

SQL workbench Queries - Report

Calculating a monthly percentage change:

```
SELECT
    Year,
    Month,
    Gold,
    Silver,
    Platinum,
    Palladium,

    ROUND(((Gold - LAG(Gold) OVER (ORDER BY Year, Month)) / LAG(Gold) OVER (ORDER BY Year,
    Month)) * 100, 2) AS Gold_Percentage_Change,

    ROUND(((Silver - LAG(Silver) OVER (ORDER BY Year, Month)) / LAG(Silver) OVER (ORDER BY Year,
    Month)) * 100, 2) AS Silver_Percentage_Change,

    ROUND(((Platinum - LAG(Platinum) OVER (ORDER BY Year, Month)) / LAG(Platinum) OVER (ORDER
    BY Year, Month)) * 100, 2) AS Platinum_Percentage_Change,

    ROUND(((Palladium - LAG(Palladium) OVER (ORDER BY Year, Month)) / LAG(Palladium) OVER
    (ORDER BY Year, Month)) * 100, 2) AS Palladium_Percentage_Change

FROM metals_with_inflation_interest_GDP;
```

WITH PercentageChanges AS (

```
SELECT
    Year,
    Month,
    Gold,
    Silver,
    Platinum,
    Palladium,

    ROUND(((Gold - LAG(Gold) OVER (ORDER BY Year, Month)) / LAG(Gold) OVER (ORDER BY Year,
    Month)) * 100, 2) AS Gold_Percentage_Change,

    ROUND(((Silver - LAG(Silver) OVER (ORDER BY Year, Month)) / LAG(Silver) OVER (ORDER BY Year,
    Month)) * 100, 2) AS Silver_Percentage_Change,
```

```

ROUND(((Platinum - LAG(Platinum) OVER (ORDER BY Year, Month)) / LAG(Platinum) OVER
(ORDER BY Year, Month)) * 100, 2) AS Platinum_Percentage_Change,

ROUND(((Palladium - LAG(Palladium) OVER (ORDER BY Year, Month)) / LAG(Palladium) OVER
(ORDER BY Year, Month)) * 100, 2) AS Palladium_Percentage_Change

FROM metals_with_inflation_interest_GDP

)

```

```

SELECT

Year,

Month,

Gold,

Silver,

Platinum,

Palladium,

COALESCE(

CASE

WHEN ROW_NUMBER() OVER (ORDER BY Year, Month) = 1 THEN

AVG(Gold_Percentage_Change) OVER ()

ELSE Gold_Percentage_Change

END,

0

) AS Gold_Percentage_Change,

COALESCE(

CASE

WHEN ROW_NUMBER() OVER (ORDER BY Year, Month) = 1 THEN

AVG(Silver_Percentage_Change) OVER ()

ELSE Silver_Percentage_Change

END,

0

) AS Silver_Percentage_Change,

COALESCE(

CASE

```

```

        WHEN ROW_NUMBER() OVER (ORDER BY Year, Month) = 1 THEN

            AVG(Platinum_Percentage_Change) OVER (

                ELSE Platinum_Percentage_Change

            END,

            0

        ) AS Platinum_Percentage_Change,

    COALESCE(

        CASE

            WHEN ROW_NUMBER() OVER (ORDER BY Year, Month) = 1 THEN

                AVG(Palladium_Percentage_Change) OVER (

                    ELSE Palladium_Percentage_Change

                END,

                0

            ) AS Palladium_Percentage_Change

    FROM PercentageChanges;

```

SCHEMAS

Filter objects

- maverick
 - Tables
 - inflation_years
 - metals_with_inflation
 - metals_with_inflation_interest_gdp
 - precious_metals
 - Views
 - Stored Procedures
 - Functions
 - phpmyadmin
 - test
 - trial_1

Administration Schemas

Information

Connection Details

Name: maverick

Host: localhost

Port: 3306

Login User: root

Current User: root@localhost

Limit to 1000 rows

64 COALESCE(

65 CASE

66 WHEN ROW_NUMBER() OVER (ORDER BY Year, Month) = 1 THEN

67 AVG(Palladium_Percentage_Change) OVER (

68 ELSE Palladium_Percentage_Change

69 END,

70 0

71) AS Palladium_Percentage_Change

72 FROM PercentageChanges;

Result Grid

Year	Month	Gold	Silver	Platinum	Palladium	Gold_Percentage_Change	Silver_Percentage_Change	Platinum
1990	4	374.59	5.06	478.01	127.07	0.456219	0.558358	0.289154
1990	5	368.95	5.07	488.09	119.54	-1.510000	0.200000	2.110000
1990	6	352.49	4.92	481.35	115.55	-4.460000	-2.960000	-1.380000
1990	7	362.18	4.87	479.09	117.08	2.750000	-1.020000	-0.470000
1990	8	394.79	5.00	491.90	115.61	9.000000	2.670000	2.670000
1990	9	389.44	4.80	462.05	105.24	-1.360000	-4.000000	-6.070000
1990	10	381.04	4.39	424.49	95.50	-2.160000	-8.540000	-8.130000
1990	11	381.80	4.17	421.72	93.78	0.200000	-5.010000	-0.650000
1990	12	377.06	4.07	421.54	89.97	-1.240000	-2.400000	-0.040000
1991	1	384.11	4.05	408.18	87.01	1.870000	-0.490000	-3.170000
1991	2	363.79	3.74	386.58	84.34	-5.290000	-7.650000	-5.290000
1991	3	363.36	3.95	400.39	86.10	-0.120000	5.610000	3.570000
1991	4	358.22	3.98	397.39	96.60	-1.410000	0.760000	-0.750000
1991	5	357.03	4.05	389.64	95.48	-0.330000	1.760000	-1.950000
1991	6	366.55	4.39	375.57	97.09	2.670000	8.400000	-3.610000
1991	7	367.85	4.35	376.05	94.62	0.350000	-0.910000	0.130000
1991	8	356.51	3.96	347.14	83.05	-3.080000	-8.970000	-7.690000
1991	9	348.60	4.04	349.09	81.93	-2.220000	2.020000	0.560000
1991	10	358.76	4.12	362.50	85.28	2.910000	1.980000	3.840000

Result Grid

Form Editor

Field Types

Query Stats

Execution Plan

Calculating the average inflation

```
SELECT AVG(inflation) AS average_inflation  
FROM metals_with_inflation;
```

The screenshot shows a database client interface with a 'SCHEMAS' panel on the left and a query editor on the right. The 'SCHEMAS' panel shows a tree view of the database structure, including tables like 'inflation_years', 'metals_with_inflation', and 'precious_metals'. The query editor contains the following SQL query:

```
1 • SELECT AVG(inflation) AS average_inflation  
2 FROM metals_with_inflation;  
3
```

Below the query editor, the 'Result Grid' shows the result of the query:

average_inflation
2.752109

Selecting all the values where inflation is higher than the average inflation.

```
SELECT *  
FROM inflation_years  
WHERE inflation > (SELECT AVG(inflation) FROM inflation_years);
```

The screenshot shows a database client interface with a 'SCHEMAS' panel on the left and a query editor on the right. The 'SCHEMAS' panel shows a tree view of the database structure, including tables like 'inflation_years', 'metals_with_inflation', and 'precious_metals'. The query editor contains the following SQL query:

```
1 • SELECT *  
2 FROM inflation_years  
3 WHERE inflation > (SELECT AVG(inflation) FROM inflation_years);  
4
```

Below the query editor, the 'Result Grid' shows the result of the query:

Year	Month	Inflation
1990	1	5.90
1990	2	6.30
1990	3	6.20
1990	4	7.70
1990	5	8.20
1990	6	8.20
1990	7	8.20
1990	8	8.90
1990	9	9.20
1990	10	9.20
1990	11	9.20
1990	12	9.20
1991	1	8.60
1991	2	8.20
1991	3	8.30
1991	4	8.00

Selecting all the values where inflation is lower than the average inflation.

```
SELECT *
```

```
FROM inflation_years
```

```
WHERE inflation < (SELECT AVG(inflation) FROM inflation_years);
```

The screenshot shows a database management interface. On the left, a 'SCHEMAS' panel lists the 'maverick' database with tables including 'inflation_years'. The main area displays a SQL query:

```
1 SELECT *
2 FROM inflation_years
3 WHERE inflation < (SELECT AVG(inflation) FROM inflation_years);
4
```

 Below the query, a 'Result Grid' shows the data for the 'inflation_years' table. The grid has columns for Year, Month, and Inflation. The data shows inflation values for the years 1993 and 1994, with the 1994 data being filtered based on the query.

Year	Month	Inflation
1993	4	2.60
1993	5	2.30
1993	6	2.20
1993	7	2.30
1993	8	2.60
1993	9	2.70
1993	10	2.40
1993	11	2.10
1993	12	2.30
1994	1	2.60
1994	2	2.40
1994	3	2.20
1994	4	2.40
1994	5	2.40
1994	6	2.30
1994	7	2.10

Connection Details:
Name: maverick
Host: localhost