Cars Data analysis in SQL

Cars Data analysis is end-to-end MySQL project

Car Dekho is an online platform designed to help users buy and sell cars. It offers detailed information, tools, and resources to facilitate informed vehicle purchases.

This report summarizes the results of complex SQL queries conducted on the "Car dekho" dataset. Each query answers a specific business question related to car listings, pricing, performance metrics, and trends in fuel types, transmission, and seller types.

Q.1 Identify the top 5 most expensive cars listed in the dataset.
Ans:

```
SELECT
   Name, selling_price
FROM
   car_dekho
ORDER BY selling_price DESC
LIMIT 5;
```

Explanation:

1. **SELECT Name, selling_price**: This selects the Name and selling_price columns from the car_dekho table.
2. **FROM car_dekho**: Specifies that the data is being selected from the car_dekho table.
3. **ORDER BY selling_price DESC**: Orders the results by selling_price in descending order (highest price first).
4. **LIMIT 5**: Limits the result to the top 5 rows.

This query will return the names of the top 5 cars with the highest selling prices.

Output:

	Name	selling_price
▶	Volvo XC90 T8 Excellence BSIV	10000000
	BMW X7 xDrive 30d DPE	7200000
	Audi A6 35 TFSI Matrix	6523000
	Audi A6 35 TFSI Matrix	6223000
	BMW 6 Series GT 630d Luxury Line	6000000

Q. 2 Find the total number of cars available for each fuel type.
Ans:

```
SELECT
   fuel, COUNT(*) AS total_cars
FROM
   car_dekho
GROUP BY fuel;
```

Explanation:

1. **SELECT fuel, COUNT(*) AS total_cars**:
   a. This selects the fuel type and counts the number of cars for each fuel type.
   b. COUNT(*) counts all the rows for each group (fuel type), and the result is labeled as total_cars.
2. **FROM car_dekho**:
   a. Specifies that the data is being selected from the car_dekho table.
3. **GROUP BY fuel**:
   a. Groups the rows based on the fuel column so that the count is calculated for each fuel type.

This query will give you the number of cars available for each fuel type.

Output:

	fuel	total_cars
▶	CNG	53
	Diesel	4304
	Electric	1
	LPG	35
	Petrol	3534

Q. 3 List the average selling price of cars for each year of manufacture.
Ans:

```
SELECT
   year, AVG(selling_price) AS avg_selling_price
FROM
   car_dekho
GROUP BY year;
```

Explanation:

1. **SELECT year, AVG(selling_price) AS avg_selling_price**:
   a. This selects the year and calculates the average (AVG) of the selling_price for each year.
   b. The result is labeled as avg_selling_price.
2. **FROM car_dekho**:
   a. Specifies the source table (car_dekho) for the query.
3. **GROUP BY year**:
   a. Groups the rows by the year column, so that the average selling price is calculated for each year.

This query will return the average selling price of cars for each year in the dataset.

Output:

year	avg_selling_price
1994	72000.0000
1995	55000.0000
1996	41000.0000
1997	86111.0000
1998	57888.8889
1999	76928.5714
2000	76682.1250
2001	46500.0000
2002	103789.4211

Q.4 Identify the car model with the highest mileage among manual transmission cars.
Ans:

```
SELECT
   Name, mileage
FROM
   car_dekho
WHERE
   transmission = 'Manual'
ORDER BY CAST(SUBSTRING_INDEX(mileage, ' ', 1) AS DECIMAL (5 , 2 )) DESC
LIMIT 1;
```

Explanation:

1. **SELECT Name, mileage**: This retrieves the Name and mileage of the cars.
2. **FROM car_dekho**: Specifies the car_dekho table as the source.
3. **WHERE transmission = 'Manual'**: Filters the results to only include cars with manual transmission.
4. **ORDER BY CAST(SUBSTRING_INDEX(mileage, ' ', 1) AS DECIMAL(5,2)) DESC**:
    a. SUBSTRING_INDEX(mileage, ' ', 1) extracts the numeric part of the mileage field (assumed to be stored as something like '15.4 kmpl').
    b. CAST(... AS DECIMAL(5,2)) converts the extracted mileage into a decimal value.
    c. DESC sorts the result in descending order (highest mileage first).
5. **LIMIT 1**: Limits the result to only the car with the highest mileage.

This query will return the name and mileage of the manual transmission car with the highest mileage in the dataset.

Output:

	Name	mileage
▶	Maruti Alto 800 CNG LXI Optional	33.44 km/kg

Q.5 Calculate the total number of kilometers driven for cars sold by dealers.
Ans:

```
SELECT
   SUM(km_driven) AS total_km_driven
FROM
   car_dekho
WHERE
   seller_type = 'Dealer';
```

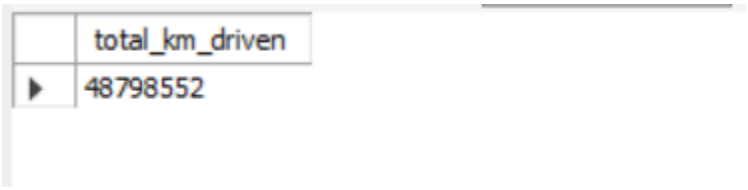
Explanation:

1. **SELECT SUM(km_driven) AS total_km_driven**:
    a. This calculates the total kilometers driven by summing up the km_driven values.
    b. The result is labeled as total_km_driven.
2. **FROM car_dekho**:
    a. Specifies the source table (car_dekho) for the query.

3. **WHERE seller_type = 'Dealer'**:
    a. Filters the results to include only those rows where the seller_type is 'Dealer'.

This query will return the total kilometers driven by cars that were sold by dealers in the dataset.

Output:



	total_km_driven
▶	48798552

Q.6 Determine the average engine capacity for cars with more than one previous owner.
Ans:

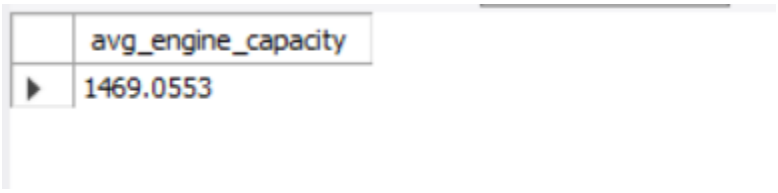
```
SELECT
   AVG(CAST(SUBSTRING_INDEX(engine, ' ', 1) AS UNSIGNED)) AS
avg_engine_capacity
FROM
   car_dekho
WHERE
   owner != 'First Owner';
```

Explanation:

1. **AVG(CAST(SUBSTRING_INDEX(engine, ' ', 1) AS UNSIGNED)) AS avg_engine_capacity**:
    a. SUBSTRING_INDEX(engine, ' ', 1) extracts the numeric part from the engine field (assuming it's in the format '1500 CC').
    b. CAST(... AS UNSIGNED) converts the extracted value to an integer.
    c. AVG(...) calculates the average engine capacity.
    d. The result is labeled as avg_engine_capacity.
2. **FROM car_dekho**: Specifies the table from which data is being queried.
3. **WHERE owner != 'First Owner'**: Filters the data to include only cars that are not owned by the first owner.

This query will return the average engine capacity of cars that are not first-owner vehicles.

Output:



	avg_engine_capacity
▶	1469.0553

Q. 7 Find the car model that has the highest torque in the dataset.

Ans:

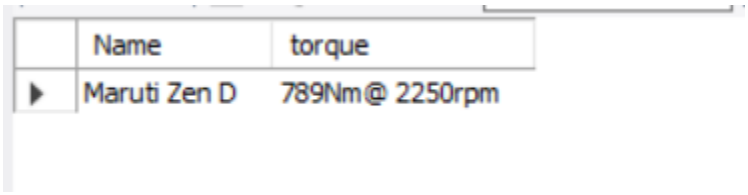
```
SELECT
   Name, torque
FROM
   car_dekho
ORDER BY CAST(SUBSTRING_INDEX(torque, 'Nm', 1) AS UNSIGNED) DESC
LIMIT 1;
```

Explanation:

1. **SELECT Name, torque**:
   a. Retrieves the Name and torque of the cars.
2. **FROM car_dekho**:
   a. Specifies the source table, car_dekho.
3. **ORDER BY CAST(SUBSTRING_INDEX(torque, 'Nm', 1) AS UNSIGNED) DESC**:
   a. SUBSTRING_INDEX(torque, 'Nm', 1) extracts the numeric part of the torque field (assuming it is stored in a format like "250Nm").
   b. CAST(... AS UNSIGNED) converts the extracted value into an unsigned integer.
   c. DESC sorts the result in descending order (highest torque first).
4. **LIMIT 1**:
   a. Limits the result to only the car with the highest torque.

This query will return the car with the highest torque in the dataset.

Output:



	Name	torque
▶	Maruti Zen D	789Nm@ 2250rpm

Q.8 List the top 3 most powerful cars (based on max_power) for each fuel type.
Ans:

```
SELECT
   fuel, Name, max_power
FROM
   car_dekho AS c1
WHERE
   (SELECT
        COUNT(*)
    FROM
        car_dekho AS c2
    WHERE
```

```
        c2.fuel = c1.fuel
          AND CAST(SUBSTRING_INDEX(c2.max_power, 'bhp', 1) AS DECIMAL (5 ,
2 )) >= CAST(SUBSTRING_INDEX(c1.max_power, 'bhp', 1) AS DECIMAL (5 , 2 ))) <=
3;
```

Explanation:

1. **Subquery**:
   a. For each row in car_dekho (referred to as c1), the subquery counts the number of cars with the same fuel type (c2.fuel = c1.fuel) and max_power greater than or equal to the current car's max_power.
   b. SUBSTRING_INDEX(c2.max_power, 'bhp', 1) extracts the numeric portion of max_power.
   c. CAST(... AS DECIMAL(5, 2)) converts the extracted value into a decimal for proper numerical comparison.
2. **WHERE Clause**:
   a. The condition (SELECT COUNT(*)) <= 3 ensures that only cars with a ranking of 3 or higher in terms of maximum power within each fuel type are returned.

This query will return up to the top 3 cars with the highest max_power for each fuel type.

Q.9 Calculate the total and average selling price of cars grouped by the number of seats.
Ans:

```
SELECT
   seats,
   SUM(selling_price) AS total_selling_price,
   AVG(selling_price) AS avg_selling_price
FROM
   car_dekho
GROUP BY seats;
```

Explanation:

1. **SELECT seats**:
   a. Selects the seats column, which is the number of seats in each car.
2. **SUM(selling_price) AS total_selling_price**:
   a. Calculates the total selling price of all cars that have the same number of seats.
   b. The result is labeled total_selling_price.
3. **AVG(selling_price) AS avg_selling_price**:
   a. Calculates the average selling price of all cars that have the same number of seats.

    b. The result is labeled avg_selling_price.
4. **FROM car_dekho**:
    a. Specifies the source table, car_dekho.
5. **GROUP BY seats**:
    a. Groups the results by the number of seats in the cars.

This query will return the total and average selling price of cars for each seating capacity.

Output:

seats	total_selling_price	avg_selling_price
2	1401000	700500.0000
4	64739700	486764.6617
5	3968136627	632473.1634
6	36059000	581596.7742
7	910473967	812923.1848
8	137755991	583711.8263
9	40005995	500074.9375
10	6537997	344105.1053
14	235000	235000.0000

Q .10 Identify cars that have a selling price higher than the average selling price of all cars in the dataset.

Ans:

```
SELECT
   Name, selling_price
FROM
   car_dekho
WHERE
   selling_price > (SELECT
         AVG(selling_price)
      FROM
         car_dekho);
```

Explanation:

1. **SELECT Name, selling_price**:
    a. Retrieves the Name and selling_price columns for each car.
2. **FROM car_dekho**:
    a. Specifies the source table, car_dekho.
3. **WHERE selling_price > (SELECT AVG(selling_price) FROM car_dekho)**:
    a. Filters the results to only include cars whose selling_price is greater than the average selling_price of all cars in the car_dekho table.

b. The subquery (SELECT AVG(selling_price) FROM car_dekho) calculates the average selling price.

This query will return all cars with a selling price above the average selling price in the dataset.

Output:

Name	selling_price
Skoda Slavia 1.0 TSI Ambition	1350000
BMW 3 Series Gran Limousine 320Ld Luxury Line	5800000
MG ZS EV Exclusive	2650000
Tata Punch Adventure	715000
Hyundai Creta SX Turbo	1895000
Renault Kiger RXT AMT Opt DT	842000
Mahindra XUV300 W8 Diesel Sunroof	1197000
Mahindra XUV700 AX5 Diesel AT	2275000
Renault Triber RXT	800000

Q. 11 Find the average mileage of petrol cars that are listed by individual sellers.
Ans:

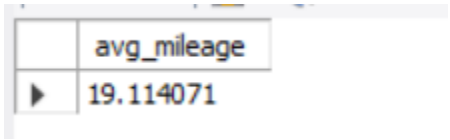
```
SELECT
   AVG(CAST(SUBSTRING_INDEX(mileage, ' ', 1) AS DECIMAL (5 , 2 ))) AS
avg_mileage
FROM
   car_dekho
WHERE
   fuel = 'Petrol'
      AND seller_type = 'Individual';
```

Explanation:

1. **SELECT AVG(CAST(SUBSTRING_INDEX(mileage, ' ', 1) AS DECIMAL(5, 2))) AS avg_mileage**:
   a. SUBSTRING_INDEX(mileage, ' ', 1) extracts the numeric part of the mileage field (assuming it's in a format like '15.4 kmpl').
   b. CAST(... AS DECIMAL(5, 2)) converts this numeric part into a decimal for accurate average calculation.
   c. AVG(...) calculates the average of these decimal values, and the result is labeled as avg_mileage.
2. **FROM car_dekho**:
   a. Specifies the table from which to retrieve the data.
3. **WHERE fuel = 'Petrol' AND seller_type = 'Individual'**:
   a. Filters the data to include only those cars with a fuel type of 'Petrol' and a seller_type of 'Individual'.

This query will return the average mileage of petrol cars sold by individual sellers.

Output:



	avg_mileage
▶	19.114071

Q. 12 Determine the distribution of cars by the type of transmission and fuel type.
Ans:

```
SELECT
   transmission, fuel, COUNT(*) AS total_cars
FROM
   car_dekho
GROUP BY transmission , fuel;
```

Explanation:

1. **SELECT transmission, fuel, COUNT(*) AS total_cars**:
   a. Selects the transmission and fuel columns.
   b. Counts the number of rows (cars) for each combination of transmission and fuel and labels this count as total_cars.
2. **FROM car_dekho**:
   a. Specifies the source table, car_dekho.
3. **GROUP BY transmission, fuel**:
   a. Groups the results by both transmission and fuel columns. This means that the count will be calculated for each unique combination of transmission and fuel.

This query will give you the total number of cars for each combination of transmission type and fuel type in the car_dekho table.

Output:

	transmission	fuel	total_cars
▶	Automatic	Diesel	3
	Automatic	Electric	1
	Automatic	Petrol	3
	Manual	Diesel	2
	Manual	Petrol	11
	Automatic	Diesel	532
	Automatic	Petrol	509
	Manual	CNG	53
	Manual	Diesel	3767

Q.13 Identify the car models that have the same engine capacity but differ in maximum power output.

Ans:

```
SELECT
   engine,
   GROUP_CONCAT(Name
      ORDER BY max_power
      SEPARATOR ', ') AS car_models
FROM
   car_dekho
GROUP BY engine
HAVING COUNT(DISTINCT max_power) > 1;
```

Explanation:

1. **SELECT engine, GROUP_CONCAT(Name ORDER BY max_power SEPARATOR ', ') AS car_models**:
   a. engine: Selects the engine type.
   b. GROUP_CONCAT(Name ORDER BY max_power SEPARATOR ', '): Concatenates the names of the cars with the same engine, ordered by max_power, and separates them with a comma and space. This is labeled as car_models.
2. **FROM car_dekho**:
   a. Specifies the source table, car_dekho.
3. **GROUP BY engine**:
   a. Groups the results by engine, so the GROUP_CONCAT function operates within each group of engine types.
4. **HAVING COUNT(DISTINCT max_power) > 1**:
   a. Filters the results to include only those engine types that have more than one distinct max_power value. This ensures that only engines with multiple power levels are included in the result.

This query will return a list of engine types and the corresponding car models that have varying max_power values for each engine type.

Output:

	engine	car_models
▶	1061 CC	Maruti Wagon R LXI DUO BSIII, Maruti Wagon ...
	1086 CC	Hyundai Santro Xing GL, Hyundai Santro Xing G...
	1120 CC	Hyundai Grand i10 CRDi Asta Option, Hyundai G...
	1150 CC	Chevrolet Aveo U-VA 1.2 LS, Chevrolet Aveo U...
	1172 CC	Tata Indica Vista Safire GLX, Tata Indica Vista A...
	1193 CC	Tata Indica V2 Emax CNG GLX, Tata Indigo CS e...
	1194 CC	Ford Freestyle Titanium Petrol BSIV, Ford Frees...
	1196 CC	Maruti Eeco CNG 5 Seater AC BSIV, Maruti Eeco...
	1197 CC	Volkswagen Polo GT TSI BSIV, Volkswagen Polo ...

Q.14 List the names and selling prices of the top 5 most recent cars (by year) that have been driven less than 10,000 km.

Ans:

```
SELECT
   Name, selling_price
FROM
   car_dekho
WHERE
   km_driven < 10000
ORDER BY year DESC
LIMIT 5;
```

Explanation:

1. **SELECT Name, selling_price**:
   a. Retrieves the Name and selling_price of the cars.
2. **FROM car_dekho**:
   a. Specifies the source table, car_dekho.
3. **WHERE km_driven < 10000**:
   a. Filters the results to include only those cars with less than 10,000 kilometers driven.
4. **ORDER BY year DESC**:
   a. Orders the results by the year column in descending order, so more recent cars appear first.
5. **LIMIT 5**:
   a. Limits the result to the top 5 rows.

This query will return the names and selling prices of the 5 most recent cars (by year) that have driven fewer than 10,000 kilometers.

Output:

	Name	selling_price
▶	BMW 3 Series Gran Limousine 320Ld Luxury Line	5800000
	Maruti S-Presso LXi	425000
	Renault Kiger RXT AMT Opt DT	842000
	Renault KWID CLIMBER	567000
	Mahindra XUV300 W8 Diesel Sunroof	1197000

Q. 15 Calculate the average selling price for each combination of fuel type and transmission type.

Ans:

```
SELECT
   fuel, transmission, AVG(selling_price) AS avg_selling_price
FROM
   car_dekho
GROUP BY fuel , transmission;
```

Explanation:

1. **SELECT fuel, transmission, AVG(selling_price) AS avg_selling_price**:
   a. **fuel**: Selects the fuel type.
   b. **transmission**: Selects the transmission type.
   c. **AVG(selling_price) AS avg_selling_price**: Calculates the average selling price for each combination of fuel and transmission. The result is labeled avg_selling_price.
2. **FROM car_dekho**:
   a. Specifies the source table, car_dekho.
3. **GROUP BY fuel, transmission**:
   a. Groups the results by both fuel and transmission, so the average selling price is calculated for each unique combination of these two columns.

This query will return the average selling price of cars for each combination of fuel type and transmission in the dataset.

Output:

	fuel	transmission	avg_selling_price
▶	CNG	Manual	313415.0377
	Diesel	Automatic	3341666.6667
	Diesel	Manual	1087000.0000
	Diesel	Automatic	2545533.8214
	Diesel	Manual	556280.7996
	Electric	Automatic	2650000.0000
	LPG	Manual	210885.7143
	Petrol	Automatic	1904000.0000
	Petrol	Manual	661545.4545

Q. 16 Find cars that have more than 5 seats and list them along with their selling price and engine capacity.

Ans:

```
SELECT
   Name, selling_price, engine
FROM
   car_dekho
WHERE
   seats > 5;
```

Explanation:

1. **SELECT Name, selling_price, engine**:
   a. Selects the Name, selling_price, and engine columns for each car.
2. **FROM car_dekho**:
   a. Specifies the source table, car_dekho.
3. **WHERE seats > 5**:
   a. Filters the results to include only cars with more than 5 seats.

This query will return the names, selling prices, and engine types of all cars in the car_dekho table that have more than 5 seats.

Output:

Name	selling_price	engine
BMW X7 xDrive 30d DPE	7200000	2993 CC
Mahindra KUV 100 D75 K6 Plus	480000	1198 CC
Mahindra KUV 100 D75 K8	550000	1198 CC
Mahindra Scorpio S11 4WD BSIV	1500000	2179 CC
Mahindra XUV500 W7	830000	2179 CC
Toyota Fortuner 2.8 4WD AT BSIV	3200000	2755 CC
Toyota Innova Crysta 2.4 G MT 8 STR	1560000	2393 CC
Toyota Innova Crysta 2.4 ZX AT	2300000	2393 CC
Toyota Innova Crysta 2.7 GX AT 8 STR	1500000	2694 CC

Q.17 Identify the least driven car for each year of manufacture.
Ans:

```
SELECT
   c1.year, c1.Name, c1.km_driven
FROM
   car_dekho c1
WHERE
   c1.km_driven = (SELECT
         MIN(c2.km_driven)
      FROM
         car_dekho c2
      WHERE
         c2.year = c1.year);
```

Explanation:

1. **SELECT c1.year, c1.Name, c1.km_driven**:
   a. Selects the year, Name, and km_driven of the cars.
2. **FROM car_dekho c1**:
   a. Specifies the source table, car_dekho, with an alias c1.
3. **WHERE c1.km_driven = (SELECT MIN(c2.km_driven) FROM car_dekho c2 WHERE c2.year = c1.year)**:
   a. The subquery (SELECT MIN(c2.km_driven) FROM car_dekho c2 WHERE c2.year = c1.year) finds the minimum km_driven for the same year as the current row in c1.
   b. The WHERE clause then filters the results to include only those cars where km_driven matches this minimum value for each year.

This query will return the names, years, and kilometers driven of the cars that have the least amount of kilometers driven for each year in the dataset.

Q.18 List all cars with a mileage greater than the average mileage of diesel cars.

Ans:

```
SELECT
   Name, mileage
FROM
   car_dekho
WHERE
   CAST(SUBSTRING_INDEX(mileage, ' ', 1) AS DECIMAL (5 , 2 )) > (SELECT
         AVG(CAST(SUBSTRING_INDEX(mileage, ' ', 1) AS DECIMAL (5 , 2 )))
      FROM
         car_dekho
      WHERE
         fuel = 'Diesel');
```

Explanation:

1. **SELECT Name, mileage**:
   a. Retrieves the Name and mileage columns of the cars.
2. **FROM car_dekho**:
   a. Specifies the source table, car_dekho.
3. **WHERE CAST(SUBSTRING_INDEX(mileage, ' ', 1) AS DECIMAL(5, 2)) > (...)**:
   a. SUBSTRING_INDEX(mileage, ' ', 1) extracts the numeric part of the mileage field (assuming it is in a format like '15.4 kmpl').
   b. CAST(... AS DECIMAL(5, 2)) converts this part to a decimal for numerical comparison.
   c. The WHERE clause filters the results to include only those cars where this mileage is greater than the average mileage of diesel cars.
4. **Subquery**:
   a. **SELECT AVG(CAST(SUBSTRING_INDEX(mileage, ' ', 1) AS DECIMAL(5, 2)))**:
      i. Computes the average mileage for diesel cars.
   b. **FROM car_dekho WHERE fuel = 'Diesel'**:
      i. Filters the dataset to consider only diesel cars.

This query will return the names and mileage of all cars with a mileage higher than the average mileage of diesel cars.

Output:

	Name	mileage
▶	MG ZS EV Exclusive	32.52 kmpl
	Mahindra XUV300 W8 Diesel Sunroof	32.52 kmpl
	Renault Triber RXT	21.01 kmpl
	Nissan Magnite XV Premium	32.52 kmpl
	Hyundai Tucson Platinum AT	21.01 kmpl
	Datsun RediGO 1.0 S	22.5 kmpl
	Honda Civic ZX Diesel BSIV	26.8 kmpl
	Honda Civic ZX Diesel BSIV	26.8 kmpl
	Hyundai Creta 1.4 EX Diesel	22.1 kmpl

Q. 19 Determine the correlation between engine capacity and selling price.
Ans:

```
SELECT
   (SUM(xy) - (SUM(x) * SUM(y)) / COUNT(*)) / (SQRT(SUM(x_squared) - (SUM(x) *
SUM(x)) / COUNT(*)) * SQRT(SUM(y_squared) - (SUM(y) * SUM(y)) / COUNT(*))) AS
correlation

FROM
   (SELECT
      CAST(SUBSTRING_INDEX(engine, ' ', 1) AS UNSIGNED) AS x,
         selling_price AS y,
         CAST(SUBSTRING_INDEX(engine, ' ', 1) AS UNSIGNED) * selling_price AS xy,
         CAST(SUBSTRING_INDEX(engine, ' ', 1) AS UNSIGNED) *
CAST(SUBSTRING_INDEX(engine, ' ', 1) AS UNSIGNED) AS x_squared,
         selling_price * selling_price AS y_squared
   FROM
      car_dekho) AS subquery;
```

Explanation:

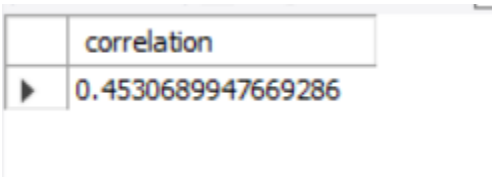
1. **Inner Subquery**:
   a. **CAST(SUBSTRING_INDEX(engine, ' ', 1) AS UNSIGNED) AS x**:
      i. Extracts and converts the numeric part of the engine capacity into an integer and aliases it as x.
   b. **selling_price AS y**:
      i. Uses the selling_price directly and aliases it as y.
   c. **CAST(SUBSTRING_INDEX(engine, ' ', 1) AS UNSIGNED) * selling_price AS xy**:
      i. Calculates the product of x and y, which will be used to compute the sum of xy.

d. **CAST(SUBSTRING_INDEX(engine, ' ', 1) AS UNSIGNED) * CAST(SUBSTRING_INDEX(engine, ' ', 1) AS UNSIGNED) AS x_squared**:
    i. Computes the square of x and aliases it as x_squared.
e. **selling_price * selling_price AS y_squared**:
    i. Computes the square of y and aliases it as y_squared.

2. **Outer Query**:
    a. **Correlation Formula**:
        i. **(SUM(xy) - (SUM(x) * SUM(y)) / COUNT(*))**:
            1. Computes the numerator of the Pearson correlation coefficient formula.
        ii. **SQRT(SUM(x_squared) - (SUM(x) * SUM(x)) / COUNT(*))**:
            1. Computes the standard deviation of x (denominator part).
        iii. **SQRT(SUM(y_squared) - (SUM(y) * SUM(y)) / COUNT(*))**:
            1. Computes the standard deviation of y (denominator part).
    b. **The full formula**:
        i. **correlation = (SUM(xy) - (SUM(x) * SUM(y)) / COUNT(*)) / (SQRT(SUM(x_squared) - (SUM(x) * SUM(x)) / COUNT(*)) * SQRT(SUM(y_squared) - (SUM(y) * SUM(y)) / COUNT(*)))**:
            1. Calculates the Pearson correlation coefficient, which measures the strength and direction of the linear relationship between x (engine capacity) and y (selling price).

This query computes the Pearson correlation coefficient between engine capacity and selling price based on the data in the car_dekho table.

Output:



	correlation
▶	0.4530689947669286

Q.20 Find the maximum, minimum, and average kilometers driven by cars with automatic transmission.

Ans:

```
SELECT
    MAX(km_driven) AS max_km_driven,
    MIN(km_driven) AS min_km_driven,
    AVG(km_driven) AS avg_km_driven
FROM
    car_dekho
WHERE
    transmission = 'Automatic';
```

Explanation:

1. **SELECT MAX(km_driven) AS max_km_driven**:
    a. Retrieves the maximum value of km_driven from the cars where the transmission is 'Automatic'.
    b. Labels this value as max_km_driven.
2. **MIN(km_driven) AS min_km_driven**:
    a. Retrieves the minimum value of km_driven from the same subset.
    b. Labels this value as min_km_driven.
3. **AVG(km_driven) AS avg_km_driven**:
    a. Calculates the average value of km_driven from the subset of cars with 'Automatic' transmission.
    b. Labels this value as avg_km_driven.
4. **FROM car_dekho**:
    a. Specifies the source table, car_dekho.
5. **WHERE transmission = 'Automatic'**:
    a. Filters the data to include only cars with 'Automatic' transmission.

This query will return the maximum, minimum, and average kilometers driven for cars in the car_dekho table that have an automatic transmission.

Output:

	max_km_driven	min_km_driven	avg_km_driven
▶	230000	1000	39849.0788

Conclusion

These analyses collectively offer valuable insights into car listings, including pricing, performance, and market trends, assisting both buyers and sellers in making informed decisions.