

Query 1:

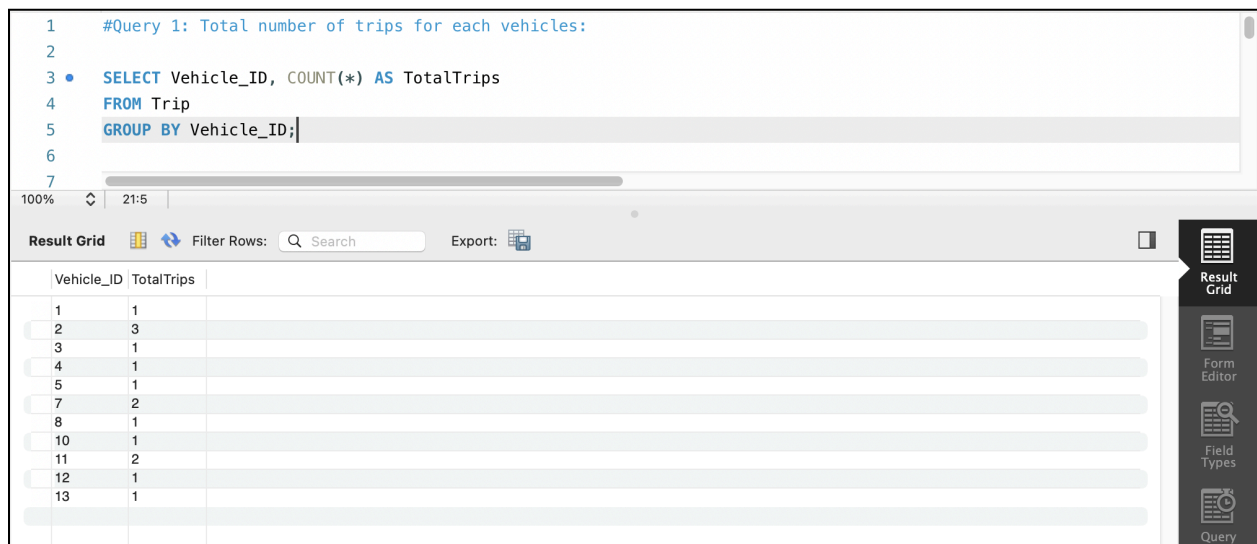
Query statement:

Total number of trips for each vehicle:

SQL command:

```
SELECT Vehicle_ID, COUNT(*) AS TotalTrips
FROM Trip
GROUP BY Vehicle_ID;
```

Output:



The screenshot shows a database query editor interface. At the top, the SQL command is entered in a text area. Below the text area, there is a toolbar with options like 'Result Grid', 'Filter Rows', and 'Export'. The main part of the interface displays the results of the query in a table. The table has two columns: 'Vehicle_ID' and 'TotalTrips'. The results are as follows:

Vehicle_ID	TotalTrips
1	1
2	3
3	1
4	1
5	1
7	2
8	1
10	1
11	2
12	1
13	1

Query 2:

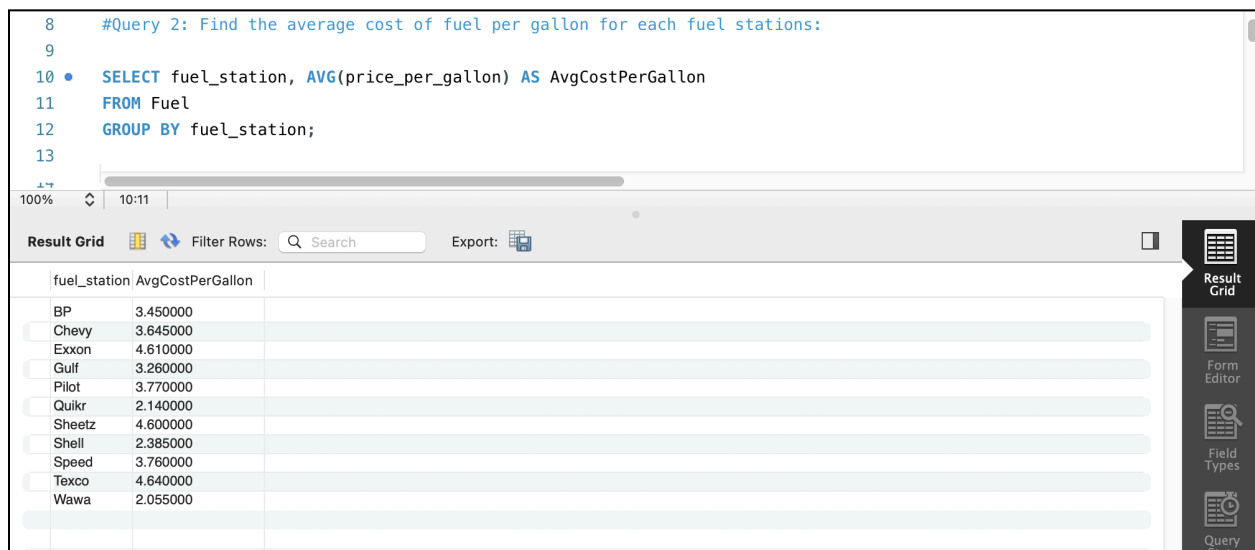
Query statement:

Find the average cost of fuel per gallon for each fuel station:

SQL command:

```
SELECT fuel_station, AVG(price_per_gallon) AS AvgCostPerGallon
FROM Fuel
GROUP BY fuel_station;
```

Output:



The screenshot shows a database query editor interface. The top section contains the SQL query: `#Query 2: Find the average cost of fuel per gallon for each fuel stations:`, `SELECT fuel_station, AVG(price_per_gallon) AS AvgCostPerGallon`, `FROM Fuel`, and `GROUP BY fuel_station;`. Below the query editor is a toolbar with options for 'Result Grid', 'Filter Rows', 'Search', and 'Export'. The 'Result Grid' is active, displaying a table with two columns: 'fuel_station' and 'AvgCostPerGallon'. The table lists ten fuel stations with their corresponding average costs per gallon. On the right side of the interface, there is a vertical sidebar with icons for 'Result Grid', 'Form Editor', 'Field Types', and 'Query'.

fuel_station	AvgCostPerGallon
BP	3.450000
Chevy	3.645000
Exxon	4.610000
Gulf	3.260000
Pilot	3.770000
Quikr	2.140000
Sheetz	4.600000
Shell	2.385000
Speed	3.760000
Texco	4.640000
Wawa	2.055000

Query 3:

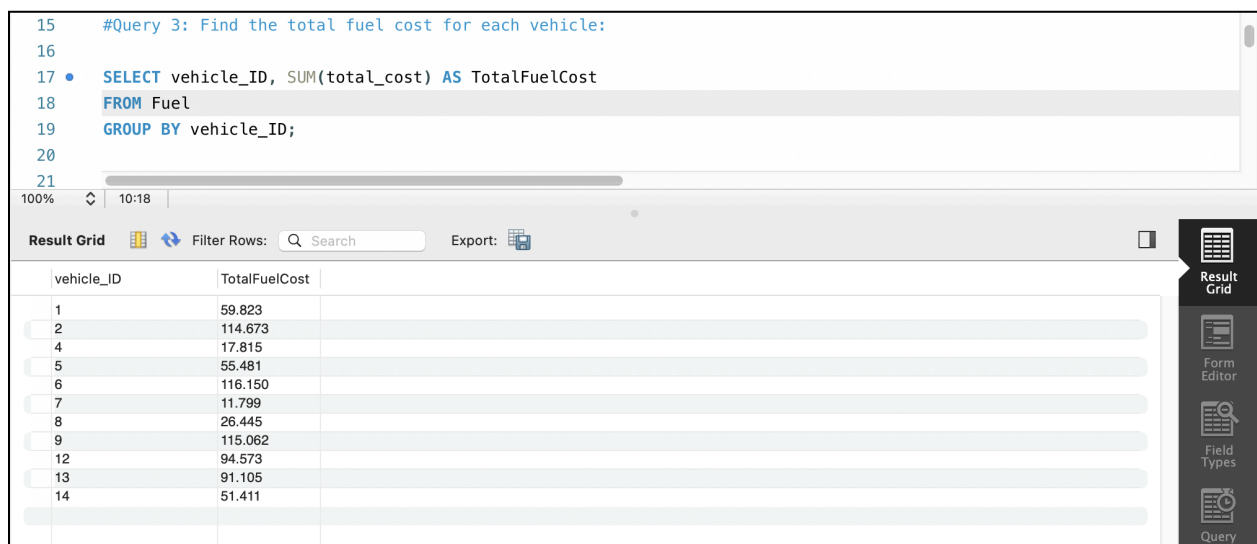
Query statement:

Find the total fuel cost for each vehicle:

SQL command:

```
SELECT vehicle_ID, SUM(total_cost) AS TotalFuelCost
FROM Fuel
GROUP BY vehicle_ID;
```

Output:



The screenshot shows a database query tool interface. At the top, a text area contains the SQL query: `#Query 3: Find the total fuel cost for each vehicle:`, `SELECT vehicle_ID, SUM(total_cost) AS TotalFuelCost`, `FROM Fuel`, and `GROUP BY vehicle_ID;`. Below the query area, a toolbar includes a 'Result Grid' button, a 'Filter Rows' search bar, and an 'Export' button. The main area displays a table with two columns: 'vehicle_ID' and 'TotalFuelCost'. The table contains 14 rows of data. On the right side, a vertical toolbar contains icons for 'Result Grid', 'Form Editor', 'Field Types', and 'Query'.

vehicle_ID	TotalFuelCost
1	59.823
2	114.673
4	17.815
5	55.481
6	116.150
7	11.799
8	26.445
9	115.062
12	94.573
13	91.105
14	51.411

Query 4:

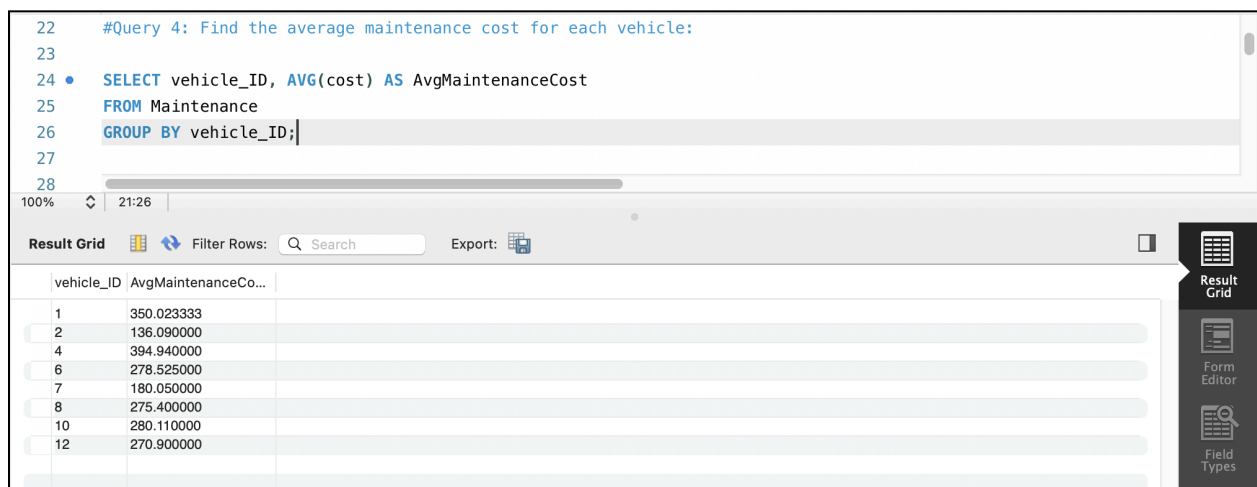
Query statement:

Find the average maintenance cost for each vehicle:

SQL command:

```
SELECT vehicle_ID, AVG(cost) AS AvgMaintenanceCost
FROM Maintenance
GROUP BY vehicle_ID;
```

Output:



The screenshot shows a database query editor interface. The top section contains the SQL query: `#Query 4: Find the average maintenance cost for each vehicle:`, `SELECT vehicle_ID, AVG(cost) AS AvgMaintenanceCost`, `FROM Maintenance`, and `GROUP BY vehicle_ID;`. Below the query editor, there is a toolbar with options like 'Filter Rows', 'Search', and 'Export'. The main area displays the 'Result Grid' with the following data:

vehicle_ID	AvgMaintenanceCo...
1	350.023333
2	136.090000
4	394.940000
6	278.525000
7	180.050000
8	275.400000
10	280.110000
12	270.900000

On the right side of the interface, there are icons for 'Result Grid', 'Form Editor', and 'Field Types'.

Query 5:

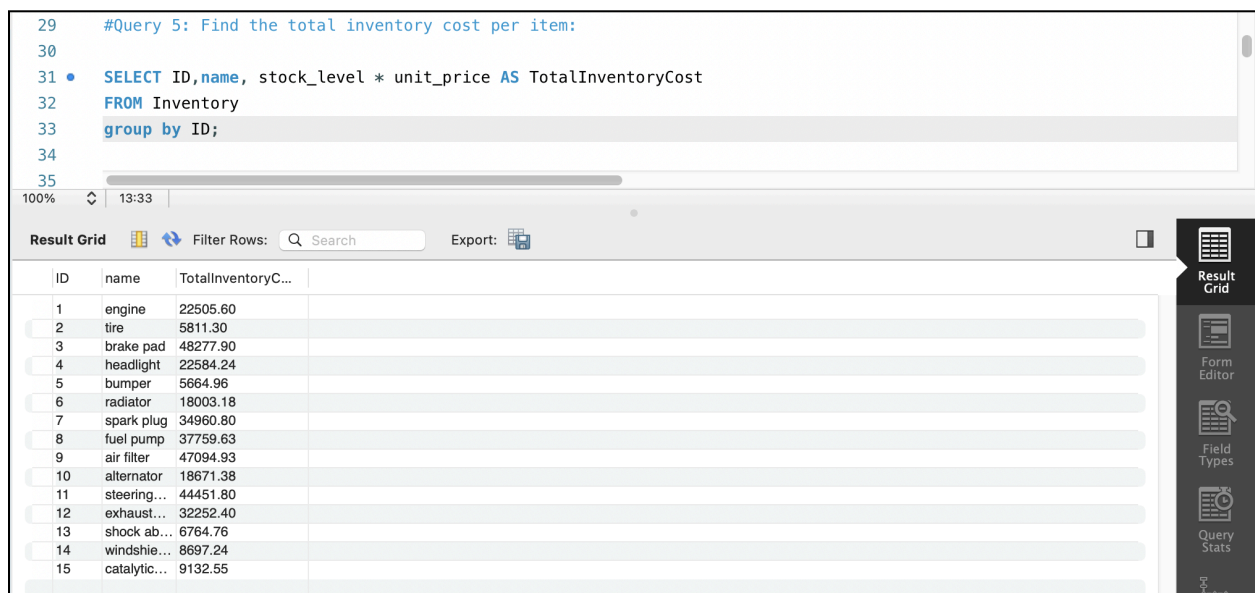
Query statement:

Find the total inventory cost per item:

SQL command:

```
SELECT ID, name, stock_level * unit_price AS TotalInventoryCost
FROM Inventory
group by ID;
```

Output:



The screenshot shows a database query interface. At the top, a text area contains the SQL query: `#Query 5: Find the total inventory cost per item: SELECT ID,name, stock_level * unit_price AS TotalInventoryCost FROM Inventory group by ID;`. Below the text area is a toolbar with options like 'Filter Rows', 'Search', and 'Export'. The main area displays a 'Result Grid' with 15 rows of data. The columns are 'ID', 'name', and 'TotalInventoryC...'. The data is as follows:

ID	name	TotalInventoryC...
1	engine	22505.60
2	tire	5811.30
3	brake pad	48277.90
4	headlight	22584.24
5	bumper	5664.96
6	radiator	18003.18
7	spark plug	34960.80
8	fuel pump	37759.63
9	air filter	47094.93
10	alternator	18671.38
11	steering...	44451.80
12	exhaust...	32252.40
13	shock ab...	6764.76
14	windshie...	8697.24
15	catalytic...	9132.55

On the right side of the interface, there is a vertical toolbar with icons for 'Result Grid', 'Form Editor', 'Field Types', and 'Query Stats'.

Query 6:

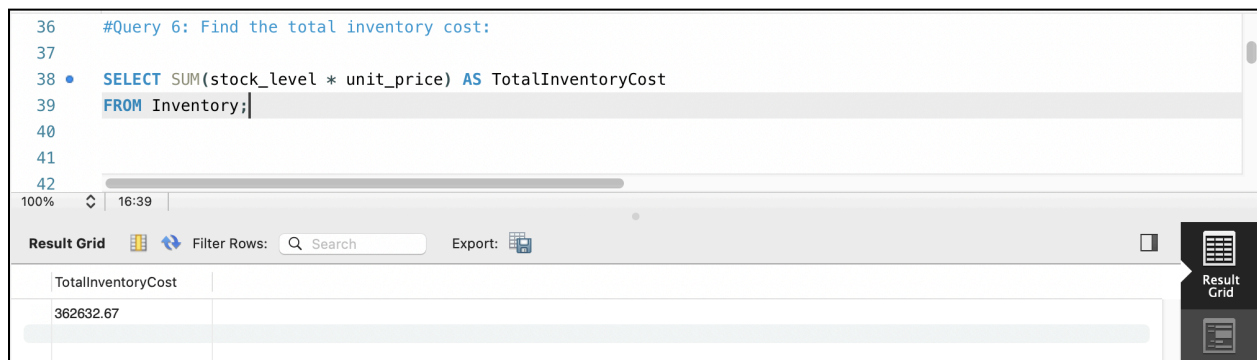
Query statement:

Find the total inventory cost:

SQL command:

```
SELECT SUM(stock_level * unit_price) AS TotalInventoryCost
FROM Inventory;
```

Output:



The screenshot shows a SQL query editor with the following text:

```
36 #Query 6: Find the total inventory cost:
37
38 • SELECT SUM(stock_level * unit_price) AS TotalInventoryCost
39 FROM Inventory;
40
41
42
```

Below the editor is a toolbar with a zoom level of 100%, a search bar, and an export button. The result grid below the toolbar shows the output of the query:

TotalInventoryCost
362632.67

On the right side of the result grid, there is a vertical toolbar with a 'Result Grid' button and a 'Result Grid' icon.

Query 7:

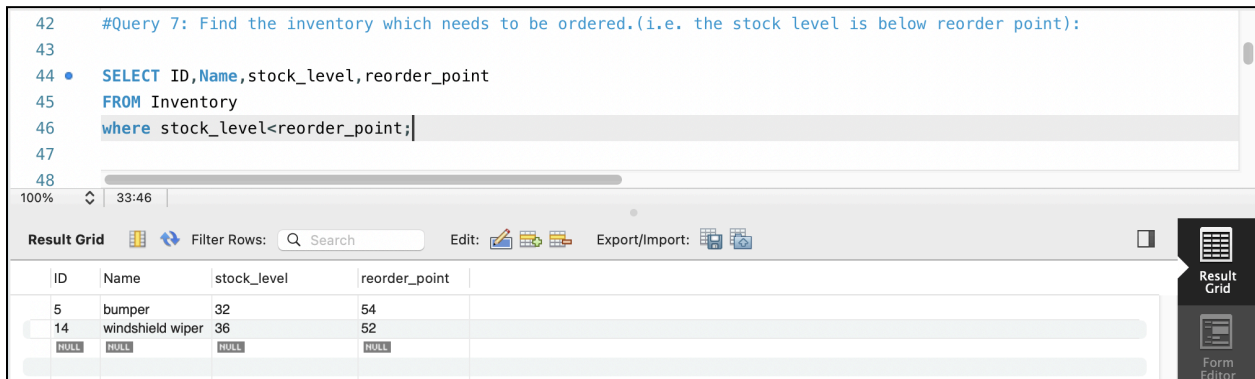
Query statement:

Find the inventory that needs to be ordered. (i.e. the stock level is below the reorder point):

SQL command:

```
SELECT ID, Name, stock_level, reorder_point
FROM Inventory
where stock_level < reorder_point;
```

Output:



The screenshot shows a database query editor interface. The top section displays the SQL query: `#Query 7: Find the inventory which needs to be ordered.(i.e. the stock level is below reorder point):`, `SELECT ID, Name, stock_level, reorder_point`, `FROM Inventory`, and `where stock_level < reorder_point;`. Below the query editor, a toolbar includes options for 'Result Grid', 'Filter Rows', 'Search', 'Edit', and 'Export/Import'. The 'Result Grid' is active, showing a table with the following data:

ID	Name	stock_level	reorder_point
5	bumper	32	54
14	windshield wiper	36	52
NULL	NULL	NULL	NULL

On the right side of the interface, there are icons for 'Result Grid' and 'Form Editor'.

Query 8:

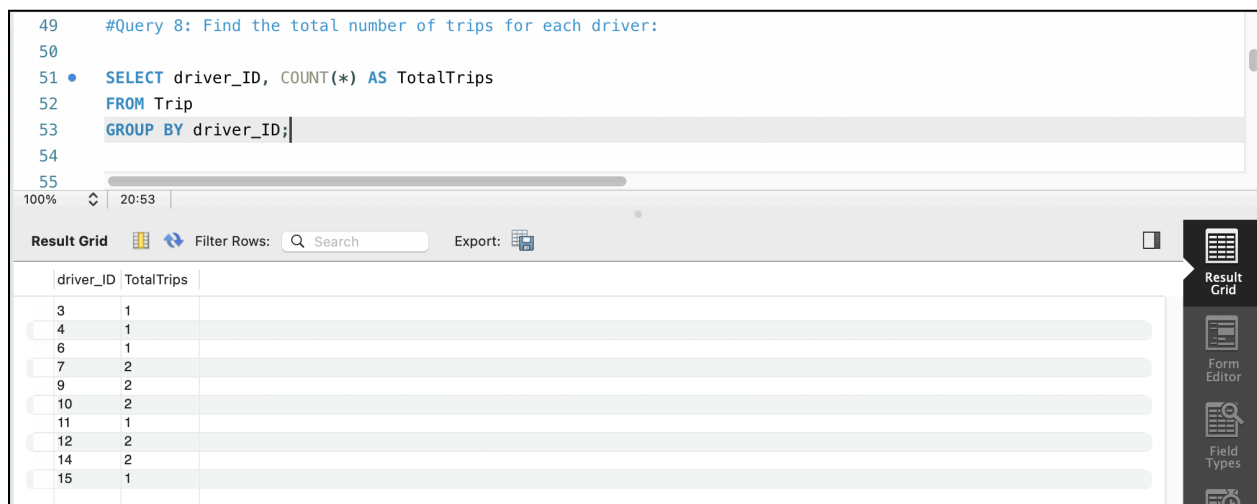
Query statement:

Find the total number of trips for each driver:

SQL command:

```
SELECT driver_ID, COUNT(*) AS TotalTrips
FROM Trip
GROUP BY driver_ID;
```

Output:



The screenshot shows a SQL query editor with the following code:

```
49 #Query 8: Find the total number of trips for each driver:
50
51 • SELECT driver_ID, COUNT(*) AS TotalTrips
52 FROM Trip
53 GROUP BY driver_ID;
54
55
```

Below the editor is a toolbar with a zoom level of 100%, a clock showing 20:53, and buttons for Filter Rows, Search, and Export. The main area displays a result grid with the following data:

driver_ID	TotalTrips
3	1
4	1
6	1
7	2
9	2
10	2
11	1
12	2
14	2
15	1

On the right side of the result grid, there is a vertical toolbar with icons for Result Grid, Form Editor, Field Types, and a refresh icon.

Query 9:

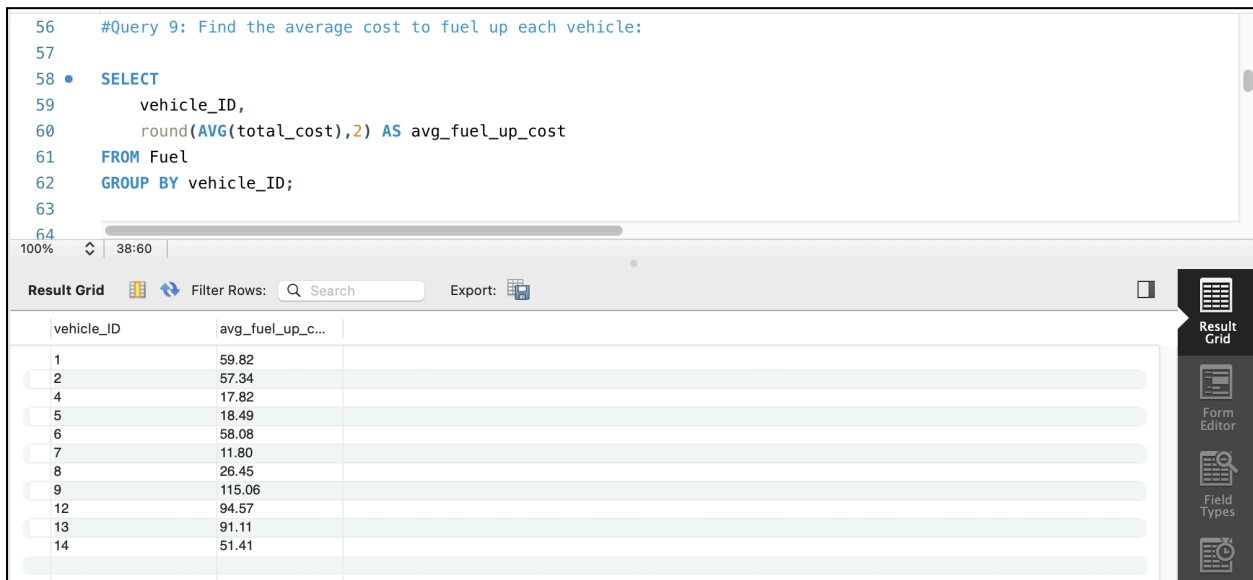
Query statement:

Find the average cost to fuel up each vehicle:

SQL command:

```
SELECT
    vehicle_ID,
    round(AVG(total_cost),2) AS avg_fuel_up_cost
FROM Fuel
GROUP BY vehicle_ID;
```

Output:



The screenshot shows a database query editor interface. The top pane contains the SQL query for Query 9. The bottom pane displays the results in a table format. The table has two columns: 'vehicle_ID' and 'avg_fuel_up_c...'. The results are as follows:

vehicle_ID	avg_fuel_up_c...
1	59.82
2	57.34
4	17.82
5	18.49
6	58.08
7	11.80
8	26.45
9	115.06
12	94.57
13	91.11
14	51.41

Query 10: (Duplicate) (Same as no.8)

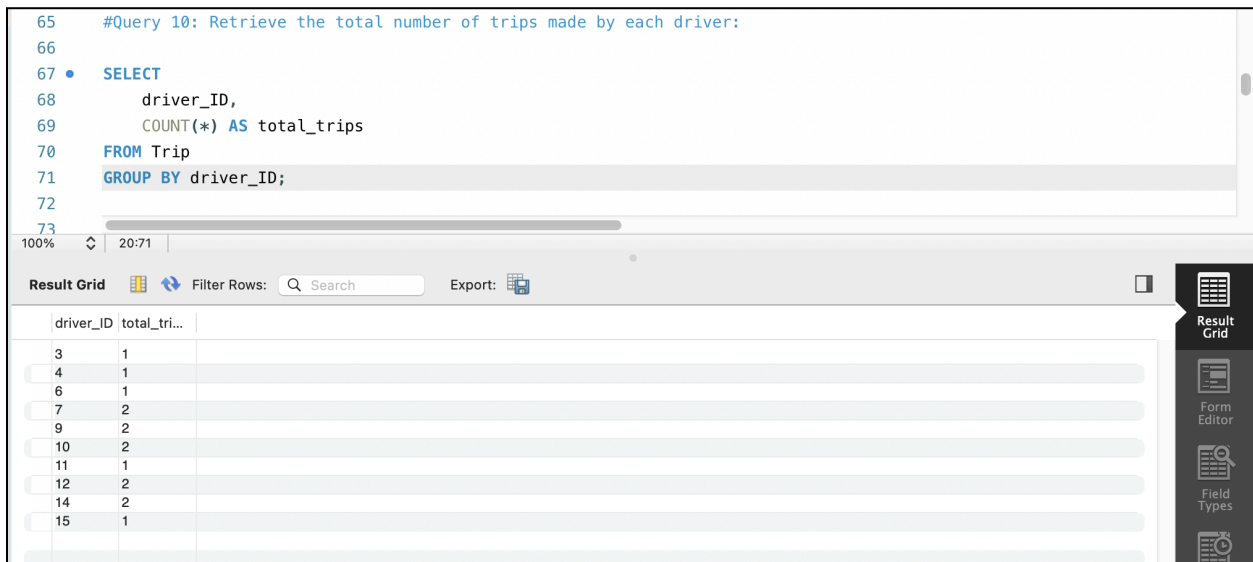
Query statement:

Retrieve the total number of trips made by each driver:

SQL command:

```
SELECT
    driver_ID,
    COUNT(*) AS total_trips
FROM Trip
GROUP BY driver_ID;
```

Output:



The screenshot shows a database query editor interface. The top pane displays the SQL query for Query 10. The bottom pane shows the results in a table format. The table has two columns: 'driver_ID' and 'total_trips'. The results are as follows:

driver_ID	total_trips
3	1
4	1
6	1
7	2
9	2
10	2
11	1
12	2
14	2
15	1

Query 11:

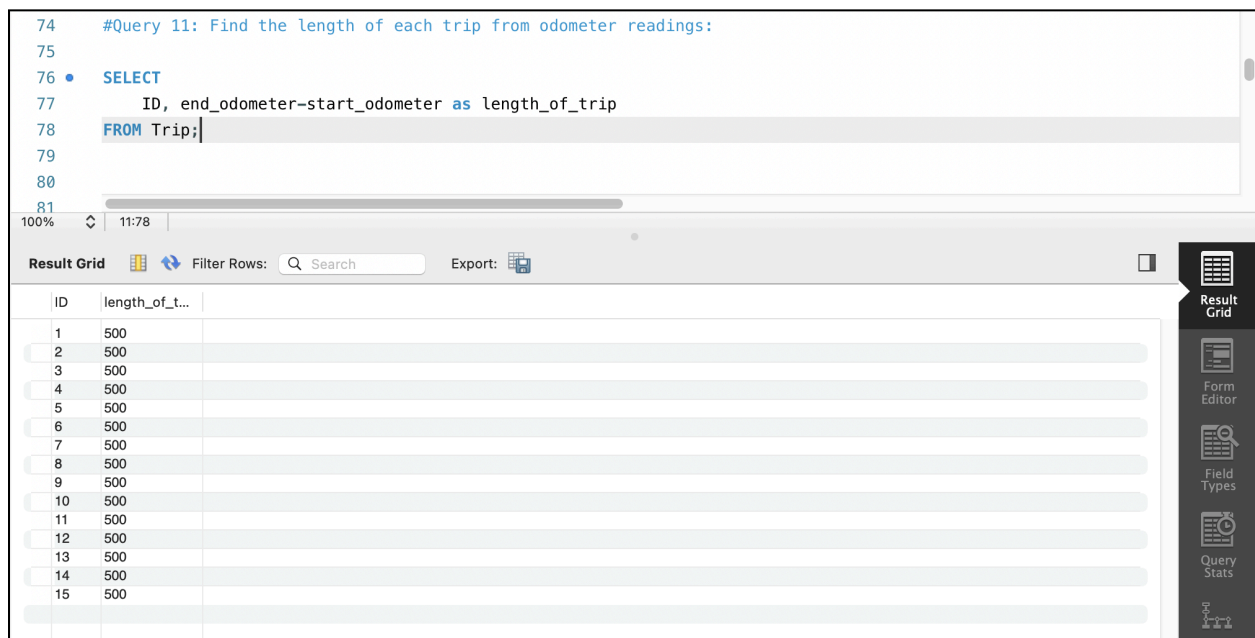
Query statement:

Find the length of each trip from odometer readings:

SQL command:

```
SELECT
    ID, end_odometer-start_odometer as length_of_trip
FROM Trip;
```

Output:



The screenshot shows a database query editor interface. The top section displays the SQL query: `SELECT ID, end_odometer-start_odometer as length_of_trip FROM Trip;`. Below the query editor, the 'Result Grid' is visible, showing a table with two columns: 'ID' and 'length_of_t...'. The table contains 15 rows, all with a value of 500 in the 'length_of_t...' column. The interface includes a search bar, an export button, and a sidebar with icons for 'Result Grid', 'Form Editor', 'Field Types', and 'Query Stats'.

ID	length_of_t...
1	500
2	500
3	500
4	500
5	500
6	500
7	500
8	500
9	500
10	500
11	500
12	500
13	500
14	500
15	500

“Unusual result, because our data is in a way that all trips have the same length. Practical/real data can give different results.”

Query 12:

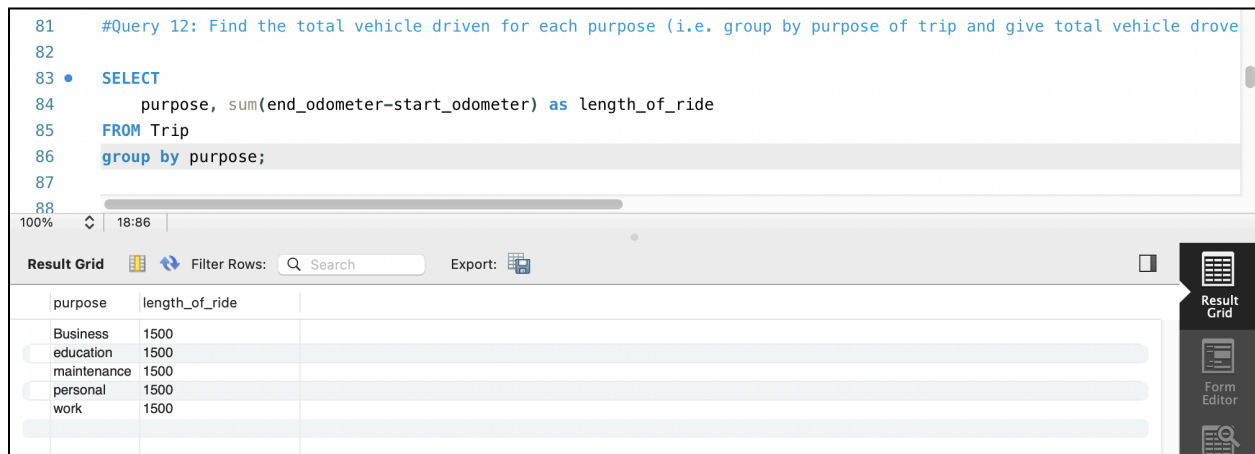
Query statement:

Find the total vehicle driven for each purpose (i.e. group by the purpose of the trip and give the total vehicle driven for that particular task):

SQL command:

```
SELECT
    purpose, sum(end_odometer-start_odometer) as length_of_ride
FROM Trip
group by purpose;
```

Output:



```
81  #Query 12: Find the total vehicle driven for each purpose (i.e. group by purpose of trip and give total vehicle drove
82
83  • SELECT
84      purpose, sum(end_odometer-start_odometer) as length_of_ride
85  FROM Trip
86  group by purpose;
87
88
```

100% 18:86

Result Grid Filter Rows: Search Export:

	purpose	length_of_ride
	Business	1500
	education	1500
	maintenance	1500
	personal	1500
	work	1500

Result Grid Form Editor

“Unusual result, because our data is in a way that all-purpose have the same number of trips and same number of trip length.

Practical/real data can give different results.”

Query 13:

Query statement:

List vehicles that have not had any maintenance yet

SQL command:

```
SELECT
    v.ID,
    v.VIN,
    v.model,
    v.year,
    v.make
FROM Vehicle v
LEFT JOIN Maintenance m ON v.ID = m.vehicle_ID
WHERE m.ID IS NULL;
```

Output:

#Query 13: List vehicles that have not had any maintenance yet

90

91 • SELECT

92 v.ID,

93 v.VIN,

94 v.model,

95 v.year,

96 v.make

97 FROM Vehicle v

98 LEFT JOIN Maintenance m ON v.ID = m.vehicle_ID

99 WHERE m.ID IS NULL;



100

100%


↕


20:99

Result Grid




Filter Rows:


Export: 




ID	VIN	model	year	make	
3	WAUED54B41N732457	300E	1993	Mercedes-Benz	
5	1G6AF5S32D0183837	7 Series	2012	BMW	
9	WBAVD53587A032582	C30	2009	Volvo	
11	5NPEB4ACXD078908	Xterra	2002	Nissan	
13	KNAFT4A22A5490080	3 Series	2007	BMW	
14	WP1AE2A2XBL100797	RDX	2012	Acura	
15	WDDHF5GB9AA337323	1500	1997	Chevrolet	



Result Grid



Form Editor



Query 14:

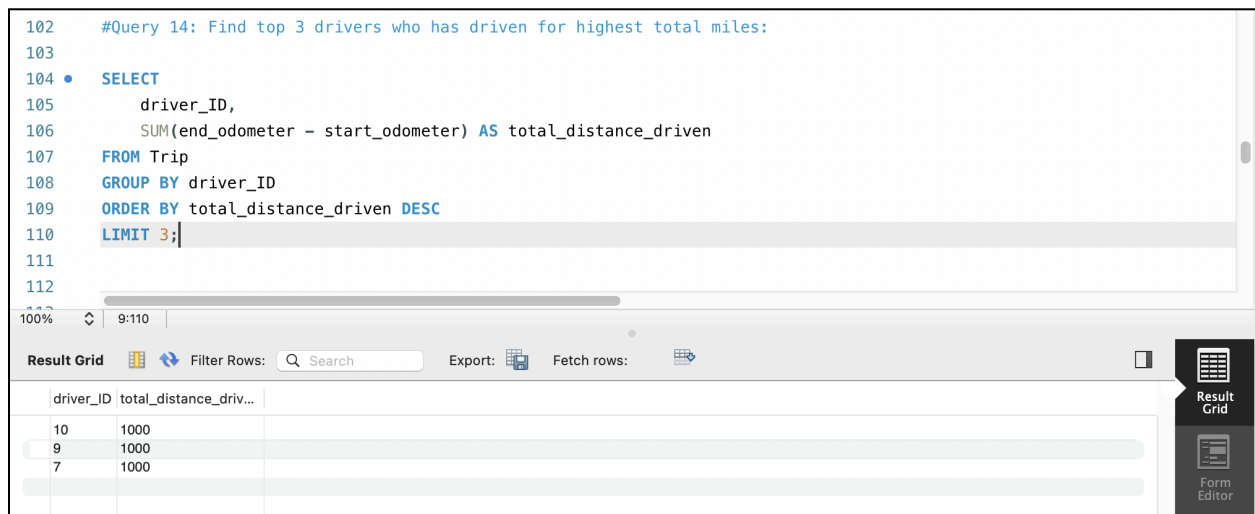
Query statement:

Find the top 3 drivers who have driven for the highest total miles:

SQL command:

```
SELECT
    driver_ID,
    SUM(end_odometer - start_odometer) AS total_distance_driven
FROM Trip
GROUP BY driver_ID
ORDER BY total_distance_driven DESC
LIMIT 3;
```

Output:



The screenshot displays a SQL query editor with the following code:

```
102 #Query 14: Find top 3 drivers who has driven for highest total miles:
103
104 • SELECT
105     driver_ID,
106     SUM(end_odometer - start_odometer) AS total_distance_driven
107 FROM Trip
108 GROUP BY driver_ID
109 ORDER BY total_distance_driven DESC
110 LIMIT 3;
111
112
```

Below the editor, the 'Result Grid' shows the output of the query. The grid has two columns: 'driver_ID' and 'total_distance_driv...'. The results are as follows:

driver_ID	total_distance_driv...
10	1000
9	1000
7	1000

The interface also includes a search bar, an 'Export' button, and a 'Fetch rows' button. A sidebar on the right contains a 'Result Grid' icon and a 'Form Editor' icon.

Query 15:

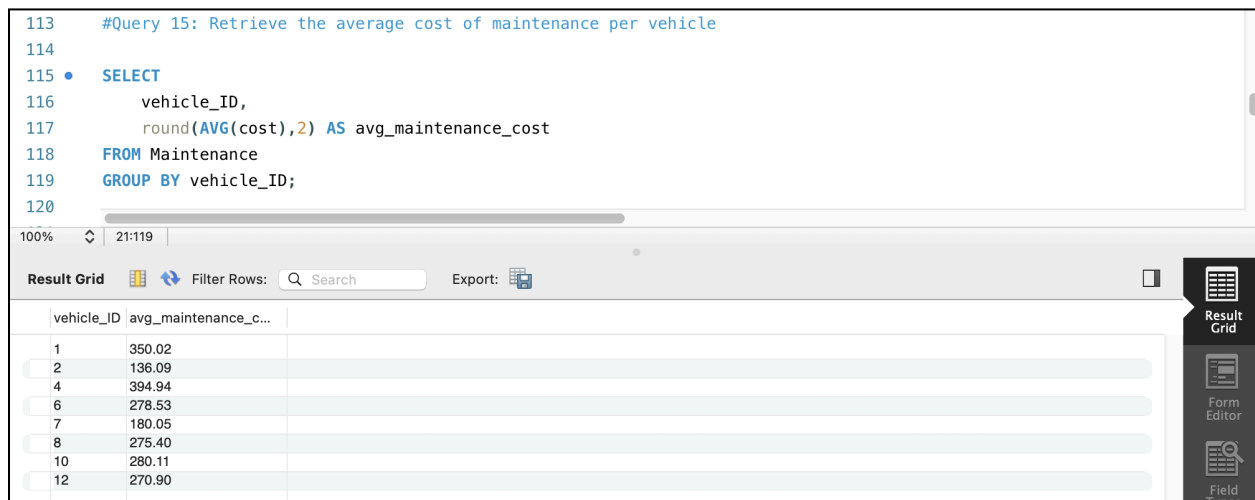
Query statement:

Retrieve the average cost of maintenance per vehicle

SQL command:

```
SELECT
    vehicle_ID,
    round(AVG(cost),2) AS avg_maintenance_cost
FROM Maintenance
GROUP BY vehicle_ID;
```

Output:



The screenshot shows a database query editor interface. The top pane displays the SQL query for Query 15. The bottom pane shows the result grid with 12 rows of data. The interface includes a search bar, an export button, and a sidebar with options for Result Grid, Form Editor, and Field Types.

vehicle_ID	avg_maintenance_c...
1	350.02
2	136.09
4	394.94
6	278.53
7	180.05
8	275.40
10	280.11
12	270.90

Query 16:

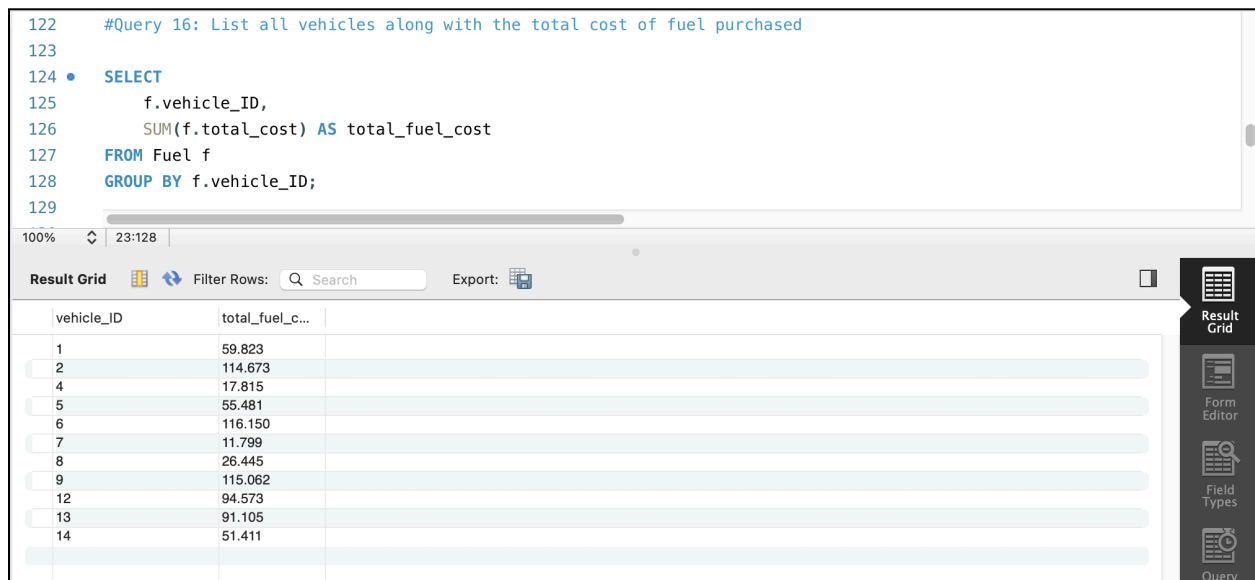
Query statement:

List all vehicles along with the total cost of fuel purchased

SQL command:

```
SELECT
    f.vehicle_ID,
    SUM(f.total_cost) AS total_fuel_cost
FROM Fuel f
GROUP BY f.vehicle_ID;
```

Output:



The screenshot shows a database query editor interface. The top pane displays the SQL query for Query 16. The bottom pane shows the 'Result Grid' with the query's output. The interface includes a toolbar with options like 'Filter Rows', 'Search', and 'Export'. A sidebar on the right contains icons for 'Result Grid', 'Form Editor', 'Field Types', and 'Query'.

```
122 #Query 16: List all vehicles along with the total cost of fuel purchased
123
124 • SELECT
125     f.vehicle_ID,
126     SUM(f.total_cost) AS total_fuel_cost
127 FROM Fuel f
128 GROUP BY f.vehicle_ID;
129
```

vehicle_ID	total_fuel_c...
1	59.823
2	114.673
4	17.815
5	55.481
6	116.150
7	11.799
8	26.445
9	115.062
12	94.573
13	91.105
14	51.411

Query 17:

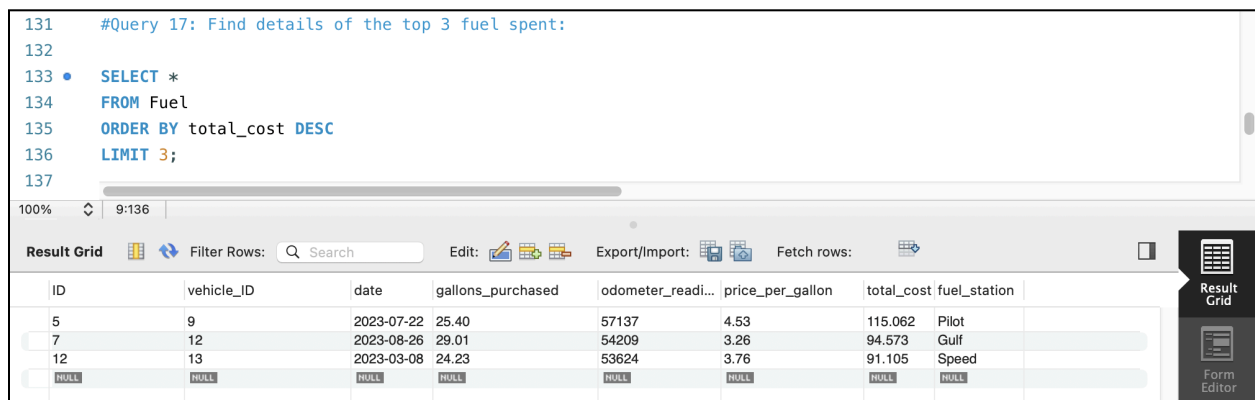
Query statement:

Find details of the top 3 fuel spent:

SQL command:

```
SELECT *  
FROM Fuel  
ORDER BY total_cost DESC  
LIMIT 3;
```

Output:



The screenshot shows a database query editor interface. At the top, a text area contains the SQL query: `#Query 17: Find details of the top 3 fuel spent:`, `SELECT *`, `FROM Fuel`, `ORDER BY total_cost DESC`, and `LIMIT 3;`. Below the query editor, a toolbar includes options for 'Result Grid', 'Filter Rows', 'Edit', 'Export/Import', and 'Fetch rows'. The main area displays a table with the following data:

ID	vehicle_ID	date	gallons_purchased	odometer_readi...	price_per_gallon	total_cost	fuel_station
5	9	2023-07-22	25.40	57137	4.53	115.062	Pilot
7	12	2023-08-26	29.01	54209	3.26	94.573	Gulf
12	13	2023-03-08	24.23	53624	3.76	91.105	Speed
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

On the right side of the interface, there are buttons for 'Result Grid' and 'Form Editor'.

Query 18:

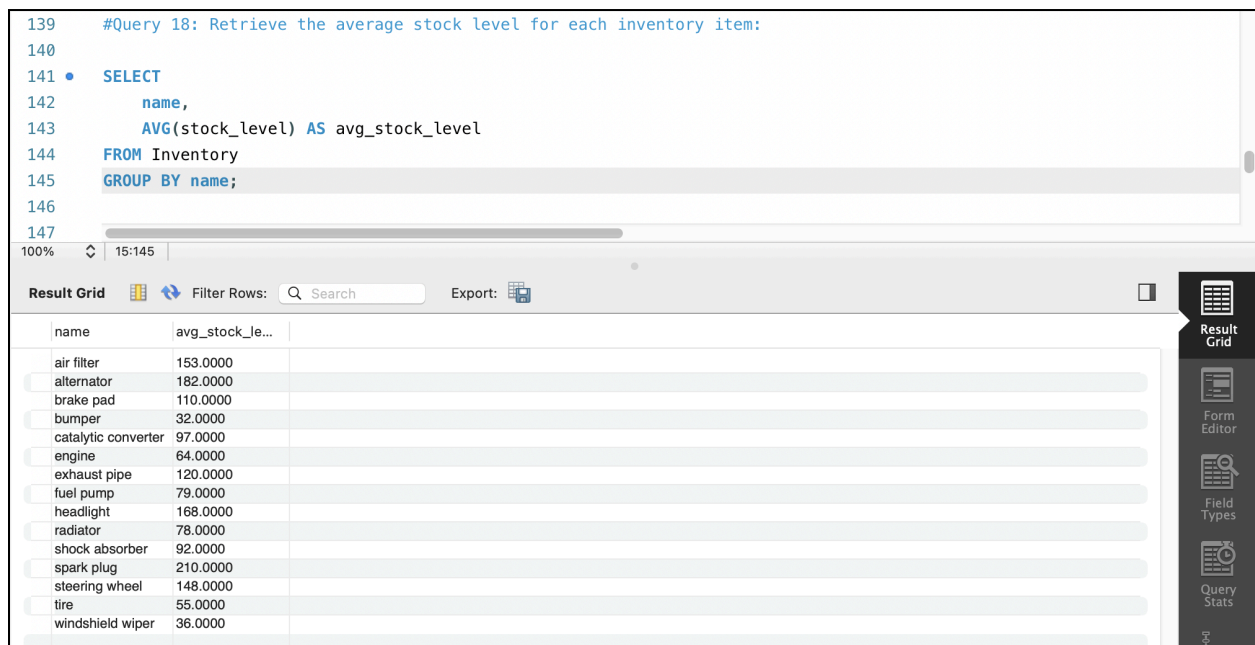
Query statement:

Retrieve the average stock level for each inventory item:

SQL command:

```
SELECT
    name,
    AVG(stock_level) AS avg_stock_level
FROM Inventory
GROUP BY name;
```

Output:



The screenshot displays a database query interface. The top section shows the SQL query being executed, which is identical to the one provided in the previous blocks. Below the query editor, a horizontal bar indicates the query execution progress. The bottom section, titled 'Result Grid', shows the output of the query as a table with two columns: 'name' and 'avg_stock_level'. The table contains 15 rows of data, listing various car parts and their average stock levels. On the right side of the interface, there are icons for 'Result Grid', 'Form Editor', 'Field Types', and 'Query Stats'.

name	avg_stock_level
air filter	153.0000
alternator	182.0000
brake pad	110.0000
bumper	32.0000
catalytic converter	97.0000
engine	64.0000
exhaust pipe	120.0000
fuel pump	79.0000
headlight	168.0000
radiator	78.0000
shock absorber	92.0000
spark plug	210.0000
steering wheel	148.0000
tire	55.0000
windshield wiper	36.0000

Query 19:

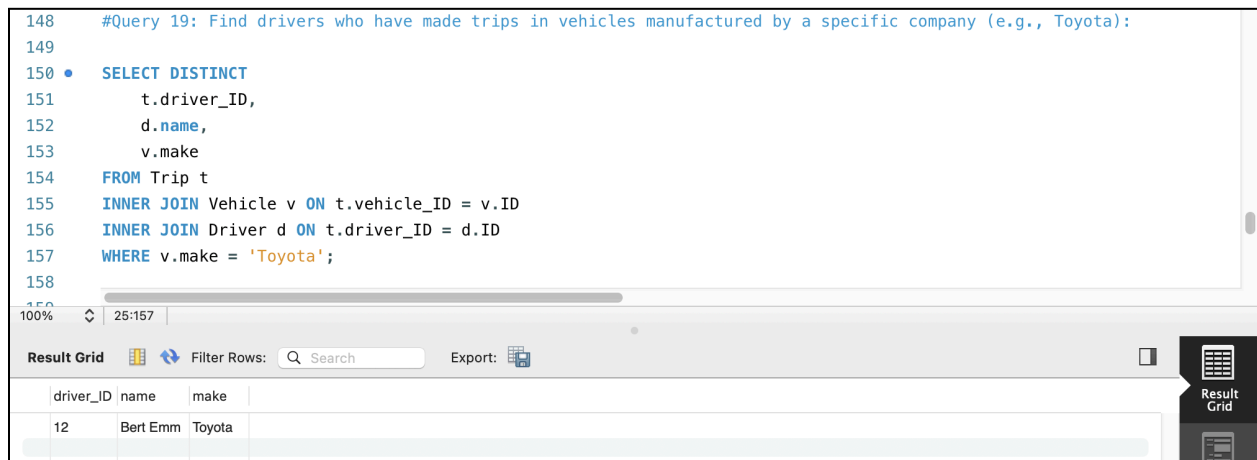
Query statement:

Find drivers who have made trips in vehicles manufactured by a specific company (e.g., Toyota):

SQL command:

```
SELECT DISTINCT
    t.driver_ID,
    d.name,
    v.make
FROM Trip t
INNER JOIN Vehicle v ON t.vehicle_ID = v.ID
INNER JOIN Driver d ON t.driver_ID = d.ID
WHERE v.make = 'Toyota';
```

Output:



The screenshot shows a SQL query editor with the following text:

```
148 #Query 19: Find drivers who have made trips in vehicles manufactured by a specific company (e.g., Toyota):
149
150 • SELECT DISTINCT
151     t.driver_ID,
152     d.name,
153     v.make
154 FROM Trip t
155 INNER JOIN Vehicle v ON t.vehicle_ID = v.ID
156 INNER JOIN Driver d ON t.driver_ID = d.ID
157 WHERE v.make = 'Toyota';
158
```

Below the query editor, there is a "Result Grid" section. It includes a "Filter Rows" search bar and an "Export" button. The results are displayed in a table with the following data:

driver_ID	name	make
12	Bert Emm	Toyota

Query 20:

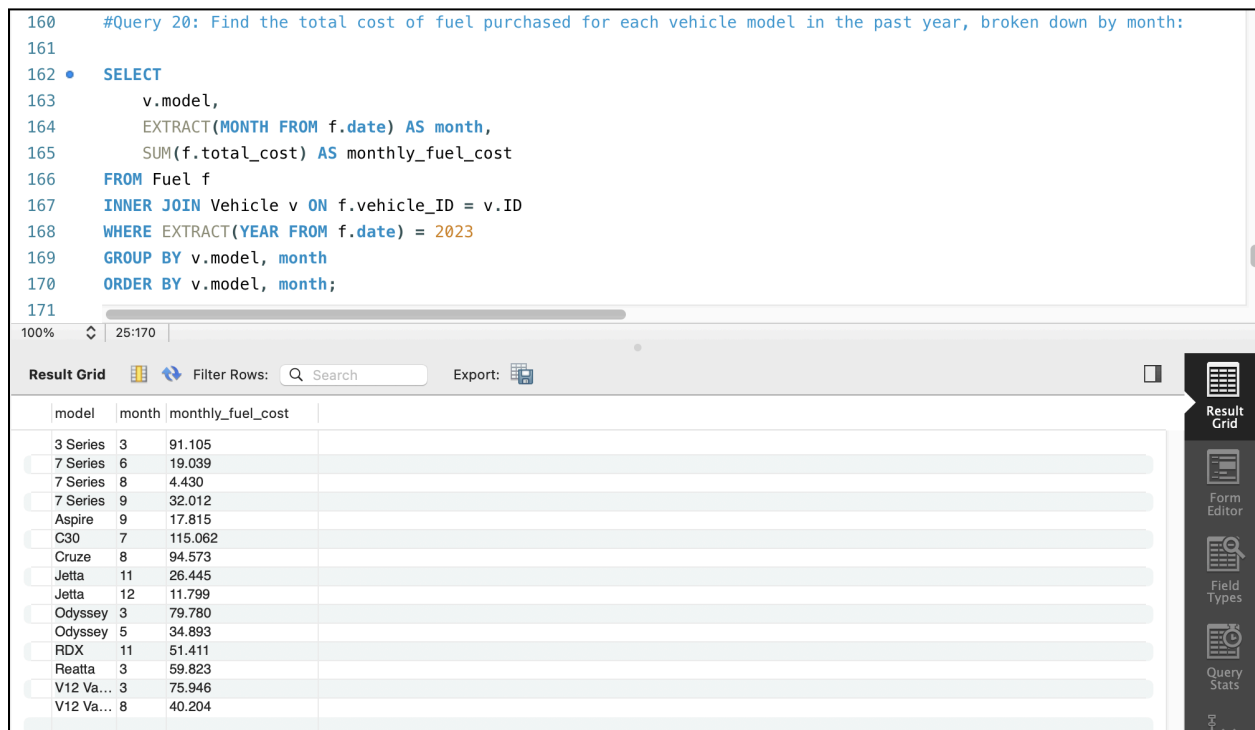
Query statement:

Find the total cost of fuel purchased for each vehicle model in the past year, broken down by month:

SQL command:

```
SELECT
    v.model,
    EXTRACT(MONTH FROM f.date) AS month,
    SUM(f.total_cost) AS monthly_fuel_cost
FROM Fuel f
INNER JOIN Vehicle v ON f.vehicle_ID = v.ID
WHERE EXTRACT(YEAR FROM f.date) = 2023
GROUP BY v.model, month
ORDER BY v.model, month;
```

Output:



The screenshot shows a database query editor interface. The top pane displays the SQL query for Query 20. The bottom pane shows the results in a table format. The table has three columns: model, month, and monthly_fuel_cost. The results are sorted by model and then by month.

model	month	monthly_fuel_cost
3 Series	3	91.105
7 Series	6	19.039
7 Series	8	4.430
7 Series	9	32.012
Aspire	9	17.815
C30	7	115.062
Cruze	8	94.573
Jetta	11	26.445
Jetta	12	11.799
Odyssey	3	79.780
Odyssey	5	34.893
RDX	11	51.411
Reatta	3	59.823
V12 Va...	3	75.946
V12 Va...	8	40.204

Query 21:

Query statement:

Calculate the average maintenance cost per vehicle made for the last two years, grouped by year and make:

SQL command:

```
SELECT
    v.make,
    EXTRACT(YEAR FROM m.date) AS year,
    AVG(m.cost) AS avg_maintenance_cost
FROM Maintenance m
INNER JOIN Vehicle v ON m.vehicle_ID = v.ID
WHERE EXTRACT(YEAR FROM m.date) >= EXTRACT(YEAR FROM
CURRENT_DATE) - 7
GROUP BY v.make, EXTRACT(YEAR FROM m.date)
ORDER BY v.make, year;
```

Output:

```
173  #Query 21: Calculate the average maintenance cost per vehicle make for the last two years, grouped by year and make:
174
175  • SELECT
176      v.make,
177      EXTRACT(YEAR FROM m.date) AS year,
178      AVG(m.cost) AS avg_maintenance_cost
179  FROM Maintenance m
180  INNER JOIN Vehicle v ON m.vehicle_ID = v.ID
181  WHERE EXTRACT(YEAR FROM m.date) >= EXTRACT(YEAR FROM CURRENT_DATE) - 7
182  GROUP BY v.make, EXTRACT(YEAR FROM m.date)
183  ORDER BY v.make, year;
184
```

100% 23:183

Result Grid Filter Rows: Search Export:

	make	year	avg_maintenance_c...
<input type="checkbox"/>	Aston Martin	2018	307.800000
<input type="checkbox"/>	Aston Martin	2020	249.250000
<input type="checkbox"/>	Buick	2018	363.810000
<input type="checkbox"/>	Chevrolet	2020	270.900000
<input type="checkbox"/>	Ford	2017	499.580000
<input type="checkbox"/>	Ford	2018	290.300000
<input type="checkbox"/>	Honda	2018	136.090000
<input type="checkbox"/>	Toyota	2017	280.250000
<input type="checkbox"/>	Toyota	2019	279.970000
<input type="checkbox"/>	Volkswagen	2017	181.040000
<input type="checkbox"/>	Volkswagen	2019	100.470000

Result Grid
Form Editor
Field Types

Query 22:

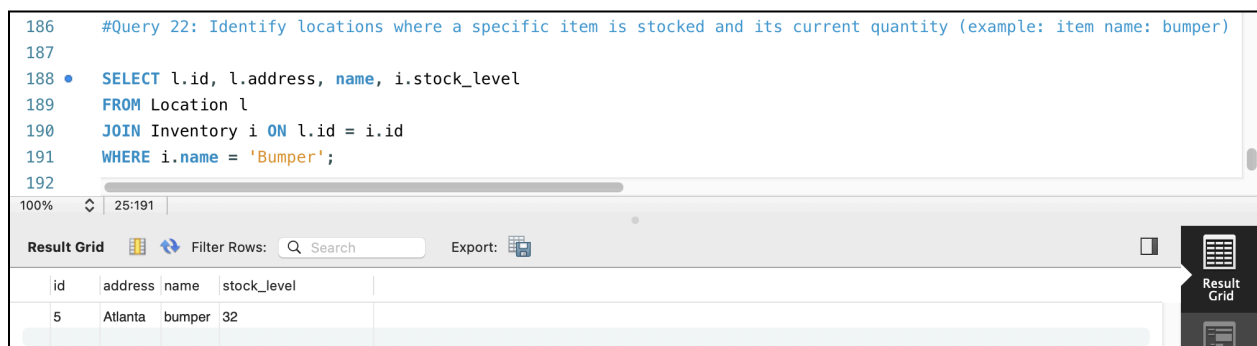
Query statement:

Identify locations where a specific item is stocked and its current quantity (example: item name: bumper)

SQL command:

```
SELECT l.id, l.address, name, i.stock_level
FROM Location l
JOIN Inventory i ON l.id = i.id
WHERE i.name = 'Bumper';
```

Output:



```
186 #Query 22: Identify locations where a specific item is stocked and its current quantity (example: item name: bumper)
187
188 • SELECT l.id, l.address, name, i.stock_level
189 FROM Location l
190 JOIN Inventory i ON l.id = i.id
191 WHERE i.name = 'Bumper';
192
```

100% 25:191

Result Grid Filter Rows: Search Export:

	id	address	name	stock_level
	5	Atlanta	bumper	32

Result Grid