```
CREATE DATABASE IF NOT EXISTS cars_data;

-- Creates a database named cars_data if it doesn't exist already

USE cars_data;

-- selects the specified database for execution of all the queries from now on

CREATE TABLE IF NOT EXISTS cars(
Manufacturer VARCHAR(50),
Varient VARCHAR(50),
Details VARCHAR(50),
India_Locations VARCHAR(50),
Model INT,
Distance_Travelled INT,
Fuel_Type VARCHAR(20),
Engine_Capacity VARCHAR(50),
Transmission VARCHAR(15),
Price VARCHAR(50)
);
-- creates a table named cars with the data- type defined for each coloumn

SELECT * FROM cars;

-- shows all the columns from the car table

LOAD DATA INFILE 'Dataset_Used Cars  (1).csv' INTO TABLE cars
FIELDS TERMINATED BY ','
IGNORE 1 LINES;

-- loads the CSV data from the specified file and its path into the cars table

CREATE TABLE IF NOT EXISTS cars_copy AS SELECT * FROM cars WHERE 1=0;
INSERT INTO cars_copy SELECT * FROM cars;

-- creates a duplicate copy of the table

SET SQL_SAFE_UPDATES = 0;

-- turns off safe mode thus allowing us to update data in the table

DESCRIBE cars;

-- describes the schema of the specified table

/* Now that we have loaded the data and specified the right data types for each column
```

*We will do some data cleaning to remove the outliers and correct some wrong entries in some of the columns */*

SELECT * FROM cars WHERE Price LIKE ' %Call% ';

/ shows all the columns where the price coloumn contains ' call ',*
here we used a wildcard %% to ignore everything around 'call' because earlier
*it was not showing any entries where there is ' call ' using the normal method */*

UPDATE cars
SET Price = NULL
WHERE Price LIKE ' %call% ';

-- updates all the 'call' entries in `Price` coloumn to NULL

SELECT * FROM cars WHERE Price IS NULL;

-- shows all the columns where the entry under the 'Price' coloumn is NULL

ALTER TABLE cars
MODIFY Price INT;

-- changes the data type of the Price coloumn to INT

/ so far we have changed the 'call' values in the `Price` coloumn because of which*
we had to make the data type of the table as varchar while importing the table
and after changing the 'call' values to null, we changed the data type of the `Price' coloumn
*to the suitable data type which is INT using the ALTER TABLE command */*

SELECT * FROM cars WHERE Price IS NULL;

SELECT * FROM cars;

SELECT Transmission, COUNT(Price) FROM cars GROUP BY Transmission;

-- groups the categories in the Transmission coloumn by their count
-- We found out that there were 4 wrong entries so in the below steps we corrected those entries too

SELECT * FROM cars WHERE Transmission IN(Lower('R') , '4');

-- shows the rows where the Transmission coloumn contains either of the specified values
-- then we note down car make and model

```
SELECT * FROM cars WHERE Manufacturer = Lower('Toyota') AND Varient = Lower('Passo') AND Details IN('X L Package S  ' , 'X S  ');

/* we use this query to show the rows wherever there are the same make and model we noted
and then we checked the Transmission details for those models and we found out that all the models were 'Automatic' and `Engine_Capacity` was 1000
so we update the 'Transmission' and `Engine_Capacity` for the wrong ones to the correct ones using the queries executed below */


UPDATE cars
SET Transmission = 'Automatic' WHERE Transmission IN( Lower('R') , '4');

-- updates the wrong value to the correct one

SELECT Engine_Capacity, COUNT(Price) FROM cars GROUP BY Engine_Capacity;

SELECT * FROM cars WHERE Engine_Capacity = 'cc';

UPDATE cars
SET Engine_Capacity = '1000'
WHERE Engine_Capacity = 'cc';

-- updates the wrong entries with the specified/correct entries

/* Now we have pretty much cleaned our dataset for aggregation and
 the two columns which have some blank entries under the make and model e will simply ignore them */

SELECT * FROM cars;
DESCRIBE cars;


SELECT @@sql_mode;
SET SESSION sql_mode = '';
-- Disables ONLY_FULL_GROUP_BY Temporarily


SELECT
   Manufacturer,
   Varient AS Model,
   AVG(Price) AS Average_Price
FROM cars
GROUP BY Manufacturer, Model
ORDER BY Manufacturer, Average_Price DESC;
-- Displays Average Price by Manufacturer and Model
```

```
SELECT
   Model AS Year,
   COUNT(*) AS Car_Count
FROM cars
GROUP BY Year
ORDER BY Year DESC;
-- Count of Cars by Year of Manufacture


SELECT
   Manufacturer,
   Varient AS Model,
   MAX(Distance_Travelled) AS Max_Distance
FROM cars
GROUP BY Manufacturer, Model
ORDER BY Max_Distance DESC;
-- Maximum Distance Travelled by Manufacturer and Model

SELECT
   Transmission,
   COUNT(*) AS Car_Count
FROM cars
GROUP BY Transmission;
-- Number of Cars by Transmission Type

SELECT
   Fuel_Type,
   COUNT(*) AS Total_Cars,
   AVG(Price) AS Average_Price
FROM cars
GROUP BY Fuel_Type
ORDER BY Total_Cars DESC;
-- Total Cars and Average Price by Fuel Type


SELECT
   Manufacturer,
   AVG(Engine_Capacity) AS Avg_Engine_Capacity,
   AVG(Price) AS Avg_Price
FROM cars
GROUP BY Manufacturer
ORDER BY Avg_Engine_Capacity DESC;
-- Average Engine Capacity and Price by Manufacturer
```

```
SELECT
   India_Locations AS Location,
   COUNT(*) AS Car_Count
FROM cars
GROUP BY Location
ORDER BY Car_Count DESC;
-- Cars Available by Location


SELECT
   Manufacturer,
   Varient AS Model,
   MAX(Price) AS Price
FROM cars
GROUP BY Manufacturer, Varient
ORDER BY Price DESC
LIMIT 5;
-- Top 5 Most Expensive Cars by Manufacturer and Model


SELECT
Manufacturer,
Varient AS Model,
COUNT(*) AS Occurences
FROM cars
GROUP BY Manufacturer, Model
ORDER BY Occurences DESC;
-- Displays the count of the most popular car makes and models


SELECT
Manufacturer AS Make,
Count(*) AS Occurences
FROM cars
GROUP BY Make
ORDER BY Occurences DESC;
-- Most popular makes by their count

SELECT
Varient AS Model,
Count(*) AS Occurences
FROM cars
GROUP BY Model
ORDER BY Occurences DESC;
-- Most Popular Models by their count
```

```
SELECT
   Manufacturer,
   Varient AS Car_Model,
   Model AS Model_Year,
   COUNT(*) AS Total_Cars,
   AVG(Price) AS Average_Price,
   MIN(Price) AS Min_Price,
   MAX(Price) AS Max_Price
FROM cars
WHERE Model IS NOT NULL  -- Ensures only valid years are included
GROUP BY Manufacturer, Varient, Model
ORDER BY Model ASC, Average_Price DESC;
-- displays a summary of pricing trends over the years, showing the average, minimum, and maximum prices for cars manufactured in each year

SELECT
India_Locations AS Region,
COUNT(*) AS Total_Cars,
AVG(Price) AS Average_Price,
MIN(Price) AS Minimun_Price,
MAX(Price) AS Maximum_Price
FROM cars
WHERE Price IS NOT NULL
GROUP BY India_Locations
ORDER BY Average_Price DESC;
-- displays data by location ( average price, min and max price)
```