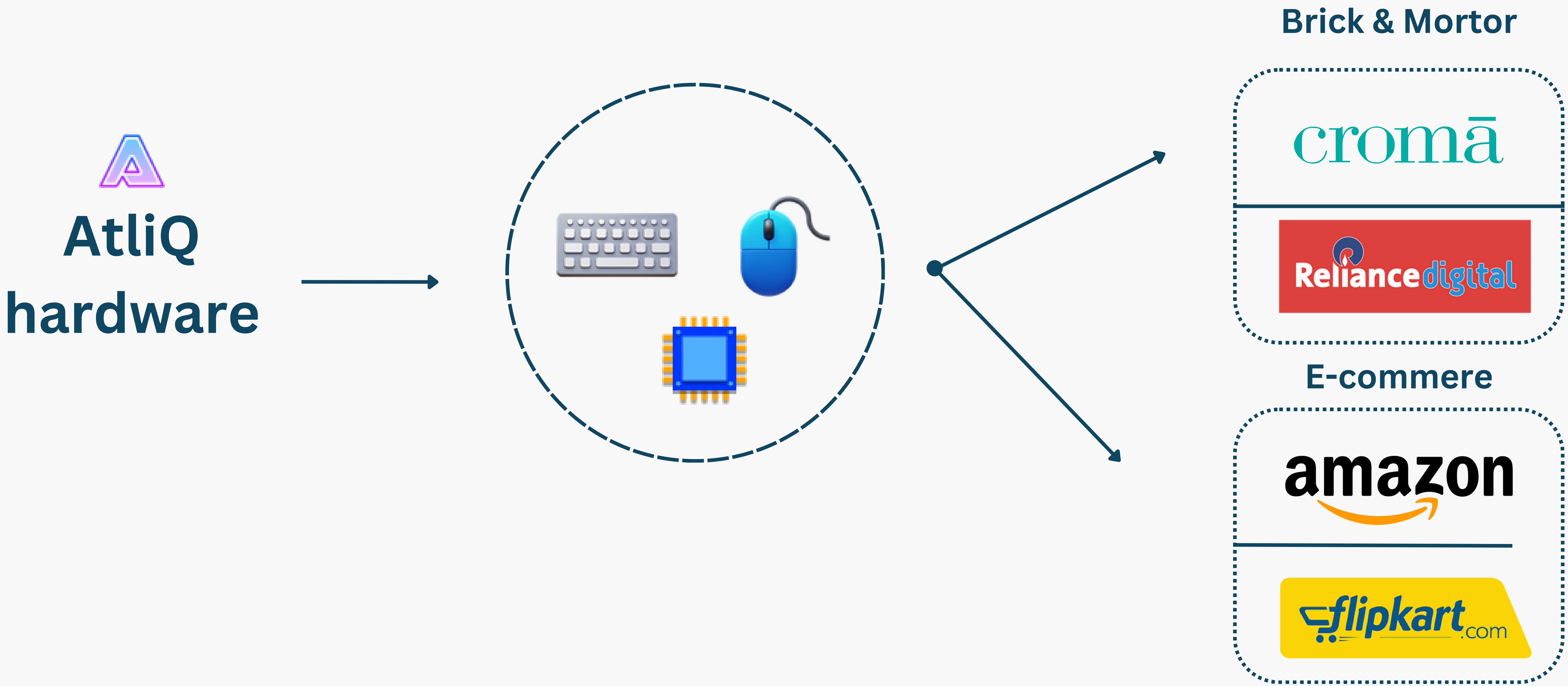
Welcome

ad-hoc project SQL

AD-HOC

In business, an ad-hoc project might be initiated to solve an **unexpected problem**, respond to a **sudden opportunity**, or **fulfill a specific, short-term objective**. Unlike regular projects, ad-hoc ones are usually one-time and may not follow the standard project management processes.

Business Model



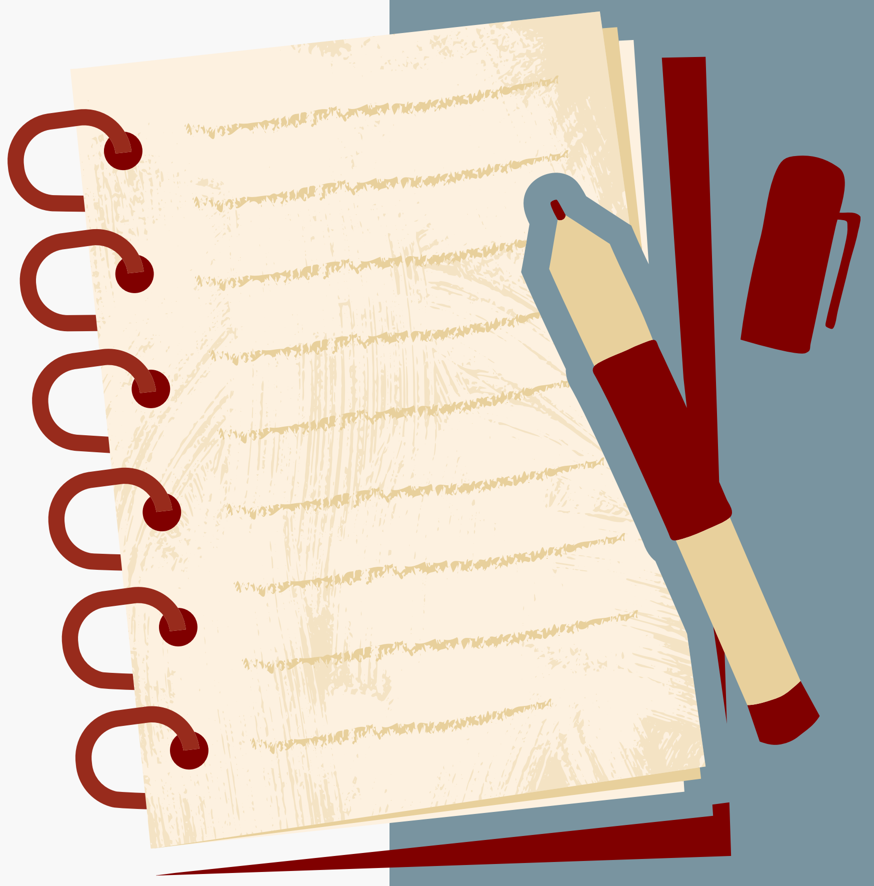
AtliQ hardware

Brick & Mortor

E-commere

Objectives

- Solve unexpected issues quickly.
- Provide a flexible response to urgent needs.
- Focus on short-term, time-sensitive goals.
- Offer tailored solutions for unique situations.
- Ensure efficiency when standard processes don't apply.



Datasets

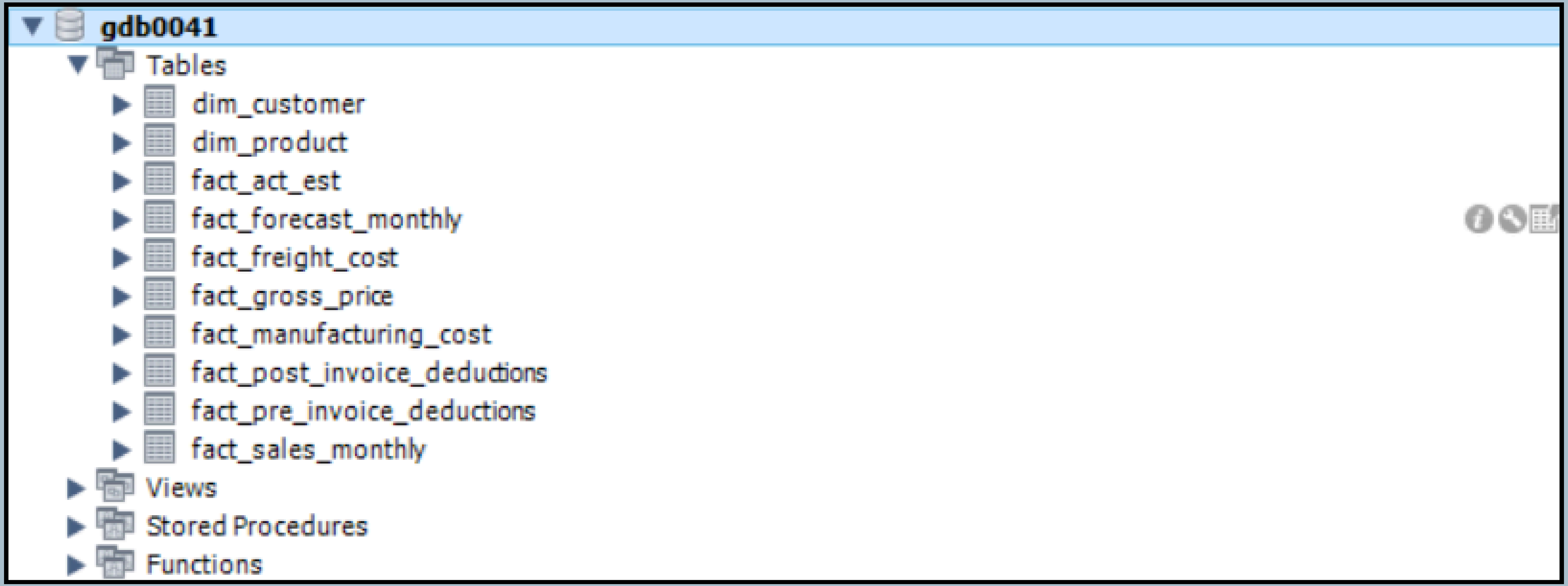
Dimension tables

- dim_customers
- dim_products

Fact tables

- fact_sales_monthly
- fact_forecast_monthly
- fact_freight_cost
- fact_gross_price
- fact_manufacturing_cost
- fact_pre_invoice_deductions
- fact_post_invoice_deductions

Datasets



- gdb0041
  - Tables
    - dim_customer
    - dim_product
    - fact_act_est
    - fact_forecast_monthly
    - fact_freight_cost
    - fact_gross_price
    - fact_manufacturing_cost
    - fact_post_invoice_deductions
    - fact_pre_invoice_deductions
    - fact_sales_monthly
  - Views
  - Stored Procedures
  - Functions

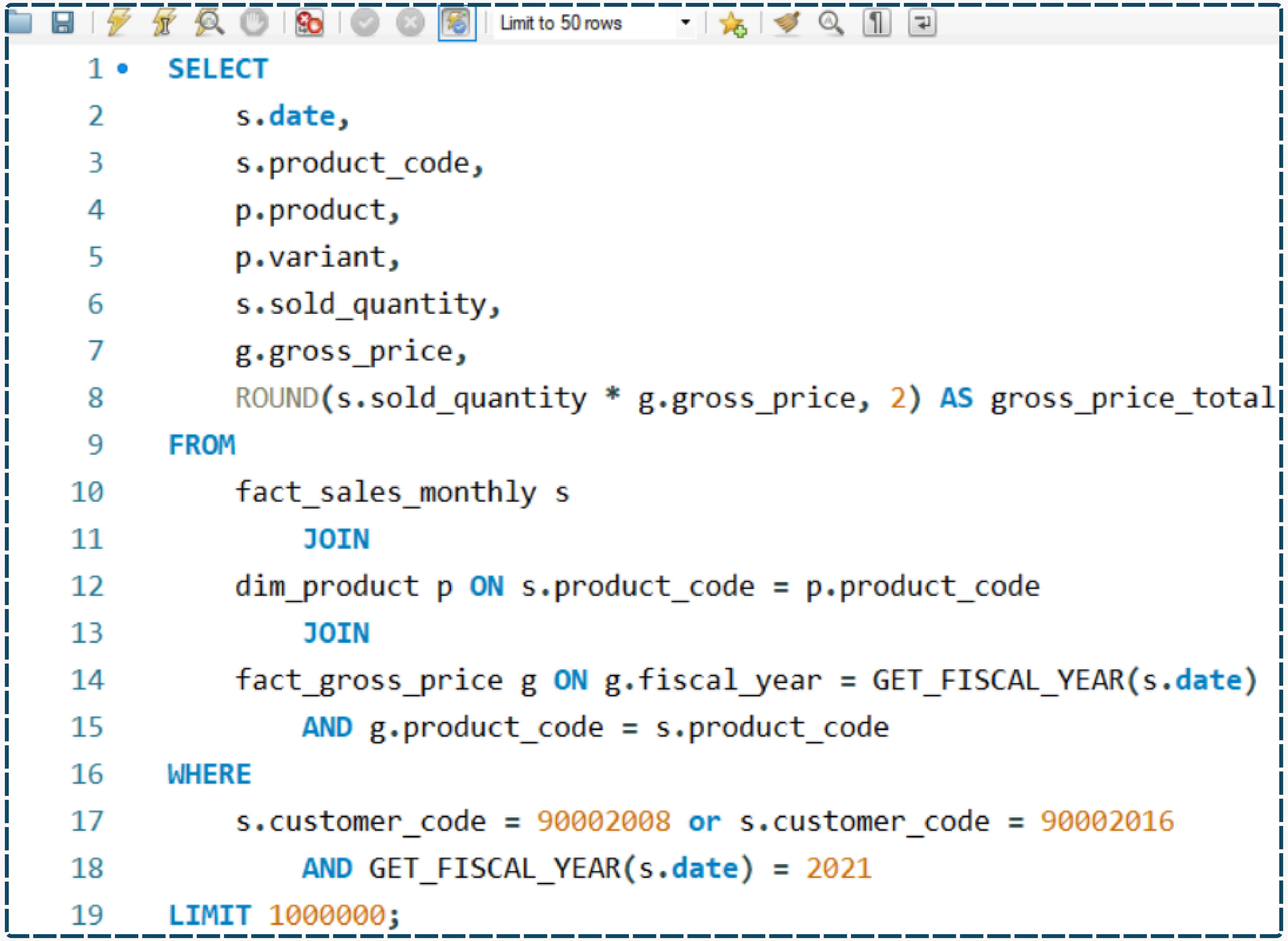
ad-hoc request 1

Description:

As a product owner, I want to generate a report of individual product sales (aggregated on a monthly basis at the product level) for **Amazon India customers for FY=2021** so that I can track individual product sales.

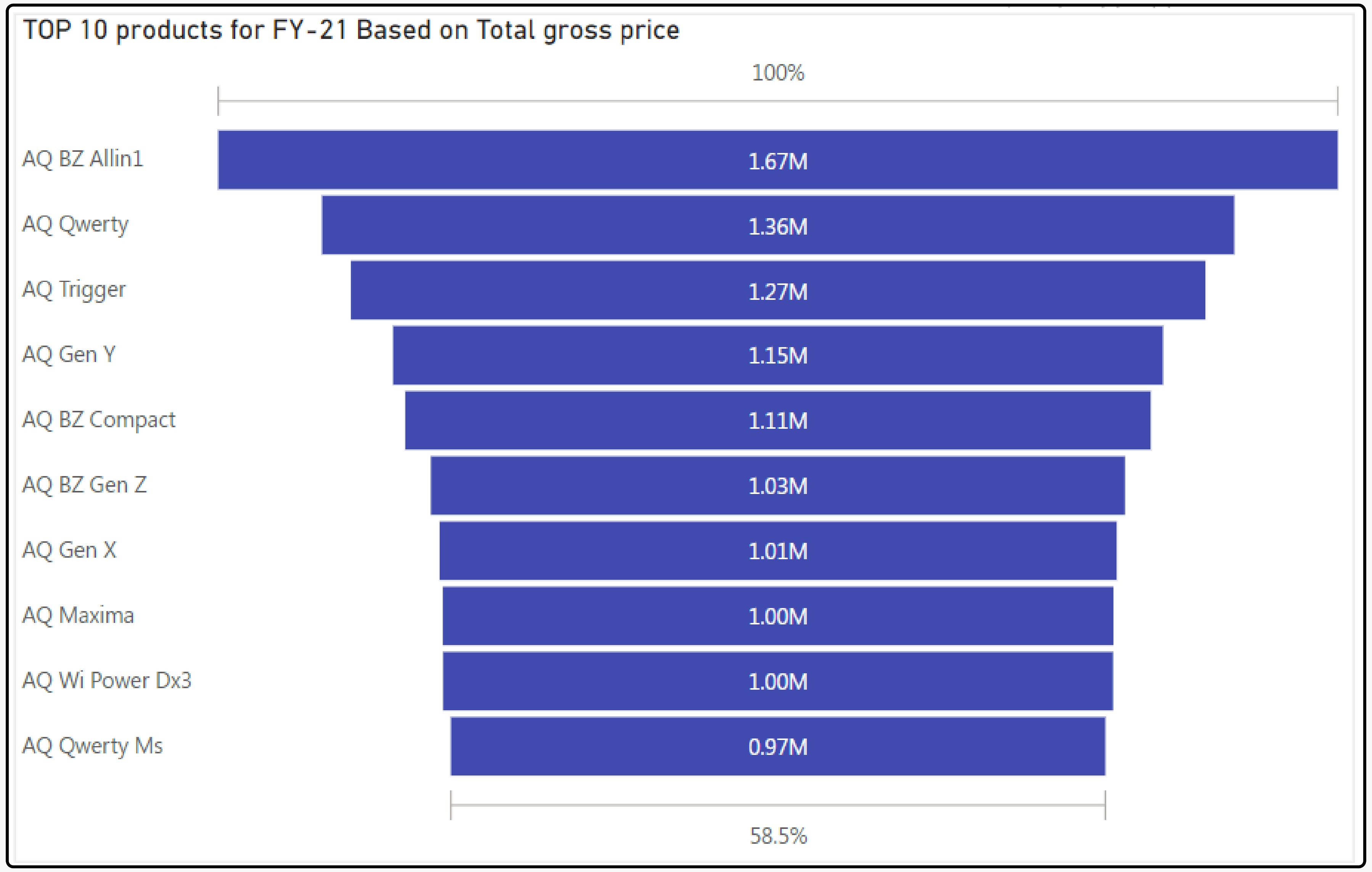
The report should have the following fields,

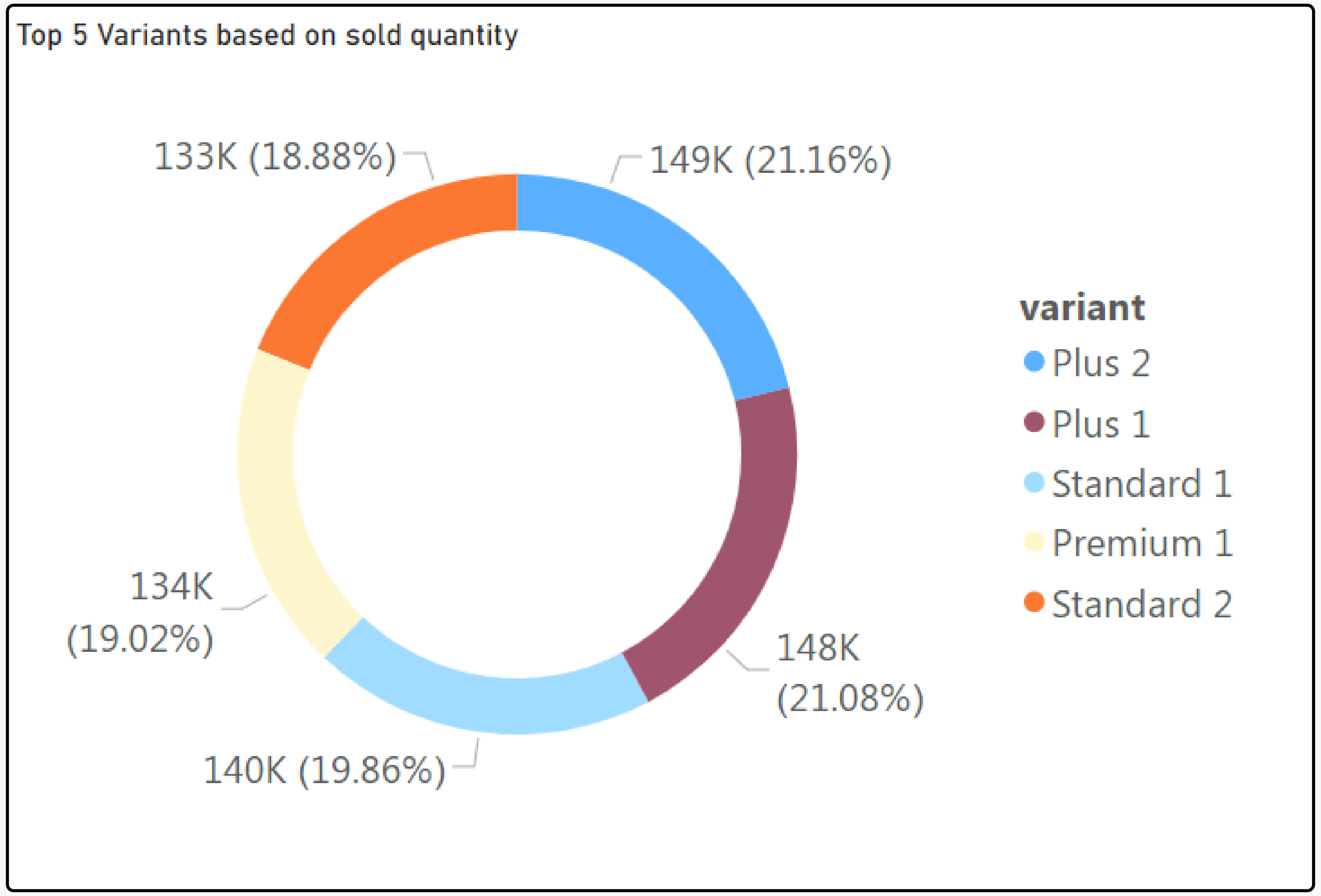
1. Month
2. Product Name & Variant
3. Sold Quantity
4. Gross Price Per Item
5. Gross Price Total



```
SELECT
    s.date,
    s.product_code,
    p.product,
    p.variant,
    s.sold_quantity,
    g.gross_price,
    ROUND(s.sold_quantity * g.gross_price, 2) AS gross_price_total
FROM
    fact_sales_monthly s
        JOIN
    dim_product p ON s.product_code = p.product_code
        JOIN
    fact_gross_price g ON g.fiscal_year = GET_FISCAL_YEAR(s.date)
        AND g.product_code = s.product_code
WHERE
    s.customer_code = 90002008 or s.customer_code = 90002016
        AND GET_FISCAL_YEAR(s.date) = 2021
LIMIT 1000000;
```

Insights & Findings


TOP 10 products for FY-21 Based on Total gross price


Top 5 Variants based on sold quantity

- The top-selling product is **AQ BZ Allin1**, with **1.67** Million.
- The second highest-selling product is **AQ Qwerty**, with **1.36** Million.

- The top selling variant is **Plus 2**, which accounts for **21.16%** of the total sold quantity.
- The second highest selling variant is **Plus 1**, which accounts for **21.08%** of the total sold quantity.

ad-hoc request 2

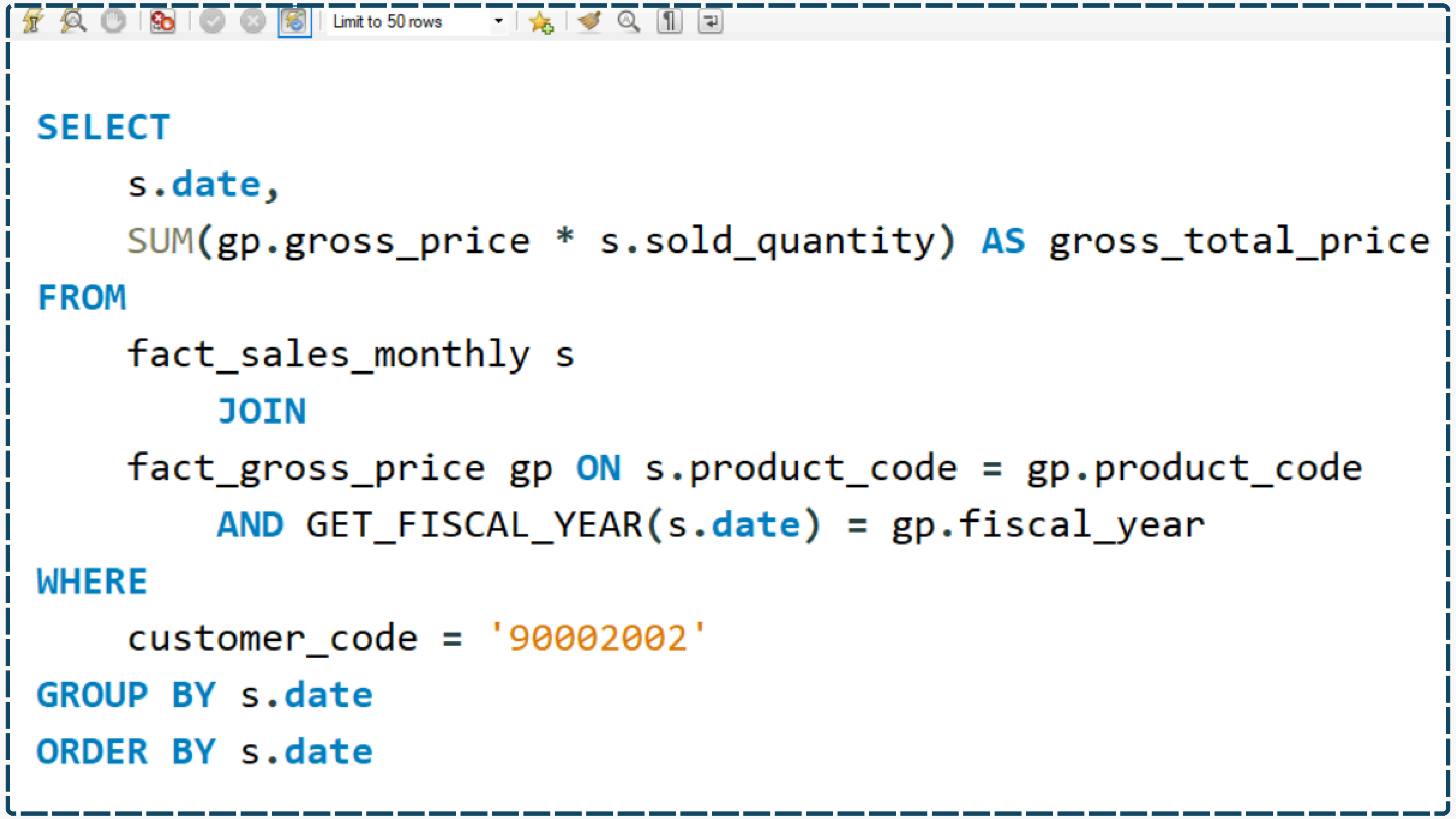
Description:

As a product owner, I need an aggregate monthly gross sales report for **Croma India** customer so that I can track how much sales this particular customer is generating for AtliQ and manage our relationships accordingly.

The report should have the following fields,

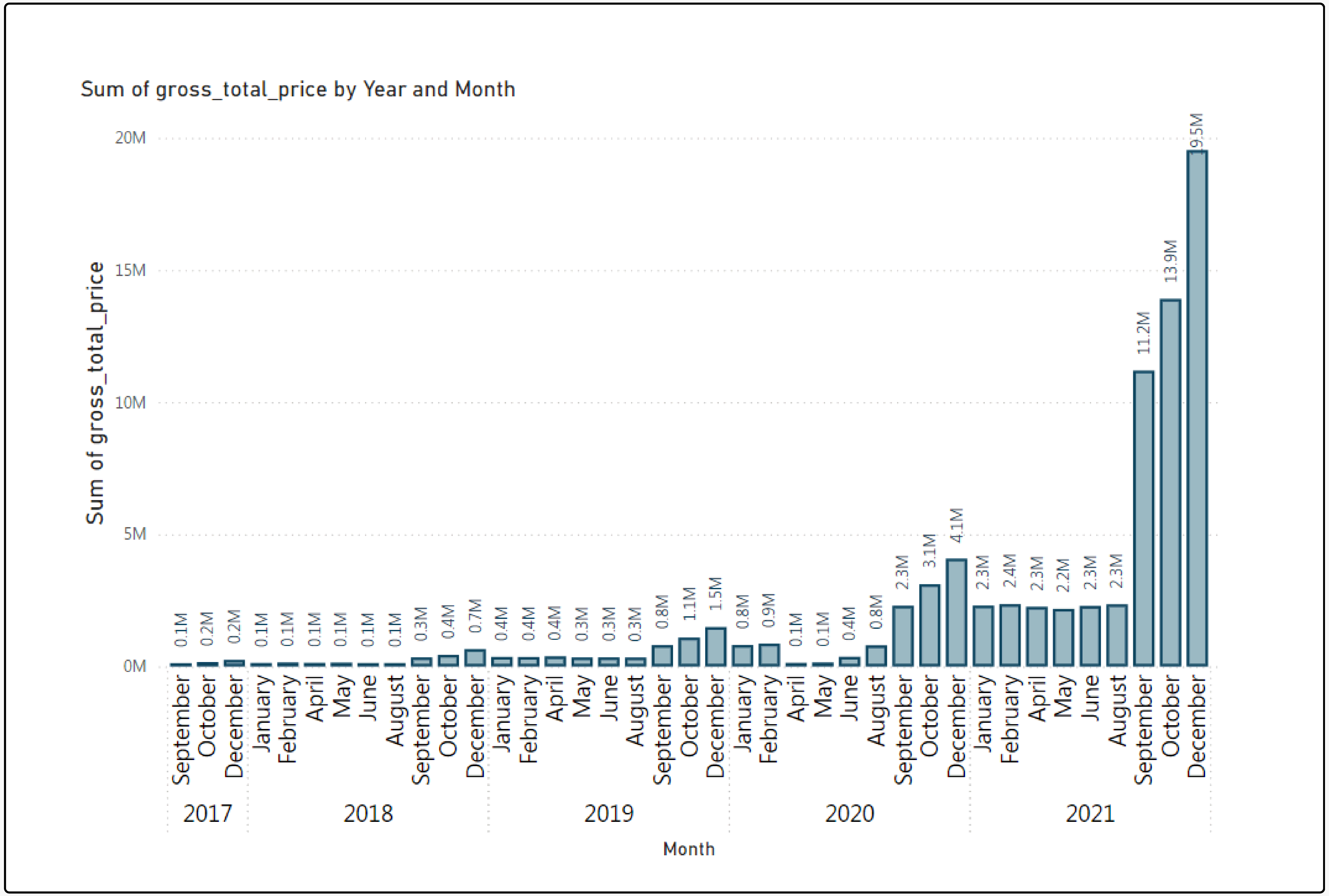
1. Month

2. Total gross sales amount to Croma India in this month



```
SELECT
    s.date,
    SUM(gp.gross_price * s.sold_quantity) AS gross_total_price
FROM
    fact_sales_monthly s
        JOIN
    fact_gross_price gp ON s.product_code = gp.product_code
        AND GET_FISCAL_YEAR(s.date) = gp.fiscal_year
WHERE
    customer_code = '90002002'
GROUP BY s.date
ORDER BY s.date
```

Insights & Findings



For **croma** the highest total price of around **19.5M** is observed in December **2021**, which is significantly higher than the other months and years.

ad-hoc request 3

Description:
Create a **stored procedure** for monthly gross sales reports so that I don't have to manually modify the query every time so that I generate this report without involving the data analytics team.

The report should have the following columns,
1. Month
2. Total gross sales in that month from a given customer

STORED PROCEDURE:

- A stored procedure in MySQL is a prepared SQL code that you can **save and reuse.**
- Allows to Automate the process
- Provides more security since users don't need to have direct access to SQL statements.

ad-hoc request 3

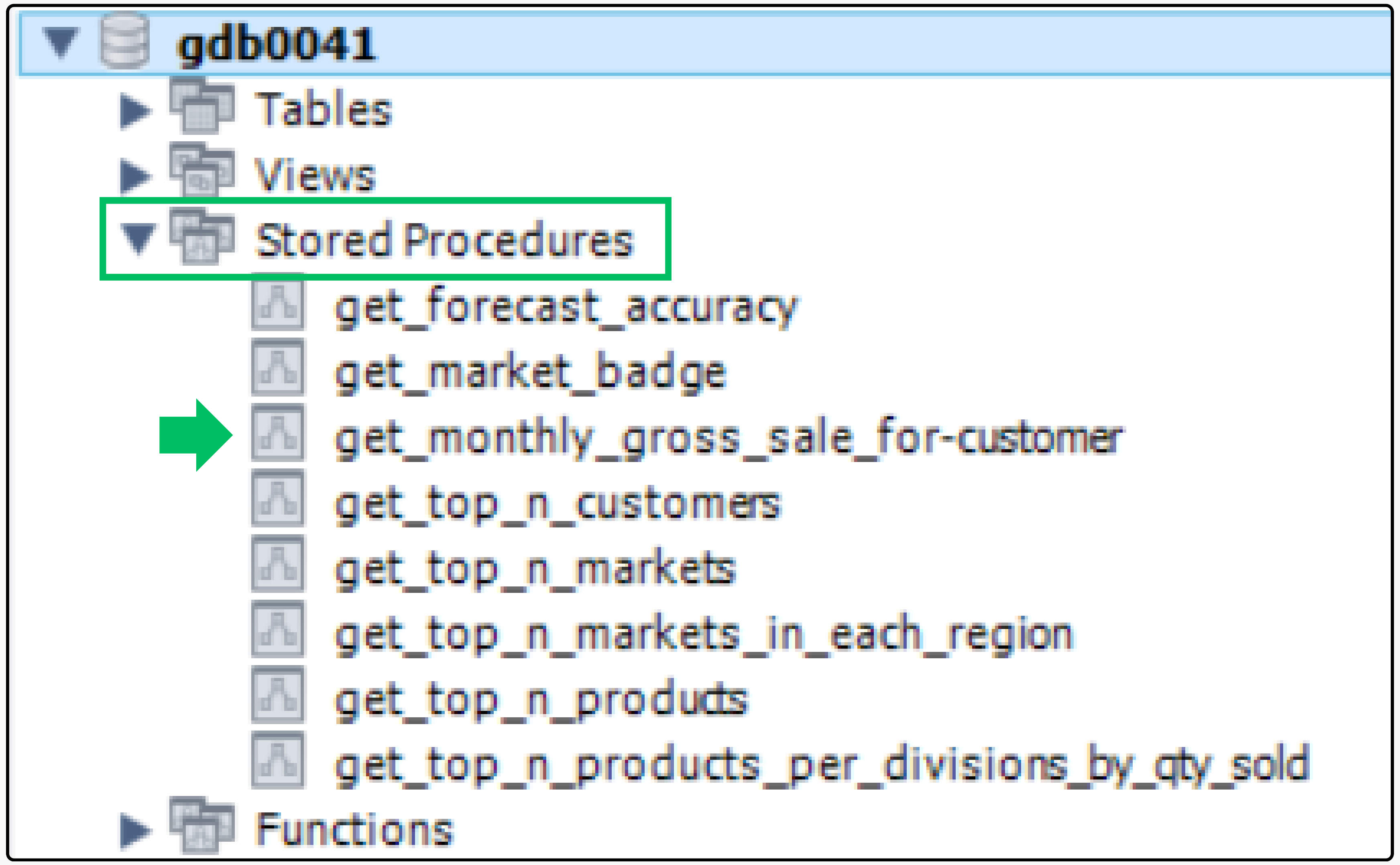
Description:

Create a **stored procedure** for monthly gross sales reports so that I don't have to manually modify the query every time so that I generate this report without involving the data analytics team.

The report should have the following columns,

1. Month

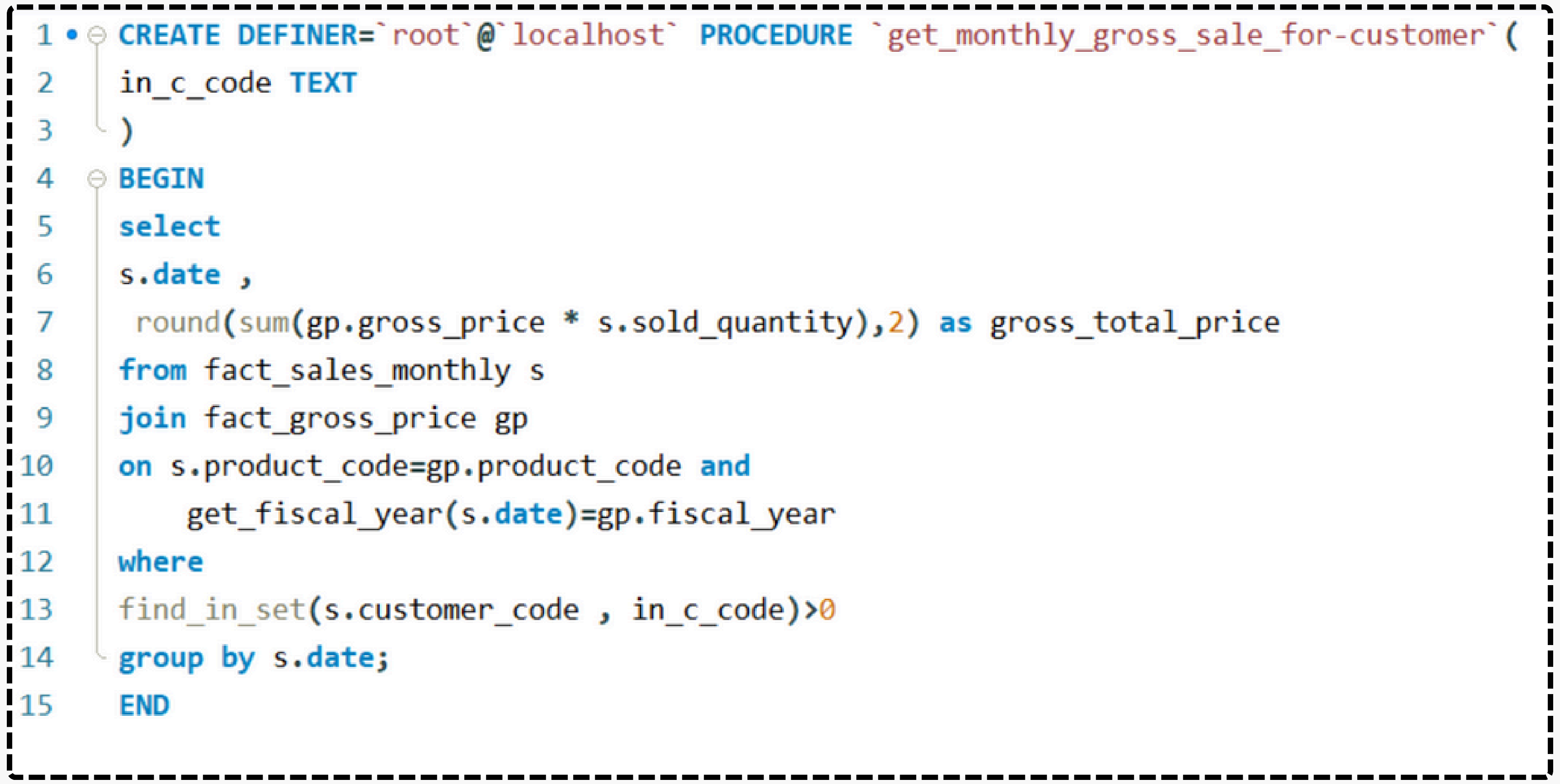
2. Total gross sales in that month from a given customer



gdb0041

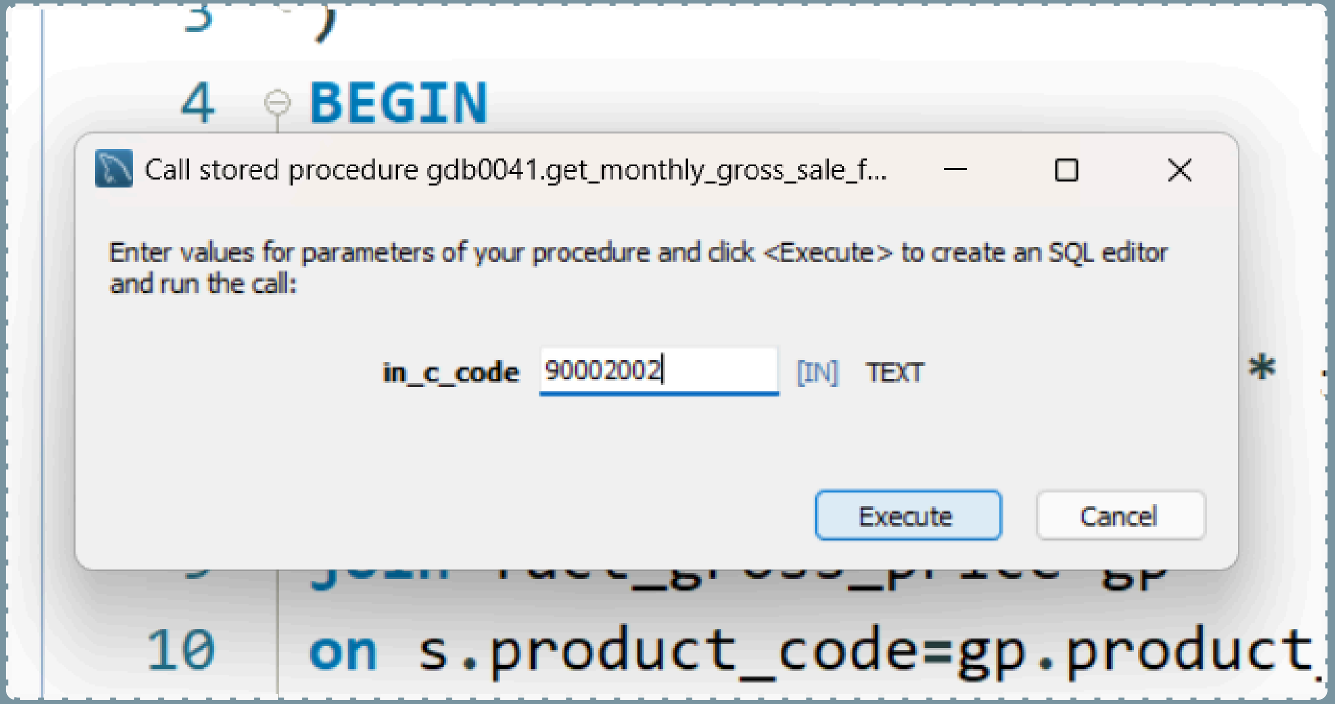
- Tables
- Views
- Stored Procedures
  - get_forecast_accuracy
  - get_market_badge
  - get_monthly_gross_sale_for-customer
  - get_top_n_customers
  - get_top_n_markets
  - get_top_n_markets_in_each_region
  - get_top_n_products
  - get_top_n_products_per_divisions_by_qty_sold
- Functions

QUERY



```
CREATE DEFINER=`root`@`localhost` PROCEDURE `get_monthly_gross_sale_for-customer`(
in_c_code TEXT
)
BEGIN
select
s.date ,
 round(sum(gp.gross_price * s.sold_quantity),2) as gross_total_price
from fact_sales_monthly s
join fact_gross_price gp
on s.product_code=gp.product_code and
    get_fiscal_year(s.date)=gp.fiscal_year
where
find_in_set(s.customer_code , in_c_code)>0
group by s.date;
END
```

Result

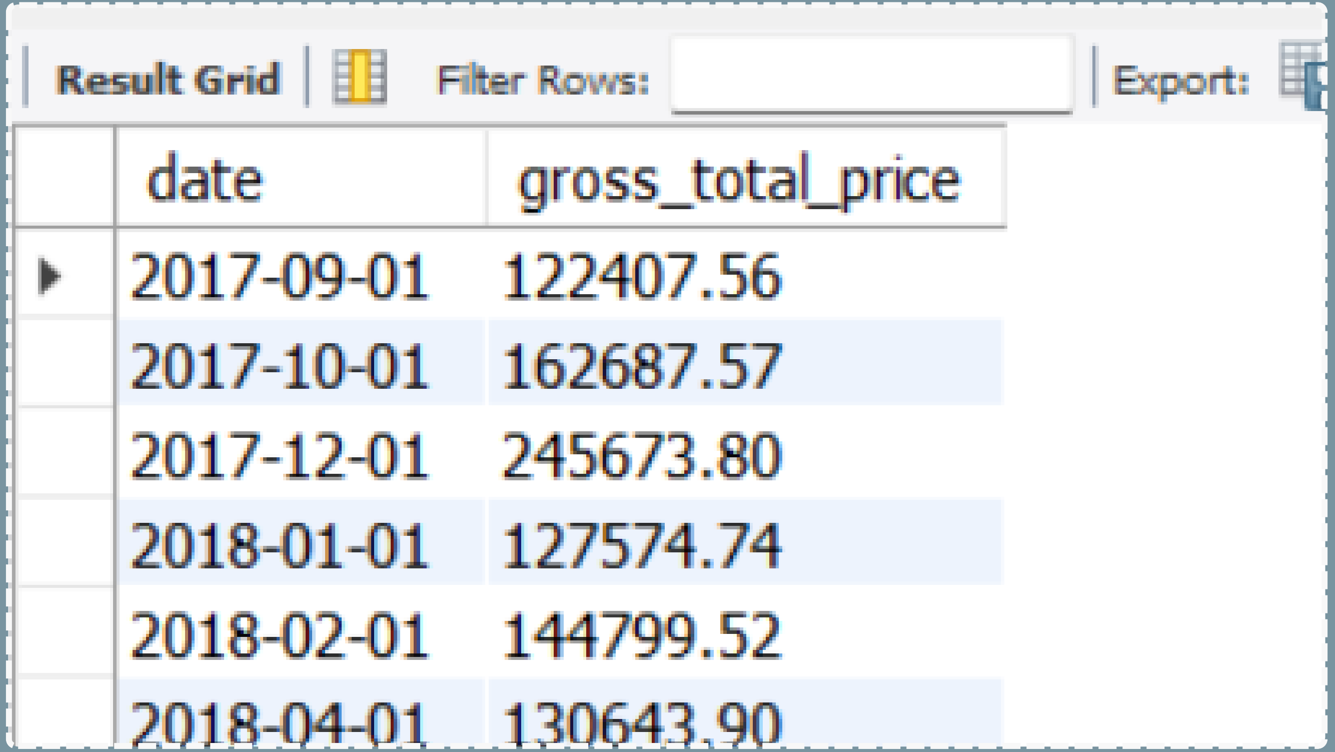


```
3  )
4  BEGIN
10 on s.product_code=gp.product
```

Call stored procedure gdb0041.get_monthly_gross_sale_f...

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

in_c_code 90002002 [IN] TEXT



date	gross_total_price
2017-09-01	122407.56
2017-10-01	162687.57
2017-12-01	245673.80
2018-01-01	127574.74
2018-02-01	144799.52
2018-04-01	130643.90

ad-hoc request 4

Description:

A **stored procedure** that can determine the market badge based on the following logic,

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

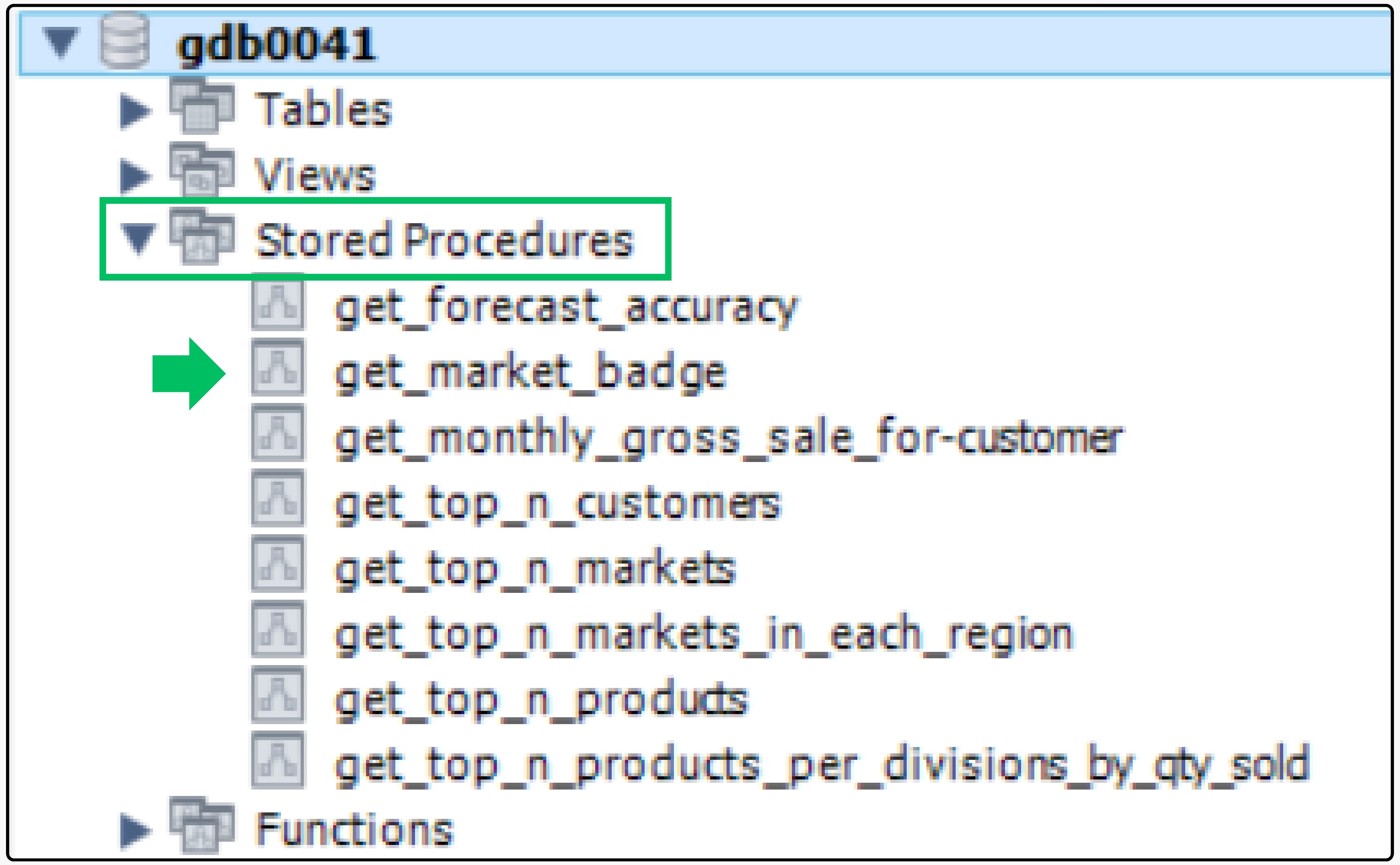
input will be,

• market

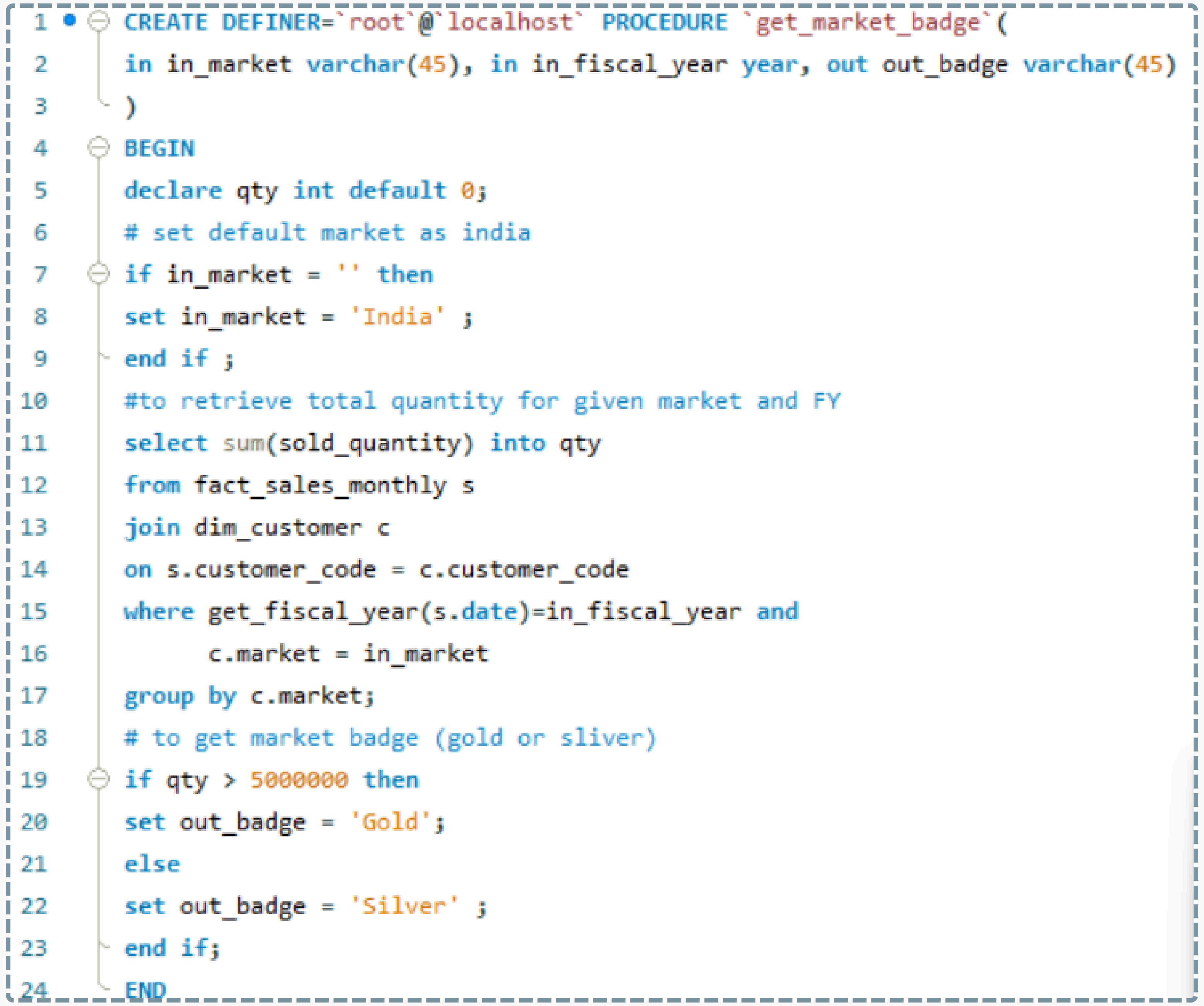
• fiscal year

Output :

• market badge

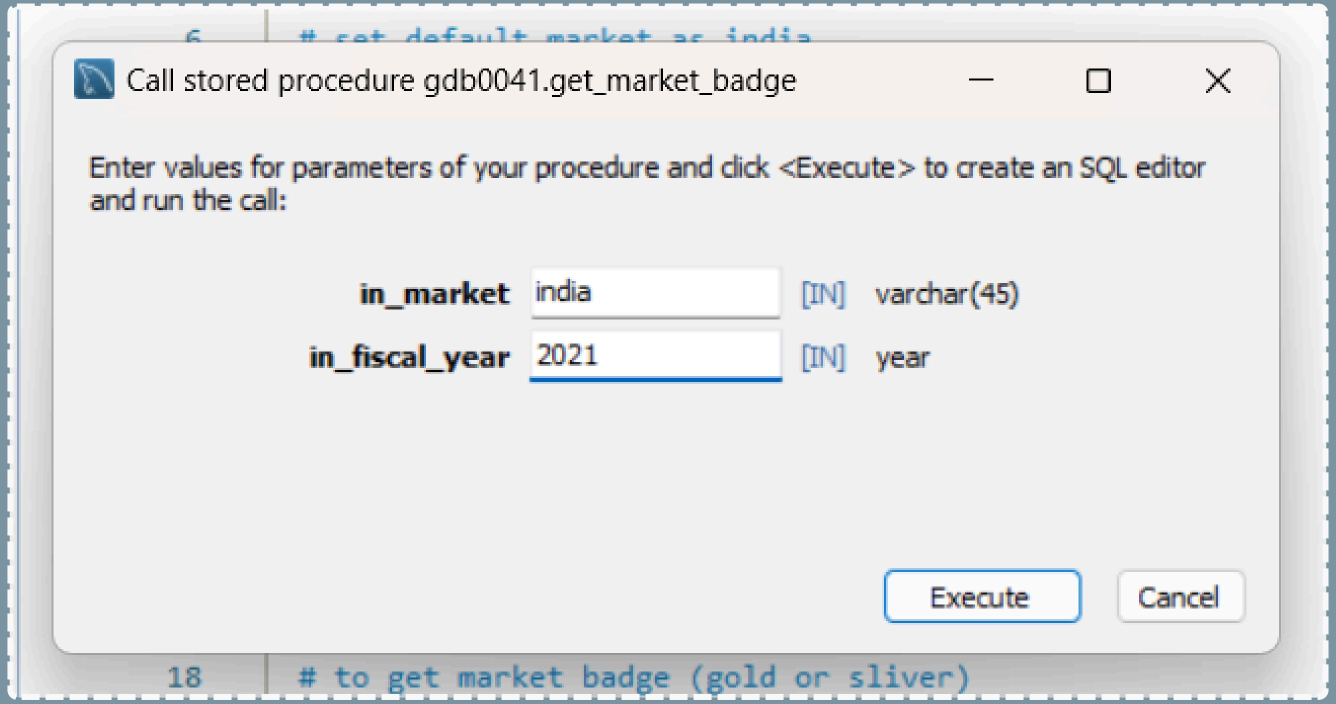


QUERY



```sql
CREATE DEFINER=`root`@`localhost` PROCEDURE `get_market_badge`(
in in_market varchar(45), in in_fiscal_year year, out out_badge varchar(45)
)
BEGIN
declare qty int default 0;
# set default market as india
if in_market = '' then
set in_market = 'India' ;
end if ;
#to retrieve total quantity for given market and FY
select sum(sold_quantity) into qty
from fact_sales_monthly s
join dim_customer c
on s.customer_code = c.customer_code
where get_fiscal_year(s.date)=in_fiscal_year and
      c.market = in_market
group by c.market;
# to get market badge (gold or sliver)
if qty > 5000000 then
set out_badge = 'Gold';
else
set out_badge = 'Silver' ;
end if;
END
```

Result



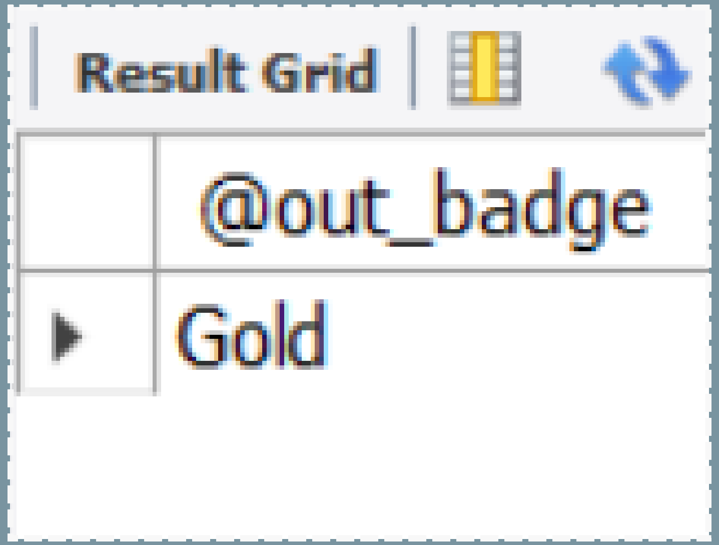
```
6    # set default market as india
```

Call stored procedure gdb0041.get_market_badge

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

in_market	india	[IN]	varchar(45)
in_fiscal_year	2021	[IN]	year

```
18   # to get market badge (gold or sliver)
```



@out_badge
Gold

ad-hoc request 5

Description:

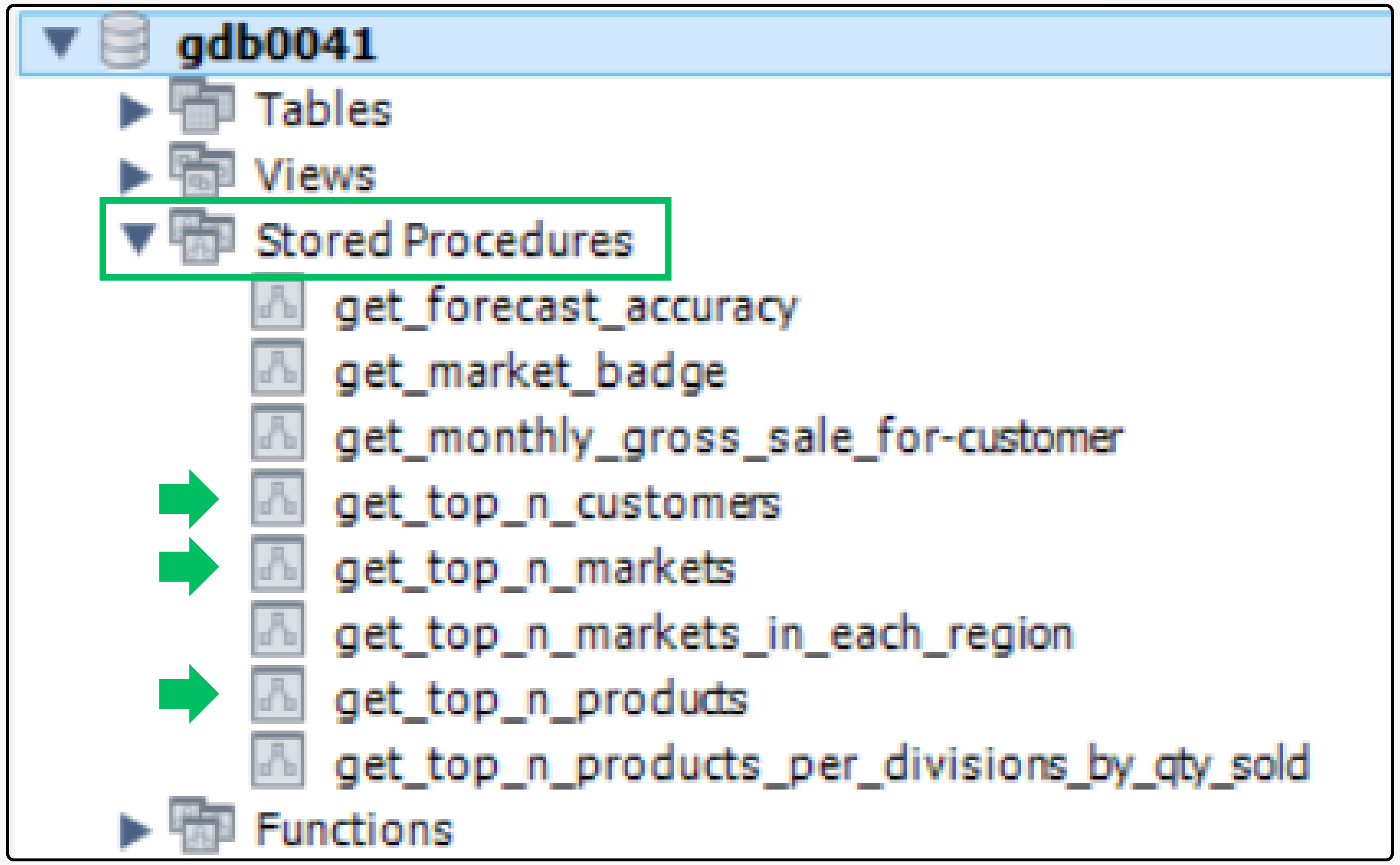
Stored procedure:

- For Top **Customers**, Top **Products** and Top **Markets**.

The report should have the following columns,

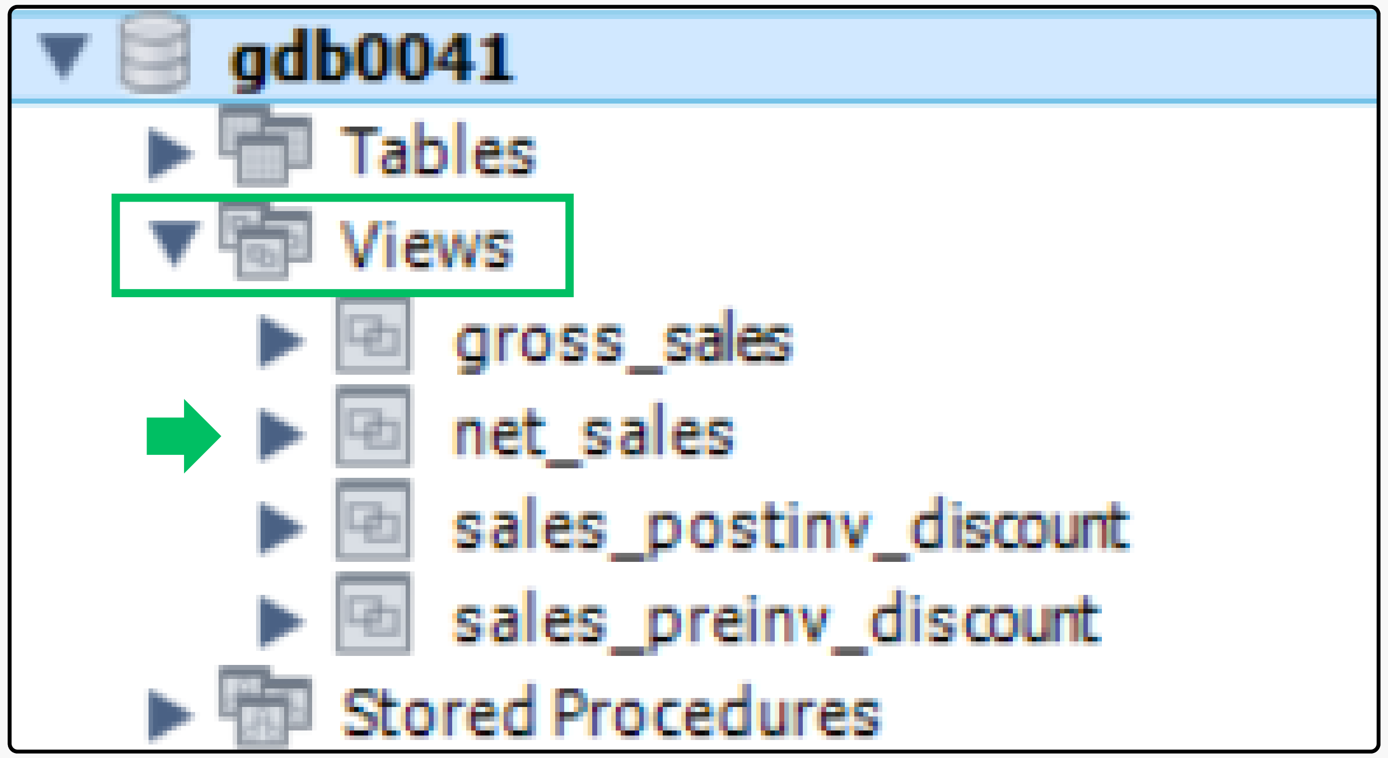
1 . Customer | Products | Markets.

2 . Net sales in million.



gdb0041
- Tables
- Views
- Stored Procedures
  - get_forecast_accuracy
  - get_market_badge
  - get_monthly_gross_sale_for-customer
  - get_top_n_customers
  - get_top_n_markets
  - get_top_n_markets_in_each_region
  - get_top_n_products
  - get_top_n_products_per_divisions_by_qty_sold
- Functions

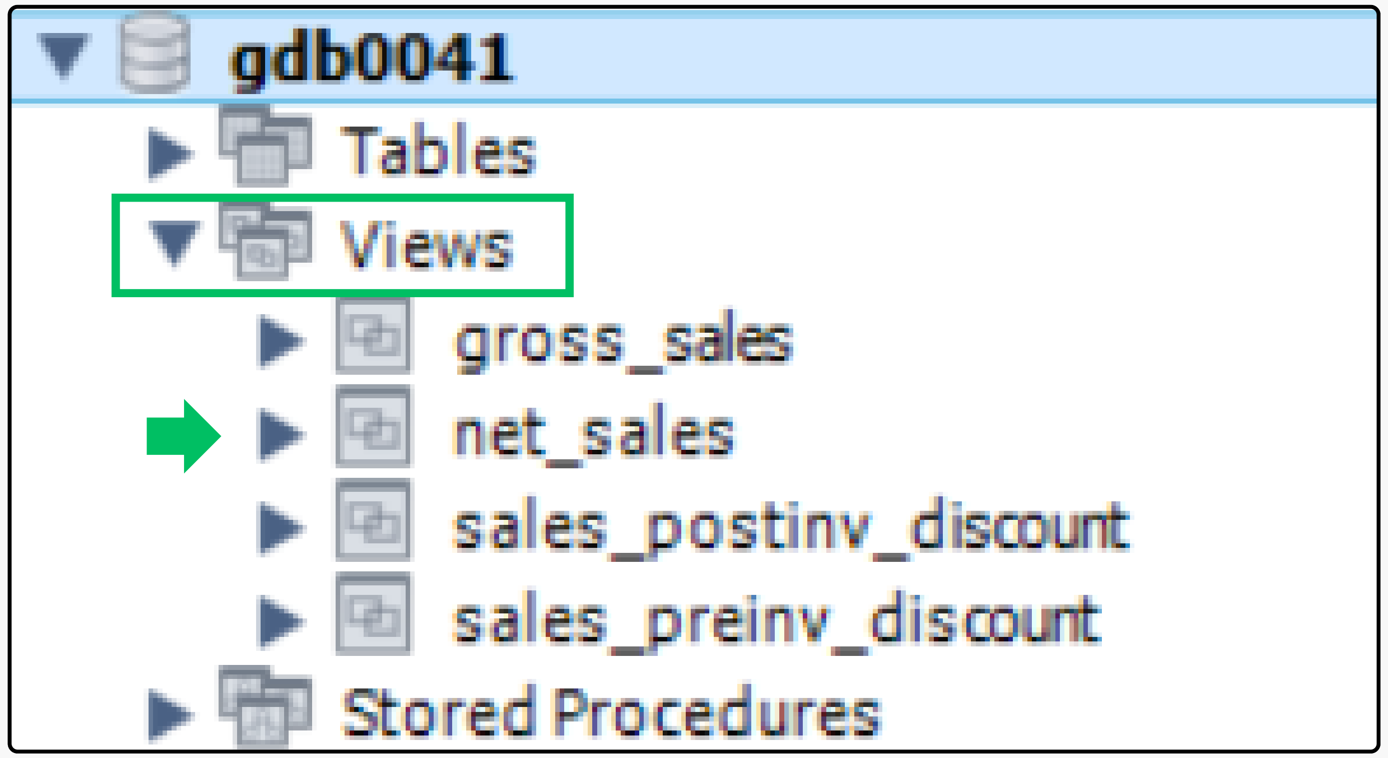
ad-hoc request 5



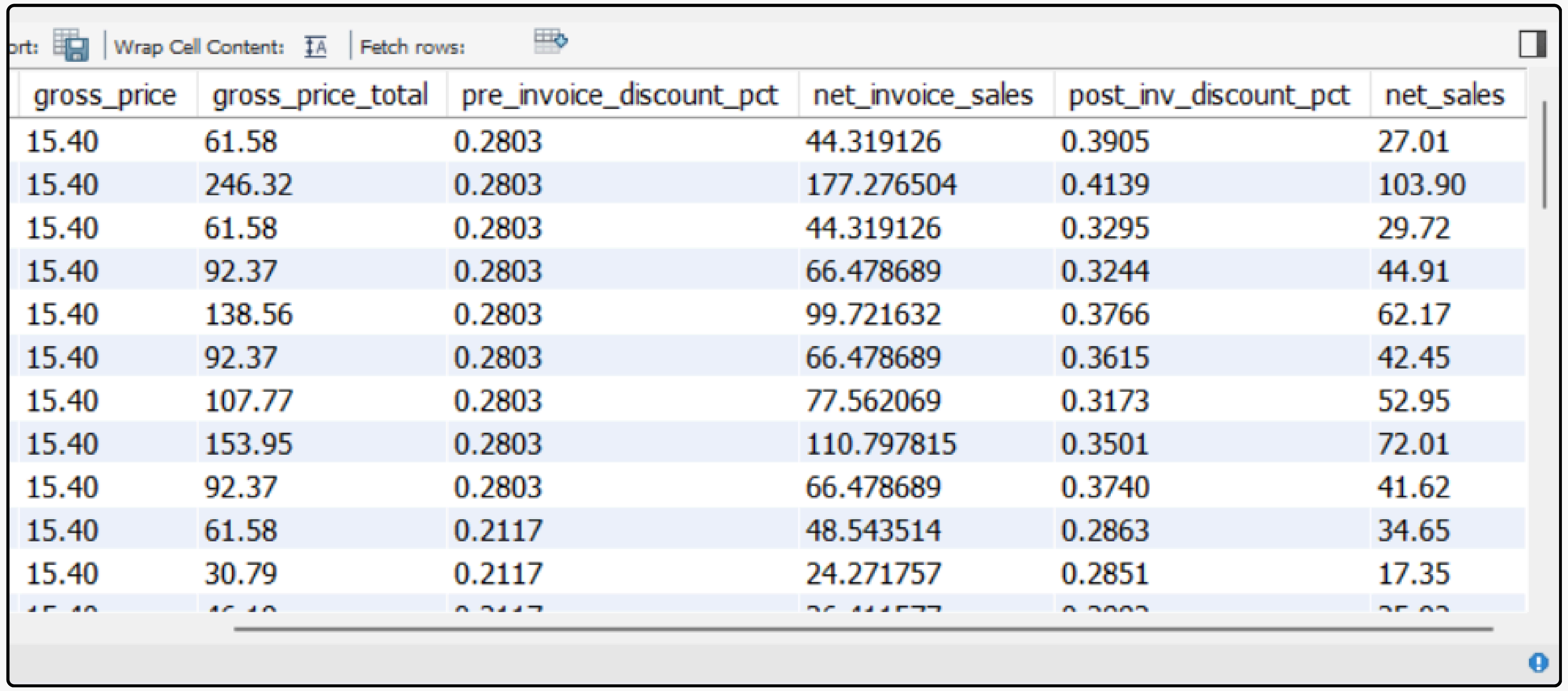
Views:

A **view** in MySQL is a **virtual table** based on the result of a SQL query. It doesn't store data itself but displays data from one or more underlying tables. Views allow you to simplify complex queries.

ad-hoc request 5

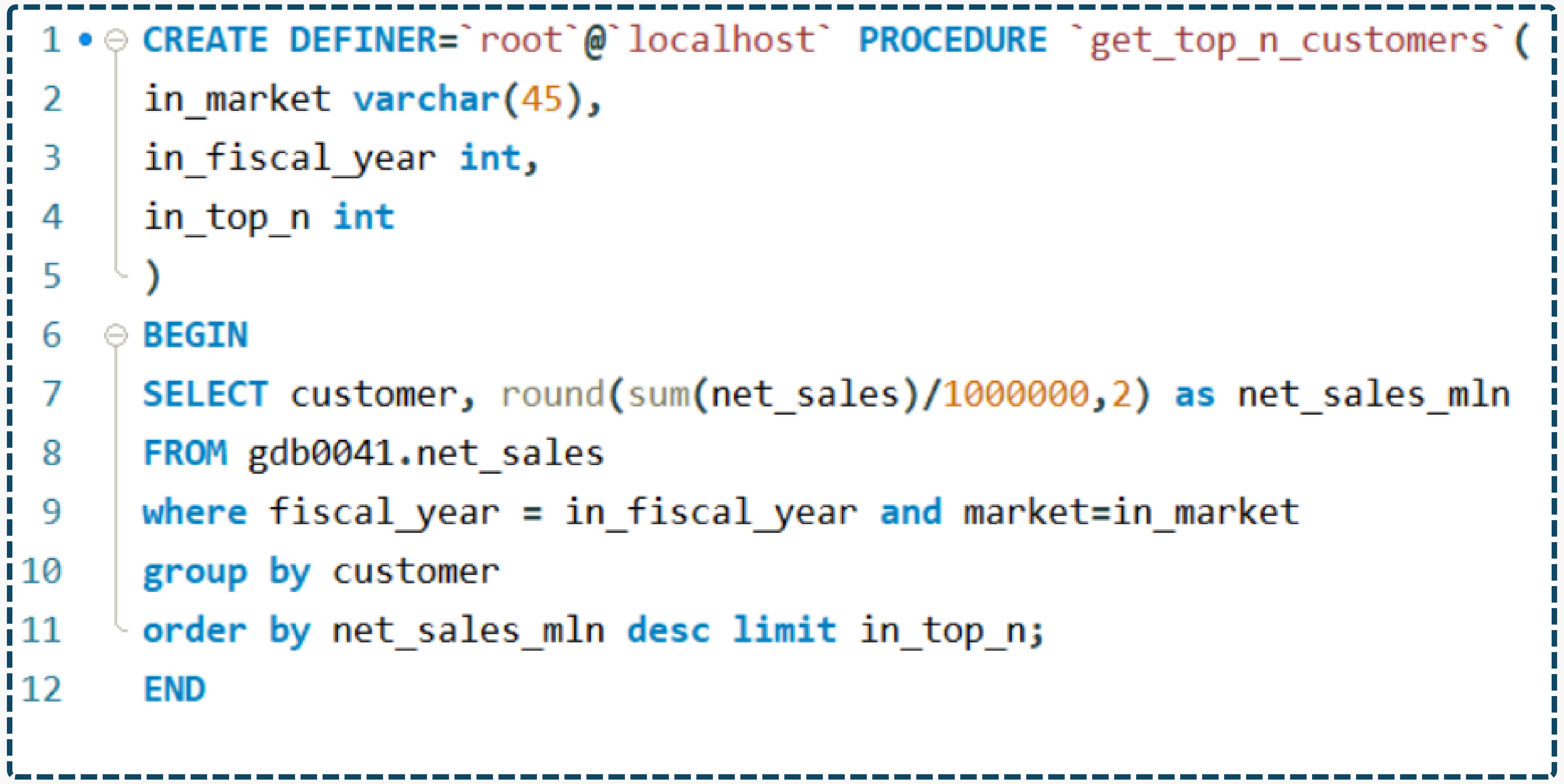


gdb0041
- Tables
- Views
  - gross_sales
  - net_sales
  - sales_postinv_discount
  - sales_preinv_discount
- Stored Procedures



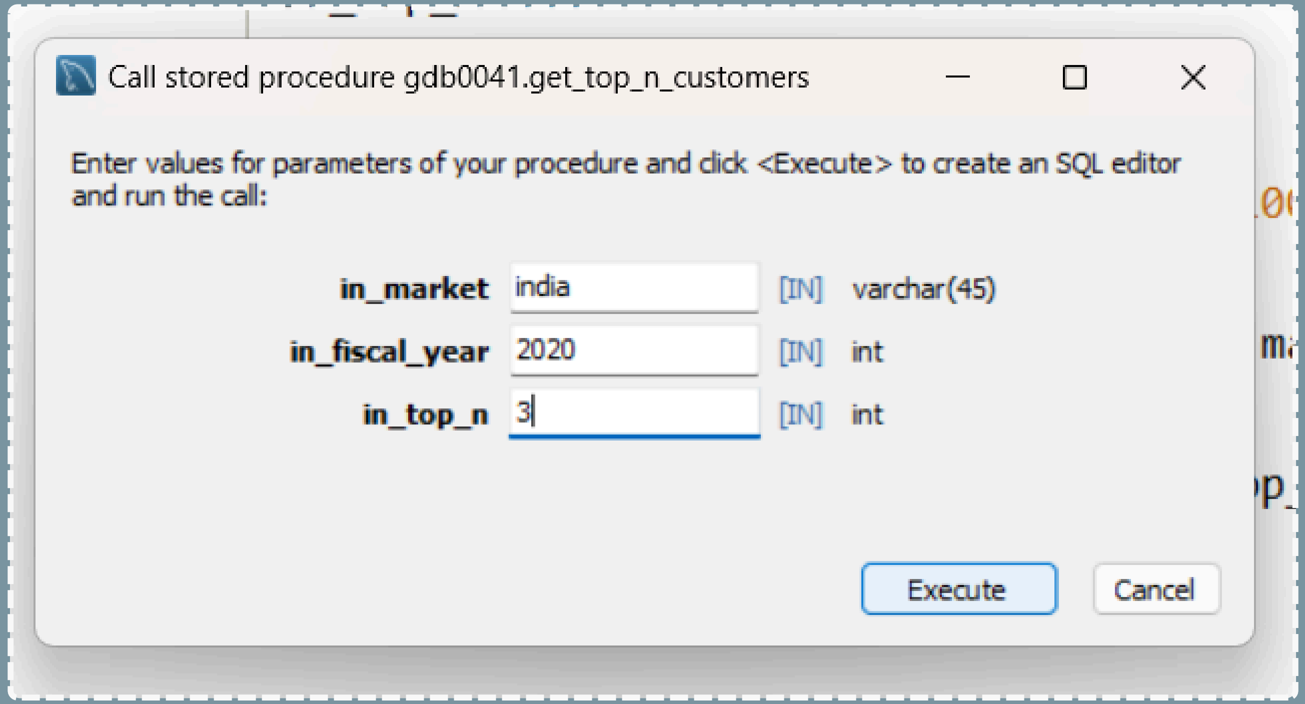
gross_price	gross_price_total	pre_invoice_discount_pct	net_invoice_sales	post_inv_discount_pct	net_sales
15.40	61.58	0.2803	44.319126	0.3905	27.01
15.40	246.32	0.2803	177.276504	0.4139	103.90
15.40	61.58	0.2803	44.319126	0.3295	29.72
15.40	92.37	0.2803	66.478689	0.3244	44.91
15.40	138.56	0.2803	99.721632	0.3766	62.17
15.40	92.37	0.2803	66.478689	0.3615	42.45
15.40	107.77	0.2803	77.562069	0.3173	52.95
15.40	153.95	0.2803	110.797815	0.3501	72.01
15.40	92.37	0.2803	66.478689	0.3740	41.62
15.40	61.58	0.2117	48.543514	0.2863	34.65
15.40	30.79	0.2117	24.271757	0.2851	17.35

QUERY



```
CREATE DEFINER=`root`@`localhost` PROCEDURE `get_top_n_customers`(
in_market varchar(45),
in_fiscal_year int,
in_top_n int
)
BEGIN
SELECT customer, round(sum(net_sales)/1000000,2) as net_sales_mln
FROM gdb0041.net_sales
where fiscal_year = in_fiscal_year and market=in_market
group by customer
order by net_sales_mln desc limit in_top_n;
END
```

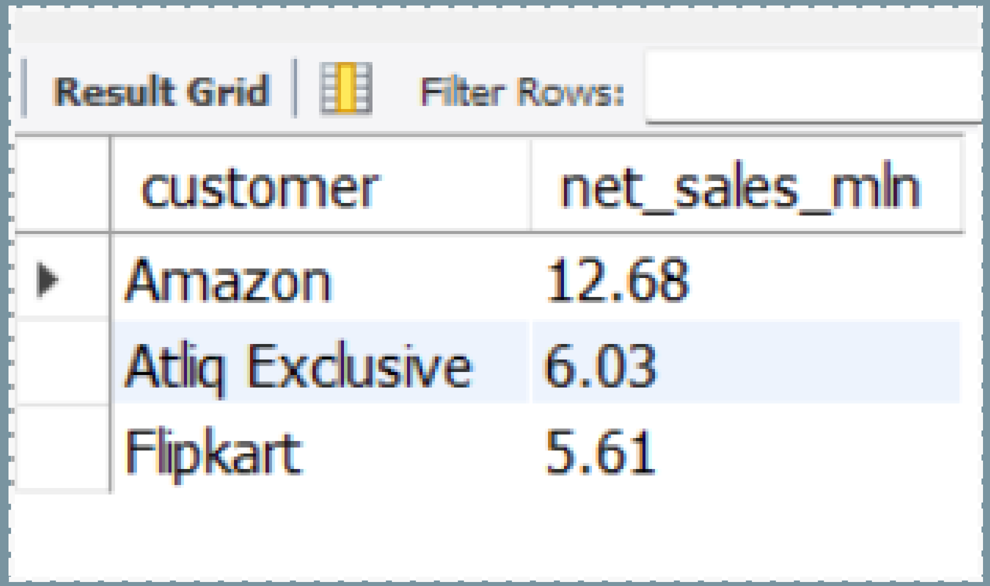
Result



Call stored procedure gdb0041.get_top_n_customers

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

in_market	india	[IN]	varchar(45)
in_fiscal_year	2020	[IN]	int
in_top_n	3	[IN]	int



customer	net_sales_mln
Amazon	12.68
Atliq Exclusive	6.03
Flipkart	5.61

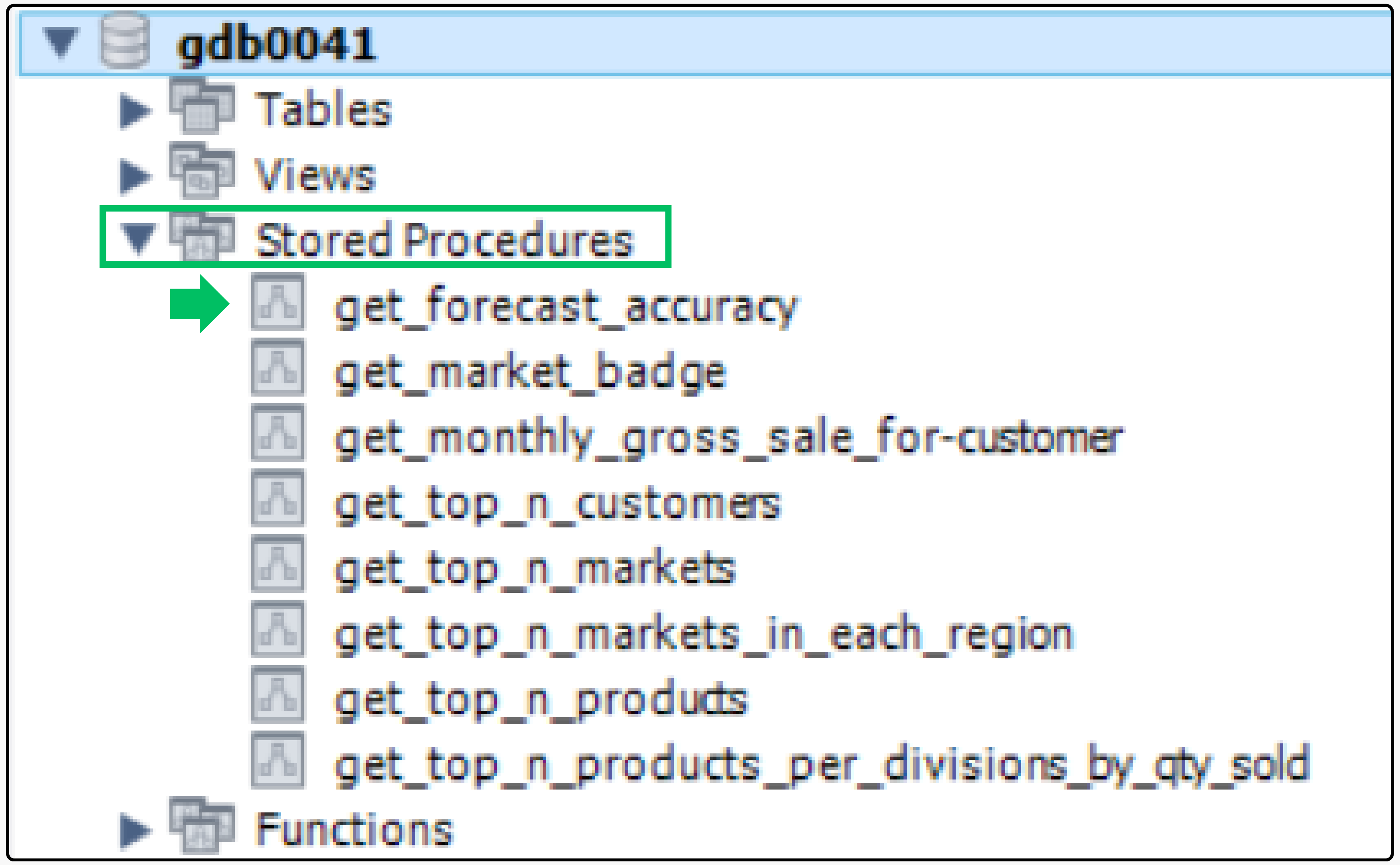
ad-hoc request 6

Description:

Need an **aggregate forecast accuracy** report for all the customers for a given fiscal year so that I can track the accuracy of the forecast we make for these customers.

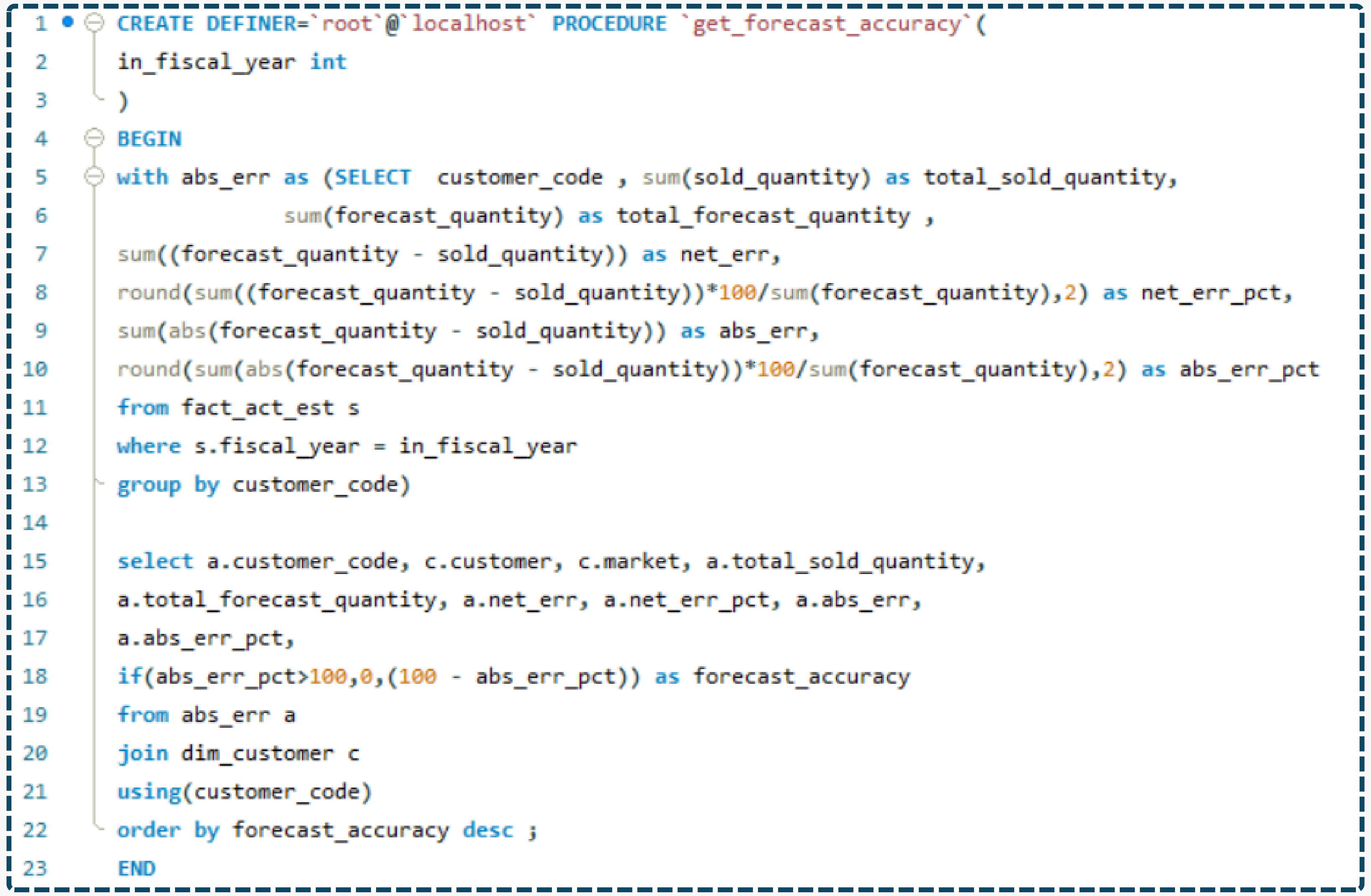
The report should have the following fields.

1. Customer Code, Name, Market
2. Total Sold Quantity
3. Total Forecast Quantity
4. Net Error
5. Absolute Error
6. Forecast Accuracy %



gdb0041
- Tables
- Views
- Stored Procedures
  - get_forecast_accuracy
  - get_market_badge
  - get_monthly_gross_sale_for-customer
  - get_top_n_customers
  - get_top_n_markets
  - get_top_n_markets_in_each_region
  - get_top_n_products
  - get_top_n_products_per_divisions_by_qty_sold
- Functions

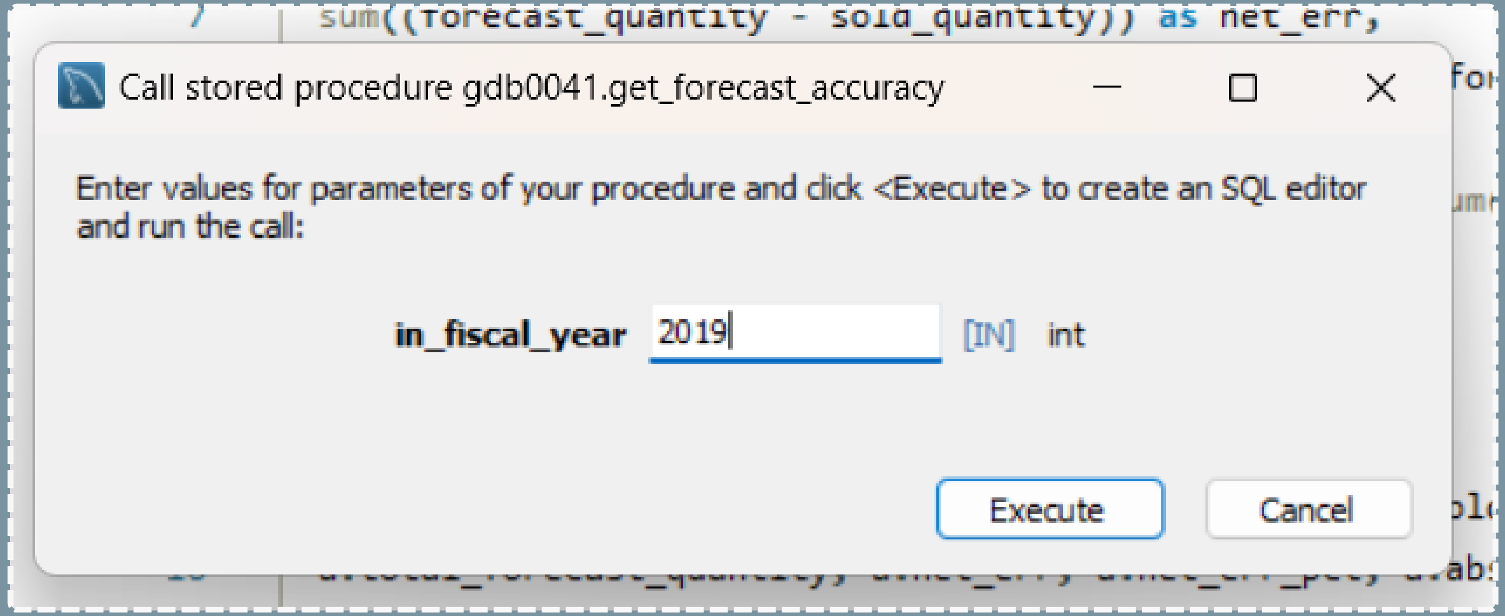
QUERY



```sql
CREATE DEFINER=`root`@`localhost` PROCEDURE `get_forecast_accuracy`(
in_fiscal_year int
)
BEGIN
with abs_err as (SELECT  customer_code , sum(sold_quantity) as total_sold_quantity,
             sum(forecast_quantity) as total_forecast_quantity ,
sum((forecast_quantity - sold_quantity)) as net_err,
round(sum((forecast_quantity - sold_quantity))*100/sum(forecast_quantity),2) as net_err_pct,
sum(abs(forecast_quantity - sold_quantity)) as abs_err,
round(sum(abs(forecast_quantity - sold_quantity))*100/sum(forecast_quantity),2) as abs_err_pct
from fact_act_est s
where s.fiscal_year = in_fiscal_year
group by customer_code)

select a.customer_code, c.customer, c.market, a.total_sold_quantity,
a.total_forecast_quantity, a.net_err, a.net_err_pct, a.abs_err,
a.abs_err_pct,
if(abs_err_pct>100,0,(100 - abs_err_pct)) as forecast_accuracy
from abs_err a
join dim_customer c
using(customer_code)
order by forecast_accuracy desc ;
END
```

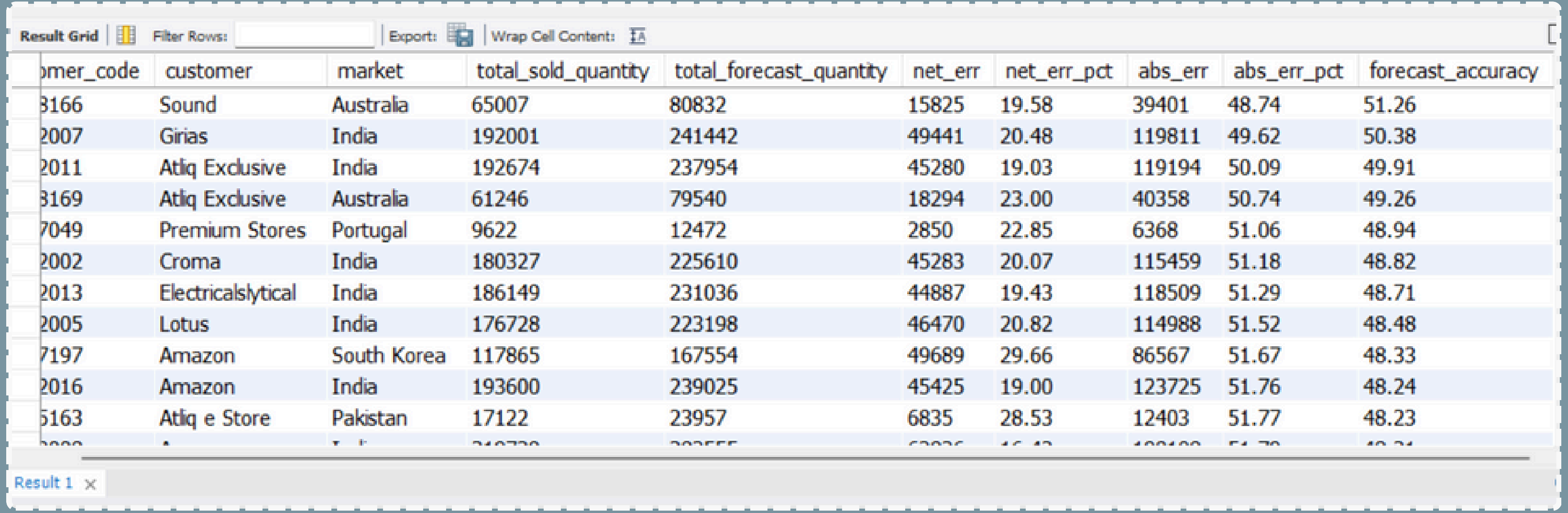
Result



Call stored procedure gdb0041.get_forecast_accuracy

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

in_fiscal_year 2019 [IN] int



omer_code	customer	market	total_sold_quantity	total_forecast_quantity	net_err	net_err_pct	abs_err	abs_err_pct	forecast_accuracy
3166	Sound	Australia	65007	80832	15825	19.58	39401	48.74	51.26
2007	Girias	India	192001	241442	49441	20.48	119811	49.62	50.38
2011	Atliq Exclusive	India	192674	237954	45280	19.03	119194	50.09	49.91
3169	Atliq Exclusive	Australia	61246	79540	18294	23.00	40358	50.74	49.26
7049	Premium Stores	Portugal	9622	12472	2850	22.85	6368	51.06	48.94
2002	Croma	India	180327	225610	45283	20.07	115459	51.18	48.82
2013	Electricalslytical	India	186149	231036	44887	19.43	118509	51.29	48.71
2005	Lotus	India	176728	223198	46470	20.82	114988	51.52	48.48
7197	Amazon	South Korea	117865	167554	49689	29.66	86567	51.67	48.33
2016	Amazon	India	193600	239025	45425	19.00	123725	51.76	48.24
5163	Atliq e Store	Pakistan	17122	23957	6835	28.53	12403	51.77	48.23

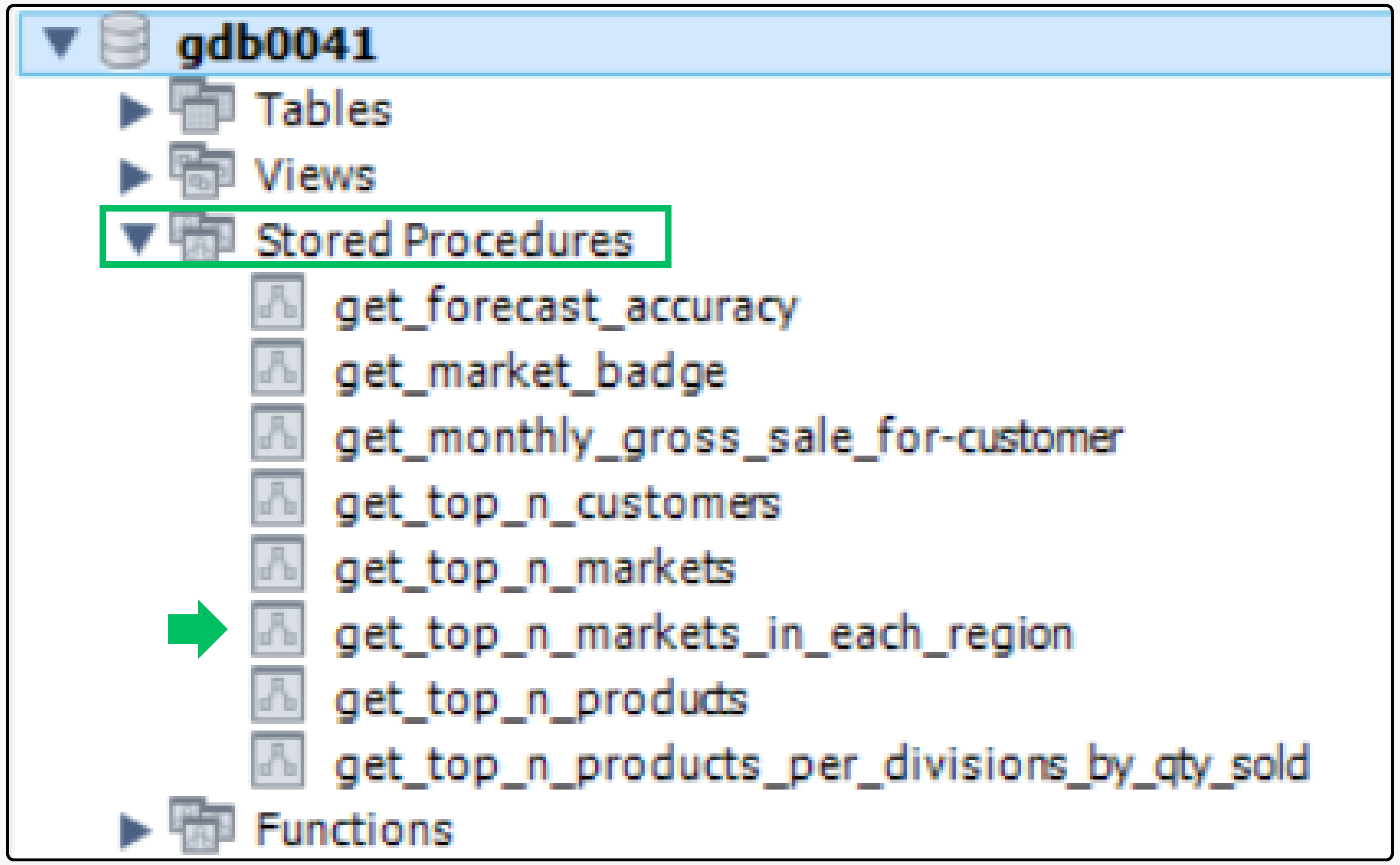
ad-hoc request 7

Description:

A store procedure for **Top markets in each region**.

The report should have the following fields.

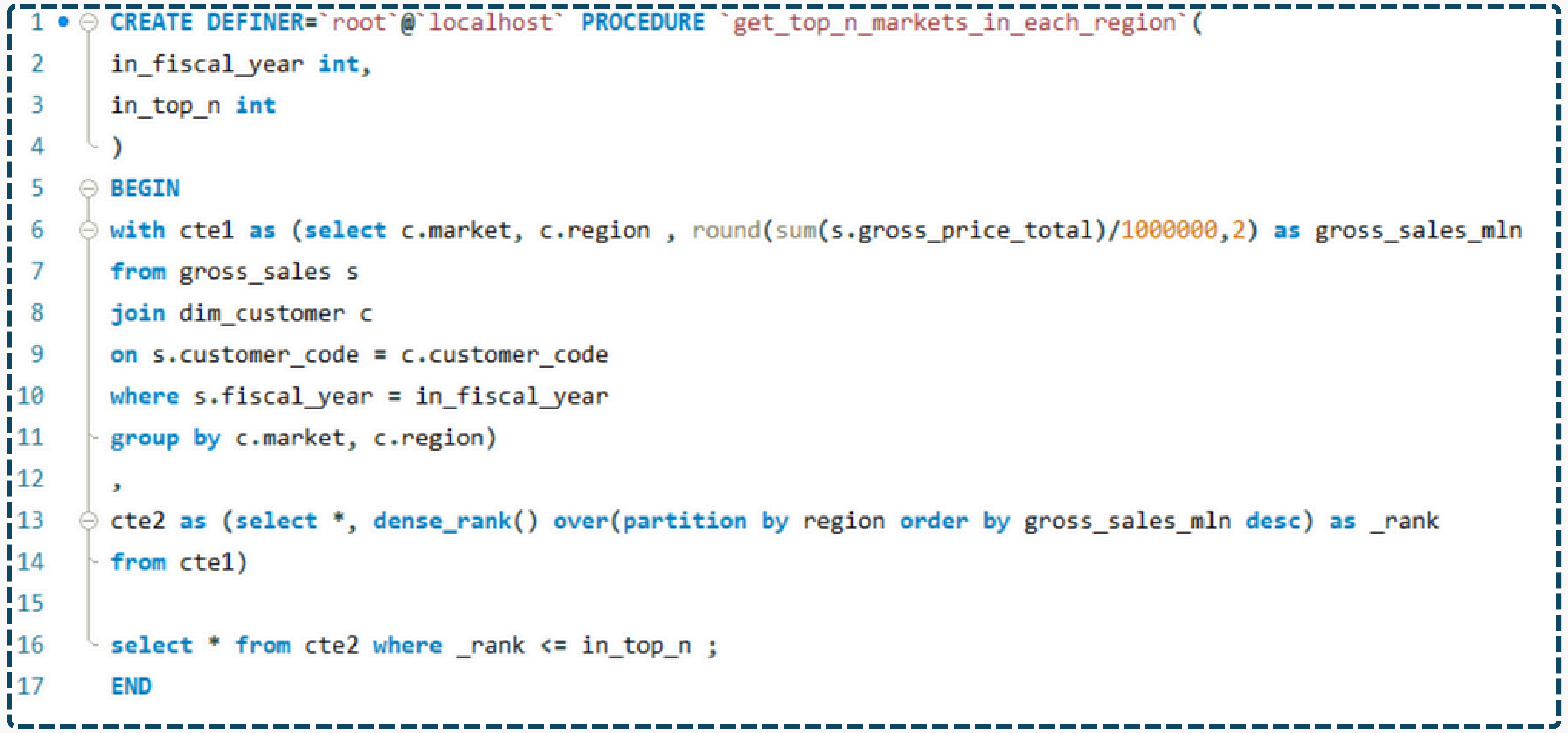
1. Market
2. Region
3. Gross sales in million
4. Rank



gdb0041

- Tables
- Views
- Stored Procedures
  - get_forecast_accuracy
  - get_market_badge
  - get_monthly_gross_sale_for-customer
  - get_top_n_customers
  - get_top_n_markets
  - get_top_n_markets_in_each_region
  - get_top_n_products
  - get_top_n_products_per_divisions_by_qty_sold
- Functions

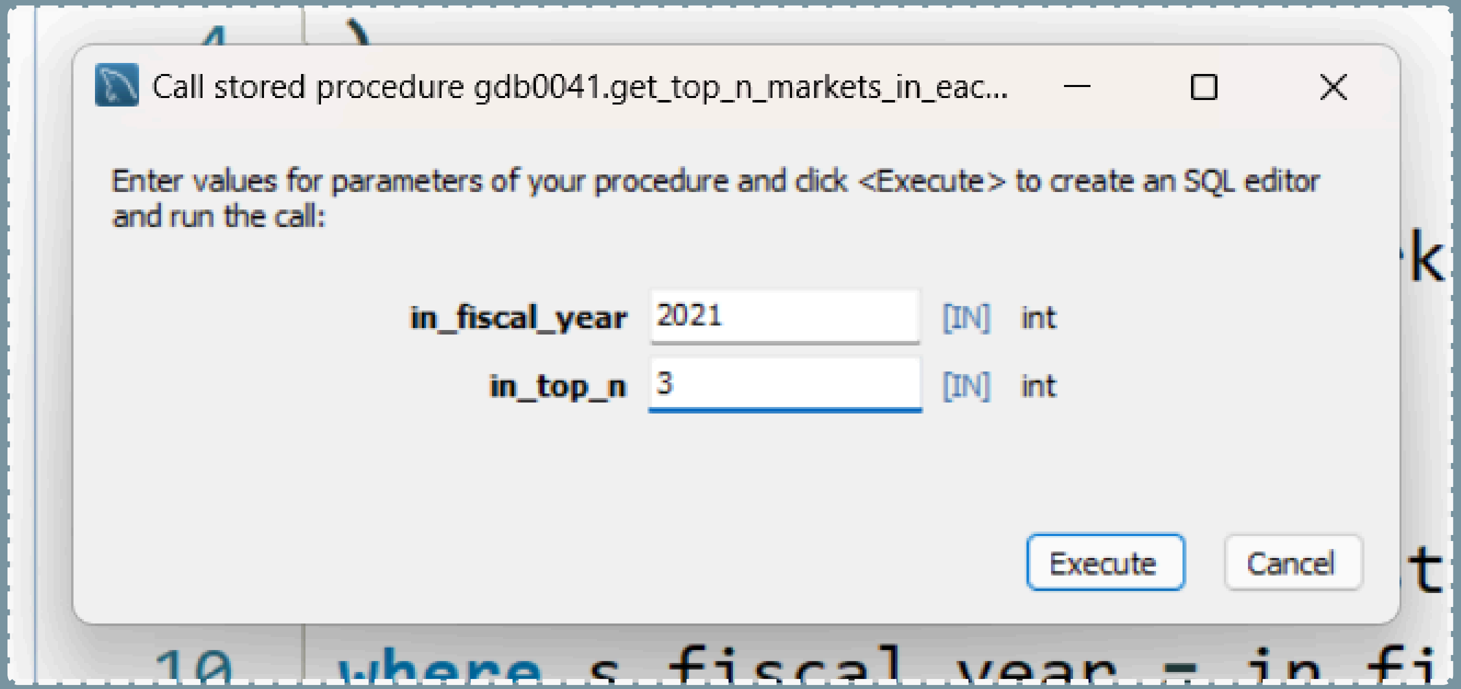
QUERY



```
CREATE DEFINER=`root`@`localhost` PROCEDURE `get_top_n_markets_in_each_region`(
in_fiscal_year int,
in_top_n int
)
BEGIN
with cte1 as (select c.market, c.region , round(sum(s.gross_price_total)/1000000,2) as gross_sales_mln
from gross_sales s
join dim_customer c
on s.customer_code = c.customer_code
where s.fiscal_year = in_fiscal_year
group by c.market, c.region)
,
cte2 as (select *, dense_rank() over(partition by region order by gross_sales_mln desc) as _rank
from cte1)

select * from cte2 where _rank <= in_top_n ;
END
```

Result

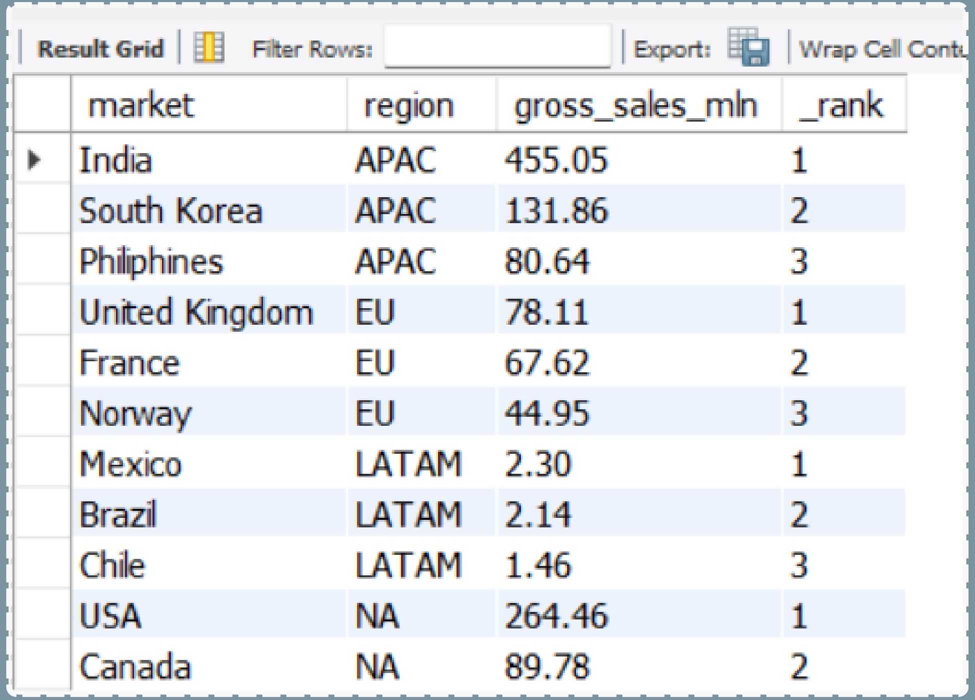


Call stored procedure gdb0041.get_top_n_markets_in_eac...

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

in_fiscal_year 2021 [IN] int

in_top_n 3 [IN] int



market	region	gross_sales_mln	_rank
India	APAC	455.05	1
South Korea	APAC	131.86	2
Philiphines	APAC	80.64	3
United Kingdom	EU	78.11	1
France	EU	67.62	2
Norway	EU	44.95	3
Mexico	LATAM	2.30	1
Brazil	LATAM	2.14	2
Chile	LATAM	1.46	3
USA	NA	264.46	1
Canada	NA	89.78	2

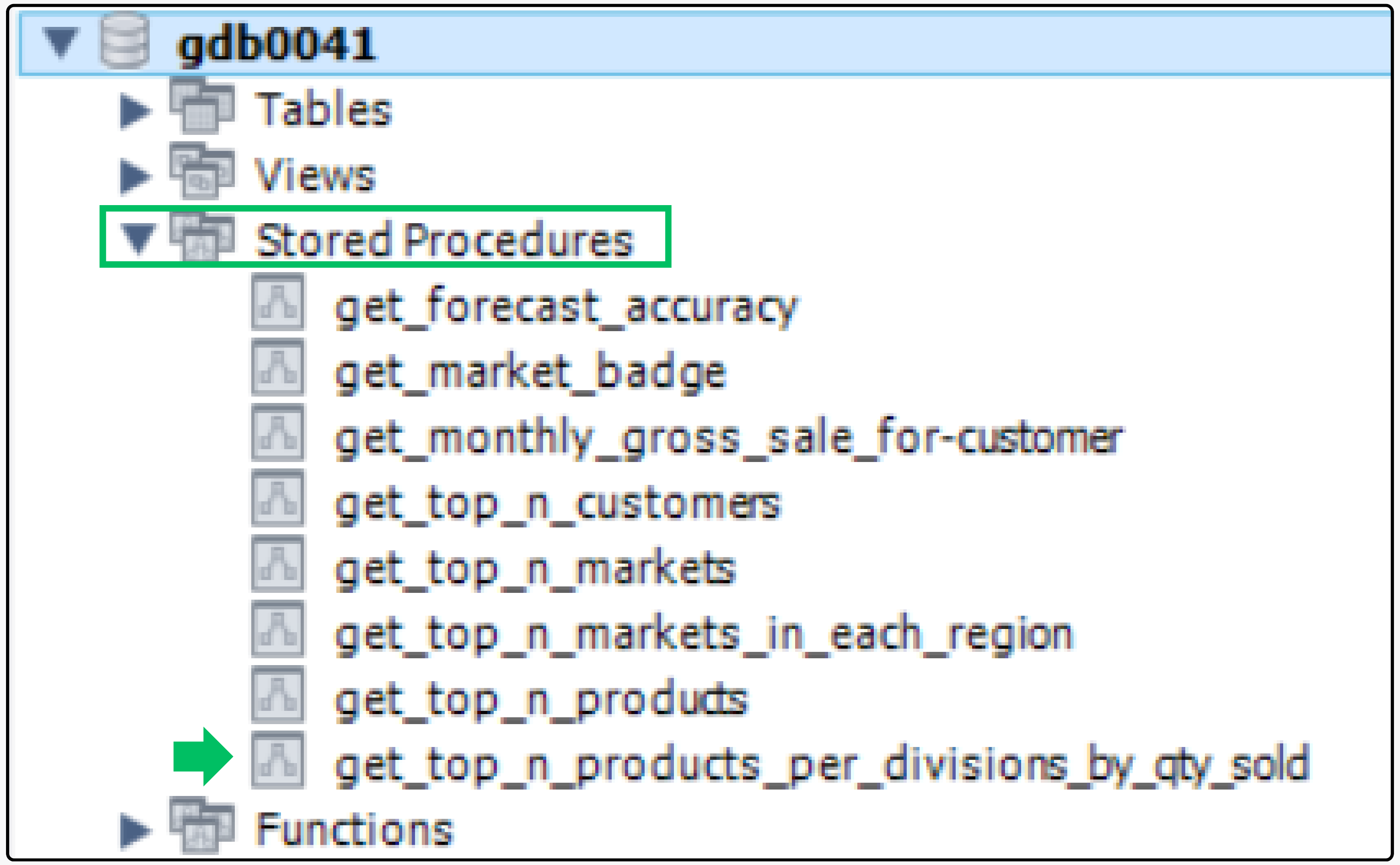
ad-hoc request 8

Description:

A store procedure for **Top Products in each division by sold quantity.**

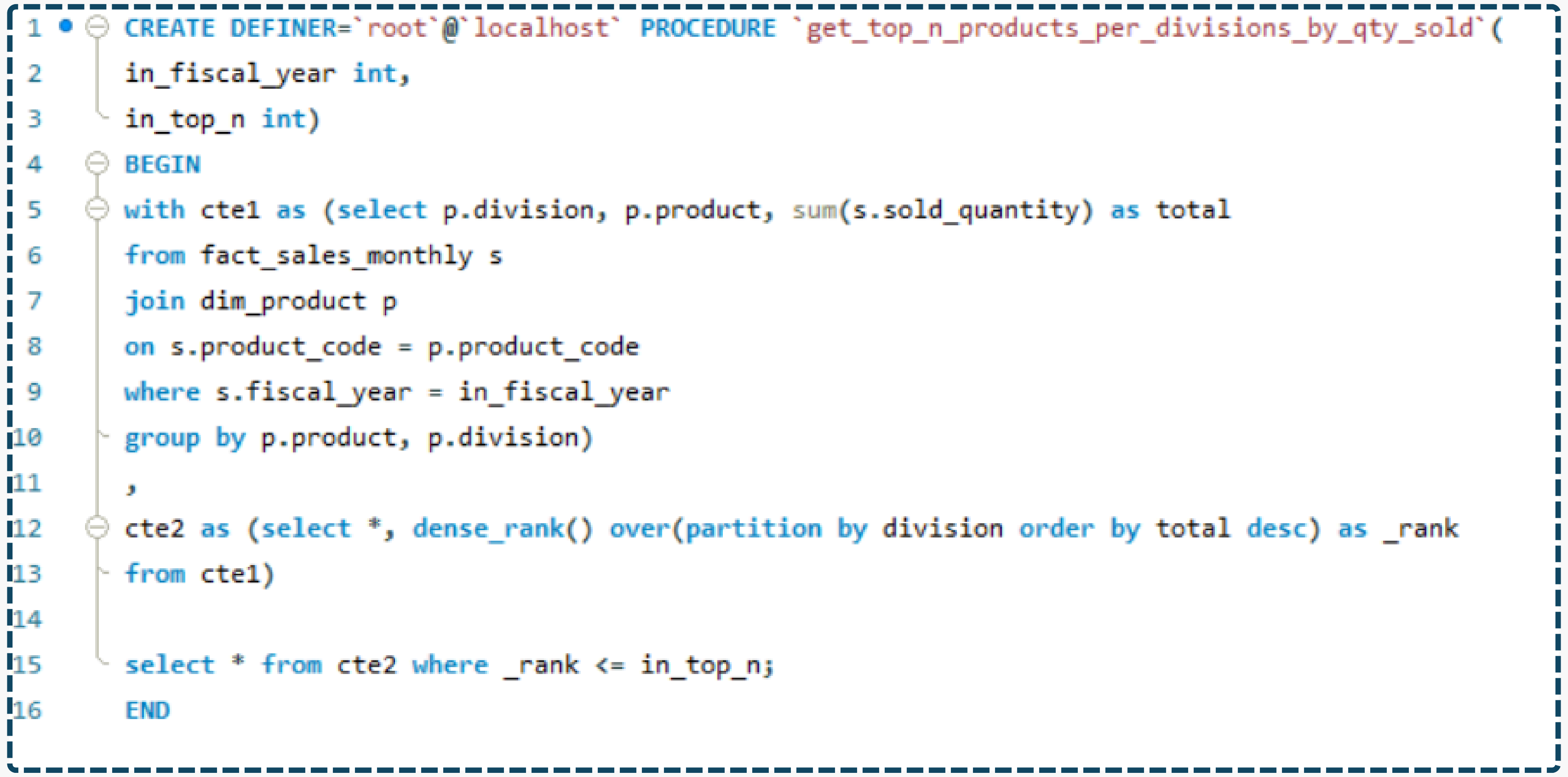
The report should have the following fields.

1. Market
2. Region
3. Gross sales in million
4. Rank



gdb0041
- Tables
- Views
- Stored Procedures
  - get_forecast_accuracy
  - get_market_badge
  - get_monthly_gross_sale_for-customer
  - get_top_n_customers
  - get_top_n_markets
  - get_top_n_markets_in_each_region
  - get_top_n_products
  - ➡ get_top_n_products_per_divisions_by_qty_sold
- Functions

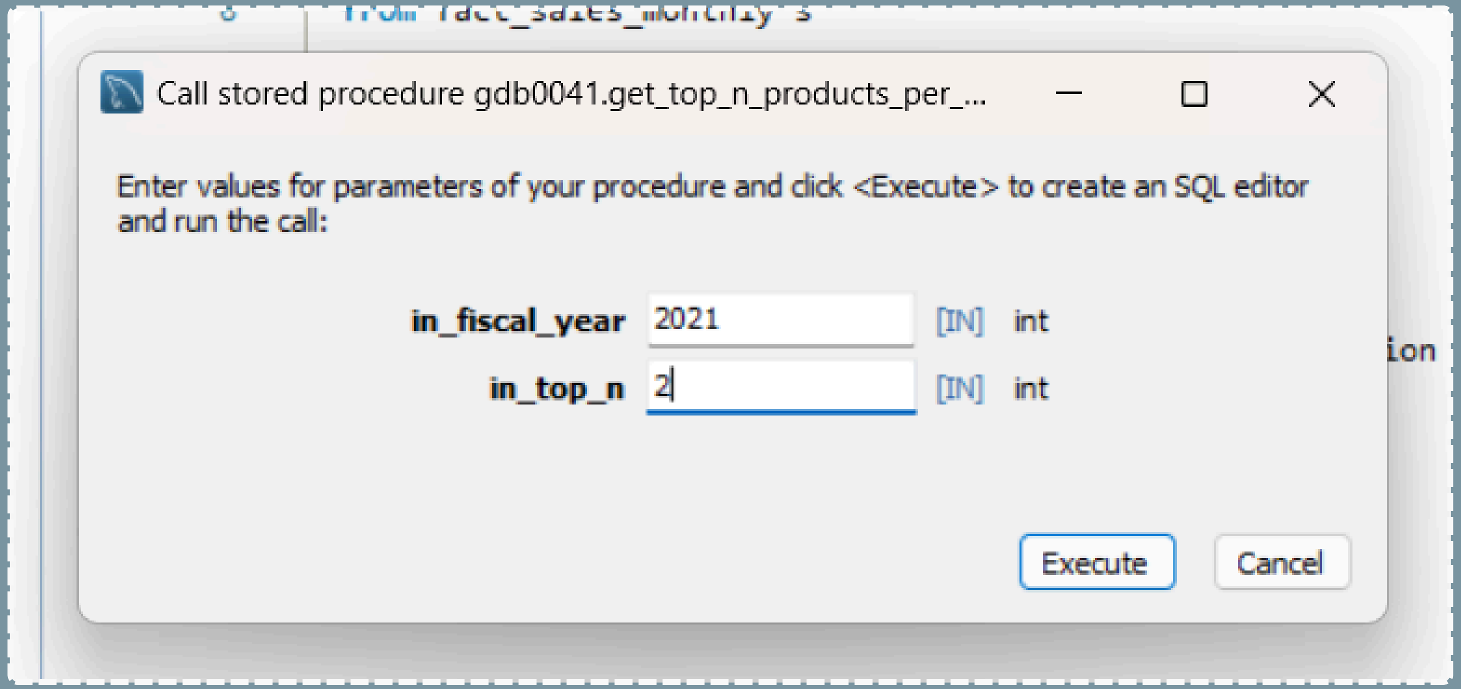
QUERY



```sql
CREATE DEFINER=`root`@`localhost` PROCEDURE `get_top_n_products_per_divisions_by_qty_sold`(
in_fiscal_year int,
in_top_n int)
BEGIN
with cte1 as (select p.division, p.product, sum(s.sold_quantity) as total
from fact_sales_monthly s
join dim_product p
on s.product_code = p.product_code
where s.fiscal_year = in_fiscal_year
group by p.product, p.division)
,
cte2 as (select *, dense_rank() over(partition by division order by total desc) as _rank
from cte1)

select * from cte2 where _rank <= in_top_n;
END
```

Result

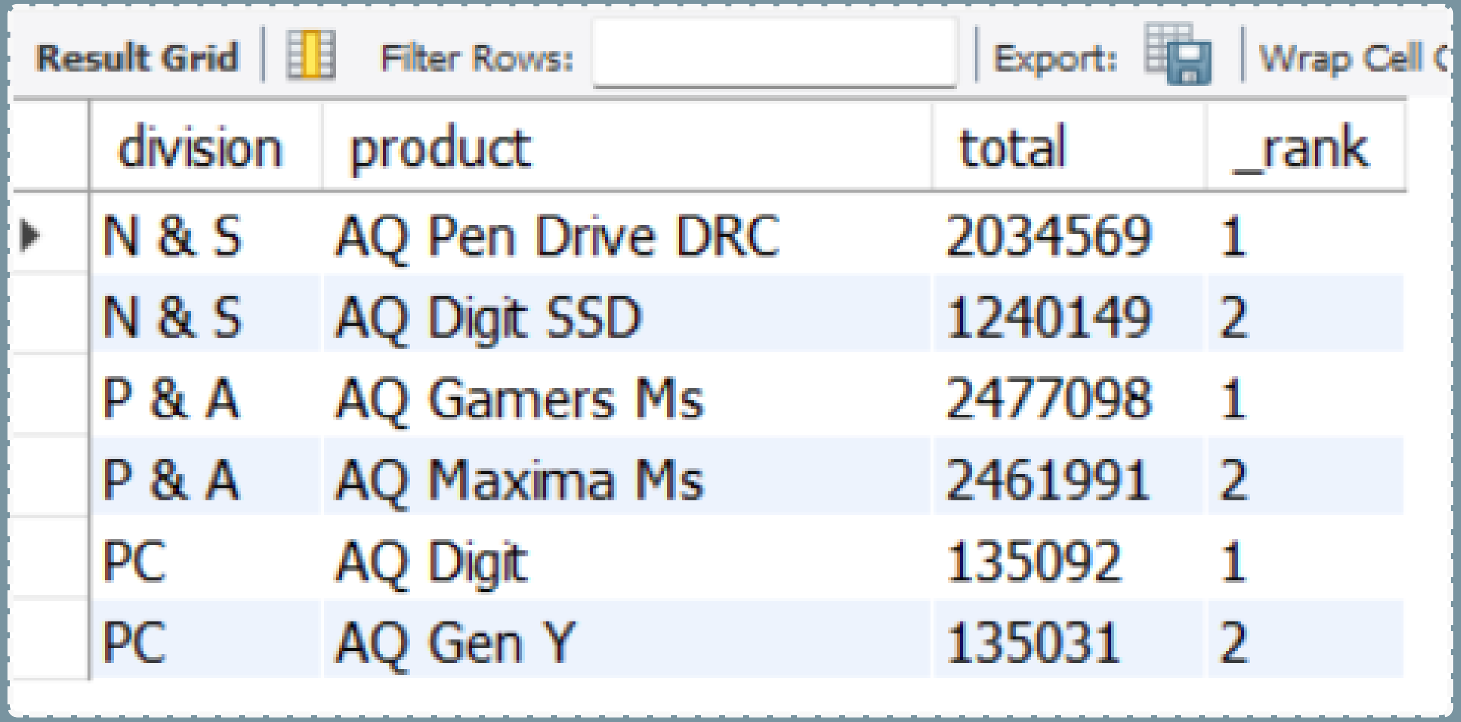


Call stored procedure gdb0041.get_top_n_products_per_...

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

in_fiscal_year 2021 [IN] int

in_top_n 2 [IN] int



division	product	total	_rank
N & S	AQ Pen Drive DRC	2034569	1
N & S	AQ Digit SSD	1240149	2
P & A	AQ Gamers Ms	2477098	1
P & A	AQ Maxima Ms	2461991	2
PC	AQ Digit	135092	1
PC	AQ Gen Y	135031	2

THANK YOU