

# Q - READ ALL DATA ?

The screenshot shows a database management interface with a left sidebar containing a 'SCHEMAS' tree. The 'car' database is expanded, showing a 'cars' table. The main area displays a SQL query editor with the following code:

```
1 • create database car;
2 • use car;
3
4 ## read data ##
5
6 • select * from cars;
7
```

Below the query editor is a 'Result Grid' showing the data from the 'cars' table. The grid has 11 columns: Name, year, selling\_price, km\_driven, fuel, seller\_type, transmission, owner, mileage, and engine. The data is as follows:

Name	year	selling_price	km_driven	fuel	seller_type	transmission	owner	mileage	engine
Maruti Alto 800 LXI Opt	2023	410000	10000	Petrol	Individual	Manual	First Owner	19.03 kmpl	999 CC
Skoda Slavia 1.0 TSI Ambition	2023	1350000	10000	Petrol	Individual	Manual	First Owner	14.08 kmpl	1956 CC
BMW 3 Series Gran Limousine 320Ld Luxury Line	2023	5800000	1000	Diesel	Dealer	Automatic	First Owner	18.15 kmpl	998 CC
MG ZS EV Exclusive	2023	2650000	10000	Electric	Dealer	Automatic	First Owner	32.52 kmpl	998 CC
Tata Punch Adventure	2023	715000	10000	Petrol	Individual	Manual	First Owner	12.15 kmpl	1451 CC
Maruti S-Presso VXi Plus	2023	450000	30171	Petrol	Individual	Manual	First Owner	19.03 kmpl	999 CC
Maruti S-Presso LXI	2022	425000	1994	Petrol	Dealer	Manual	First Owner	19.47 kmpl	999 CC

Below the result grid is an 'Output' section showing the execution of the queries. The output is as follows:

#	Time	Action	Message
38	18:05:31	use car	0 row(s) affected
39	18:05:47	select * from cars LIMIT 0, 1000	1000 row(s) returned
40	18:14:32	select * from cars LIMIT 0, 1000	1000 row(s) returned

# Q - TOTAL CARS ?

The screenshot shows a database management interface with a query editor and a results pane. The query editor contains the following SQL code:

```
7  
8  ## TOTAL CARS : TO GET COUNT OF TOTAL RECORDS  
9  
10 • SELECT COUNT(*) FROM CARS;  
11  
12  
13  
14
```

The results pane shows a single row with the count of records:

COUNT(*)
7927

The bottom pane shows the execution log with the following entries:

#	Time	Action	Message
40	18:14:32	select * from cars LIMIT 0, 1000	1000 row(s) returned
41	18:18:13	SELECT COUNT(*) FROM CARS LIMIT 0, 1000	1 row(s) returned

The left sidebar shows the database schema structure, including the 'car' database and the 'cars' table. The 'cars' table has the following columns:

Column Name	Column Type
Name	text
year	int
selling_price	int
km_driven	int
fuel	text
seller_type	text
transmission	text
owner	text
mileage	text

# Q - WHICH YEAR HAD LESS THAN 50 CARS ?

The screenshot shows a database management interface with a left sidebar, a central query editor, and a bottom output pane.

**Left Sidebar:**

- Navigation:** Schemas, Filter objects, car (Tables, cars, Columns, Indexes, Foreign Keys, Triggers, Views, Stored Procedures, Functions), sys, transactoins.
- Administration:** Schemas.
- Information:** Table: cars. Columns: Name (text), year (int), selling\_price (int), km\_driven (int), fuel (text), seller\_type (text), transmission (text), owner (text), mileage (text).

**Central Query Editor:**

Query 1: car\_dekho

Limit to 1000 rows

```
29 ## WHICH YEAR HAD MORE THAN 100 CARS ##
30
31 • SELECT YEAR, COUNT(*) FROM CARS GROUP BY YEAR
32 HAVING COUNT(*) < 50;
33
34
```

**Result Grid:**

YEAR	COUNT(*)
2023	6
2022	7
2021	7
2003	37
2002	19
2001	6
2000	16
1999	14
1998	9

**Bottom Output Pane:**

Output: Action Output

#	Time	Action	Message
68	19:05:46	SELECT YEAR, COUNT(*) FROM CARS GROUP BY YEAR HAVING COUNT(*) > 100 LIMIT ...	14 row(s) returned
69	19:06:43	SELECT YEAR, COUNT(*) FROM CARS GROUP BY YEAR HAVING COUNT(*) < 50 LIMIT ...	13 row(s) returned



# Q - WHICH YEAR HAD MORE THAN 100 CARS ?

The screenshot shows a database management interface with a query editor and a results grid. The query editor contains the following SQL code:

```
## WHICH YEAR HAD MORE THAN 100 CARS ##  
  
SELECT YEAR, COUNT(*) FROM CARS GROUP BY YEAR  
HAVING COUNT(*) > 100;
```

The results grid displays the following data:

YEAR	COUNT(*)
2019	583
2018	806
2017	1010
2016	856
2015	775
2014	620
2013	668
2012	621
2011	570

The interface also shows a sidebar with a tree view of database objects, including 'car' (Tables, Columns, Indexes, Foreign Keys, Triggers, Views, Stored Procedures, Functions) and 'sys' (transactoins). The bottom panel shows the 'Output' tab with a log of actions:

#	Time	Action	Message
67	19:02:51	SELECT YEAR, COUNT(*) FROM CARS WHERE FUEL = "CNG" GROUP BY YEAR LIMIT ...	13 row(s) returned
68	19:05:46	SELECT YEAR, COUNT(*) FROM CARS GROUP BY YEAR HAVING COUNT(*) > 100 LIM...	14 row(s) returned

# Q - ALL CARS COUNT DETAILS BETWEEN 2015 TO 2023 ?

The screenshot shows a database management interface with a query editor and a results pane. The query editor displays a SQL query to count cars between 2015 and 2023. The results pane shows a single row with the count 4124. The bottom pane shows a log of actions, including the execution of the query and the return of 1000 rows.

**Navigator:**

- SCHEMAS
  - Filter objects
  - car
    - Tables
      - cars
        - Columns
        - Indexes
        - Foreign Keys
        - Triggers
      - Views
      - Stored Procedures
      - Functions
    - sys
    - transactoins

**Query 1** **car\_dekho**

Limit to 1000 rows

```
33  ## ALL CARS COUNT DETAILS 2015 TO 2023 WITH COMPLETE DETAILS ##
34
35  •  SELECT COUNT(*) FROM CARS WHERE YEAR BETWEEN 2015 AND 2023;
36
37
38
```

**Result Grid** | Filter Rows: | Export: | Wrap Cell Content: |

COUNT(*)
4124

**Table: cars**

**Columns:**

Column	Type
Name	text
year	int
selling_price	int
km_driven	int
fuel	text
seller_type	text
transmission	text
owner	text
mileage	text

**Result 28** **×** **Read Only**

**Output**

Action Output

#	Time	Action	Message
71	19:10:20	SELECT * FROM CARS WHERE YEAR BETWEEN 2015 AND 2023 ORDER BY YEAR AS...	1000 row(s) returned
72	19:11:44	SELECT COUNT(*) FROM CARS WHERE YEAR BETWEEN 2015 AND 2023 LIMIT 0, 1000	1 row(s) returned

Object Info | Session



# Q - HOW MANY CNG CARS IN ALL YEARS ?

The screenshot shows a database management interface with a Navigator pane on the left, a central query editor, and a results/output pane at the bottom.

**Navigator:**

- SCHEMAS**
  - Filter objects
  - car
    - Tables
      - cars
        - Columns
        - Indexes
        - Foreign Keys
        - Triggers
      - Views
      - Stored Procedures
      - Functions
    - sys
    - transactoins

**Query 1: car\_dekho**

```
24 • SELECT COUNT(*) FROM CARS WHERE YEAR = 2020 AND FUEL = "DIESEL";
25 ## HOW MANY PETROL CARS IN 2020 ##
26
27 • SELECT YEAR, COUNT(*) FROM CARS WHERE FUEL = "CNG"
28 GROUP BY YEAR;
29
```

**Result Grid:**

YEAR	COUNT(*)
2020	3
2019	7
2018	5
2017	9
2016	6
2015	2
2014	4
2013	3
2012	5

**Table: cars**

**Columns:**

Column Name	Data Type
Name	text
year	int
selling_price	int
km_driven	int
fuel	text
seller_type	text
transmission	text
owner	text
mileage	text

**Output:**

#	Time	Action	Message
66	19:02:46	SELECT COUNT(*) FROM CARS WHERE YEAR = 2020 AND FUEL = "DIESEL" LIMIT 0, ...	1 row(s) returned
67	19:02:51	SELECT YEAR, COUNT(*) FROM CARS WHERE FUEL = "CNG" GROUP BY YEAR LIMIT ...	13 row(s) returned



# Q - HOW MANY PETROL CARS IN ALL YEARS ?

The screenshot shows a database management interface with a left-hand sidebar containing a 'Navigator' pane. The 'Schemas' section is expanded, showing a 'car' schema with a 'cars' table. The 'Columns' section for the 'cars' table is also expanded, listing columns: Name (text), year (int), selling\_price (int), km\_driven (int), fuel (text), seller\_type (text), transmission (text), owner (text), and mileage (text). The main area displays two SQL queries in a text editor. The first query is: `SELECT COUNT(*) FROM CARS WHERE YEAR = 2020 AND FUEL = "DIESEL";` followed by a comment: `## HOW MANY PETROL CARS IN 2020 ##`. The second query is: `SELECT YEAR, COUNT(*) FROM CARS WHERE FUEL = "PETROL" GROUP BY YEAR;`. Below the queries, a 'Result Grid' is shown with a table containing two columns: 'YEAR' and 'COUNT(\*)'. The table has 10 rows of data. At the bottom, an 'Output' pane shows a list of actions and their results. Action 62 is a failed query: `SELECT YEAR, FUEL FROM CARS GROUP BY YEAR LIMIT 0, 1000` with an error message: 'Error Code: 1055. Expression #2 of SELECT list is not in GROUP BY clause'. Action 63 is a successful query: `SELECT YEAR, COUNT(*) FROM CARS WHERE FUEL = "PETROL" GROUP BY YEAR L...` with a message: '30 row(s) returned'.

Query 1 car\_dekho x

Limit to 1000 rows

```
24 • SELECT COUNT(*) FROM CARS WHERE YEAR = 2020 AND FUEL = "DIESEL";
25 ## HOW MANY PETROL CARS IN 2020 ##
26
27 • SELECT YEAR, COUNT(*) FROM CARS WHERE FUEL = "PETROL"
28 GROUP BY YEAR;
29
```

Result Grid

YEAR	COUNT(*)
2023	4
2022	5
2021	5
2020	51
2019	352
2018	394
2017	432
2016	429
2015	278

Result 19 x

Output

#	Time	Action	Message
62	18:57:43	SELECT YEAR, FUEL FROM CARS GROUP BY YEAR LIMIT 0, 1000	Error Code: 1055. Expression #2 of SELECT list is not in GROUP BY clause
63	19:01:20	SELECT YEAR, COUNT(*) FROM CARS WHERE FUEL = "PETROL" GROUP BY YEAR L...	30 row(s) returned

# Q - HOW MANY DIESEL CARS IN ALL YEARS ?

The screenshot shows a database management interface with a left sidebar, a central query editor, and a bottom results panel.

**Left Sidebar (Navigator):**

- SCHEMAS
- Filter objects
- car
  - Tables
    - cars
      - Columns
      - Indexes
      - Foreign Keys
      - Triggers
    - Views
    - Stored Procedures
    - Functions
  - sys
  - transactoins

**Central Query Editor (Query 1):**

```
24 • SELECT COUNT(*) FROM CARS WHERE YEAR = 2020 AND FUEL = "DIESEL";
25 ## HOW MANY PETROL CARS IN 2020 ##
26
27 • SELECT YEAR, COUNT(*) FROM CARS WHERE FUEL = "DIESEL"
28 GROUP BY YEAR;
29
```

**Result Grid:**

YEAR	COUNT(*)
2023	1
2022	2
2021	2
2020	20
2019	224
2018	407
2017	569
2016	421
2015	493

**Bottom Panel (Output):**

Result 21 x

Action Output

#	Time	Action	Message
64	19:01:51	SELECT YEAR, COUNT(*) FROM CARS WHERE FUEL = "DIESEL" GROUP BY YEAR LIM...	0 row(s) returned
65	19:02:16	SELECT YEAR, COUNT(*) FROM CARS WHERE FUEL = "DIESEL" GROUP BY YEAR LI...	25 row(s) returned

**Table: cars (Bottom Left):**

Columns:

- Name: text
- year: int
- selling\_price: int
- km\_driven: int
- fuel: text
- seller\_type: text
- transmission: text
- owner: text
- mileage: text



# Q - CLIENT ASKED TO PRINT TOTAL OF ALL CARS BY YEARS ?

The screenshot shows a database client interface with a query editor and a results pane. The query editor contains the following SQL code:

```
17  ## CLIENT ASKED ME TO PRINT TOTAL OF ALL CARS BY YEARS ##
18
19  • select year , count(*) from cars
20  group by year;
21
22
```

The results pane displays a table with the following data:

year	count(*)
2023	6
2022	7
2021	7
2020	74
2019	583
2018	806
2017	1010
2016	856
2015	775

The interface also shows a sidebar with a tree view of the database schema, including a 'car' schema with a 'cars' table. The bottom pane shows the 'Action Output' tab with the following log entries:

#	Time	Action	Message
54	18:47:25	select year , count(*) from cars LIMIT 0, 1000	Error Code: 1140. In aggregated query without GROUP BY, expression #1 b
55	18:47:45	select year , count(*) from cars group by year LIMIT 0, 1000	30 row(s) returned

# Q - HOW MANY CARS WILL BE AVAILABLE IN 2020 , 2021 , 2022 ?

The screenshot shows a database management interface with a query editor and a results pane. The query editor contains the following SQL code:

```
12  ## HOW MANY CARS WILL BE AVAILABLE IN 2020 ,2021,2022 ##
13
14  •  SELECT COUNT(*) FROM CARS
15  WHERE YEAR IN (2020 , 2021 , 2022)
16
17
```

The results pane shows a single row with the count of cars:

COUNT(*)
88

The interface also includes a sidebar with a schema tree and a table information pane.

**Table: cars**

**Columns:**

Column Name	Column Type
Name	text
year	int
selling_price	int
km_driven	int
fuel	text
seller_type	text
transmission	text
owner	text
mileage	text

**Output**

#	Time	Action	Message
49	18:30:33	SELECT COUNT(*) FROM CARS WHERE YEAR IN (2020 , 2021 , 2022) GROUP BY YEA...	3 row(s) returned
50	18:30:45	SELECT COUNT(*) FROM CARS WHERE YEAR IN (2020 , 2021 , 2022) LIMIT 0, 1000	1 row(s) returned

