

DATA BASE MANAGEMENT SYSTEM PROJECT

(Structured query language)

Topic: Retail price of the products in Markets year-wise

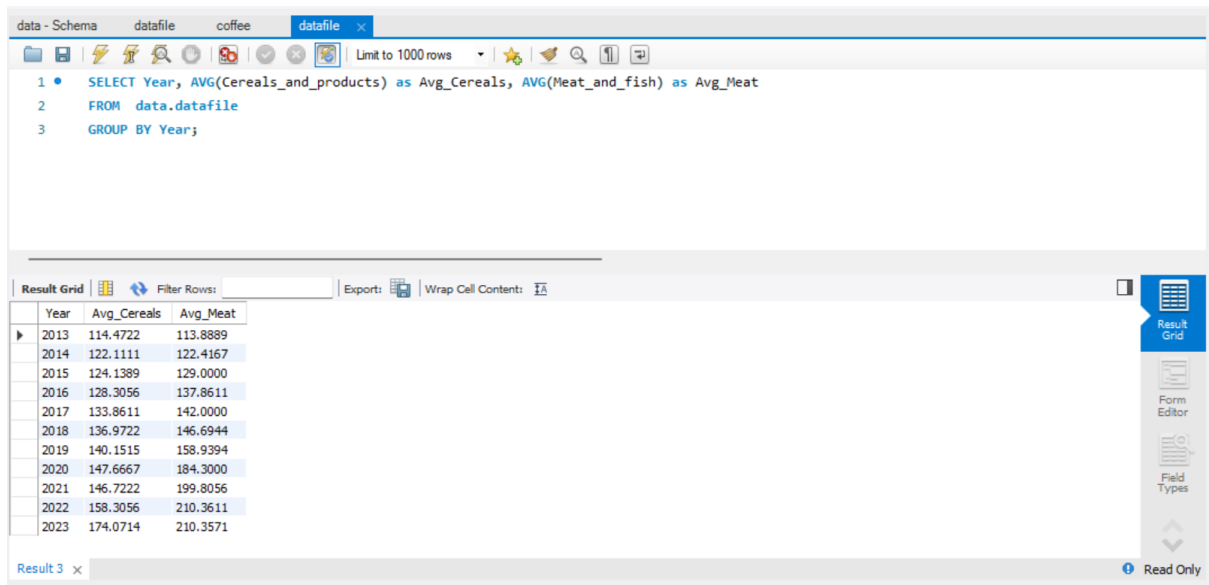
SOURCE OF DATA:

<https://data.gov.in/catalogs?ministry=Department%20of%20Water%20Resources,%20River%20Development%20%26%20Ganga%20Rejuvenation>

SUBMITTED BY – LAVANYA PUROHIT

SUBMITTED TO – MR. CHINTAN PATEL

1. Calculate the average price for each food category in a specific year



The screenshot shows a database query interface with a SQL query editor and a result grid. The query is as follows:

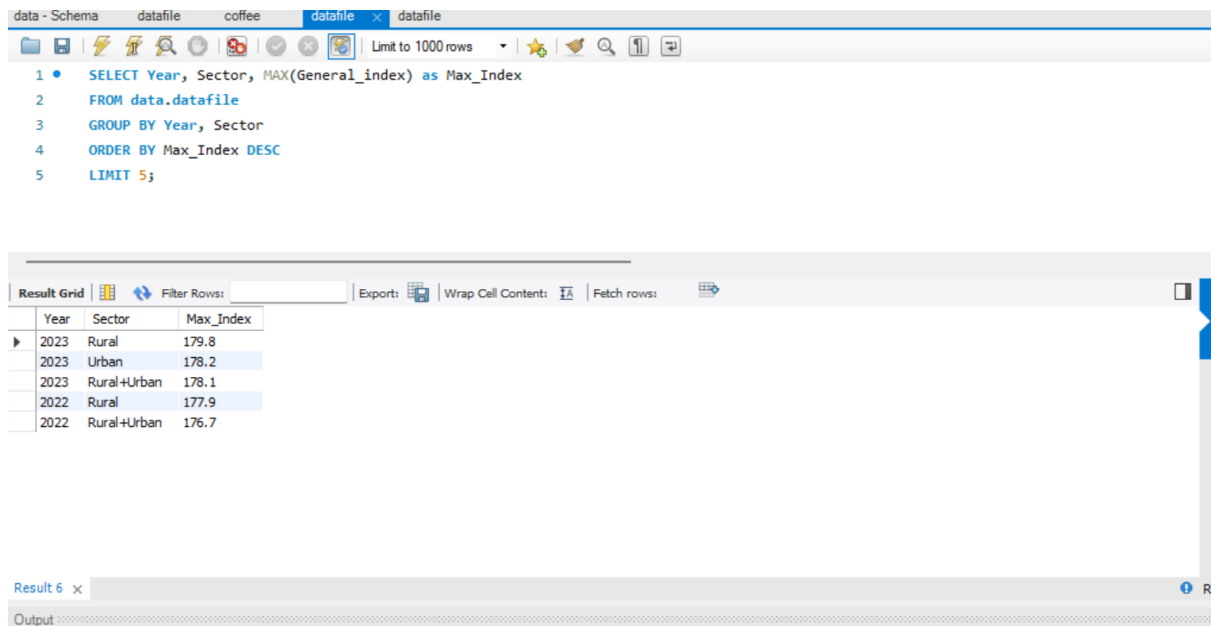
```
1 • SELECT Year, AVG(Cereals_and_products) as Avg_Cereals, AVG(Meat_and_fish) as Avg_Meat
2 FROM data.datafile
3 GROUP BY Year;
```

The result grid displays the following data:

Year	Avg_Cereals	Avg_Meat
2013	114.4722	113.8889
2014	122.1111	122.4167
2015	124.1389	129.0000
2016	128.3056	137.8611
2017	133.8611	142.0000
2018	136.9722	146.6944
2019	140.1515	158.9394
2020	147.6667	184.3000
2021	146.7222	199.8056
2022	158.3056	210.3611
2023	174.0714	210.3571

The interface also includes a toolbar with icons for various actions, a 'Limit to 1000 rows' dropdown, and a 'Read Only' status indicator.

2. Retrieve the top 5 sectors with the highest index for a specific year



The screenshot shows a database query interface with a SQL query editor and a result grid. The query is as follows:

```
1 • SELECT Year, Sector, MAX(General_index) as Max_Index
2 FROM data.datafile
3 GROUP BY Year, Sector
4 ORDER BY Max_Index DESC
5 LIMIT 5;
```

The result grid displays the following data:

Year	Sector	Max_Index
2023	Rural	179.8
2023	Urban	178.2
2023	Rural+Urban	178.1
2022	Rural	177.9
2022	Rural+Urban	176.7

The interface also includes a toolbar with icons for various actions, a 'Limit to 1000 rows' dropdown, and a 'Fetch rows' button.

3. Calculate the total spending on clothing and footwear for each year

The screenshot shows a data tool interface with a SQL editor and a result grid. The SQL query is as follows:

```
1 • SELECT Year, SUM(Clothing_and_footwear) as Total_Spending
2 FROM data.datafile
3 GROUP BY Year;
```

The result grid displays the following data:

Year	Total_Spending
2013	3957.7999999999997
2014	4265.4000000000001
2015	4513.7999999999998
2016	4744.4
2017	4954.6
2018	5189.0000000000001
2019	4852.6
2020	4527.2000000000002
2021	5766.7000000000001
2022	6315.3999999999999
2023	2579.7000000000003

The interface includes a toolbar with icons for file operations, a 'Limit to 1000 rows' dropdown, and a 'Filter Rows' field. The result grid has an 'Export' button and a 'Wrap Cell Content' checkbox. The status bar at the bottom indicates 'Result 9' and 'Read Only'.

4. Find the sector with the lowest index in a specific month and year

The screenshot shows a data tool interface with a SQL editor and a result grid. The SQL query is as follows:

```
4
5 • SELECT Sector, Month, MIN(General_index) as Min_Index
6 FROM data.datafile
7 WHERE Year = 2023 AND Month = 'February'
8 GROUP BY Sector, Month
```

The result grid displays the following data:

Sector	Month	Min_Index
Rural	February	178
Urban	February	176.3
Rural+Urban	February	177.2

The interface includes a toolbar with icons for file operations, a 'Limit to 1000 rows' dropdown, and a 'Filter Rows' field. The result grid has an 'Export' button and a 'Wrap Cell Content' checkbox. The status bar at the bottom indicates 'Result 10' and 'Read Only'.

5. Calculate the total spending on housing and transportation for each year:

The screenshot shows a database query editor with a SQL query and its results. The query is as follows:

```
1 • SELECT Year, SUM(Housing) as Total_Housing, SUM(Transport_and_communication) as Total_Transportation
2 FROM data.datafile
3 GROUP BY Year;
```

The results are displayed in a table with the following columns: Year, Total_Housing, and Total_Transportation. The data is grouped by year from 2013 to 2023.

Year	Total_Housing	Total_Transportation
2013	2534.5999999999995	3840.0000000000005
2014	2752.1999999999994	4014.6000000000004
2015	2883.8000000000006	4001.7000000000007
2016	3035.3999999999996	4084.8999999999996
2017	3208.2	4227.5
2018	3451.6000000000001	4440.1000000000001
2019	3316.7999999999997	4150.9999999999999
2020	3114.6000000000004	4088.4999999999995
2021	3875.5999999999995	5425.1999999999999
2022	4033.2	5804.6999999999999
2023	1564.2	2301.4

6. Calculate the total spending on all sectors for a specific year and month

The screenshot shows a database query editor with a SQL query and its results. The query is as follows:

```
1 • SELECT Year, Month, SUM(General_index) as Total_Spending
2 FROM data.datafile
3 WHERE Year = 2023 AND Month = 'April'
4 GROUP BY Year, Month;
```

The results are displayed in a table with the following columns: Year, Month, and Total_Spending. The data is grouped by year and month for the year 2023 and month April.

Year	Month	Total_Spending
2023	April	534.3000000000001

7. Identify sectors where spending on Housing is in the top 5 for a specific year

The screenshot shows a data tool interface with a SQL editor and a results grid. The SQL query is as follows:

```
1 WITH RankedSectors AS (  
2   SELECT  
3     Sector,  
4     Year,  
5     Housing,  
6     RANK() OVER (PARTITION BY Year ORDER BY Housing DESC) AS Spending_Rank  
7   FROM data.datafile  
8 )  
9 SELECT Sector, Year, Housing  
10 FROM RankedSectors  
11 WHERE Spending_Rank <= 5;
```

The results grid displays the following data:

	Sector	Year	Housing
▶	Rural	2013	NA
	Rural	2013	NA
	Rural	2013	NA
	Rural	2013	NA
	Rural	2013	NA
	Rural	2013	NA

Below the grid, it says "Result 2 x" and "Output:".

8. Calculate the cumulative sum of spending on Recreation and Amusement over time:

The screenshot shows a data tool interface with a SQL editor and a results grid. The SQL query is as follows:

```
13  
14 • SELECT  
15   Sector,  
16   Year,  
17   Month,  
18   Recreation_and_amusement,  
19   SUM(Recreation_and_amusement) OVER (PARTITION BY Sector ORDER BY Year, Month) AS Cumulative_Spending  
20 FROM data.datafile;
```

The results grid displays the following data:

	Sector	Year	Month	Recreation_and_amusement	Cumulative_Spending
▶	Rural	2013	April	104.5	104.5
	Rural	2013	August	106.8	211.3
	Rural	2013	December	109.2	320.5
	Rural	2013	February	104	424.5
	Rural	2013	January	103.4	527.9
	Rural	2013	July	106.4	634.3
	Rural	2013	June	105.6	739.9
	Rural	2013	March	104	843.9
	Rural	2013	May	105	948.9
	Rural	2013	November	108.7	1057.6
	Rural	2013	October	108.3	1165.8999999999999

Below the grid, it says "Result 3 x" and "Output:". On the right side, there are buttons for "Result Grid", "Form Editor", "Field Types", and "Read Only".

9. Calculate the weighted average of spending on Health and Education, giving more weight to Health

The screenshot shows a data analysis tool interface. At the top, there's a toolbar with various icons and a 'Limit to 1000 rows' dropdown. Below the toolbar is a SQL query editor with the following code:

```
22
23 • SELECT
24     Sector,
25     Year,
26     AVG(Health * 0.7 + Education * 0.3) AS Weighted_Avg_Health_Education
27 FROM data.datafile
28 GROUP BY Sector, Year;
```

Below the query editor is a 'Result Grid' section. It includes a 'Filter Rows' input, an 'Export' button, and a 'Wrap Cell Content' checkbox. The grid displays the following data:

Sector	Year	Weighted_Avg_Health_Education
Rural	2013	106.96249999999999
Urban	2013	107.11583333333334
Rural+Urban	2013	107.05166666666666
Rural	2014	113.68666666666667
Urban	2014	112.96583333333332
Rural+Urban	2014	113.51416666666665
Rural	2015	120.76583333333332
Urban	2015	118.325
Rural+Urban	2015	119.95583333333332
Rural	2016	127.70333333333333
Urban	2016	123.37416666666662

On the right side of the interface, there are buttons for 'Result Grid', 'Form Editor', and 'Field Types'. At the bottom right, there is a 'Read Only' indicator.

10. Calculate the exponential moving average of the General Index for each sector:

The screenshot shows a data analysis tool interface. At the top, there's a toolbar with various icons and a 'Limit to 1000 rows' dropdown. Below the toolbar is a SQL query editor with the following code:

```
31 • SELECT
32     Sector,
33     Year,
34     Month,
35     General_index,
36     AVG(General_index) OVER (PARTITION BY Sector ORDER BY Year, Month ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS E
37 FROM data.datafile;
```

Below the query editor is a 'Result Grid' section. It includes a 'Filter Rows' input, an 'Export' button, and a 'Wrap Cell Content' checkbox. The grid displays the following data:

Sector	Year	Month	General_index	EMA
Rural	2013	April	106.4	106.4
Rural	2013	August	112.1	109.25
Rural	2013	December	115.5	111.33333333333333
Rural	2013	February	105.8	109.95
Rural	2013	January	105.1	108.97999999999999
Rural	2013	July	110.7	109.26666666666667
Rural	2013	June	108.9	109.21428571428571
Rural	2013	March	106	108.8125
Rural	2013	May	107.2	108.63333333333334
Rural	2013	November	117.4	109.51000000000002
Rural	2013	October	115.5	110.05454545454546

On the right side of the interface, there are buttons for 'Result Grid', 'Form Editor', and 'Field Types'. At the bottom right, there is a 'Read Only' indicator.

11. Calculate the month-over-month percentage change in the General Index:

datafile

```

1 • SELECT
2     Sector,
3     Year,
4     Month,
5     General_index,
6     LAG(General_index) OVER (PARTITION BY Sector ORDER BY Year, Month) AS Prev_Index,
7     ((General_index - LAG(General_index) OVER (PARTITION BY Sector ORDER BY Year, Month)) / LAG(General_index) OVER (PARTITION BY Sector ORDER BY Year, Month)) AS Percent_Change
8 FROM data.datafile;

```

Result Grid

Sector	Year	Month	General_index	Prev_Index	Percent_Change
Rural	2013	February	105.8	115.5	-8.398268398268401
Rural	2013	January	105.1	105.8	-0.6616257088846907
Rural	2013	July	110.7	105.1	5.328258801141779
Rural	2013	June	108.9	110.7	-1.626016260162599
Rural	2013	March	106	108.9	-2.662993572084486
Rural	2013	May	107.2	106	1.1320754716981158
Rural	2013	November	117.4	107.2	9.51492537313433
Rural	2013	October	115.5	117.4	-1.6183986371379946
Rural	2013	September	114.2	115.5	-1.125541125541123
Rural	2014	April	115.4	114.2	1.0507880910683036
Rural	2014	August	120.7	115.4	4.592720970537259

Result 6 ×

Output