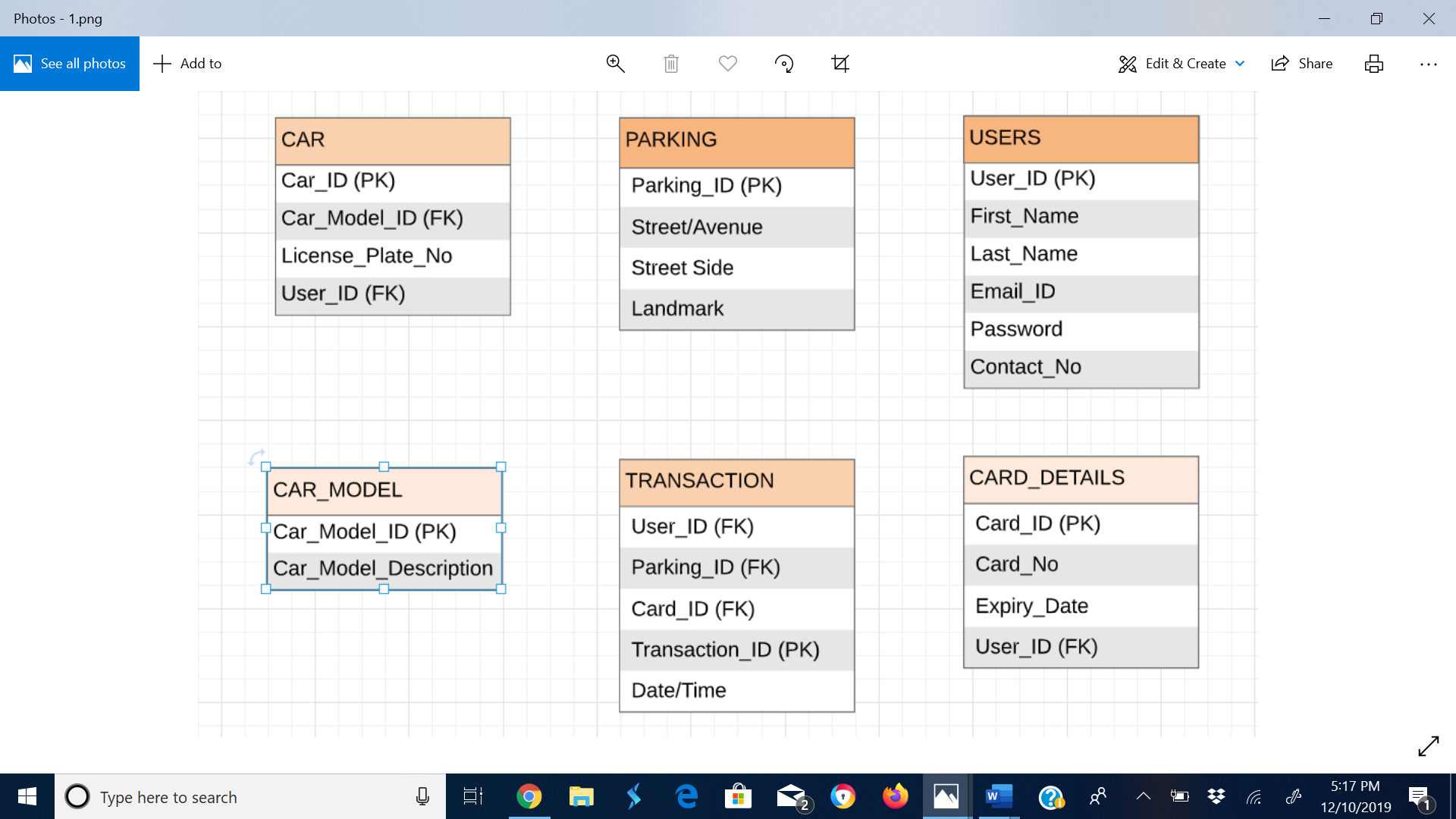
**Car\_Model** – In the world of data, it will be better to have a separate table for the various models of cars that have been parked using the app. It will save storage space, reduce redundant data and avoid repetition of data.

**Parking** – All the information associated with a parking spot, like the location/address is stored in this table with a unique identifier.

**Transaction** – To keep a track of all the payments or spot booking made using the SmartPark app at one place.

**Card (Details)** – To suggest the user a way to save card details for quick checkout during next booking.

Here are the entities and associated attributes:



**Entity Relationships**

After the identification of the various entity types and attributes, the next step is to determine how each of the entity types are related to each other. The following types of relationships are included in the ER model:

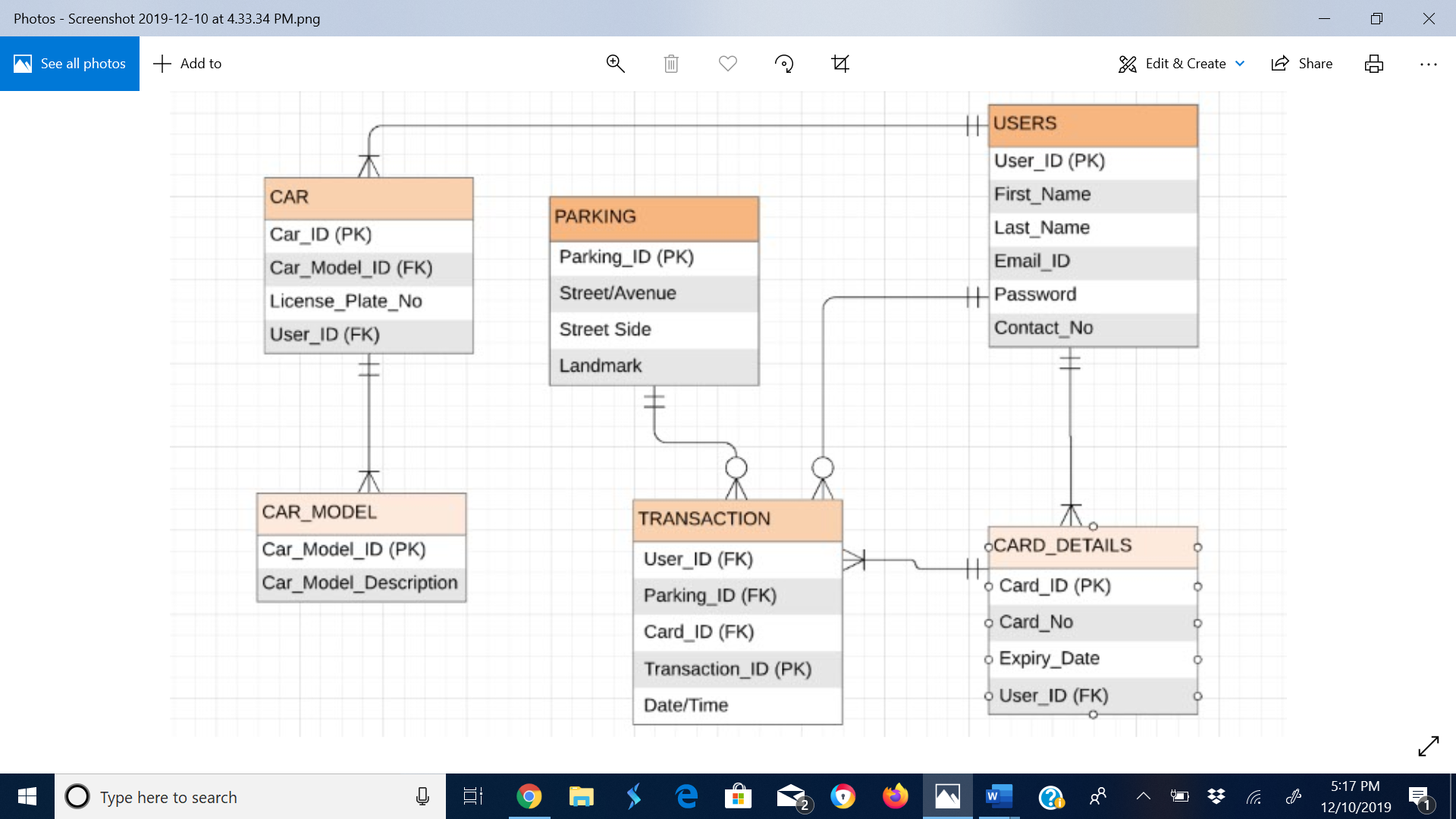
1:1 (One to One relationship)

1: N (One to Many relationship)

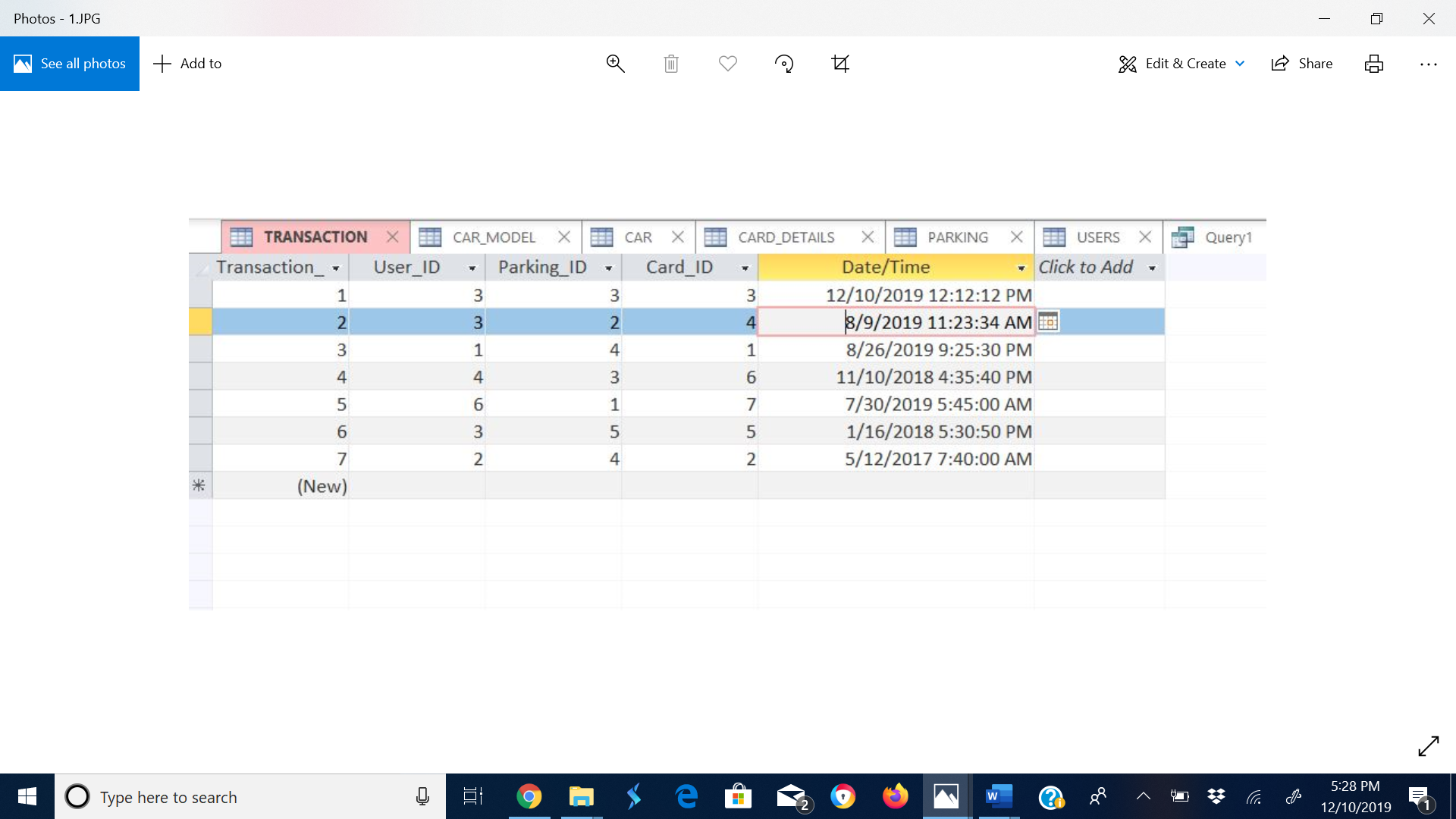
N:M (Many to Many relationship)

Here are the Identifiable Relations between the entities. Every booking made by the USER is a TRANSACTION made in the system, stored in a separate table using the FKs. Hence every relation is in association with the TRANSACTION table.

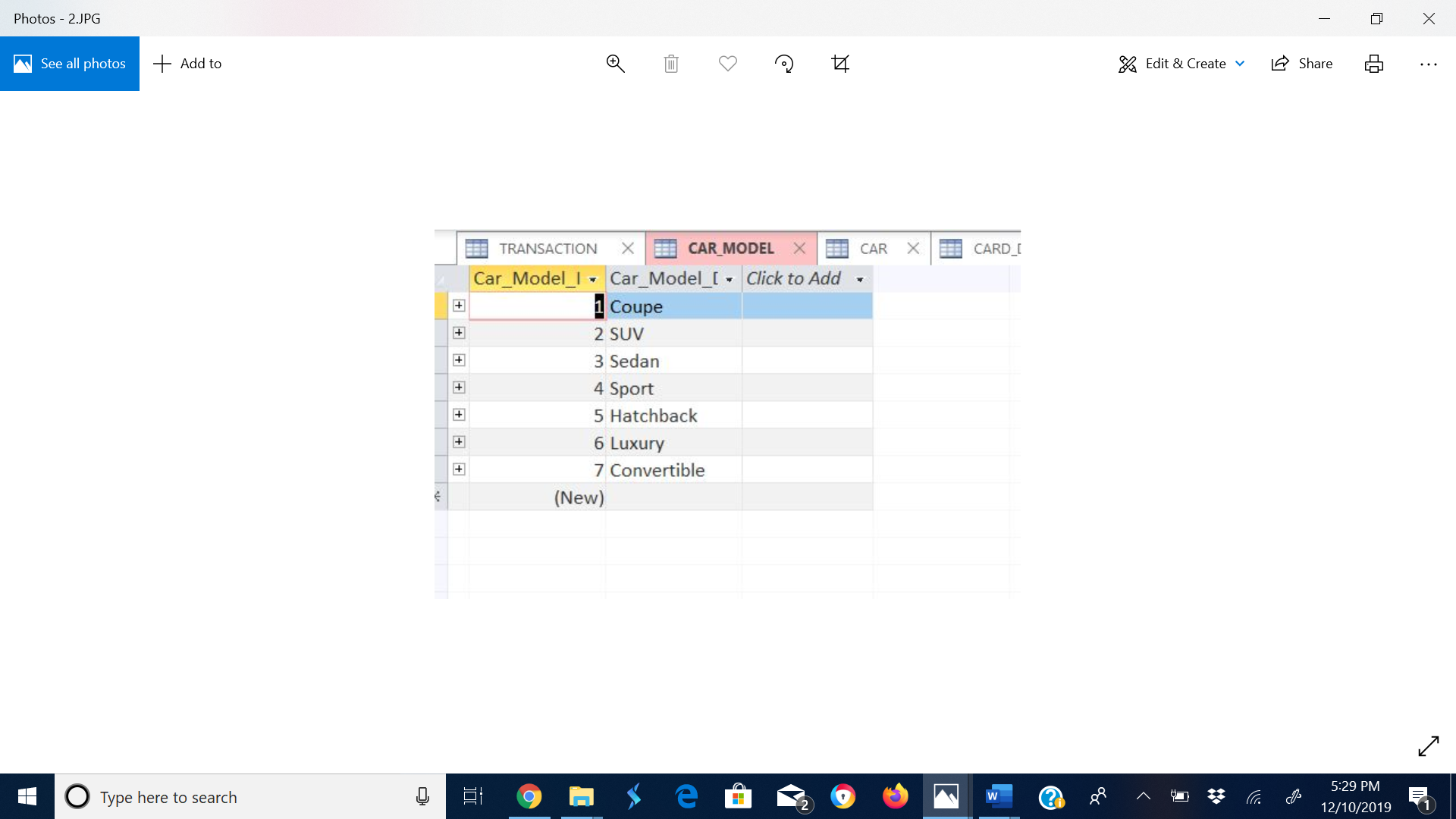
1. A USER can have **0 or many** TRANSACTIONS. But a TRANSACTION can have **one and only one** USER.
2. A TRANSACTION can be associated with **one and only one** PARKING spot. But a PARKING spot can be associated with **0 or many** TRANSACTIONS.
3. A TRANSACTION can use **one and only one** CARD. But a CARD can be used for **one or many** TRANSACTIONS
4. A USER can have **one or many** CARDs. But a CARD will have **one and only one** USER as an owner.
5. A CAR can have **one or many** CAR MODELS. But a CAR MODEL will be associated with be only associated with just **one and only one** type of CAR.
6. A USER can own **one or many** CARS. But a CAR can be owned by **one and only one** USER.

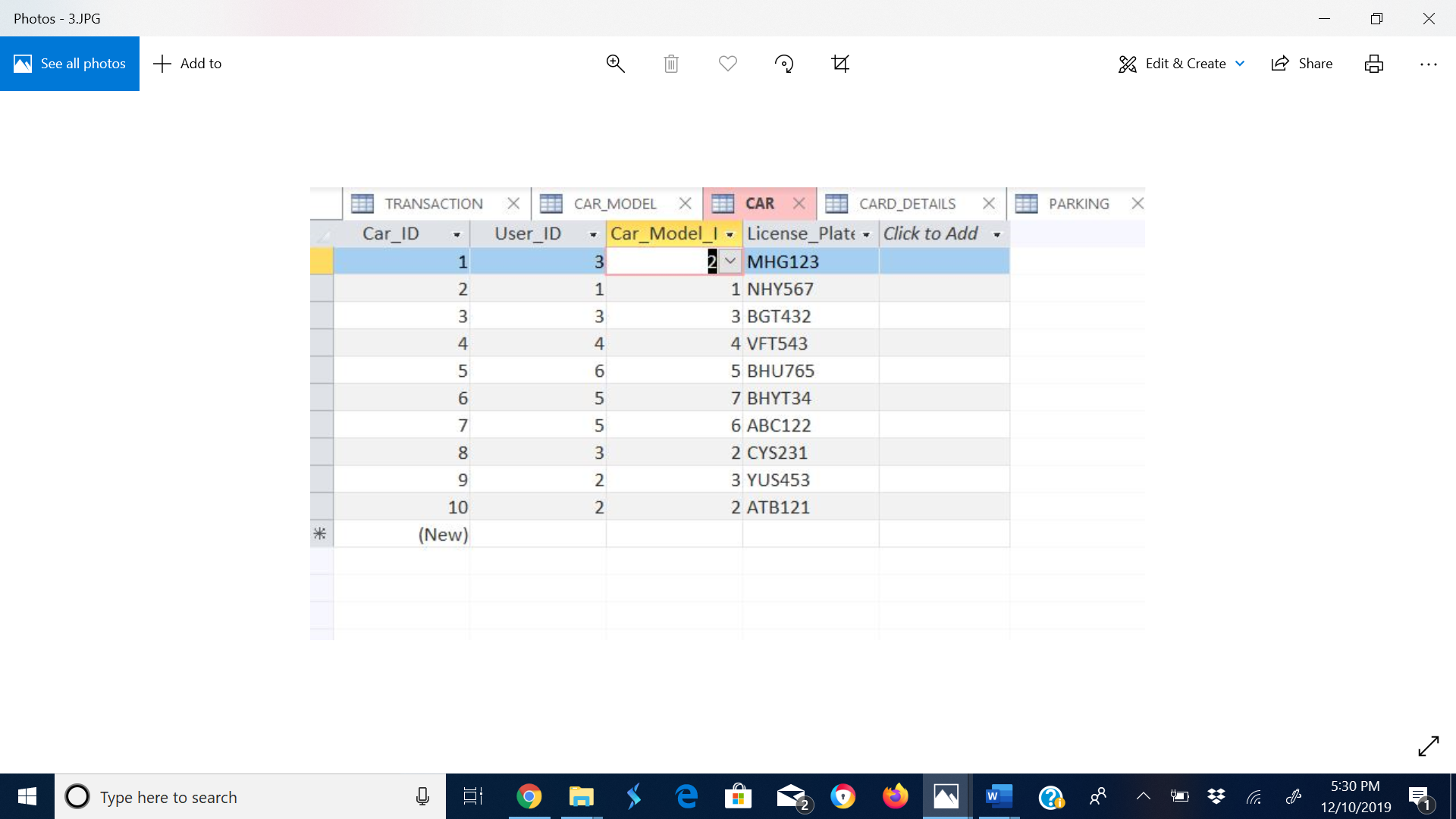


Transaction Table-

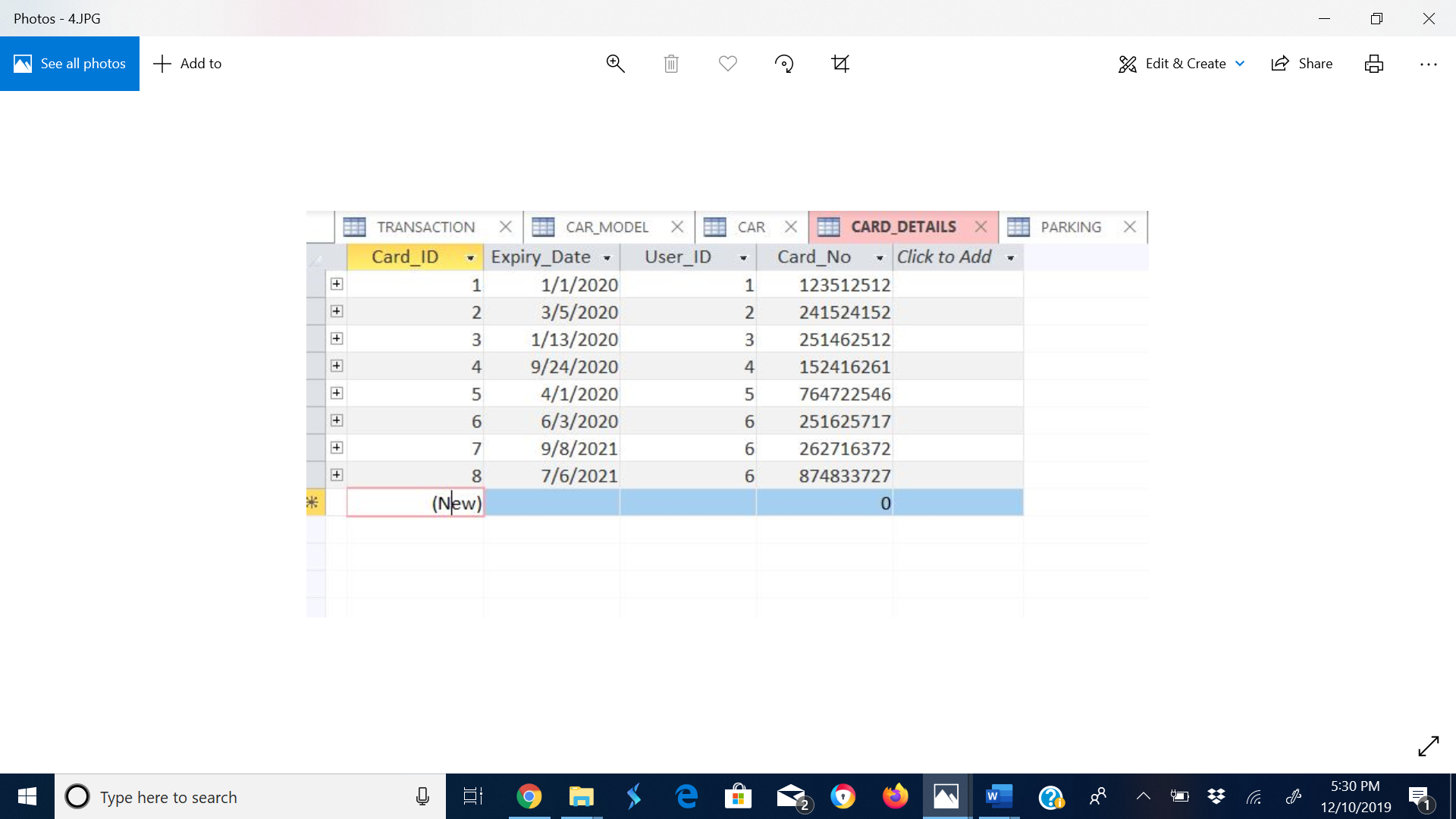


Car\_Model Table-

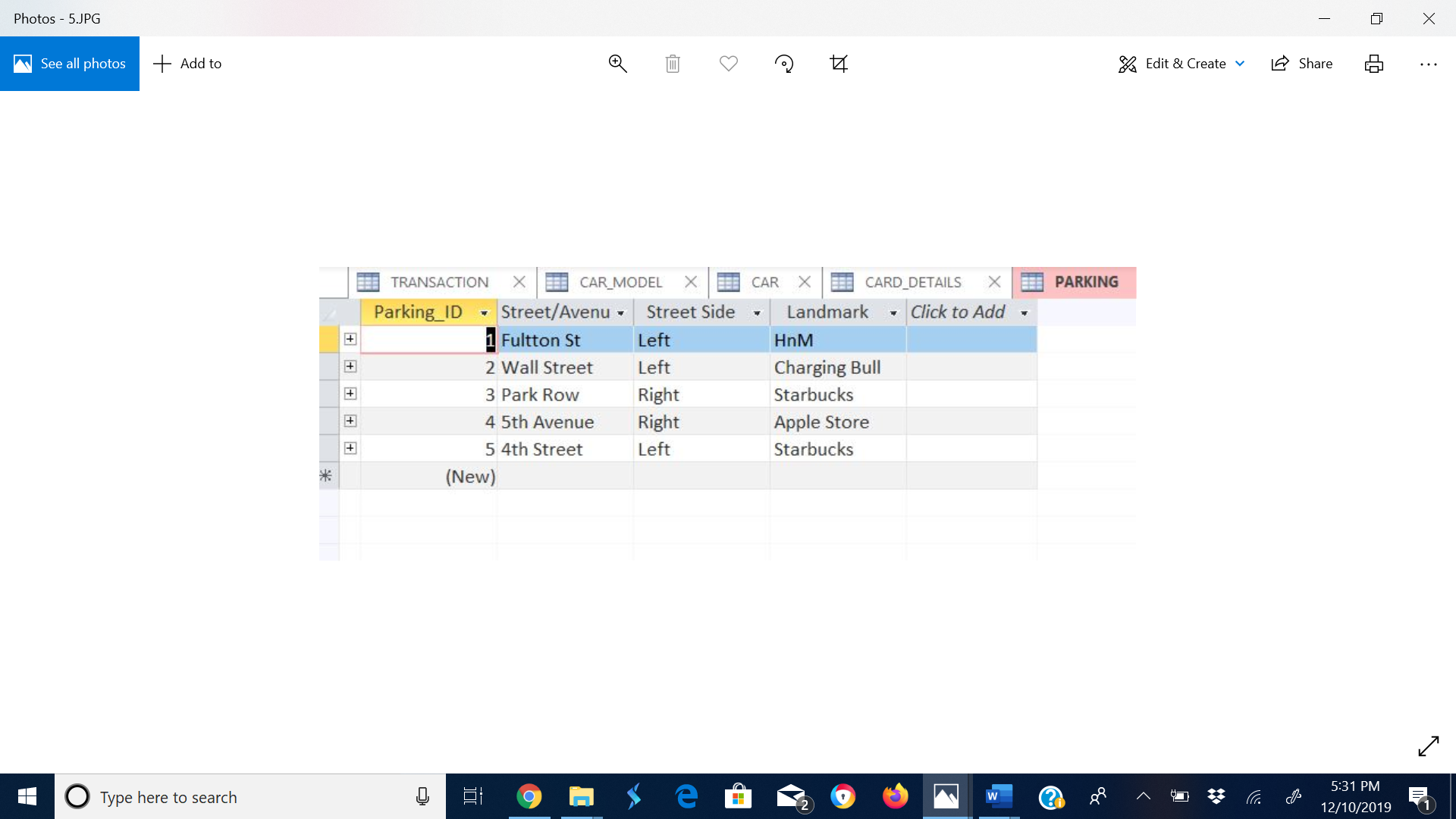


Car Table-

Card\_Details Table-



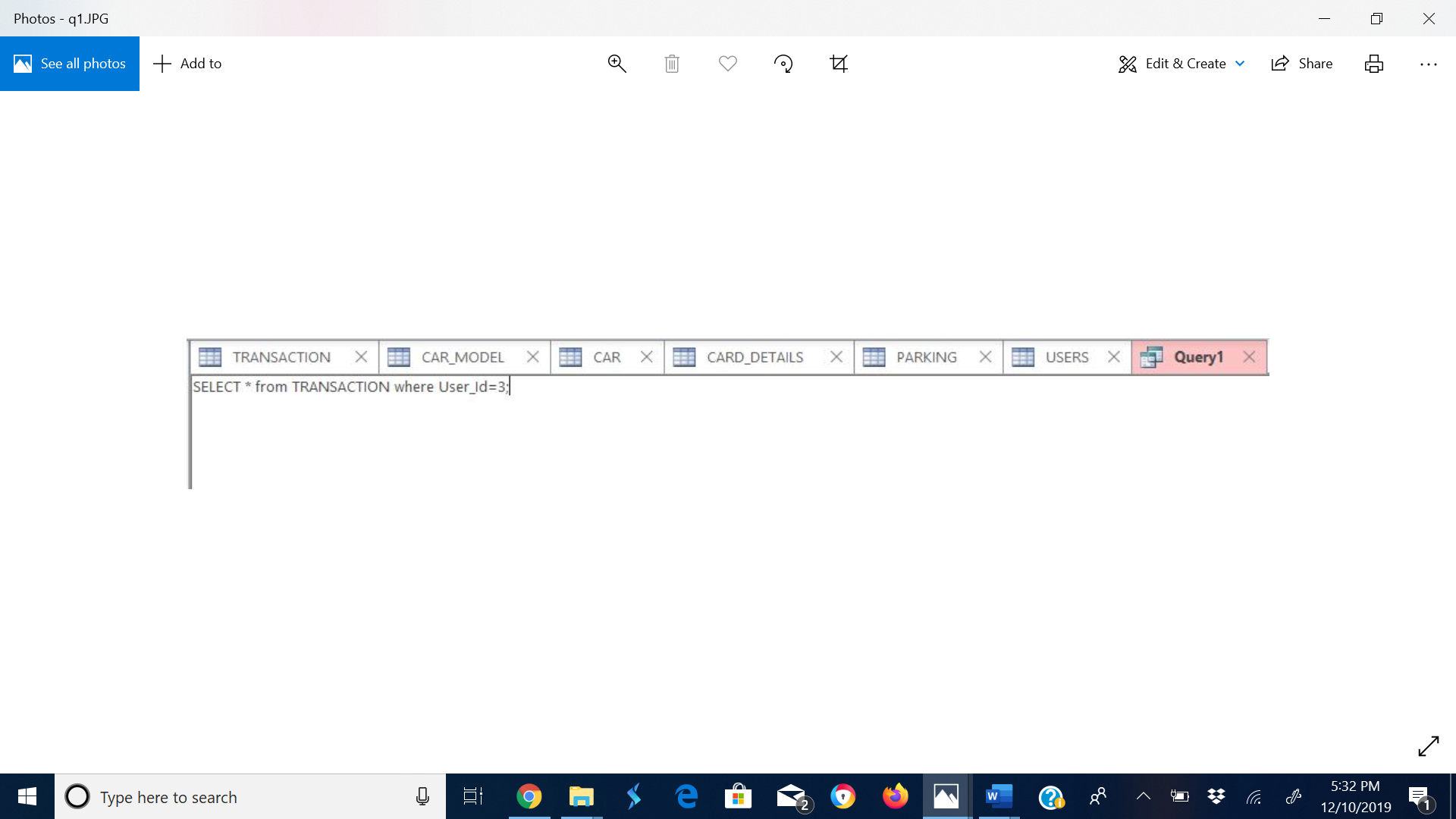
Parking Table-



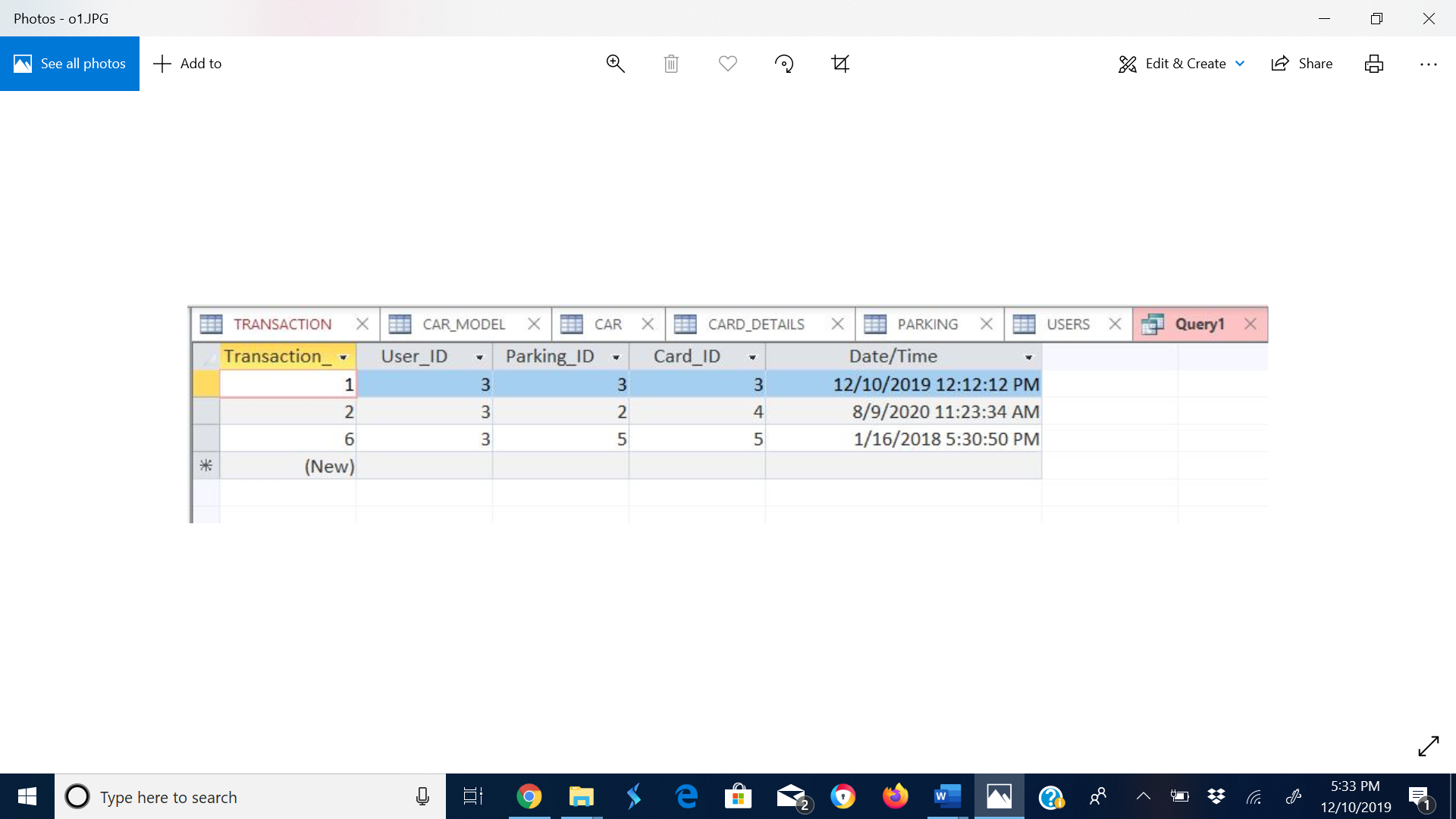
Users Table



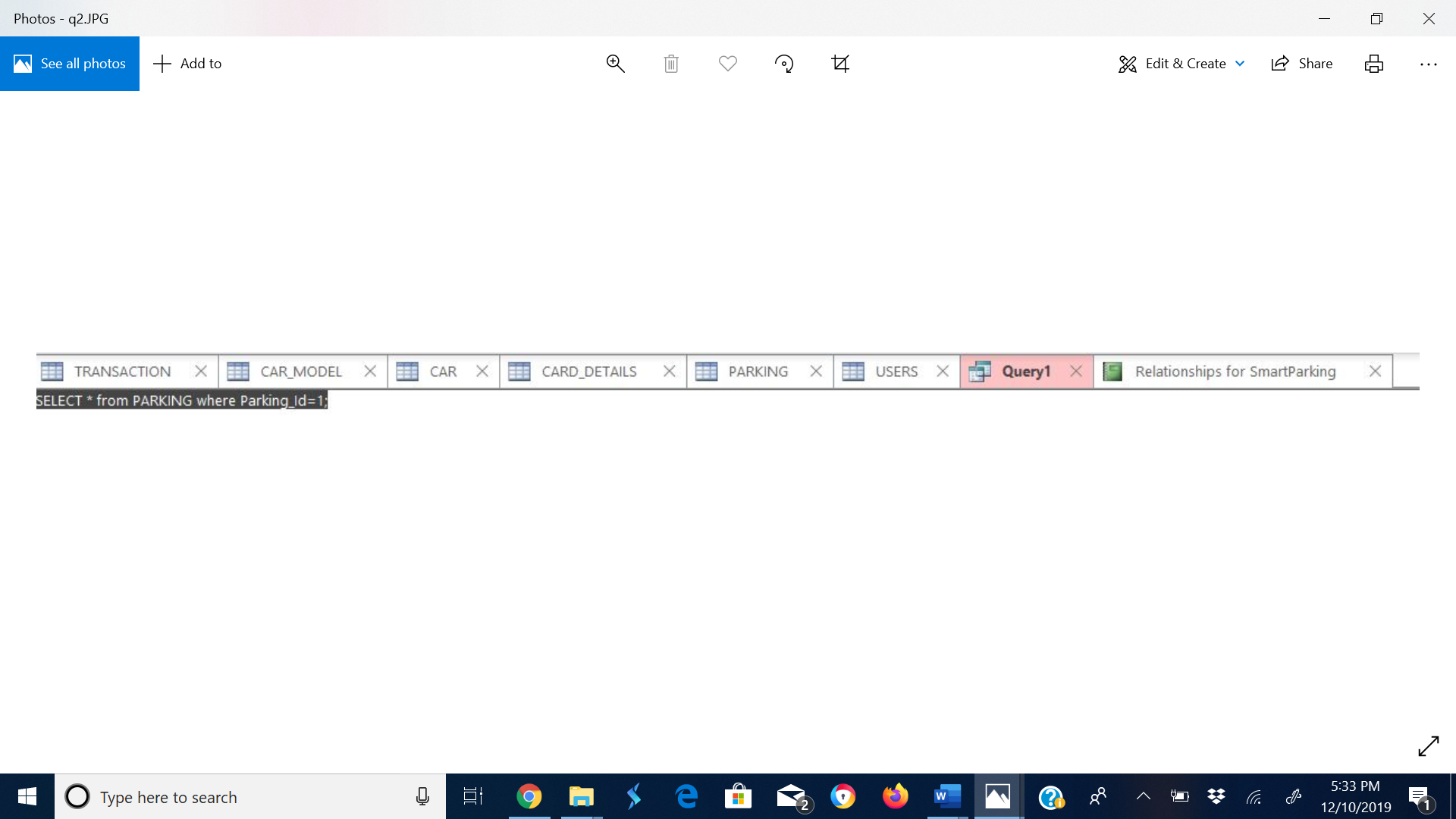
Query 1



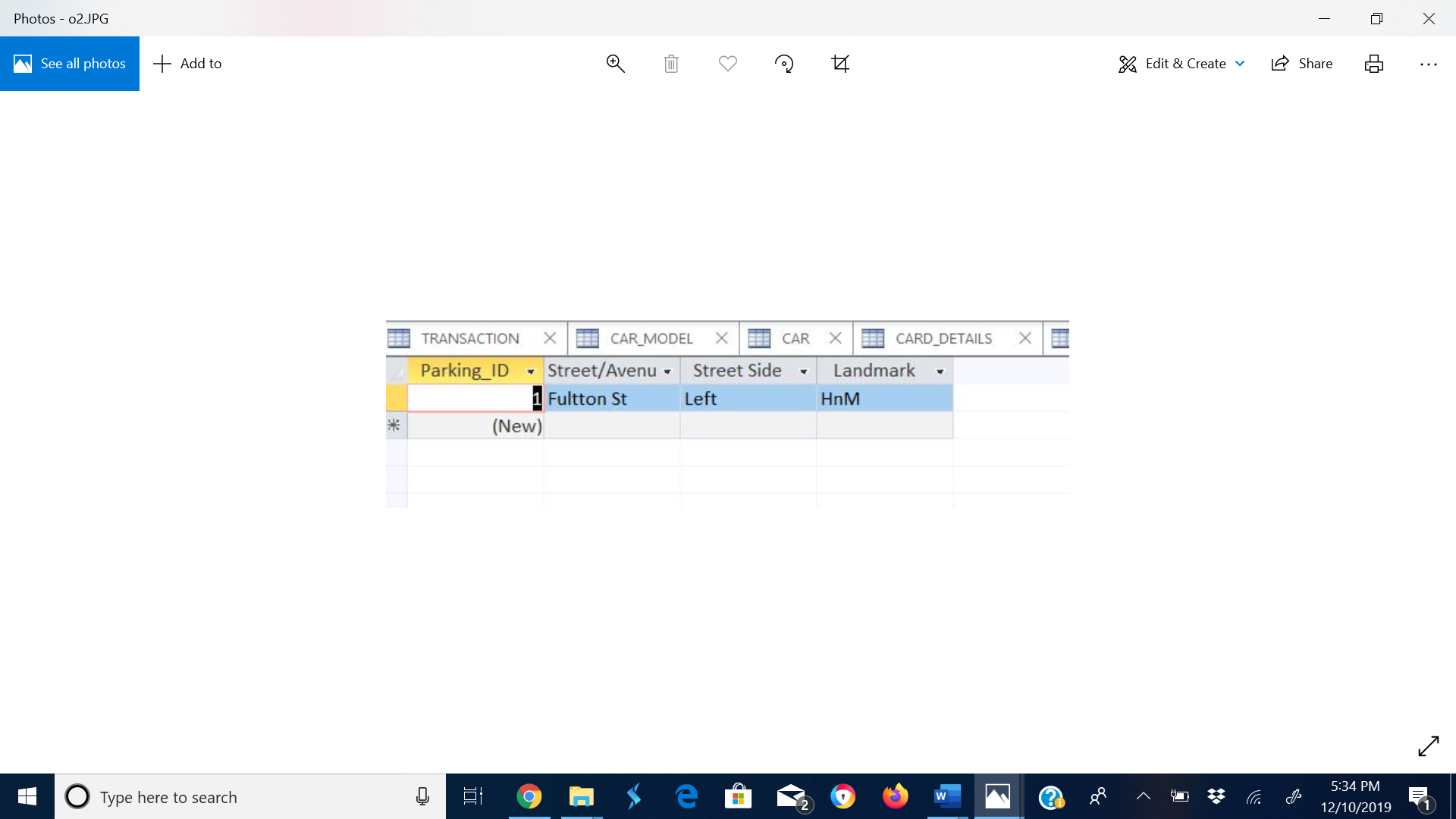
Output



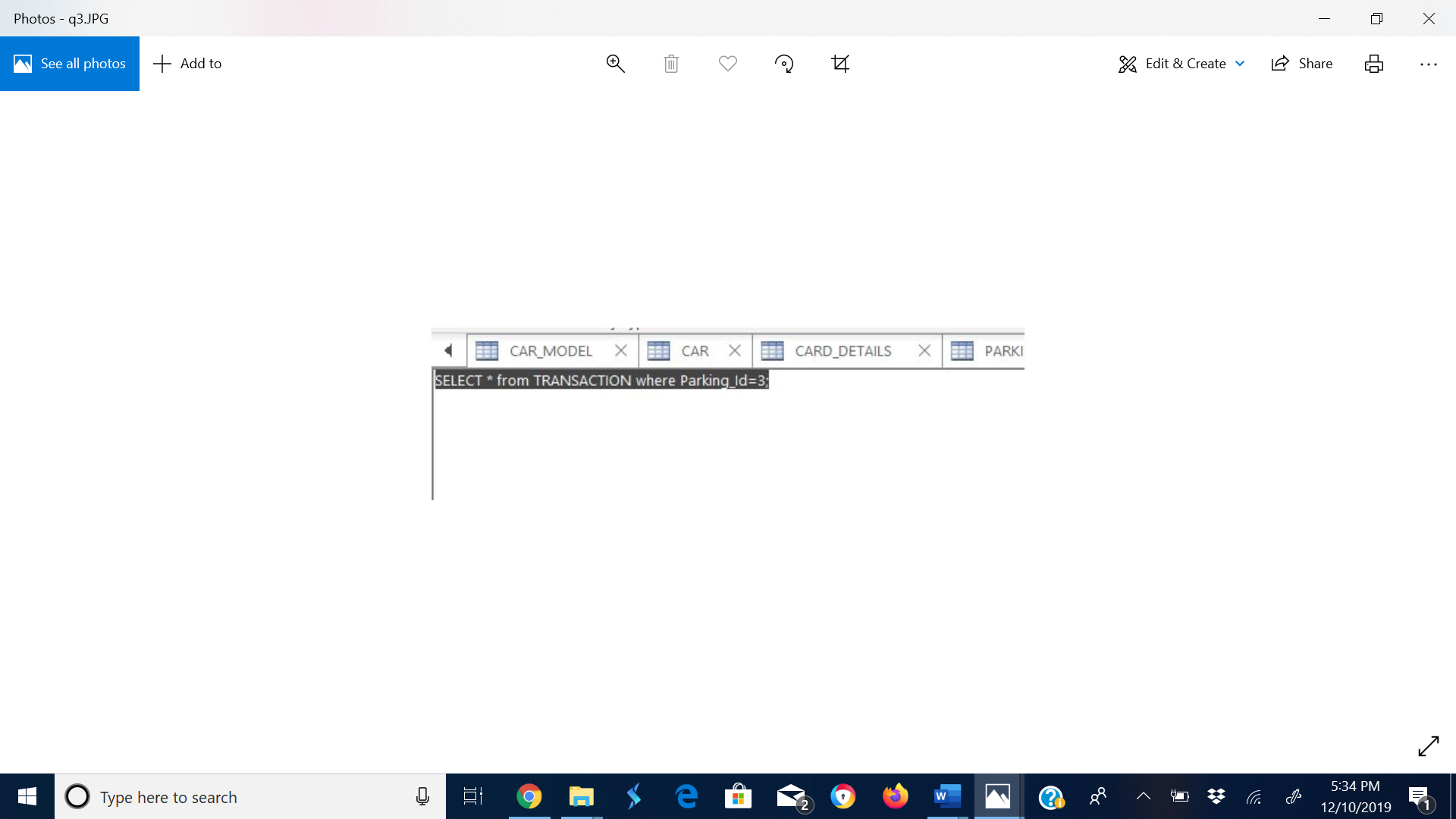
Query 2



Output



Query 3



Output

