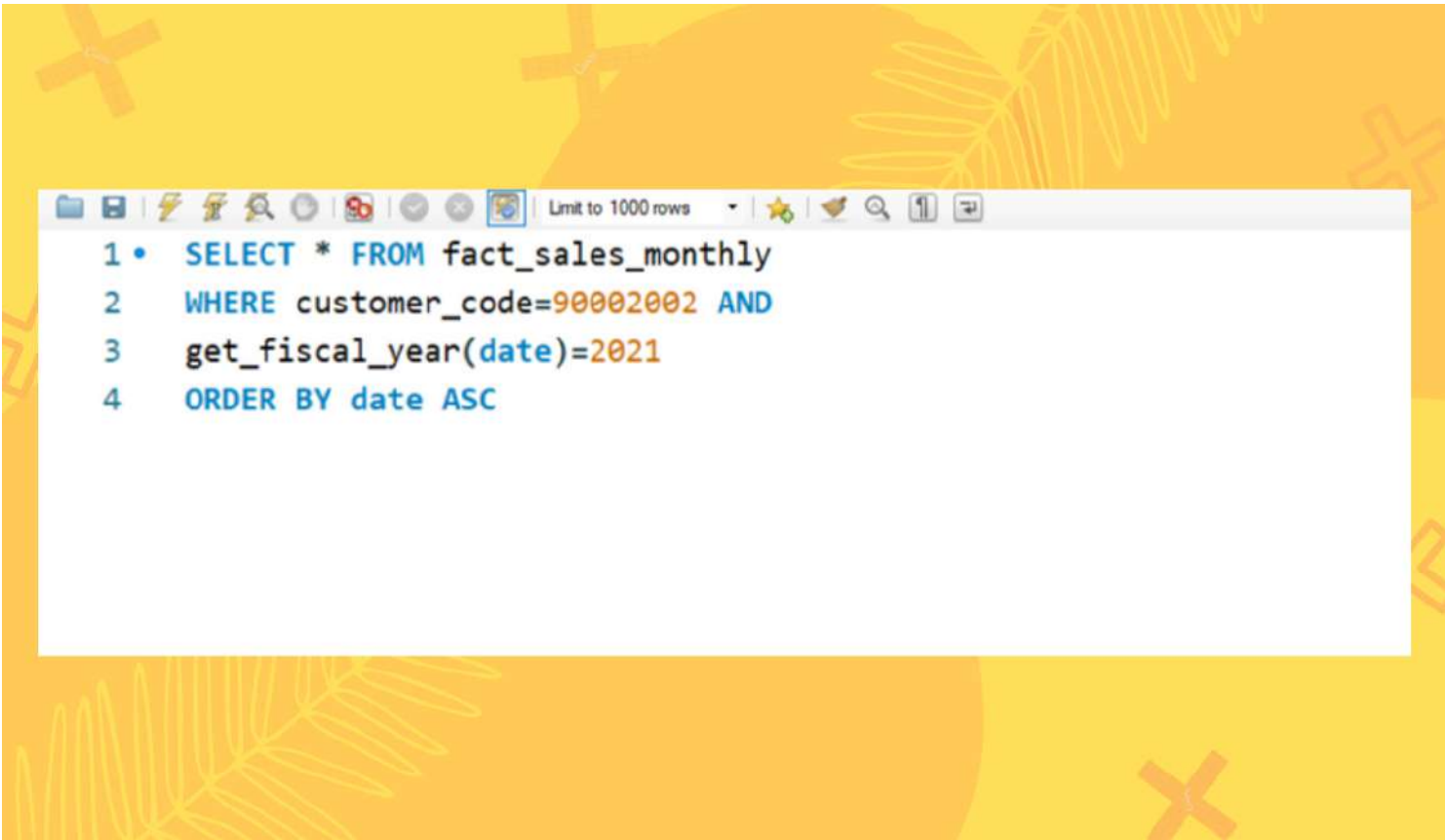


**Generate a report of  
individual product sales  
(aggregated on a monthly  
basis at the product code  
level) for Fiscal Year 2021**





```
1 • SELECT * FROM fact_sales_monthly
2   WHERE customer_code=90002002 AND
3     get_fiscal_year(date)=2021
4   ORDER BY date ASC
```



Name: `get_fiscal_year`

The name of the routine is parsed automatically from the DDL statement. The DDL is parsed automatically while you type.

DDL:

```
1 • CREATE DEFINER=`root`@`localhost`  
2 FUNCTION `get_fiscal_year`  
3 (calender_date DATE) RETURNS INT  
4     DETERMINISTIC  
5 BEGIN  
6     DECLARE fiscal_year INT;  
7     SET fiscal_year= YEAR DATE_ADD(calender_date,  
8     INTERVAL 4 MONTH));  
9     RETURN fiscal_year;  
10 END
```



Name: get\_fiscal\_year

The name of the routine is parsed automatically from the DDL statement. The DDL is parsed automatically while you type.

DDL:



```
1 • CREATE DEFINER=`root`@`localhost` FUNCTION `ge  
2 DETERMINISTIC
```

Call stored function gdb0041.get\_fiscal\_year

Enter values for parameters of your function and click <Execute> to create an SQL editor and run the call:

**calender\_date**  DATE

alender\_date,

1 • `select gdb0041.get_fiscal_year('2020-09-01');`

Result Grid

Filter Rows:

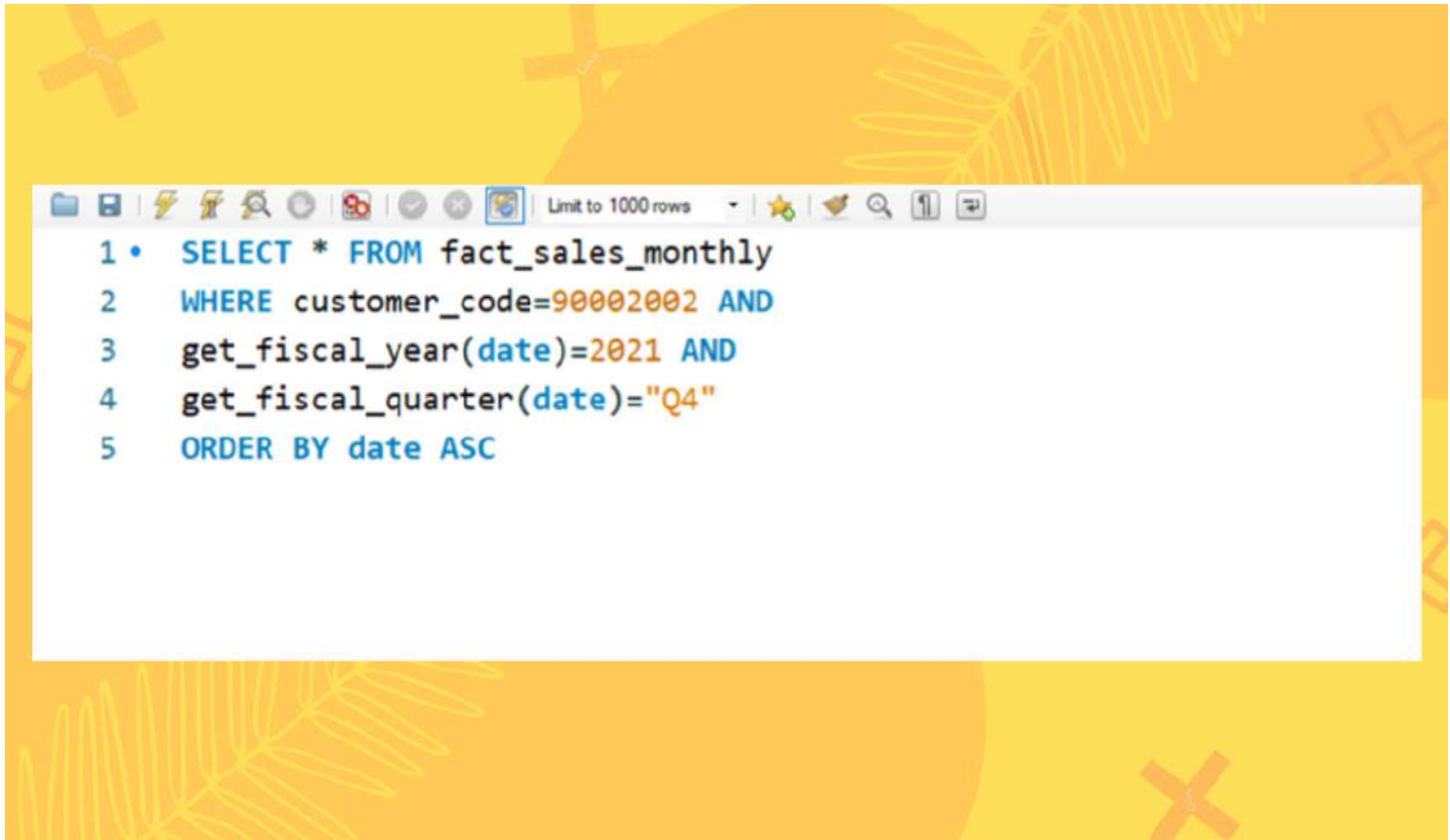
Export:

Wrap Cell Content:

gdb0041.get_fiscal_year('2020-09-01')	
▶	2021

Result 1 x

Read Only



The image shows a screenshot of a database client application window. The window has a title bar with standard OS icons and a toolbar with various database-related icons. Below the toolbar, the text "Limit to 1000 rows" is visible. The main area of the window contains a SQL query, which is numbered 1 through 5 on the left side. The query is as follows:

```
1 • SELECT * FROM fact_sales_monthly
2 WHERE customer_code=90002002 AND
3 get_fiscal_year(date)=2021 AND
4 get_fiscal_quarter(date)="Q4"
5 ORDER BY date ASC
```



Name: get\_quarter\_year

The name of the routine is parsed automatically from the DDL statement. The DDL is parsed automatically while you type.

DDL:



```
1 • CREATE DEFINER='root'@'localhost' FUNCTION `get_quarter_year`  
2   (calender_date DATE) RETURNS char(2) CHARSET utf8mb4  
3   DETERMINISTIC  
4   BEGIN  
5     DECLARE m TINYINT;  
6     DECLARE qtr CHAR(2);  
7     SET m= MONTH(calender_date);  
8     CASE WHEN m IN (9,10,11) THEN  
9       SET qtr="Q1";  
10    WHEN m IN (12,1,2) THEN  
11      SET qtr="Q2";  
12    WHEN m IN (3,4,5) THEN  
13      SET qtr="Q3";  
14    ELSE SET qtr="Q4";  
15    END CASE;  
16    RETURN qtr;  
17  END
```



Name:

The name of the routine is parsed automatically from the DDL statement. The DDL is parsed automatically while you type.

DDL:



```
1 • CREATE DEFINER='root'@'localhost' FUNCTION `get_quarter_year`  
2 (calender_date DATE) RETURNS char(2) CHARSET utf8mb4
```

Call stored function gdb0041.get\_quarter\_year

Enter values for parameters of your function and click <Execute> to create an SQL editor and run the call:

**calender\_date**  **DATE**

```
12 WHEN m IN (3,4,5) THEN  
13 SET qtr="Q3";  
14 ELSE SET qtr="Q4";  
15 END CASE;  
16 RETURN qtr;  
17 END
```



Limit to 1000 rows

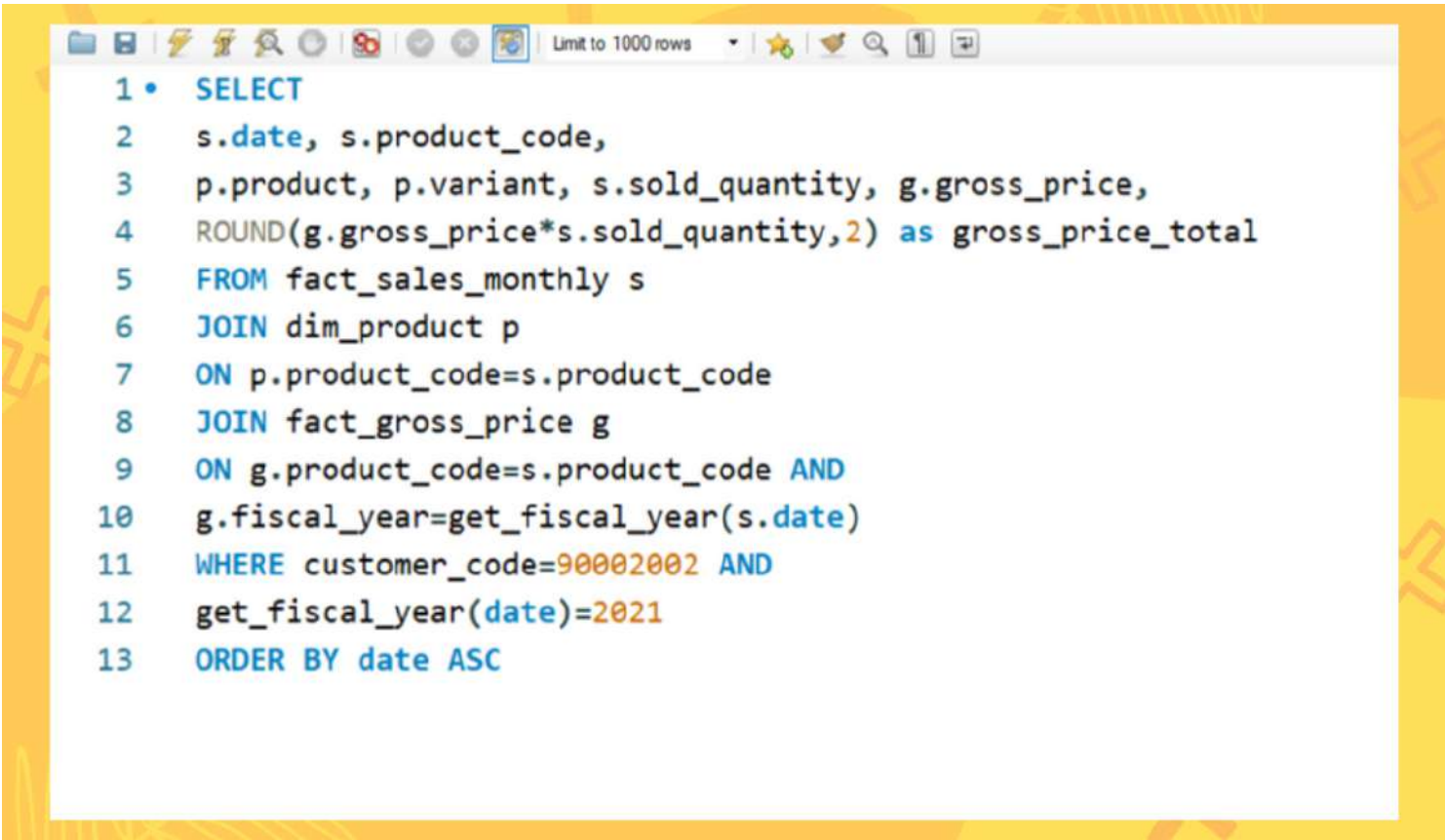
```
1 • select gdb0041.get_quarter_year('2020-09-01');
```

Result Grid

	<code>gdb0041.get_quarter_year('2020-09-01')</code>
▶	Q1

Result 1 x

Read Only



```
1 • SELECT
2   s.date, s.product_code,
3   p.product, p.variant, s.sold_quantity, g.gross_price,
4   ROUND(g.gross_price*s.sold_quantity,2) as gross_price_total
5 FROM fact_sales_monthly s
6 JOIN dim_product p
7 ON p.product_code=s.product_code
8 JOIN fact_gross_price g
9 ON g.product_code=s.product_code AND
10 g.fiscal_year=get_fiscal_year(s.date)
11 WHERE customer_code=90002002 AND
12 get_fiscal_year(date)=2021
13 ORDER BY date ASC
```

```

5 FROM fact_sales_monthly s
6 JOIN dim_product p
7 ON p.product_code=s.product_code
8 TOTAL fact_gross_price p

```

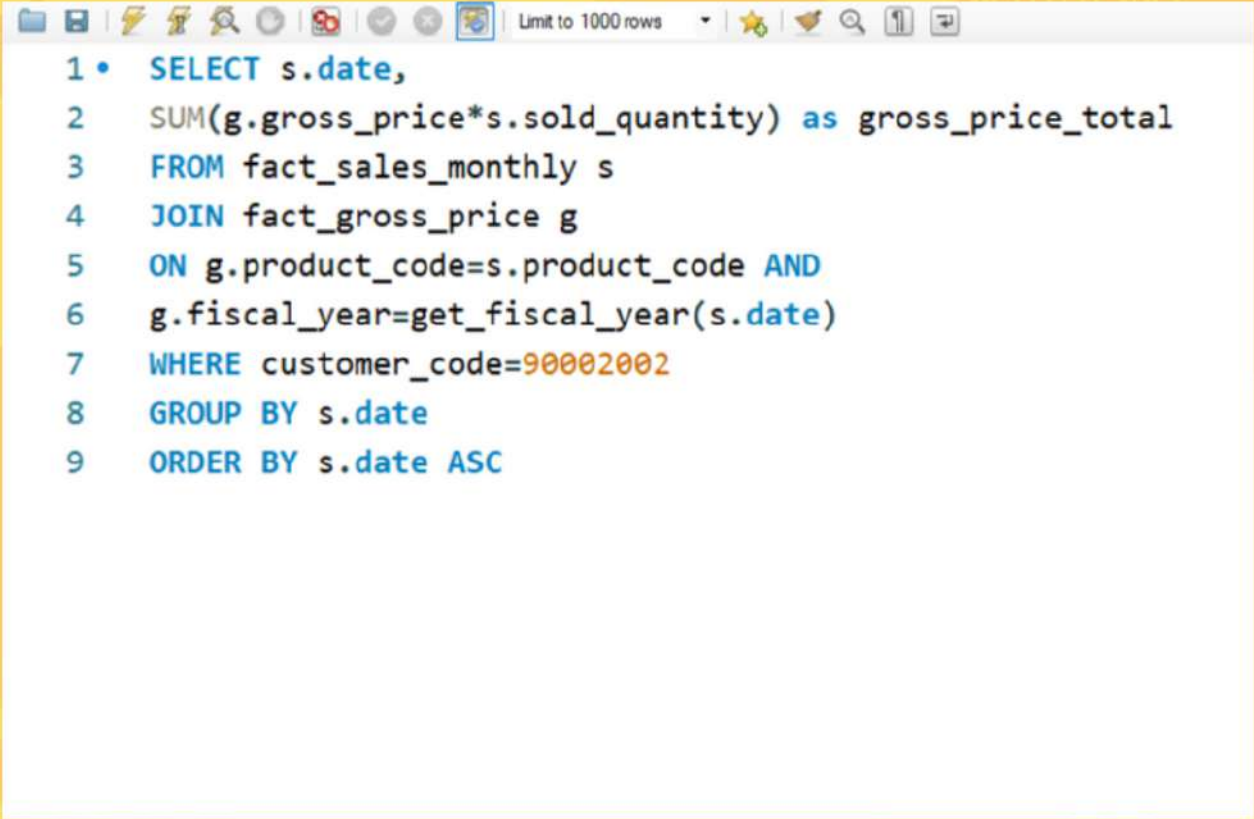
date	product_code	product	variant	sold_quantity	gross_price	gross_price_total
2020-09-01	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 R...	Standard	202	19.0573	3849.57
2020-09-01	A4419110403	AQ Elite	Standard Red	16	288.0503	4608.80
2020-09-01	A2720150701	AQ Trigger Ms	Standard 1	822	17.0917	14049.38
2020-09-01	A4218110204	AQ Digit	Plus Grey	27	232.1038	6266.80
2020-09-01	A5419110205	AQ Gamer 2	Plus Cool Blue	7	570.7578	3995.30
2020-09-01	A5419110206	AQ Gamer 2	Plus Black	4	601.6398	2406.56
2020-09-01	A3220150401	AQ Lite	Standard 1	197	18.4943	3643.38
2020-09-01	A5419110204	AQ Gamer 2	Plus Firey Red	5	602.9200	3014.60
2020-09-01	A2620150606	AQ Qwerty Ms	Premium 2	688	16.7850	11548.08
2020-09-01	A0118150102	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 R...	Plus	162	21.4565	3475.95
2020-09-01	A4319110304	AQ Velocity	Plus Grey	40	267.0636	10682.54
2020-09-01	A5419110207	AQ Gamer 2	Premium Black	5	599.2302	2996.15
2020-09-01	A2721150702	AQ Trigger Ms	Standard 2	171	17.2368	2947.49
2020-09-01	A2021150503	AQ MB Lito 2	Plus 2	17	45.4377	772.44
2020-09-01	A5419110208	AQ Gamer 2	Premium Mist...	4	608.4070	2433.63
2020-09-01	A3718150103	AQ LION x1	Plus 2	28	17.5697	491.95

Result 1 x

Read Only

**Generate an aggregate  
monthly gross sales report**





```
1 • SELECT s.date,  
2 SUM(g.gross_price*s.sold_quantity) as gross_price_total  
3 FROM fact_sales_monthly s  
4 JOIN fact_gross_price g  
5 ON g.product_code=s.product_code AND  
6 g.fiscal_year=get_fiscal_year(s.date)  
7 WHERE customer_code=90002002  
8 GROUP BY s.date  
9 ORDER BY s.date ASC
```

1 • **SELECT** s.date,  
2 SUM(g.gross\_price\*s.sold\_quantity) **as** gross\_price\_total  
3 **FROM** fact\_sales\_monthly s  
4 **JOIN** fact\_gross\_price g  
5 **ON** g.product\_code=s.product\_code **AND**  
6 g.fiscal\_year=get\_fiscal\_year(s.date)  
7 **WHERE** customer\_code=90002002  
8 **GROUP BY** s.date  
9 **ORDER BY** s.date ASC

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

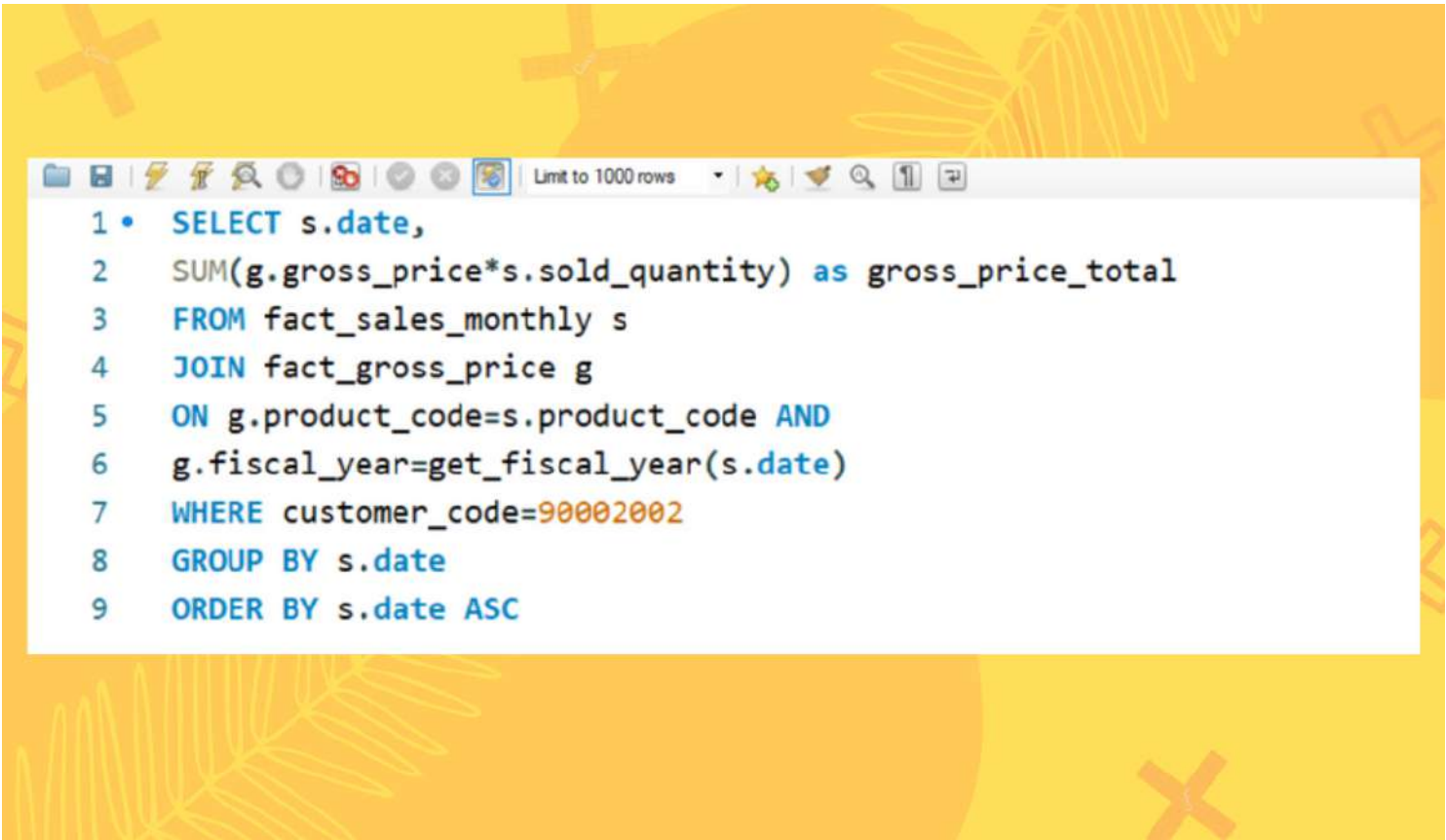
	date	gross_price_total
▶	2017-09-01	122407.5582
	2017-10-01	162687.5716
	2017-12-01	245673.8042
	2018-01-01	127574.7372
	2018-02-01	144799.5182
	2018-04-01	130643.8976
	2018-05-01	139165.0975

Result 1 x

Result Grid

Form Editor

Read Only



The image shows a screenshot of a SQL query editor window. The window has a light gray title bar with various icons on the left and a dropdown menu showing "Limit to 1000 rows" on the right. The main area of the window is white and contains a SQL query. The query is numbered 1 through 9 on the left side. The query text is as follows:

```
1 • SELECT s.date,  
2 SUM(g.gross_price*s.sold_quantity) as gross_price_total  
3 FROM fact_sales_monthly s  
4 JOIN fact_gross_price g  
5 ON g.product_code=s.product_code AND  
6 g.fiscal_year=get_fiscal_year(s.date)  
7 WHERE customer_code=90002002  
8 GROUP BY s.date  
9 ORDER BY s.date ASC
```



Name: `get_monthly_gross_sales_for_customer`

The name of the routine is parsed automatically from the DDL statement. The DDL is parsed automatically while you type.

DDL:



```
1 • CREATE DEFINER=`root`@`localhost`  
2 PROCEDURE `get_monthly_gross_sales_for_customer`  
3 (in_customer_codes TEXT)  
4 BEGIN  
5 SELECT s.date,  
6 SUM(ROUND(s.sold_quantity*g.gross_price,2)) as monthly_sales  
7 FROM fact_sales_monthly s  
8 JOIN fact_gross_price g  
9 ON g.fiscal_year=get_fiscal_year(s.date)  
10 AND g.product_code=s.product_code  
11 WHERE FIND_IN_SET(s.customer_code, in_customer_codes)>0  
12 GROUP BY date;  
13 END
```





Name: `get_monthly_gross_sales_for_customer`

The name of the routine is parsed automatically from the DDL statement. The DDL is parsed automatically while you type.

DDL:



```
1 • CREATE DEFINER=`root`@`localhost`  
2 PROCEDURE `get_monthly_gross_sales_for_customer`
```

Call stored procedure gdb0041.get\_monthly\_gross\_sales\_f... — □ ×

Enter values for parameters of your procedure and click <Execute> to create an SQL editor and run the call:

`in_customer_codes`  [IN] TEXT

Execute

Cancel

```
11 WHERE FIND_IN_SET(s.customer_code, in_customer_codes)>0  
12 GROUP BY date;  
13 END
```

as monthly\_sales

Limit to 1000 rows

```
1 • gdb0041.get_monthly_gross_sales_for_customer('90002002');
```

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

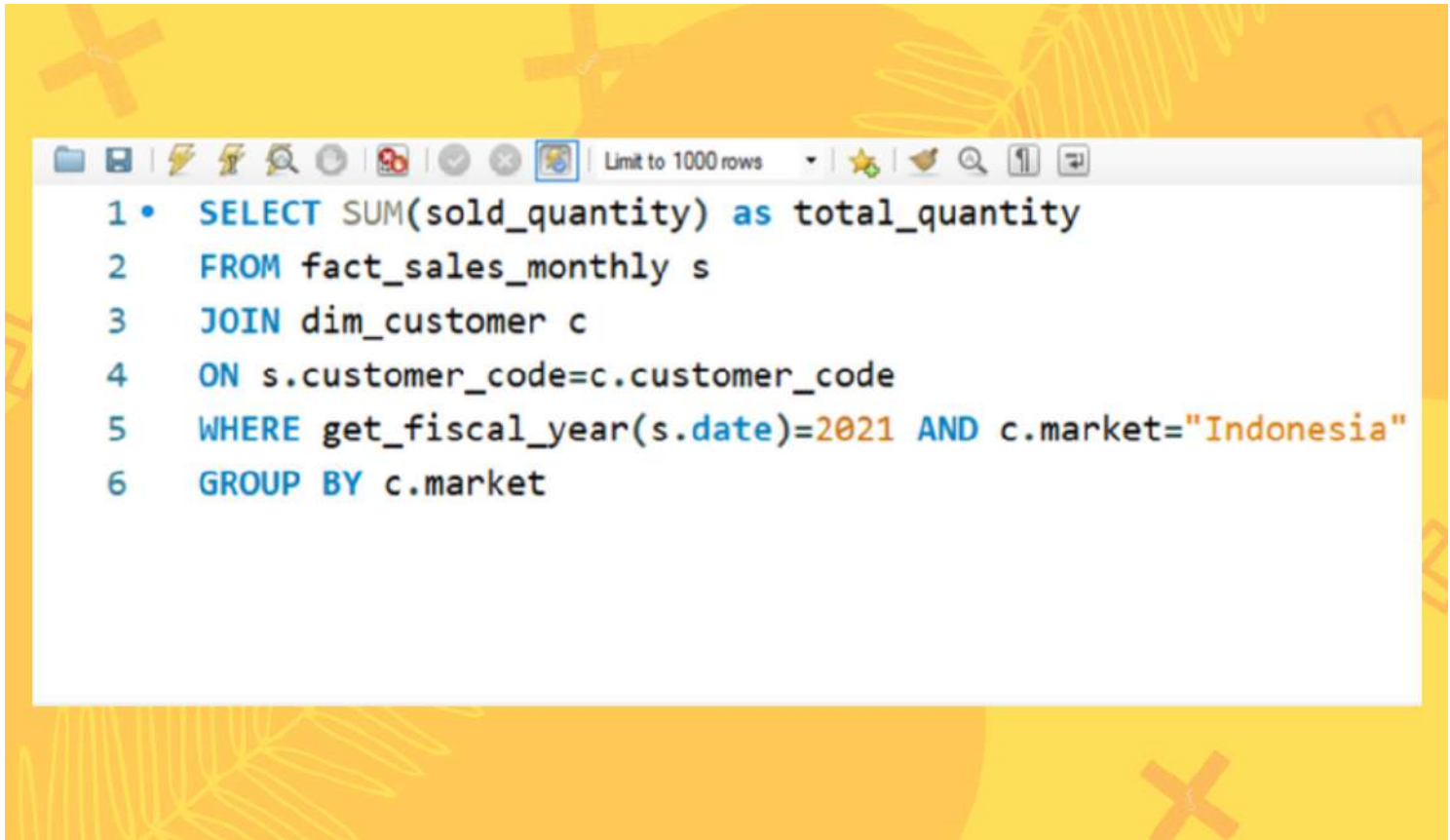
	date	monthly_sales
▶	2017-09-01	122407.57
	2017-10-01	162687.56
	2017-12-01	245673.84
	2018-01-01	127574.73
	2018-02-01	144799.54
	2018-04-01	130643.92
	2018-05-01	139165.06

Result 1 x | Read Only

**Create a stored procedure  
thatt can determine the  
market badge on following  
logic.**

**If total quantity > 5 million  
that market is considered  
Gold else it is Silver**





```
1 • SELECT SUM(sold_quantity) as total_quantity
2 FROM fact_sales_monthly s
3 JOIN dim_customer c
4 ON s.customer_code=c.customer_code
5 WHERE get_fiscal_year(s.date)=2021 AND c.market="Indonesia"
6 GROUP BY c.market
```