# Part 2 - Creating the Dealership

```
MariaDB [(none)]> USE u1195258;
Database changed
MariaDB [u1195258]> SHOW tables;
Empty set (0.001 sec)
MariaDB [u1195258]> CREATE TABLE Cars (
    -> VIN VARCHAR(225) PRIMARY KEY,
    -> make VARCHAR(225),
    -> model VARCHAR(225),
    -> year INT UNSIGNED,
    -> color VARCHAR(225)
    -> );
Query OK, 0 rows affected (0.008 sec)
MariaDB [u1195258]> SHOW tables;
+--------------------+
| Tables_in_u1195258 |
+--------------------+
| Cars               |
+--------------------+
1 row in set (0.001 sec)
MariaDB [u1195258]> SELECT * FROM CARS
    -> ;
ERROR 1146 (42S02): Table 'u1195258.CARS' doesn't exist
MariaDB [u1195258]> SELECT * FROM Cars;
Empty set (0.001 sec)
MariaDB [u1195258]> CREATE TABLE SalesPeople (
    -> SSN VARCHAR(225) PRIMARY KEY,
    -> Name VARCHAR(225)
    -> );
Query OK, 0 rows affected (0.007 sec)
MariaDB [u1195258]> show tables
    -> ;
+--------------------+
| Tables_in_u1195258 |
+--------------------+
| Cars               |
| SalesPeople        |
+--------------------+
2 rows in set (0.001 sec)
MariaDB [u1195258]> CREATE TABLE CheckedOut {
    -> VIN VARCHAR(225),
    -> SSN VARCHAR(225),
    -> FOREIGN KEY (VIN) REFERENCES Car(VIN),
    -> FOREIGN KEY (SSN) REFERENCES SalesPeople(SSN),
    -> );
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server
version for the right syntax to use near '{
VIN VARCHAR(225),
SSN VARCHAR(225),
FOREIGN KEY (VIN) REFERENCES Car(VIN),
...' at line 1
MariaDB [u1195258]> CREATE TABLE CheckedOut ( VIN VARCHAR(225), SSN VARCHAR(225), FOREIGN KEY (VIN)
REFERENCES Car(VIN), FOREIGN KEY (SSN) REFERENCES SalesPeople(SSN), );
```

```
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server
version for the right syntax to use near ')' at line 1
MariaDB [u1195258]> CREATE TABLE CheckedOut ( VIN VARCHAR(225), SSN VARCHAR(225), FOREIGN KEY (VIN)
REFERENCES Car(VIN), FOREIGN KEY (SSN) REFERENCES SalesPeople(SSN) );
ERROR 1005 (HY000): Can't create table `u1195258`.`CheckedOut` (errno: 150 "Foreign key constraint is incorrectly formed")
MariaDB [u1195258]> CREATE TABLE CheckedOut ( VIN VARCHAR(225), SSN VARCHAR(225), FOREIGN KEY (VIN)
REFERENCES Car(VIN), FOREIGN KEY (SSN) REFERENCES SalesPeople(SSN) );
ERROR 1005 (HY000): Can't create table `u1195258`.`CheckedOut` (errno: 150 "Foreign key constraint is incorrectly formed")
MariaDB [u1195258]> CREATE TABLE CheckedOut ( VIN VARCHAR(225), SSN VARCHAR(225), FOREIGN KEY (VIN)
REFERENCES Cars(VIN), FOREIGN KEY (SSN) REFERENCES SalesPeople(SSN) );
Query OK, 0 rows affected (0.012 sec)
MariaDB [u1195258]> show tables;
+--------------------+
| Tables_in_u1195258 |
+--------------------+
| Cars               |
| CheckedOut         |
| SalesPeople        |
+--------------------+
3 rows in set (0.001 sec)
MariaDB [u1195258]> INSERT INTO Cars (VIN, make, model, year, color) VALUES
    -> ('1FTFW1EF1', 'Toyota', 'Tacoma', 2008, 'Red'),
    -> ('1FTFW2EF2', 'Toyota', 'Tacoma', 1999, 'Green'),
    -> ('1FTFW3EF3', 'Tesla', 'Model 3', 2018, 'White'),
    -> ('1FTFW4EF4', 'Subaru', 'WRX', 2016, 'Blue'),
    -> ('1FTFW5EF5', 'Ford', 'F150', 2004, 'Red');
Query OK, 5 rows affected (0.002 sec)
Records: 5  Duplicates: 0  Warnings: 0
MariaDB [u1195258]> SELECT * FROM Cars
    -> ;
+-----------+--------+---------+------+-------+
| VIN       | make   | model   | year | color |
+-----------+--------+---------+------+-------+
| 1FTFW1EF1 | Toyota | Tacoma  | 2008 | Red   |
| 1FTFW2EF2 | Toyota | Tacoma  | 1999 | Green |
| 1FTFW3EF3 | Tesla  | Model 3 | 2018 | White |
| 1FTFW4EF4 | Subaru | WRX     | 2016 | Blue  |
| 1FTFW5EF5 | Ford   | F150    | 2004 | Red   |
+-----------+--------+---------+------+-------+
5 rows in set (0.001 sec)
MariaDB [u1195258]> INSERT INTO SalesPeople (SSN, Name) VALUES
    -> ('111-11-1111', 'Arnold'),
    -> (222-22-2222', 'Hannah'),
    '> ('333-33-3333', 'Steve');
    '>
    '> ;
    '>
    '> ;
    '> exit
    '> quit
    '> escape
    '> esc
    '>
    '>
    '>
```

```
   '> EXIT;
   '> Ctrl-C -- exit!
Aborted
[u1195258@lab1-11 ~]$ hello
hello: Command not found.
[u1195258@lab1-11 ~]$ mysql -h cs-db.eng.utah.edu -u u1195258 -p --ssl-verify-server-cert
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 242
Server version: 10.11.3-MariaDB-log MariaDB Server
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
MariaDB [(none)]> show tables
   -> ;
ERROR 1046 (3D000): No database selected
MariaDB [(none)]> USE u1195258;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
Database changed
MariaDB [u1195258]> SHOW TABLES;
+--------------------+
| Tables_in_u1195258 |
+--------------------+
| Cars               |
| CheckedOut         |
| SalesPeople        |
+--------------------+
3 rows in set (0.001 sec)
MariaDB [u1195258]> SELECT * FROM Cars;
+-----------+--------+---------+------+-------+
| VIN       | make   | model   | year | color |
+-----------+--------+---------+------+-------+
| 1FTFW1EF1 | Toyota | Tacoma  | 2008 | Red   |
| 1FTFW2EF2 | Toyota | Tacoma  | 1999 | Green |
| 1FTFW3EF3 | Tesla  | Model 3 | 2018 | White |
| 1FTFW4EF4 | Subaru | WRX     | 2016 | Blue  |
| 1FTFW5EF5 | Ford   | F150    | 2004 | Red   |
+-----------+--------+---------+------+-------+
5 rows in set (0.001 sec)
MariaDB [u1195258]> SELECT * FROM SalesPeople;
Empty set (0.001 sec)
MariaDB [u1195258]> INSERT INTO CheckedOut (VIN, SSN) VALUES
   -> ('1FTFW1EF1', '111-11-1111'),
   -> ('1FTFW2EF2', '111-11-1111'),
   -> ('1FTFW1EF1', '222-22-2222'),
   -> ('1FTFW5EF5', '222-22-2222'),
   -> ('1FTFW3EF3', '333-33-3333');
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails (`u1195258`.`CheckedOut`,
CONSTRAINT `CheckedOut_ibfk_2` FOREIGN KEY (`SSN`) REFERENCES `SalesPeople` (`SSN`))
MariaDB [u1195258]> INSERT INTO SALESPEOPLE (SSN, Name) VALUES
   -> ('111-11-1111', 'Arnold'),
   -> ('222-22-2222', 'Hannah'),
   -> ('333-33-3333', 'Steve');
ERROR 1146 (42S02): Table 'u1195258.SALESPEOPLE' doesn't exist
```

```
MariaDB [u1195258]> INSERT INTO SalesPeople (SSN, Name) VALUES ('111-11-1111', 'Arnold'), ('222-22-2222', 'Hannah'),
('333-33-3333'
Query OK, 3 rows affected (0.001 sec)
Records: 3  Duplicates: 0  Warnings: 0
MariaDB [u1195258]> SELECT * FROM SalesPeople
   -> ;
+-------------+--------+
| SSN         | Name   |
+-------------+--------+
| 111-11-1111 | Arnold |
| 222-22-2222 | Hannah |
| 333-33-3333 | Steve  |
+-------------+--------+
3 rows in set (0.001 sec)
MariaDB [u1195258]> INSERT INTO CheckedOut (VIN, SSN) VALUES
   -> ('1FTFW1EF1', '111-11-1111'),
   -> ('1FTFW2EF2', '111-11-1111'),
   -> ('1FTFW1EF1', '222-22-2222'),
   -> ('1FTFW5EF5', '222-22-2222'),
   -> ('1FTFW3EF3', '333-33-3333');
Query OK, 5 rows affected (0.001 sec)
Records: 5  Duplicates: 0  Warnings: 0
MariaDB [u1195258]> SELECT * FROM CheckedOut
   -> ;
+-----------+-------------+
| VIN       | SSN         |
+-----------+-------------+
| 1FTFW1EF1 | 111-11-1111 |
| 1FTFW2EF2 | 111-11-1111 |
| 1FTFW1EF1 | 222-22-2222 |
| 1FTFW5EF5 | 222-22-2222 |
| 1FTFW3EF3 | 333-33-3333 |
+-----------+-------------+
5 rows in set (0.001 sec)
MariaDB [u1195258]>
```

| VIN | make | model | year | color |
|---|---|---|---|---|
| 1FTFW1EF1 | Toyota | Tacoma | 2008 | Red |
| 1FTFW2EF2 | Toyota | Tacoma | 1999 | Green |
| 1FTFW3EF3 | Tesla | Model 3 | 2018 | White |
| 1FTFW4EF4 | Subaru | WRX | 2016 | Blue |
| 1FTFW5EF5 | Ford | F150 | 2004 | Red |

| ssn | name |
|---|---|
| 111-11-1111 | Arnold |
| 222-22-2222 | Hannah |
| 333-33-3333 | Steve |

| VIN | ssn |
|---|---|
| 1FTFW1EF1 | 111-11-1111 |
| 1FTFW2EF2 | 111-11-1111 |
| 1FTFW1EF1 | 222-22-2222 |
| 1FTFW5EF5 | 222-22-2222 |
| 1FTFW3EF3 | 333-33-3333 |

# Part 3 - Simple Retrieval Queries

Connect to the Library database, and create queries to find the specified information below. Some of these are quite simple, and we have already seen them. Note that the Library has some additional information in it to make the queries more interesting. The slides shown in class do not contain the full instances.

1. Get the ISBNs of all books by <Author>

```
SELECT * FROM Titles WHERE Author = '<Author>';
```

2. Get Serial numbers (descending order) of all books by <ISBN>

```
SELECT * FROM Titles ORDER BY ISBN DESC;
```

3. Get the Serial numbers and ISBNs of all books checked out by <Patron's name>

```
SELECT Inventory.*
  -> FROM Inventory
  -> JOIN CheckedOut ON Inventory.Serial = CheckedOut.Serial
  -> JOIN Patrons ON CheckedOut.CardNum = Patrons.CardNum
  -> WHERE Patrons.Name = '<Patron's Name>';
```

4. Get phone number(s) and Name of anyone with <ISBN> checked out

```
SELECT Patrons.Name, Phones.Phone
-> FROM Phones
-> JOIN Patrons ON Phones.CardNum = Patrons.CardNum
-> JOIN CheckedOut ON Patrons.CardNum = CheckedOut.CardNum
->JOIN Inventory ON CheckedOut.Serial = Inventory.Serial
-> WHERE Inventory.ISBN = '<ISBN>'';
```

When testing out your queries, you can replace the variables such as <Author> with an author that you know is in the database, such as "Herbert". However, when you hand in your solution, put the placeholder back into the query.

# Part 4 - Intermediate Retrieval Queries

1. Find the Authors of the library's oldest <N> books. Assume the lowest serial number is the oldest book.

```
SELECT Titles.Author
  -> FROM Titles
  -> JOIN Inventory ON Titles.ISBN = Inventory.ISBN
  -> ORDER BY Inventory.Serial ASC
  -> LIMIT <N>;
```

2. Find the name and phone number of the person who has checked out the most recent book. Assume higher serial numbers are newer. **Note that this query is not concerned with the absolute highest serial number, it is concerned with the highest one that has been checked out.**

```
SELECT Patrons.Name, Phones.Phone
  -> FROM Phones
  -> JOIN Patrons ON Phones.CardNum = Patrons.CardNum
  -> JOIN CheckedOut ON Patrons.CardNum = CheckedOut.CardNum
  -> JOIN Inventory ON CheckedOut.Serial = Inventory.Serial
  -> WHERE Inventory.Serial = (SELECT MAX(Serial) FROM CheckedOut);
```

3. Find the phone number(s) and name of anyone who has checked out any book.

```
SELECT Patrons.Name, Phones.Phone
  -> FROM Phones
  -> JOIN Patrons ON Phones.CardNum = Patrons.CardNum
  -> WHERE Patrons.CardNum NOT IN (
  -> SELECT CardNum FROM CheckedOut
  -> );
```

4. Find the Authors and Titles of the books who have NOT been checked out by anyone. The query should not return duplicates.

```
SELECT Titles.Author, Titles.Title
        -> FROM Inventory
        ->JOIN Titles ON Inventory.ISBN = Titles.ISBN
        -> WHERE Inventory.Serial NOT IN ( SELECT Serial FROM
        CheckedOut );
```

# Part 5 - Chess Queries

For this part, switch to the Chess database.

```
MariaDB [Chess]> DESCRIBE Players;
+-------+------------------+------+-----+---------+----------------+
| Field | Type             | Null | Key | Default | Extra          |
+-------+------------------+------+-----+---------+----------------+
| Name  | varchar(255)     | YES  | UNI | NULL    |                |
| Elo   | int(10) unsigned | YES  |     | NULL    |                |
| pID   | int(10) unsigned | NO   | PRI | NULL    | auto_increment |
+-------+------------------+------+-----+---------+----------------+
3 rows in set (0.001 sec)
MariaDB [Chess]> DESCRIBE Events;
+-------+------------------+------+-----+---------+----------------+
| Field | Type             | Null | Key | Default | Extra          |
+-------+------------------+------+-----+---------+----------------+
| Name  | varchar(255)     | NO   | MUL | NULL    |                |
| Site  | varchar(255)     | NO   |     | NULL    |                |
| Date  | date             | NO   |     | NULL    |                |
| eID   | int(10) unsigned | NO   | PRI | NULL    | auto_increment |
+-------+------------------+------+-----+---------+----------------+
4 rows in set (0.001 sec)
MariaDB [Chess]> DESCRIBE Games;
+-------------+------------------+------+-----+---------+-------+
| Field       | Type             | Null | Key | Default | Extra |
+-------------+------------------+------+-----+---------+-------+
| Round       | varchar(10)      | NO   | PRI | NULL    |       |
| Result      | char(1)          | NO   |     | NULL    |       |
| Moves       | varchar(2000)    | NO   |     | NULL    |       |
| BlackPlayer | int(10) unsigned | NO   | PRI | NULL    |       |
| WhitePlayer | int(10) unsigned | NO   | PRI | NULL    |       |
| eID         | int(10) unsigned | NO   | PRI | NULL    |       |
+-------------+------------------+------+-----+---------+-------+
6 rows in set (0.002 sec)
```

Provide SQL for the queries below:

1. Find the names and IDs of any player with the 10 highest Elo ratings.

```
SELECT Players.Name, Players.pID
-> FROM Players
-> ORDER BY Players.Elo DESC
->LIMIT 10;
```

2. Find the names and Elo ratings of any player who has ever played a game as black.

```
SELECT Players.Name, Players.Elo
-> FROM Games
-> JOIN Players ON Games.BlackPlayer = Players.pID;
```

3. Find the names of any player who has ever won a game as white.

```
SELECT Players.Name
-> FROM Players JOIN Games ON Players.pID = Games.WhitePlayer
-> WHERE Games.Result = 'W';
```

4. Find the names of any player who played any games between 2014 and 2018 in Budapest HUN .

```
SELECT DISTINCT Players.Name
  -> FROM Players
  -> JOIN Games ON (Players.pID = Games.WhitePlayer OR Players.pID =
Games.BlackPlayer)
  -> JOIN Events ON Games.eID = Events.eID
  -> WHERE Events.Site = 'Budapest HUN'
  -> AND YEAR(Events.Date) BETWEEN 2014 AND 2018;
```

5. Find the Sites and dates of any event in which Garry Kasparov won a game.

```
SELECT DISTINCT Events.Date, Events.Site
FROM Players
JOIN Games ON (Players.pID = Games.WhitePlayer OR Players.pID =
Games.BlackPlayer)
JOIN Events ON Games.eID = Events.eID
WHERE Players.Name = 'Kasparov, Garry'
AND ((Games.WhitePlayer = Players.pID AND Games.Result = 'W') OR
   (Games.BlackPlayer = Players.pID AND Games.Result = 'B'));
```

6. Find the names of all opponents of Magnus Carlsen. An opponent is someone who he has played a game against. **Hint:** Both Magnus and his opponents could play as white or black.

```
SELECT DISTINCT p.Name
  -> FROM Players p
  -> JOIN Games g ON (p.pID = g.WhitePlayer OR p.pID = g.BlackPlayer)
  -> WHERE (g.WhitePlayer = (SELECT pID FROM Players WHERE Name =
'Carlsen, Magnus') AND p.pID <> (SELECT pID FROM Players WHERE
Name = 'Carlsen, Magnus'))
  ->   OR (g.BlackPlayer = (SELECT pID FROM Players WHERE Name =
'Carlsen, Magnus') AND p.pID <> (SELECT pID FROM Players WHERE
Name = 'Carlsen, Magnus'));
```