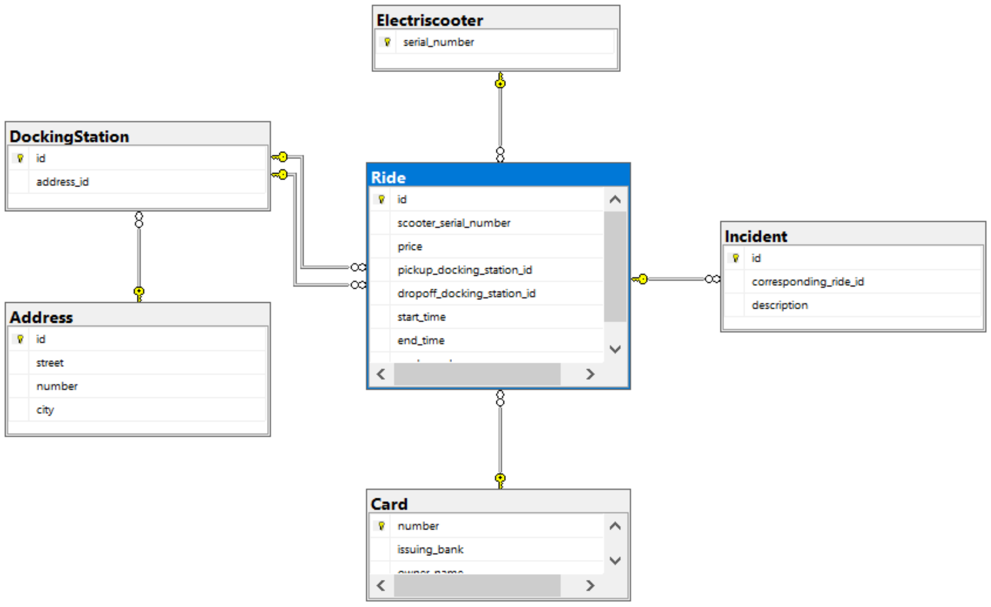
```sql
-- part II
use practic_DB

-- a.
-- creating the table for Electricscooter
CREATE TABLE Electriscooter
(
	serial_number int PRIMARY KEY
);

-- additional table for address, (it's more elegant this way)
CREATE TABLE [Address]
(
	id int PRIMARY KEY,
	street varchar(100),
	number int,
	city varchar(100)
);

-- we are using an id as primary key
CREATE TABLE DockingStation
(
	id int PRIMARY KEY,
	address_id int,

	CONSTRAINT FK_address_id FOREIGN KEY(address_id) REFERENCES [Address](id)
);

CREATE TABLE [Card]
(
	number int PRIMARY KEY,
	issuing_bank varchar(100),
	owner_name varchar(100)
);

-- the ride has 4 foreign keys
CREATE TABLE Ride
(
	id int PRIMARY KEY,
	scooter_serial_number int,
	price int,
	pickup_docking_station_id int,
	dropoff_docking_station_id int,

	start_time DATETIME,
	end_time DATETIME,

	card_number int,


	CONSTRAINT pickup_docking_station_id FOREIGN KEY (pickup_docking_station_id) 
REFERENCES DockingStation(id),
	CONSTRAINT dropoff_docking_station_id FOREIGN KEY (dropoff_docking_station_id) 
REFERENCES DockingStation(id),
	CONSTRAINT card_number FOREIGN KEY (card_number) REFERENCES [Card](number),
	CONSTRAINT FK_scooter_serial_number FOREIGN KEY (scooter_serial_number) REFERENCES 
Electriscooter(serial_number)
);
```

```
CREATE TABLE Incident
(
	id int PRIMARY KEY,
	corresponding_ride_id int,
	[description] varchar(100)
	CONSTRAINT FK_corresponding_ride_id FOREIGN KEY (corresponding_ride_id) REFERENCES
Ride(id)
);



-- populate data (starting from tables with no FK and which are required; use dully data)
INSERT INTO [Address] SELECT 1,'s1',1,'c1';

INSERT INTO DockingStation SELECT 1,1;
INSERT INTO DockingStation SELECT 2,1;
INSERT INTO DockingStation SELECT 3,1;

INSERT INTO [Card] SELECT 1,'B1','O1';

INSERT INTO Electriscooter SELECT 1;
INSERT INTO Electriscooter SELECT 2;

INSERT INTO Ride SELECT 1,1,15,1,2,GETDATE(),GETDATE(),1;
INSERT INTO Ride SELECT 2,1,15,1,3,GETDATE(),GETDATE(),1;

--c.
GO
CREATE OR ALTER VIEW MyView AS
-- we choose the information we are required
SELECT [DockingStation].id, street, number, city
-- we join the tables to have access to all the needed information
FROM Ride INNER JOIN DockingStation ON Ride.pickup_docking_station_id = DockingStation.id
INNER JOIN [Address] ON DockingStation.id = [Address].id
-- we group by id because we are interested in the docking stations with the most rides
and id is the PK for the relation
GROUP BY DockingStation.id,street, number, city
-- the condition is to have the number of rides equal to the maximum number of rides
HAVING COUNT(*) = (
-- here we take the maximum
				SELECT MAX(AMMOUNT)
				FROM
				(SELECT DockingStation.id, COUNT(*) AS AMMOUNT
				FROM Ride INNER JOIN DockingStation ON
Ride.pickup_docking_station_id = DockingStation.id
				-- we need an aditional group by here to make sure we take all
rides for each doking station
				GROUP BY DockingStation.id) AS R)
GO

SELECT *
FROM MyView;

GO
CREATE OR ALTER PROCEDURE deleteCardOwnersAndRides @givenStr varchar(100)
-- we have to delete from the rides first, because it contains FK_card_number
```

```
EXECUTE        ('DELETE FROM Ride WHERE Ride.card_number = ( SELECT [Card].number FROM
[Card] WHERE [Card].owner_name = +' + @givenStr +')' );
EXECUTE        ('DELETE FROM [Card] WHERE [Card].owner_name =' + @givenStr + ')');
GO

-- we use these lines to drop the table if needed
DROP TABLE Ride;
DROP TABLE DockingStation;
DROP TABLE Electriscooter;
DROP TABLE [Address];
DROP TABLE [Card];
DROP TABLE Incident;
```


Electriscooter
serial_number
DockingStation
id
address_id
Ride
id
scooter_serial_number
price
pickup_docking_station_id
dropoff_docking_station_id
start_time
end_time
Incident
id
corresponding_ride_id
description
Address
id
street
number
city
Card
number
issuing_bank