OPTIMIZING FINANCIAL ANALYTICS WITH MYSQL

KEY INSIGHTS: TOP CUSTOMERS, MARKETS, PRODUCTS

PROJECT OVERVIEW

- AtliQ Hardware is a global computer hardware manufacturer that specializes in producing high-performance computer components and peripherals.
- The company operates across major international markets, including APAC (Asia-Pacific), EU (Europe), NA (North America), and LATAM (Latin America).
- AtliQ's product portfolio includes processors, motherboards, graphics cards, RAM, storage solutions (SSDs, HDDs), and peripherals such as keyboards, mice, and monitors.
- With a commitment to innovation and quality, the company caters to both individual consumers and businesses, ensuring reliable and efficient hardware solutions for diverse computing needs.

PROBLEM STATEMENT

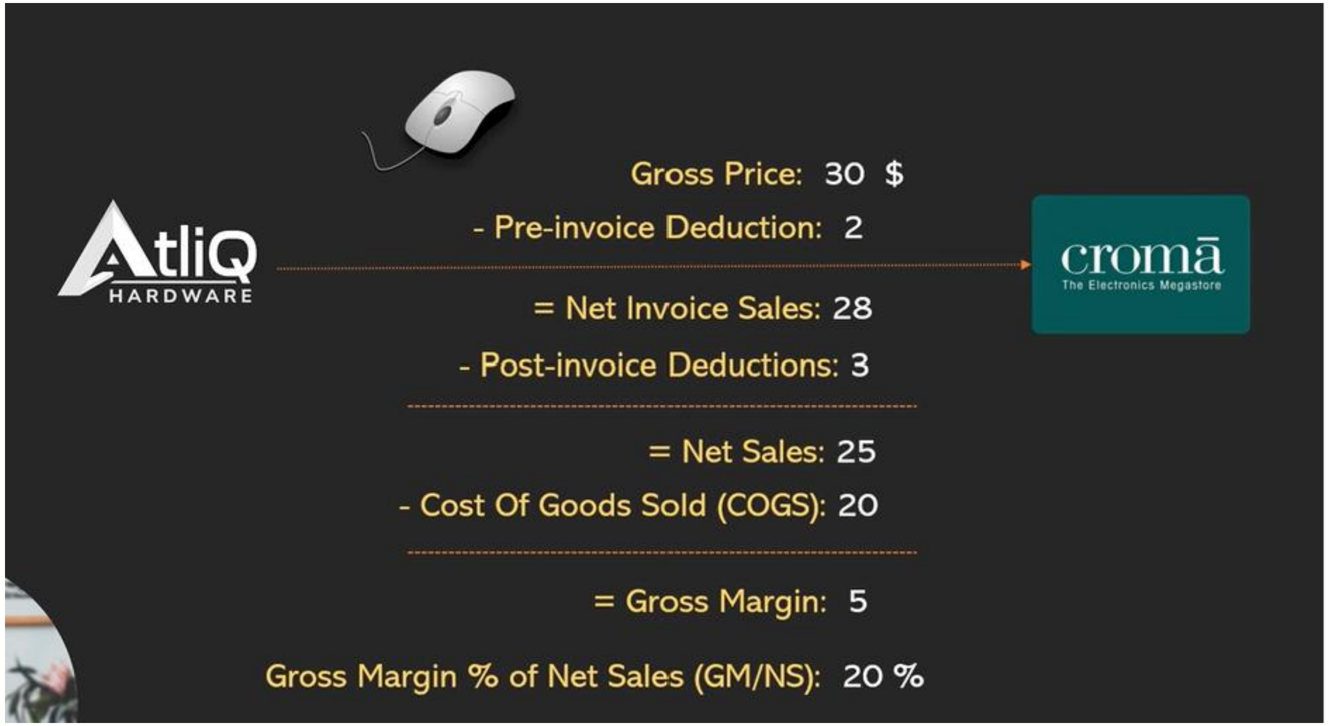
- **Performance Challenges with Large Excel Files** – AtliQ Hardware is facing inefficiencies
- and slow performance due to the increasing size of Excel files.

- **Expanding Data Analytics Team** – To address this issue, the company is hiring junior data analysts to strengthen its data analytics capabilities.

- **Transition to MySQL** – AtliQ is utilizing MySQL as its database management system to analyze data, track financial trends, and extract meaningful insights.

- **Business Optimization Goals** – The project aims to enhance decision-making, streamline operations, and improve overall company performance.

P & L STATEMENT



Gross Price: 30 $

- Pre-invoice Deduction: 2

= Net Invoice Sales: 28

- Post-invoice Deductions: 3

= Net Sales: 25

- Cost Of Goods Sold (COGS): 20

= Gross Margin: 5

Gross Margin % of Net Sales (GM/NS): 20 %

TASK - 1

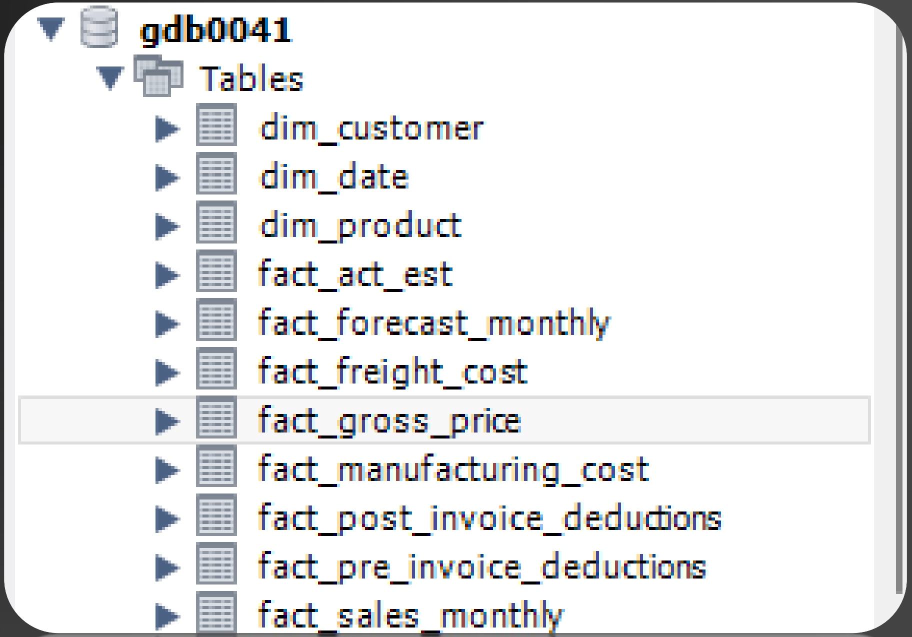
In order to track individual product sales and perform additional product analytics on it using Excel, I, as the product owner, would like to create a report of Croma India customers' sales of individual products (aggregated monthly at the product code level) for FY-2021.

The following fields should be included in the report:

1. Month
2. Product Name
3. Variant
4. Sold Quantity
5. Gross Price Per Item
6. Gross Price Tota1

DATASET

- Loaded a million rows of data into a MYSQL database and executed multiple queries.
- Analyzed the following tables to derive the SQL concepts presented.



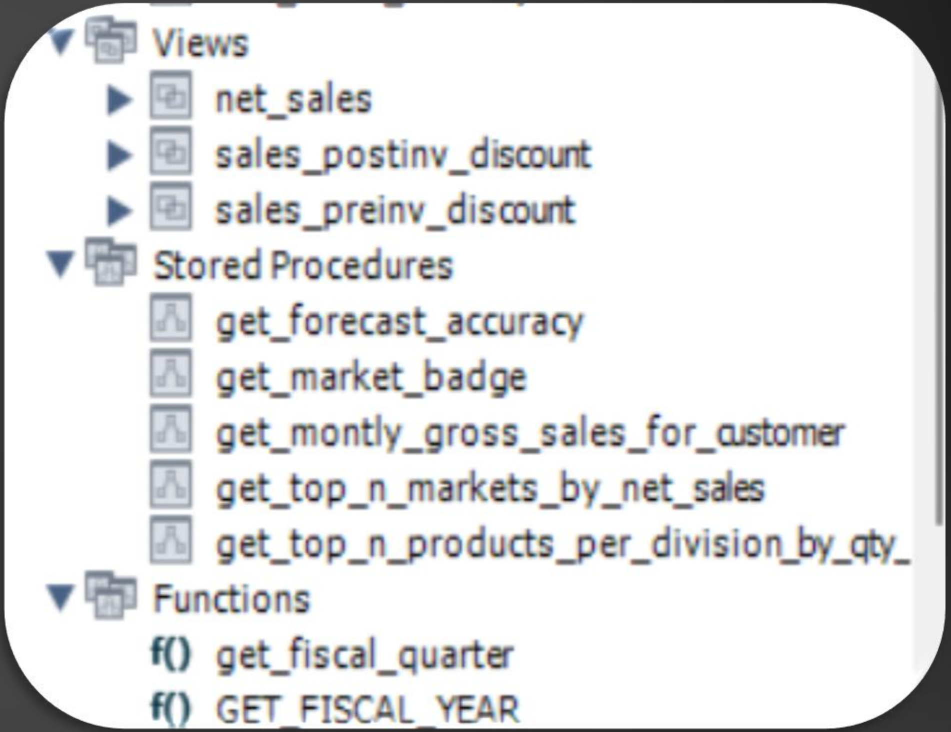
```
▼ gdb0041
    ▼ Tables
        ▶ dim_customer
        ▶ dim_date
        ▶ dim_product
        ▶ fact_act_est
        ▶ fact_forecast_monthly
        ▶ fact_freight_cost
        ▶ fact_gross_price
        ▶ fact_manufacturing_cost
        ▶ fact_post_invoice_deductions
        ▶ fact_pre_invoice_deductions
        ▶ fact_sales_monthly
```

PROJECT CONTAINS

1.Views

2.Stored Procedure

3.Functions

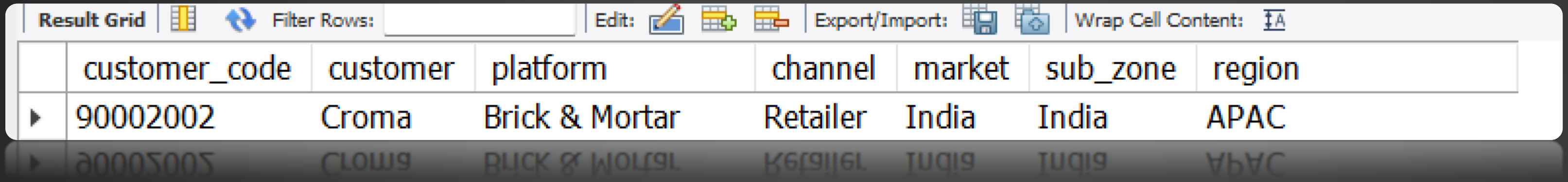


- Views
  - net_sales
  - sales_postinv_discount
  - sales_preinv_discount
- Stored Procedures
  - get_forecast_accuracy
  - get_market_badge
  - get_montly_gross_sales_for_customer
  - get_top_n_markets_by_net_sales
  - get_top_n_products_per_division_by_qty_
- Functions
  - f() get_fiscal_quarter
  - f() GET_FISCAL_YEAR

FINANCE ANALYTICS

1: First grab customer codes for Croma india

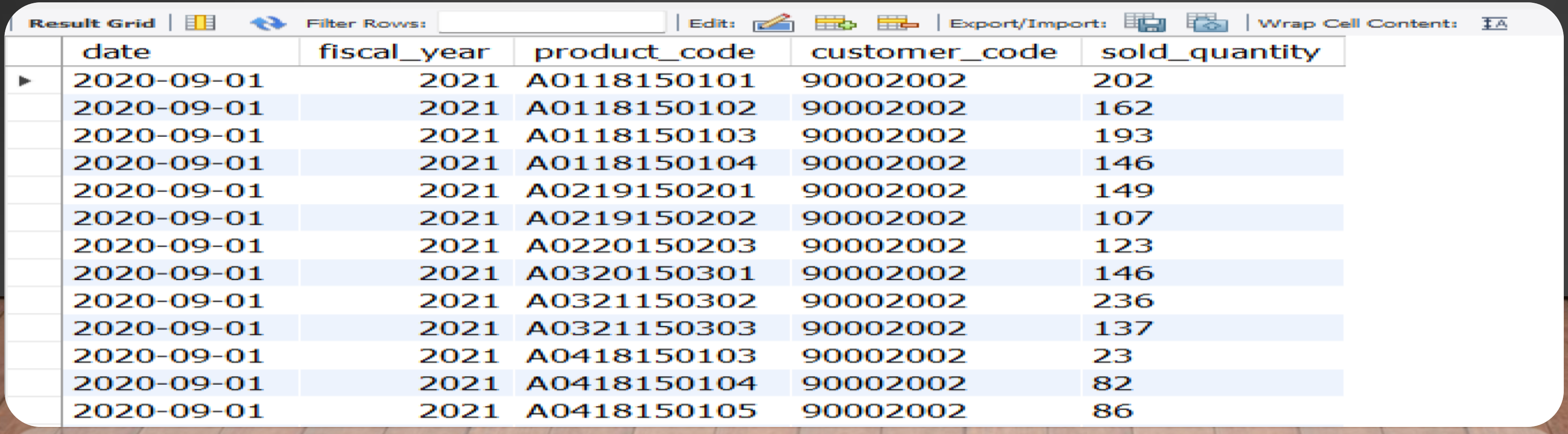
Query: SELECT * FROM dim_customer WHERE customer like "%croma%" AND market="India";



customer_code	customer	platform	channel	market	sub_zone	region
90002002	Croma	Brick & Mortar	Retailer	India	India	APAC

2: Get all the sales transaction data from fact_sales_monthly table for that customer(croma: 90002002) in the fiscal_year 2021.

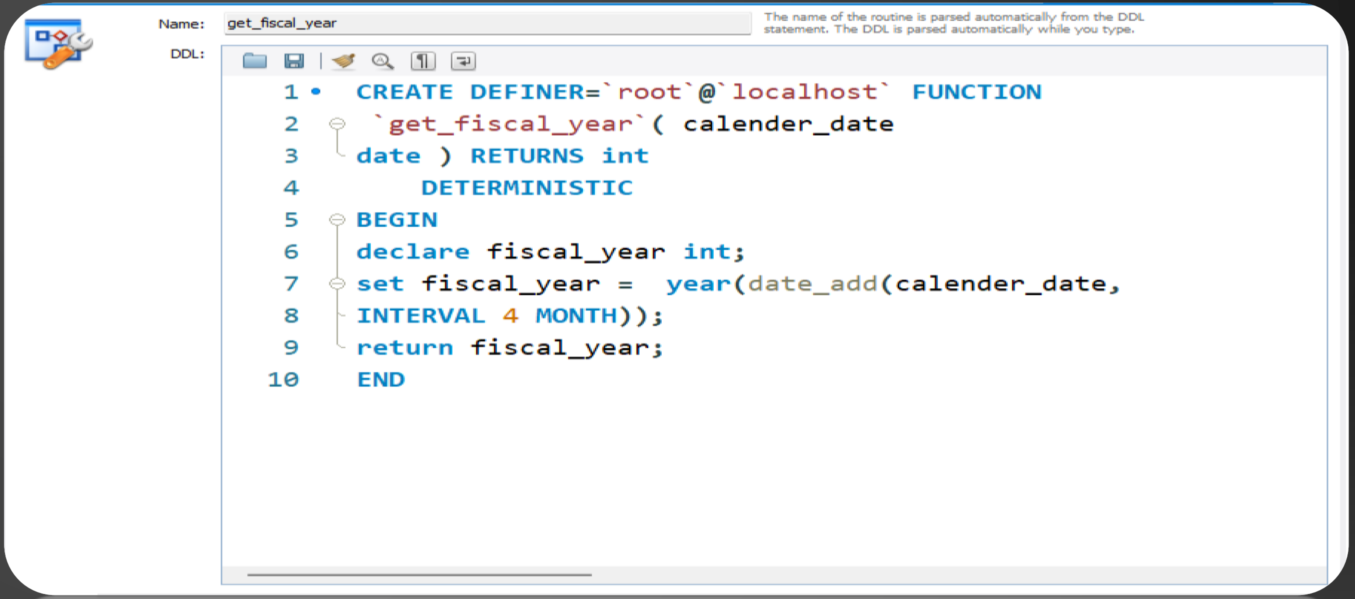
Query: SELECT * FROM fact_sales_monthly WHERE customer_code=90002002 AND YEAR(DATE_ADD(date, INTERVAL 4 MONTH))=2021 ORDER BY date asc LIMIT 100000;



date	fiscal_year	product_code	customer_code	sold_quantity
2020-09-01	2021	A0118150101	90002002	202
2020-09-01	2021	A0118150102	90002002	162
2020-09-01	2021	A0118150103	90002002	193
2020-09-01	2021	A0118150104	90002002	146
2020-09-01	2021	A0219150201	90002002	149
2020-09-01	2021	A0219150202	90002002	107
2020-09-01	2021	A0220150203	90002002	123
2020-09-01	2021	A0320150301	90002002	146
2020-09-01	2021	A0321150302	90002002	236
2020-09-01	2021	A0321150303	90002002	137
2020-09-01	2021	A0418150103	90002002	23
2020-09-01	2021	A0418150104	90002002	82
2020-09-01	2021	A0418150105	90002002	86

3. create a function 'get_fiscal_year' to get fiscal year by passing the date

```
CREATE FUNCTION `get_fiscal_year`(calendar_date DATE) RETURNS int
DETERMINISTIC
BEGIN
DECLARE fiscal_year INT;
SET fiscal_year = YEAR
(DATE_ADD(calendar_date,
INTERVAL 4 MONTH));
 RETURN fiscal_year;
END
```



Name: get_fiscal_year

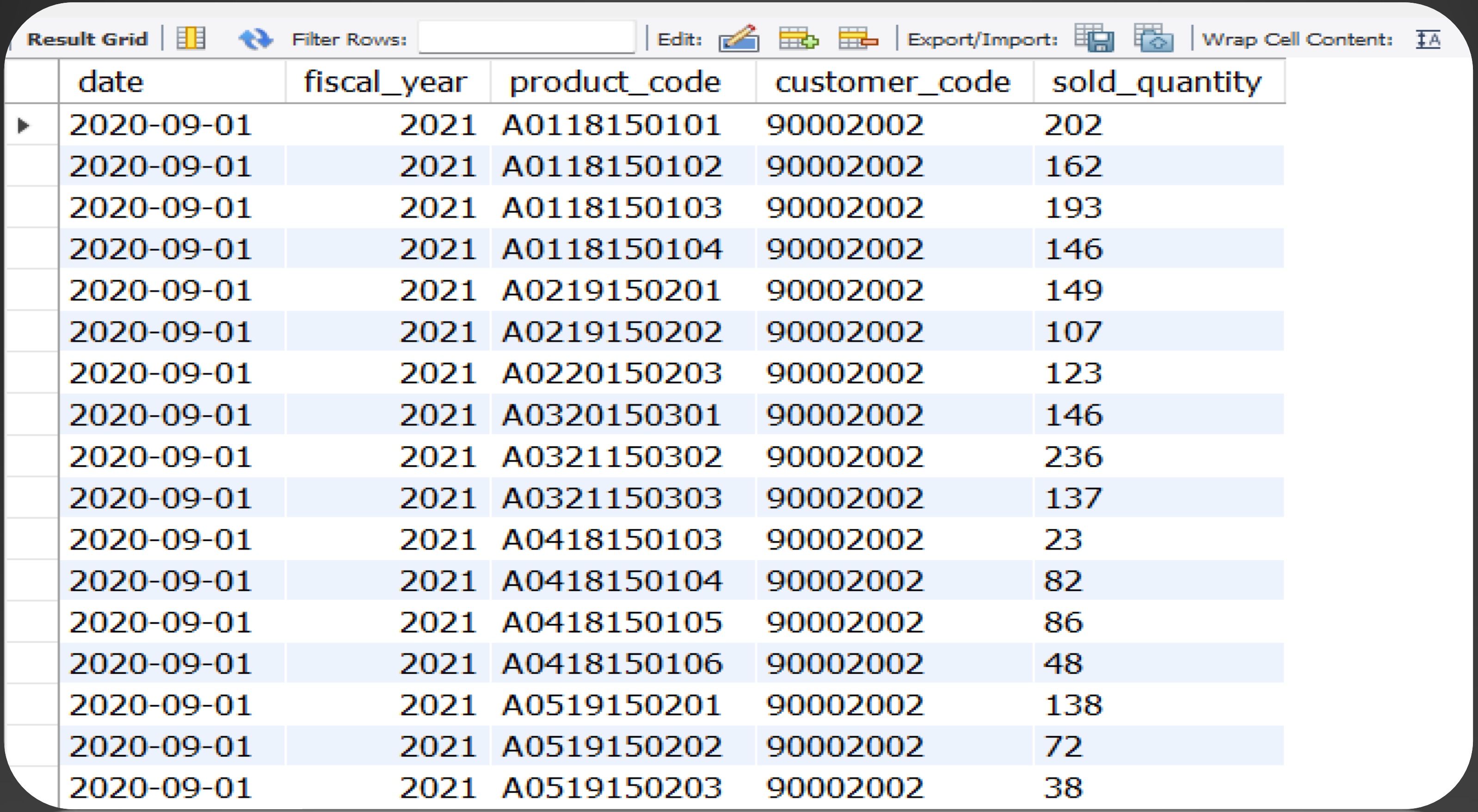
The name of the routine is parsed automatically from the DDL statement. The DDL is parsed automatically while you type.

DDL:

```
CREATE DEFINER=`root`@`localhost` FUNCTION
 `get_fiscal_year`( calender_date
date ) RETURNS int
    DETERMINISTIC
BEGIN
declare fiscal_year int;
set fiscal_year =  year(date_add(calender_date,
INTERVAL 4 MONTH));
return fiscal_year;
END
```

- **4.** Replacing the function created in the step:b

Query: SELECT * FROM fact_sales_monthly WHERE customer_code=90002002 AND get_fiscal_year(date)=2021 ORDER BY date asc LIMIT 100000;

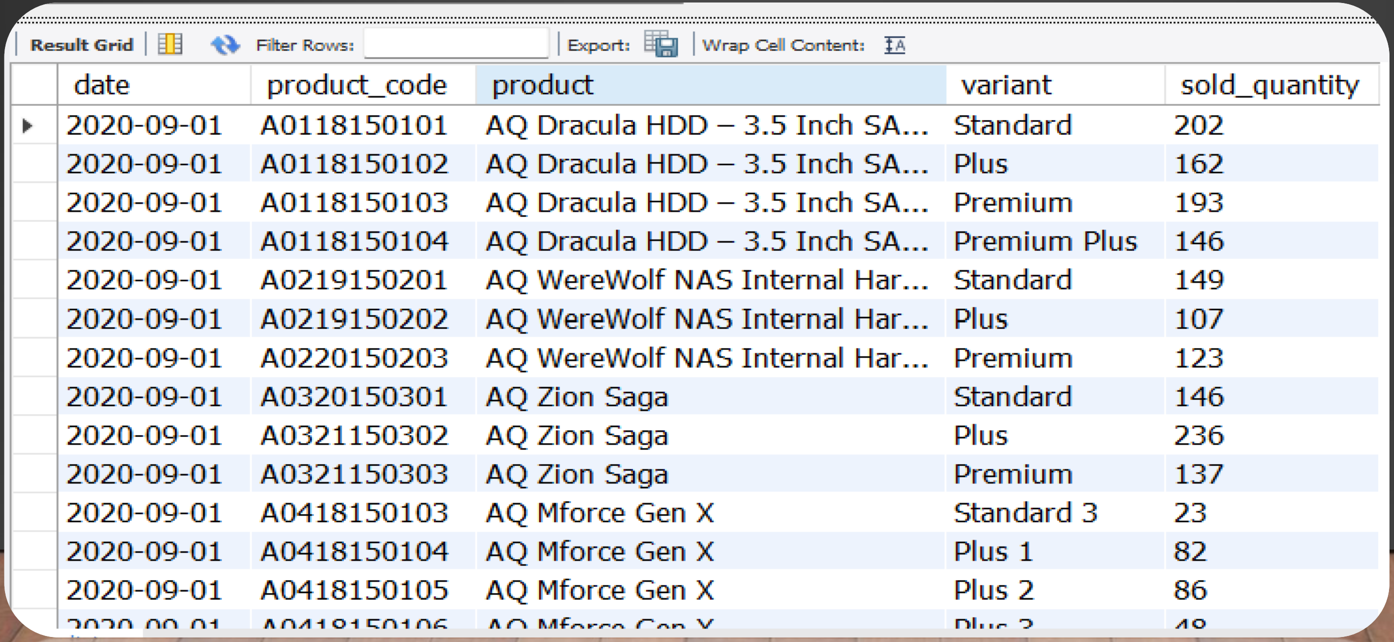


date	fiscal_year	product_code	customer_code	sold_quantity
2020-09-01	2021	A0118150101	90002002	202
2020-09-01	2021	A0118150102	90002002	162
2020-09-01	2021	A0118150103	90002002	193
2020-09-01	2021	A0118150104	90002002	146
2020-09-01	2021	A0219150201	90002002	149
2020-09-01	2021	A0219150202	90002002	107
2020-09-01	2021	A0220150203	90002002	123
2020-09-01	2021	A0320150301	90002002	146
2020-09-01	2021	A0321150302	90002002	236
2020-09-01	2021	A0321150303	90002002	137
2020-09-01	2021	A0418150103	90002002	23
2020-09-01	2021	A0418150104	90002002	82
2020-09-01	2021	A0418150105	90002002	86
2020-09-01	2021	A0418150106	90002002	48
2020-09-01	2021	A0519150201	90002002	138
2020-09-01	2021	A0519150202	90002002	72
2020-09-01	2021	A0519150203	90002002	38

❑ **Use Case:** Gross Sales Report: Monthly Product Transactions

A. Perform joins to pull product information

```
SELECT s.date, s.product_code, p.product, p.variant, s.sold_quantity FROM
fact_sales_monthly s JOIN dim_product p ON s.product_code=p.product_code WHERE
customer_code=90002002 AND get_fiscal_year(date)=2021 LIMIT 1000000;
```

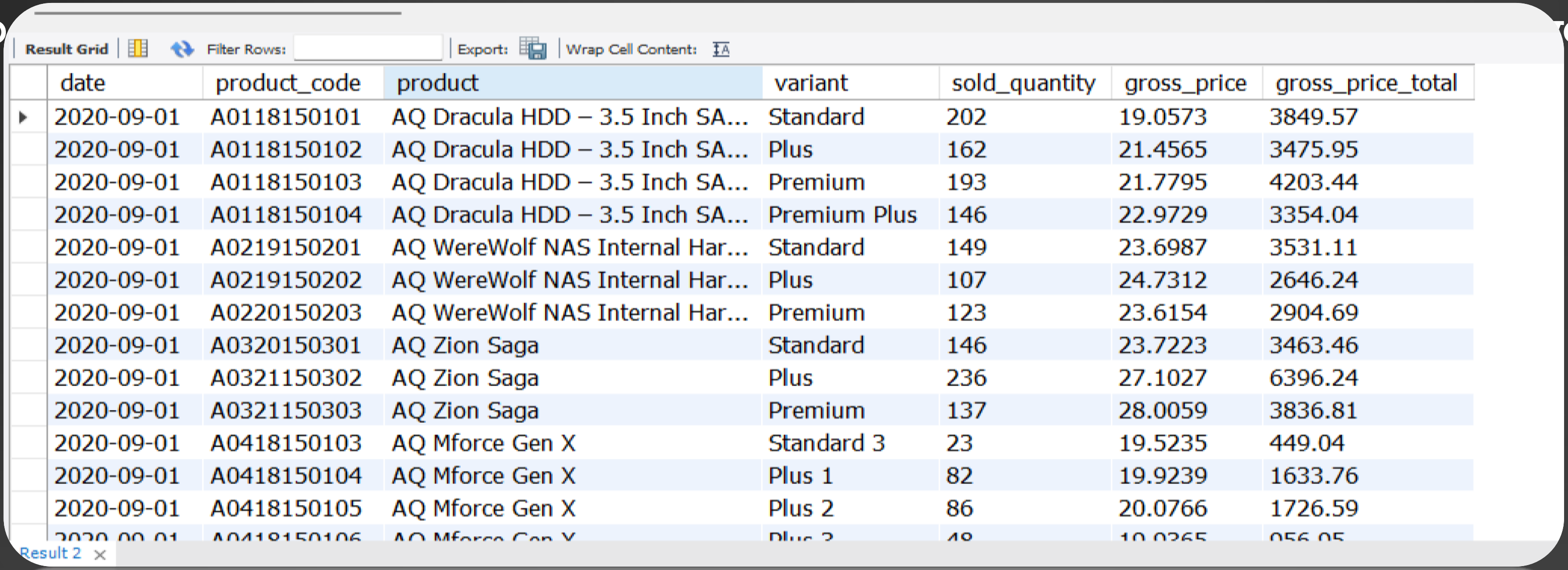


date	product_code	product	variant	sold_quantity
2020-09-01	A0118150101	AQ Dracula HDD – 3.5 Inch SA...	Standard	202
2020-09-01	A0118150102	AQ Dracula HDD – 3.5 Inch SA...	Plus	162
2020-09-01	A0118150103	AQ Dracula HDD – 3.5 Inch SA...	Premium	193
2020-09-01	A0118150104	AQ Dracula HDD – 3.5 Inch SA...	Premium Plus	146
2020-09-01	A0219150201	AQ WereWolf NAS Internal Har...	Standard	149
2020-09-01	A0219150202	AQ WereWolf NAS Internal Har...	Plus	107
2020-09-01	A0220150203	AQ WereWolf NAS Internal Har...	Premium	123
2020-09-01	A0320150301	AQ Zion Saga	Standard	146
2020-09-01	A0321150302	AQ Zion Saga	Plus	236
2020-09-01	A0321150303	AQ Zion Saga	Premium	137
2020-09-01	A0418150103	AQ Mforce Gen X	Standard 3	23
2020-09-01	A0418150104	AQ Mforce Gen X	Plus 1	82
2020-09-01	A0418150105	AQ Mforce Gen X	Plus 2	86
2020-09-01	A0418150106	AQ Mforce Gen X	Plus 3	48

B. Performing join with 'fact_gross_price' table with the above query and generating required fields

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_co

ear(s.date)=2021 LIMIT 1000000;

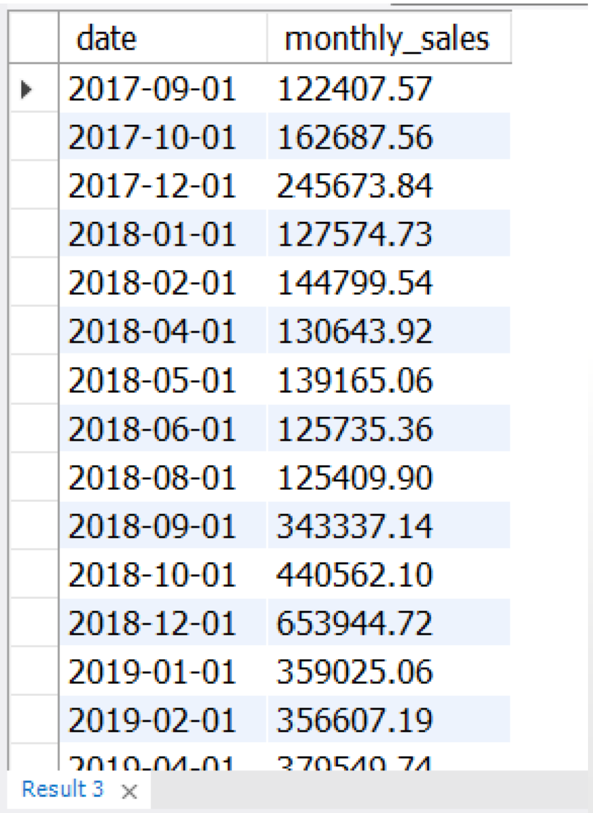


date	product_code	product	variant	sold_quantity	gross_price	gross_price_total
2020-09-01	A0118150101	AQ Dracula HDD – 3.5 Inch SA...	Standard	202	19.0573	3849.57
2020-09-01	A0118150102	AQ Dracula HDD – 3.5 Inch SA...	Plus	162	21.4565	3475.95
2020-09-01	A0118150103	AQ Dracula HDD – 3.5 Inch SA...	Premium	193	21.7795	4203.44
2020-09-01	A0118150104	AQ Dracula HDD – 3.5 Inch SA...	Premium Plus	146	22.9729	3354.04
2020-09-01	A0219150201	AQ WereWolf NAS Internal Har...	Standard	149	23.6987	3531.11
2020-09-01	A0219150202	AQ WereWolf NAS Internal Har...	Plus	107	24.7312	2646.24
2020-09-01	A0220150203	AQ WereWolf NAS Internal Har...	Premium	123	23.6154	2904.69
2020-09-01	A0320150301	AQ Zion Saga	Standard	146	23.7223	3463.46
2020-09-01	A0321150302	AQ Zion Saga	Plus	236	27.1027	6396.24
2020-09-01	A0321150303	AQ Zion Saga	Premium	137	28.0059	3836.81
2020-09-01	A0418150103	AQ Mforce Gen X	Standard 3	23	19.5235	449.04
2020-09-01	A0418150104	AQ Mforce Gen X	Plus 1	82	19.9239	1633.76
2020-09-01	A0418150105	AQ Mforce Gen X	Plus 2	86	20.0766	1726.59
2020-09-01	A0418150106	AQ Mforce Gen X	Plus 3	48	19.9365	956.95

❑ **Use Case:** Gross Sales Report: Total Sales Amount

Generate monthly gross sales report for Croma India for all the years

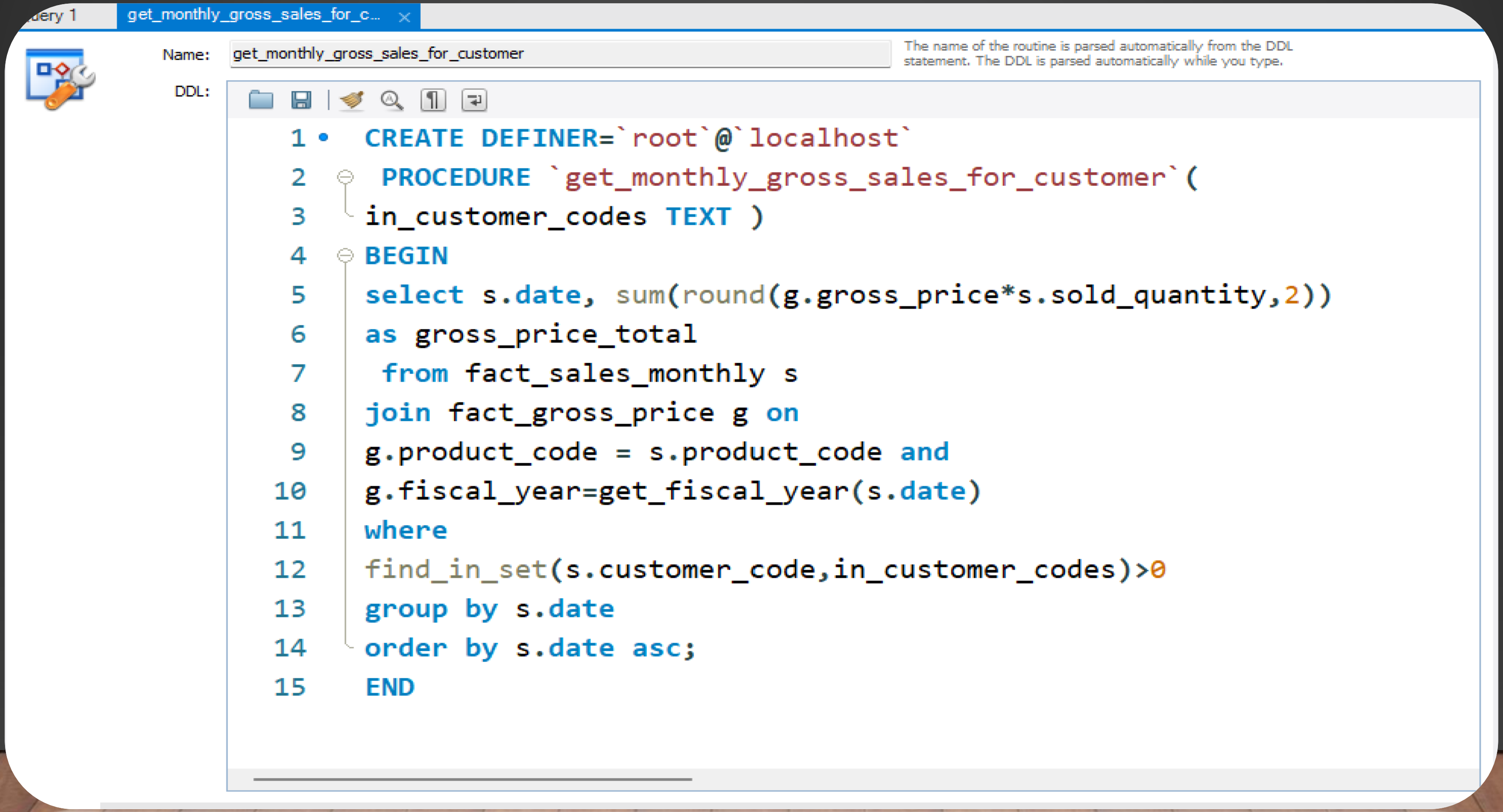
```
SELECT s.date,
SUM(ROUND(s.sold_quantity*g.gross_price,2)) as
monthly_sales FROM fact_sales_monthly s JOIN
fact_gross_price g ON
g.fiscal_year=get_fiscal_year(s.date) AND
g.product_code=s.product_code WHERE
customer_code=90002002 GROUP BY date;
```



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
2018-06-01	125735.36
2018-08-01	125409.90
2018-09-01	343337.14
2018-10-01	440562.10
2018-12-01	653944.72
2019-01-01	359025.06
2019-02-01	356607.19
2019-04-01	370540.74

❑ **Use Case:** Stored Procedures: Monthly Gross Sales Report Generate monthly

Gross sales report for any customer using **stored procedure**

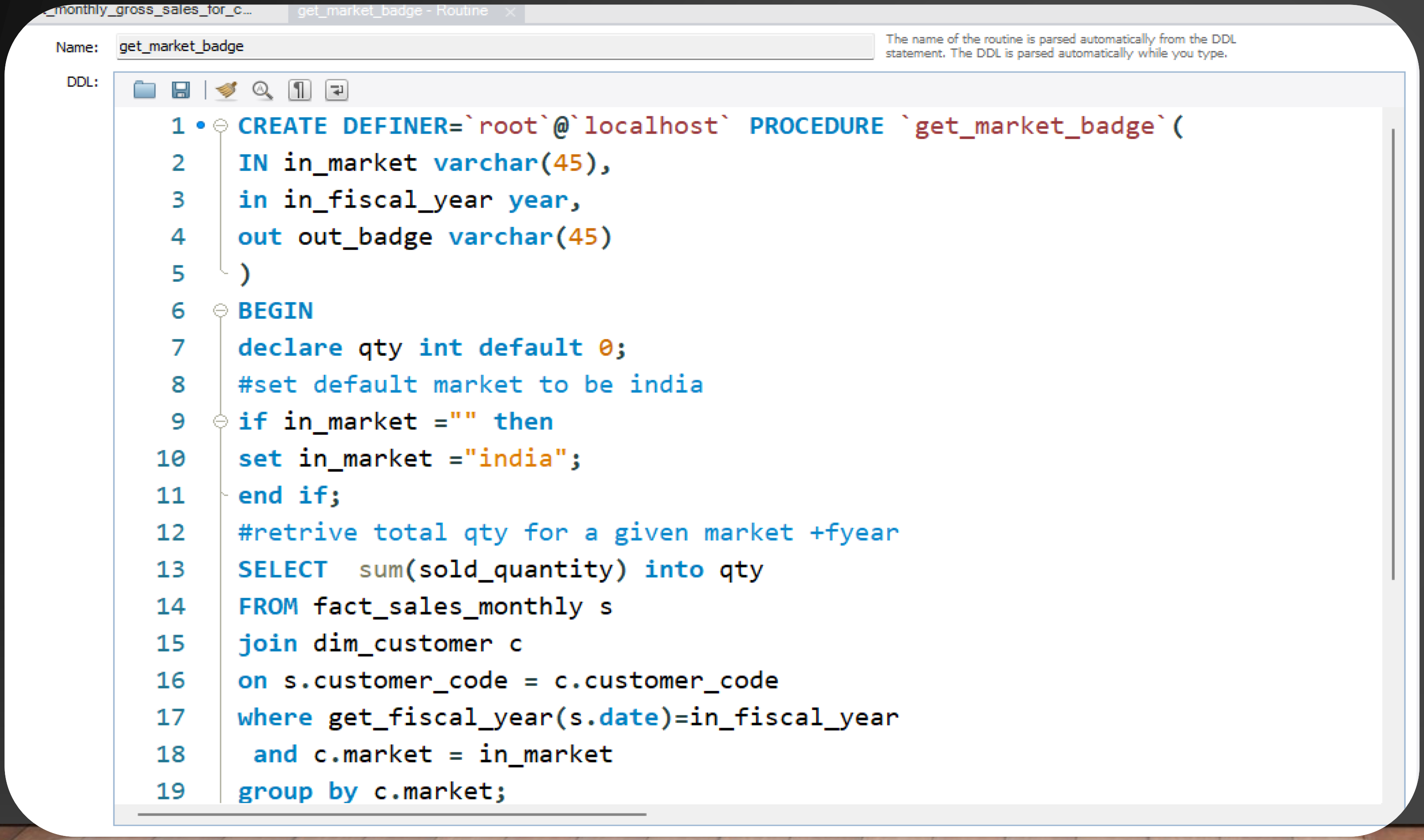


Name: get_monthly_gross_sales_for_customer

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

```
CREATE DEFINER=`root`@`localhost`
 PROCEDURE `get_monthly_gross_sales_for_customer`(
in_customer_codes TEXT )
BEGIN
select s.date, sum(round(g.gross_price*s.sold_quantity,2))
as gross_price_total
 from fact_sales_monthly s
join fact_gross_price g on
g.product_code = s.product_code and
g.fiscal_year=get_fiscal_year(s.date)
where
find_in_set(s.customer_code,in_customer_codes)>0
group by s.date
order by s.date asc;
END
```

❑ **Use Case :** Stored Procedure: Market Badge

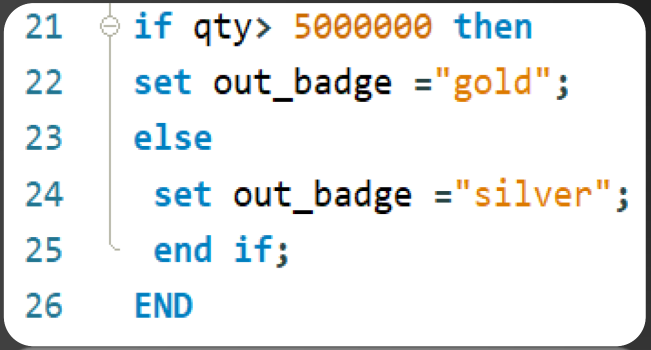


Name: get_market_badge

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

DDL:

```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 to be india
if in_market ="" then
set in_market ="india";
end if;
#retrive total qty for a given market +fyear
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;
```



```sql
if qty> 5000000 then
set out_badge ="gold";
else
 set out_badge ="silver";
 end if;
END
```

TOP CUSTOMERS, PRODUCTS, MARKETS

TASK :- 2

1. REPORT FOR TOP MARKET
2. REPORT FOR TOP PRODUCTS
3. REPORT FOR TOP CUSTOMERS

❑ Use Case: Problem Statement and Pre-Invoice Discount Report

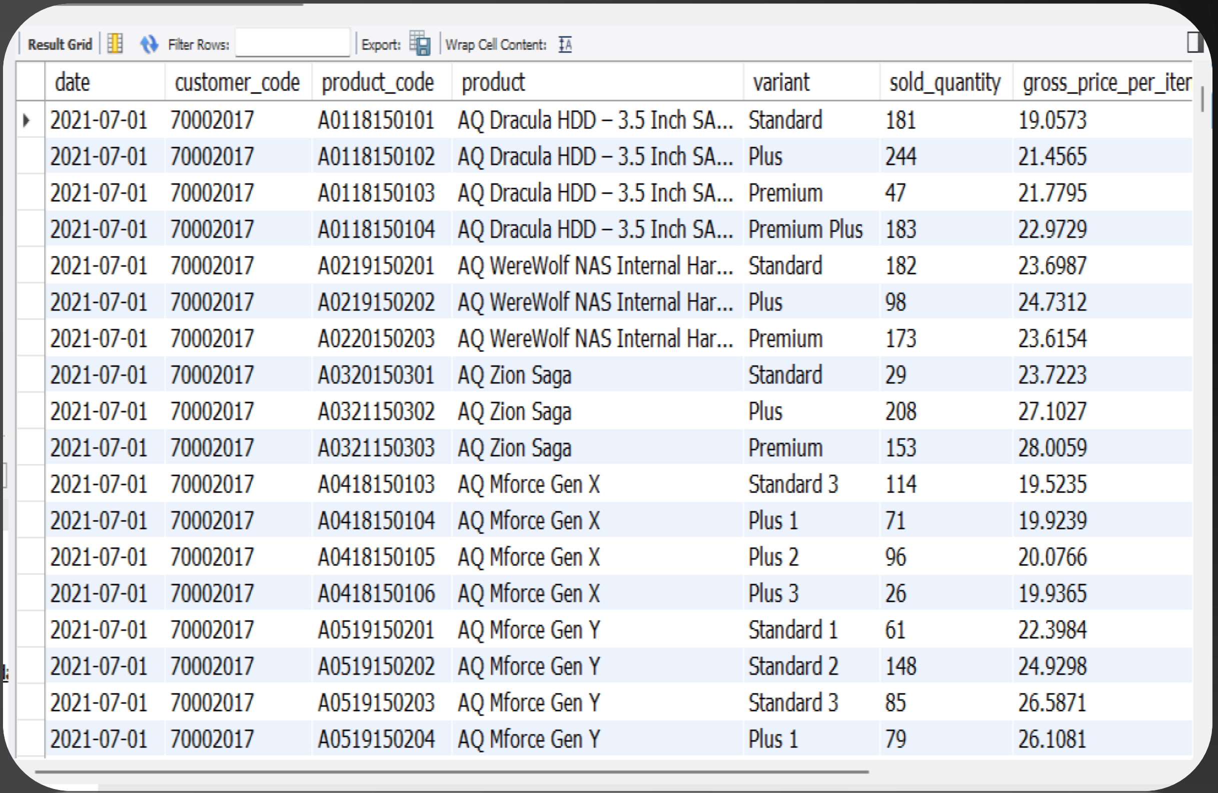
Include pre-invoice deductions in Croma detailed report

```
→ SELECT s.date, s.product_code, p.product, p.variant, s.sold_quantity, g.gross_price as gross_price_per_item,

ROUND(s.sold_quantity*g.gross_price,2) as gross_price_total, pre.pre_invoice_discount_pct

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

JOIN fact_pre_invoice_deductions as pre
ON pre.customer_code = s.customer_code AND
 pre.fiscal_year=get_fiscal_year(s.date)  WHERE s.customer_code=90002002 AND get_fiscal_year(s.date)=2021
LIMIT 1000000;
```

❑Use Case: Performance Improvement # 1

creating dim_date and joining with this table and avoid using the function 'get_fiscal_year()' to reduce the amount of time taking to run the query

```
SELECT
s.date, s.customer_code,  s.product_code, p.product, p.variant,
s.sold_quantity, g.gross_price as gross_price_per_item,
 ROUND(s.sold_quantity*g.gross_price,2) as gross_price_total,
 pre.pre_invoice_discount_pct
 FROM fact_sales_monthly s
 JOIN dim_date dt  ON dt.calendar_date = s.date
 JOIN dim_product p ON s.product_code=p.product_code
 JOIN fact_gross_price g  ON g.fiscal_year=dt.fiscal_year
 AND g.product_code=s.product_code
 JOIN fact_pre_invoice_deductions as pre
 ON pre.customer_code = s.customer_code AND
 pre.fiscal_year=dt.fiscal_year
```

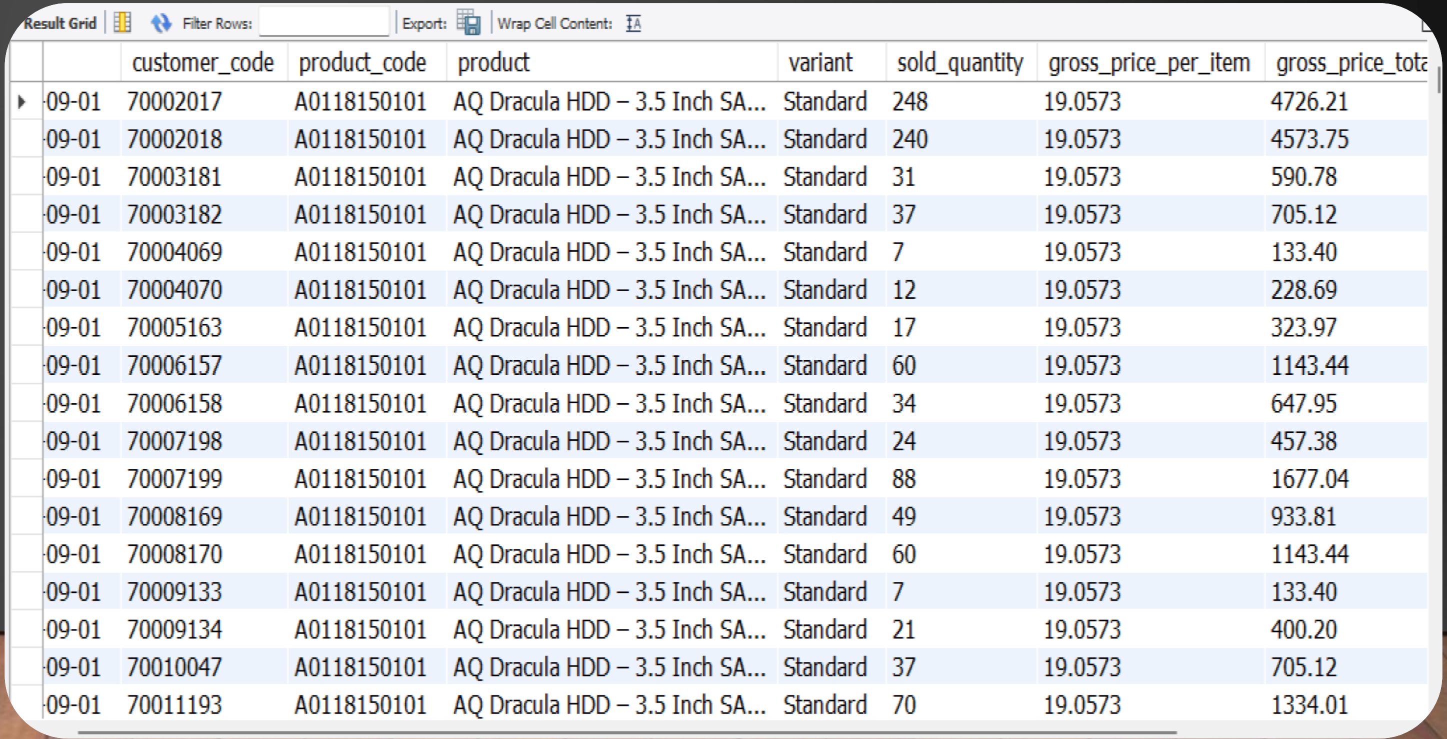


date	customer_code	product_code	product	variant	sold_quantity	gross_price_per_iter
2021-07-01	70002017	A0118150101	AQ Dracula HDD – 3.5 Inch SA...	Standard	181	19.0573
2021-07-01	70002017	A0118150102	AQ Dracula HDD – 3.5 Inch SA...	Plus	244	21.4565
2021-07-01	70002017	A0118150103	AQ Dracula HDD – 3.5 Inch SA...	Premium	47	21.7795
2021-07-01	70002017	A0118150104	AQ Dracula HDD – 3.5 Inch SA...	Premium Plus	183	22.9729
2021-07-01	70002017	A0219150201	AQ WereWolf NAS Internal Har...	Standard	182	23.6987
2021-07-01	70002017	A0219150202	AQ WereWolf NAS Internal Har...	Plus	98	24.7312
2021-07-01	70002017	A0220150203	AQ WereWolf NAS Internal Har...	Premium	173	23.6154
2021-07-01	70002017	A0320150301	AQ Zion Saga	Standard	29	23.7223
2021-07-01	70002017	A0321150302	AQ Zion Saga	Plus	208	27.1027
2021-07-01	70002017	A0321150303	AQ Zion Saga	Premium	153	28.0059
2021-07-01	70002017	A0418150103	AQ Mforce Gen X	Standard 3	114	19.5235
2021-07-01	70002017	A0418150104	AQ Mforce Gen X	Plus 1	71	19.9239
2021-07-01	70002017	A0418150105	AQ Mforce Gen X	Plus 2	96	20.0766
2021-07-01	70002017	A0418150106	AQ Mforce Gen X	Plus 3	26	19.9365
2021-07-01	70002017	A0519150201	AQ Mforce Gen Y	Standard 1	61	22.3984
2021-07-01	70002017	A0519150202	AQ Mforce Gen Y	Standard 2	148	24.9298
2021-07-01	70002017	A0519150203	AQ Mforce Gen Y	Standard 3	85	26.5871
2021-07-01	70002017	A0519150204	AQ Mforce Gen Y	Plus 1	79	26.1081

❑Use Case: Performance Improvement # 2

Added the fiscal year in the fact_sales_monthly table itself

```
SELECT
s.date, s.customer_code, s.product_code, p.product,
p.variant, s.sold_quantity, g.gross_price as gross_price_per_item,
 ROUND(s.sold_quantity*g.gross_price,2) as gross_price_total,
 pre.pre_invoice_discount_pct
 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=s.fiscal_year
 AND g.product_code=s.product_code
 JOIN fact_pre_invoice_deductions as pre
 ON pre.customer_code = s.customer_code AND
 pre.fiscal_year=s.fiscal_year
 WHERE
s.fiscal_year=2021
LIMIT 1500000
```

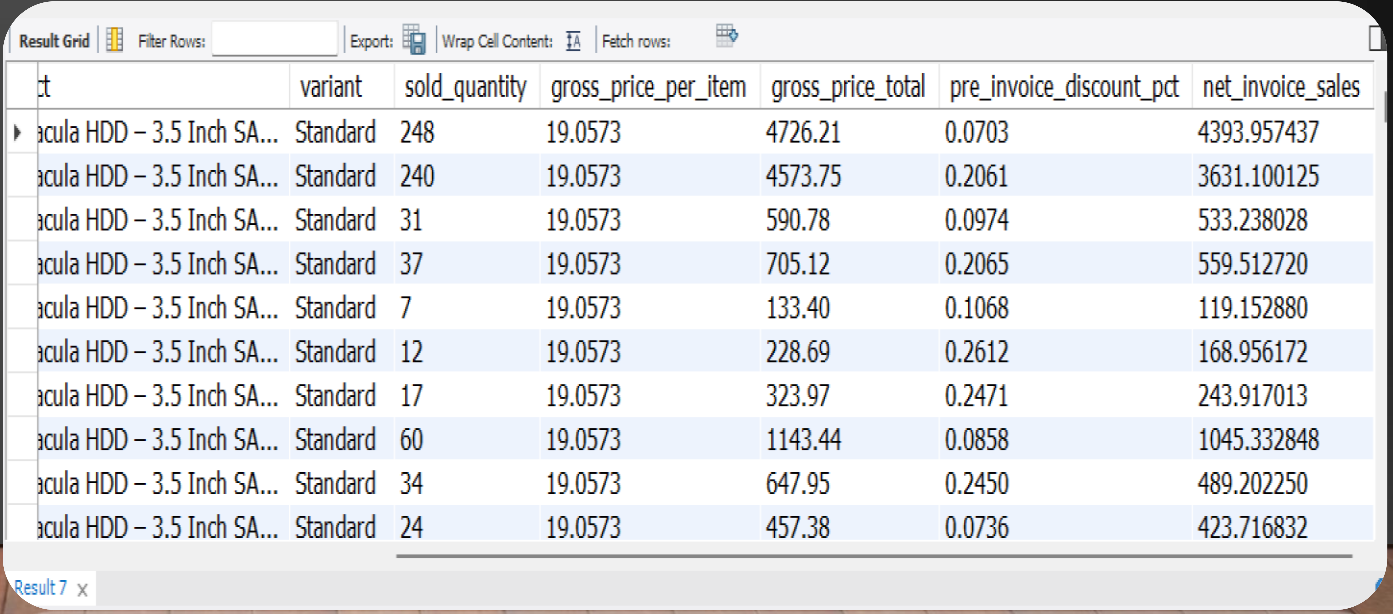


	customer_code	product_code	product	variant	sold_quantity	gross_price_per_item	gross_price_tota
09-01	70002017	A0118150101	AQ Dracula HDD – 3.5 Inch SA...	Standard	248	19.0573	4726.21
09-01	70002018	A0118150101	AQ Dracula HDD – 3.5 Inch SA...	Standard	240	19.0573	4573.75
09-01	70003181	A0118150101	AQ Dracula HDD – 3.5 Inch SA...	Standard	31	19.0573	590.78
09-01	70003182	A0118150101	AQ Dracula HDD – 3.5 Inch SA...	Standard	37	19.0573	705.12
09-01	70004069	A0118150101	AQ Dracula HDD – 3.5 Inch SA...	Standard	7	19.0573	133.40
09-01	70004070	A0118150101	AQ Dracula HDD – 3.5 Inch SA...	Standard	12	19.0573	228.69
09-01	70005163	A0118150101	AQ Dracula HDD – 3.5 Inch SA...	Standard	17	19.0573	323.97
09-01	70006157	A0118150101	AQ Dracula HDD – 3.5 Inch SA...	Standard	60	19.0573	1143.44
09-01	70006158	A0118150101	AQ Dracula HDD – 3.5 Inch SA...	Standard	34	19.0573	647.95
09-01	70007198	A0118150101	AQ Dracula HDD – 3.5 Inch SA...	Standard	24	19.0573	457.38
09-01	70007199	A0118150101	AQ Dracula HDD – 3.5 Inch SA...	Standard	88	19.0573	1677.04
09-01	70008169	A0118150101	AQ Dracula HDD – 3.5 Inch SA...	Standard	49	19.0573	933.81
09-01	70008170	A0118150101	AQ Dracula HDD – 3.5 Inch SA...	Standard	60	19.0573	1143.44
09-01	70009133	A0118150101	AQ Dracula HDD – 3.5 Inch SA...	Standard	7	19.0573	133.40
09-01	70009134	A0118150101	AQ Dracula HDD – 3.5 Inch SA...	Standard	21	19.0573	400.20
09-01	70010047	A0118150101	AQ Dracula HDD – 3.5 Inch SA...	Standard	37	19.0573	705.12
09-01	70011193	A0118150101	AQ Dracula HDD – 3.5 Inch SA...	Standard	70	19.0573	1334.01

❑Get the net_invoice_sales amount using the CTE’s

```
WITH cte1 AS (
 SELECT
s.date, s.customer_code,   s.product_code, p.product, p.variant,
s.sold_quantity, g.gross_price as gross_price_per_item,
 ROUND(s.sold_quantity*g.gross_price,2) as gross_price_total,
 pre.pre_invoice_discount_pct
 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=s.fiscal_year
 AND g.product_code=s.product_code
 JOIN fact_pre_invoice_deductions as pre
 ON pre.customer_code = s.customer_code AND
 pre.fiscal_year=s.fiscal_year WHERE s.fiscal_year=2021)
```

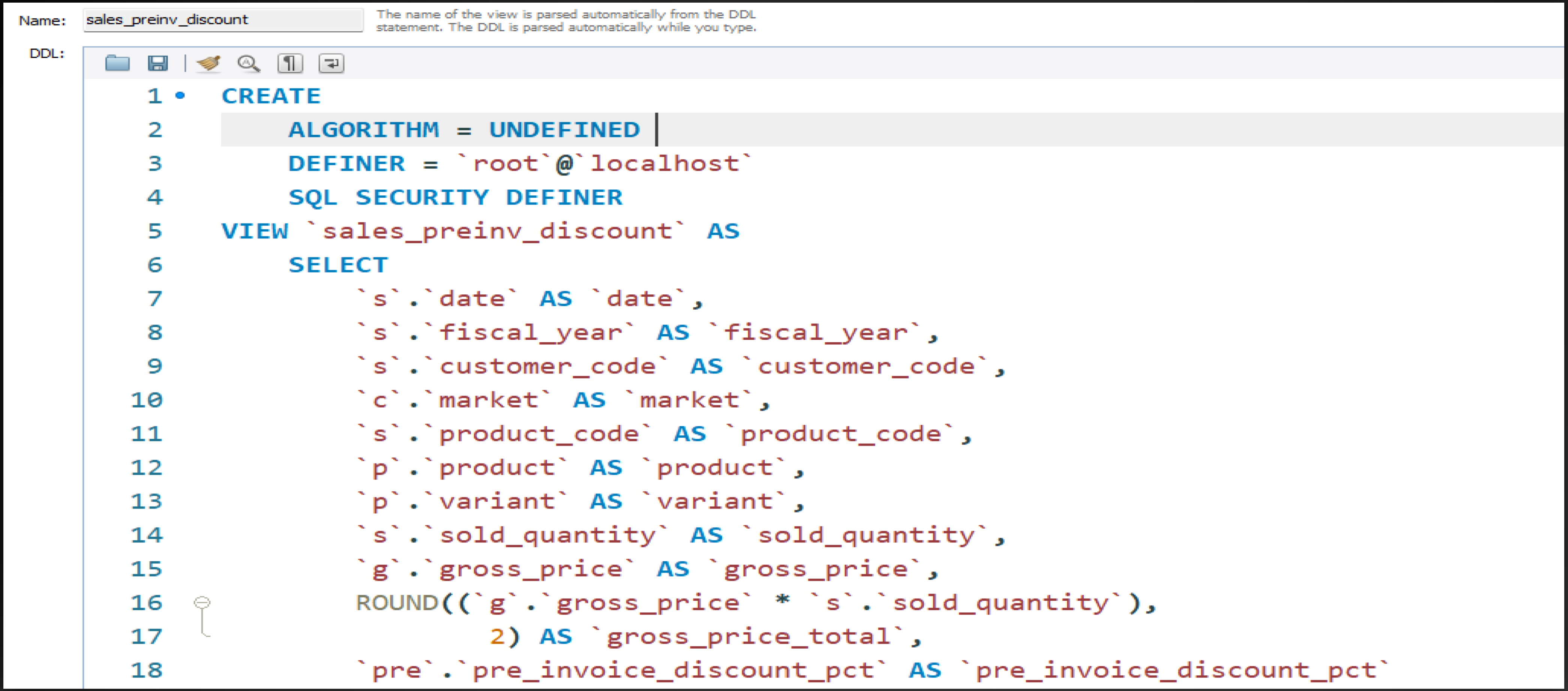
```
SELECT
*,
(gross_price_total
pre_invoice_discount_pct*gross_price_total) as
 net_invoice_sales
 FROM cte1
```



ct	variant	sold_quantity	gross_price_per_item	gross_price_total	pre_invoice_discount_pct	net_invoice_sales
acula HDD – 3.5 Inch SA...	Standard	248	19.0573	4726.21	0.0703	4393.957437
acula HDD – 3.5 Inch SA...	Standard	240	19.0573	4573.75	0.2061	3631.100125
acula HDD – 3.5 Inch SA...	Standard	31	19.0573	590.78	0.0974	533.238028
acula HDD – 3.5 Inch SA...	Standard	37	19.0573	705.12	0.2065	559.512720
acula HDD – 3.5 Inch SA...	Standard	7	19.0573	133.40	0.1068	119.152880
acula HDD – 3.5 Inch SA...	Standard	12	19.0573	228.69	0.2612	168.956172
acula HDD – 3.5 Inch SA...	Standard	17	19.0573	323.97	0.2471	243.917013
acula HDD – 3.5 Inch SA...	Standard	60	19.0573	1143.44	0.0858	1045.332848
acula HDD – 3.5 Inch SA...	Standard	34	19.0573	647.95	0.2450	489.202250
acula HDD – 3.5 Inch SA...	Standard	24	19.0573	457.38	0.0736	423.716832

❑ **Creating the view `sales_preinv_discount` and store all the data in like a virtual table**

```
CREATE  VIEW `sales_preinv_discount` AS
 SELECT
s.date, s.fiscal_year, s.customer_code,
 c.market, s.product_code, p.product,
p.variant, s.sold_quantity, g.gross_price as gross_price_per_item,
 ROUND(s.sold_quantity*g.gross_price,2) as gross_price_total,
 pre.pre_invoice_discount_pct
 FROM fact_sales_monthly s
 JOIN dim_customer c ON s.customer_code = c.customer_code
 JOIN dim_product p ON s.product_code=p.product_code
 JOIN fact_gross_price ON g.fiscal_year=s.fiscal_year
 AND g.product_code=s.product_code
JOIN fact_pre_invoice_deductions as pre  ON pre.customer_code = s.customer_code
 AND pre.fiscal_year=s.fiscal_year
```

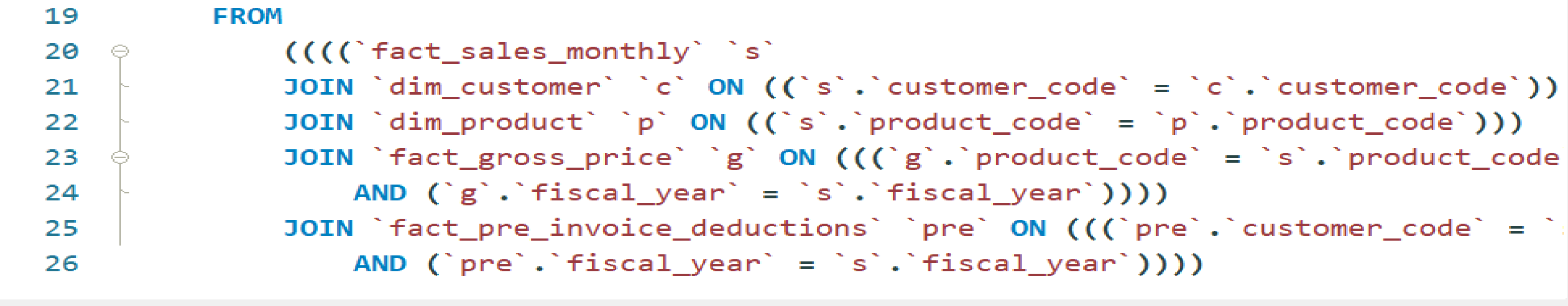


Name: sales_preinv_discount

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

DDL:

```sql
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `sales_preinv_discount` AS
    SELECT
        `s`.`date` AS `date`,
        `s`.`fiscal_year` AS `fiscal_year`,
        `s`.`customer_code` AS `customer_code`,
        `c`.`market` AS `market`,
        `s`.`product_code` AS `product_code`,
        `p`.`product` AS `product`,
        `p`.`variant` AS `variant`,
        `s`.`sold_quantity` AS `sold_quantity`,
        `g`.`gross_price` AS `gross_price`,
        ROUND((`g`.`gross_price` * `s`.`sold_quantity`),
                2) AS `gross_price_total`,
        `pre`.`pre_invoice_discount_pct` AS `pre_invoice_discount_pct`
```

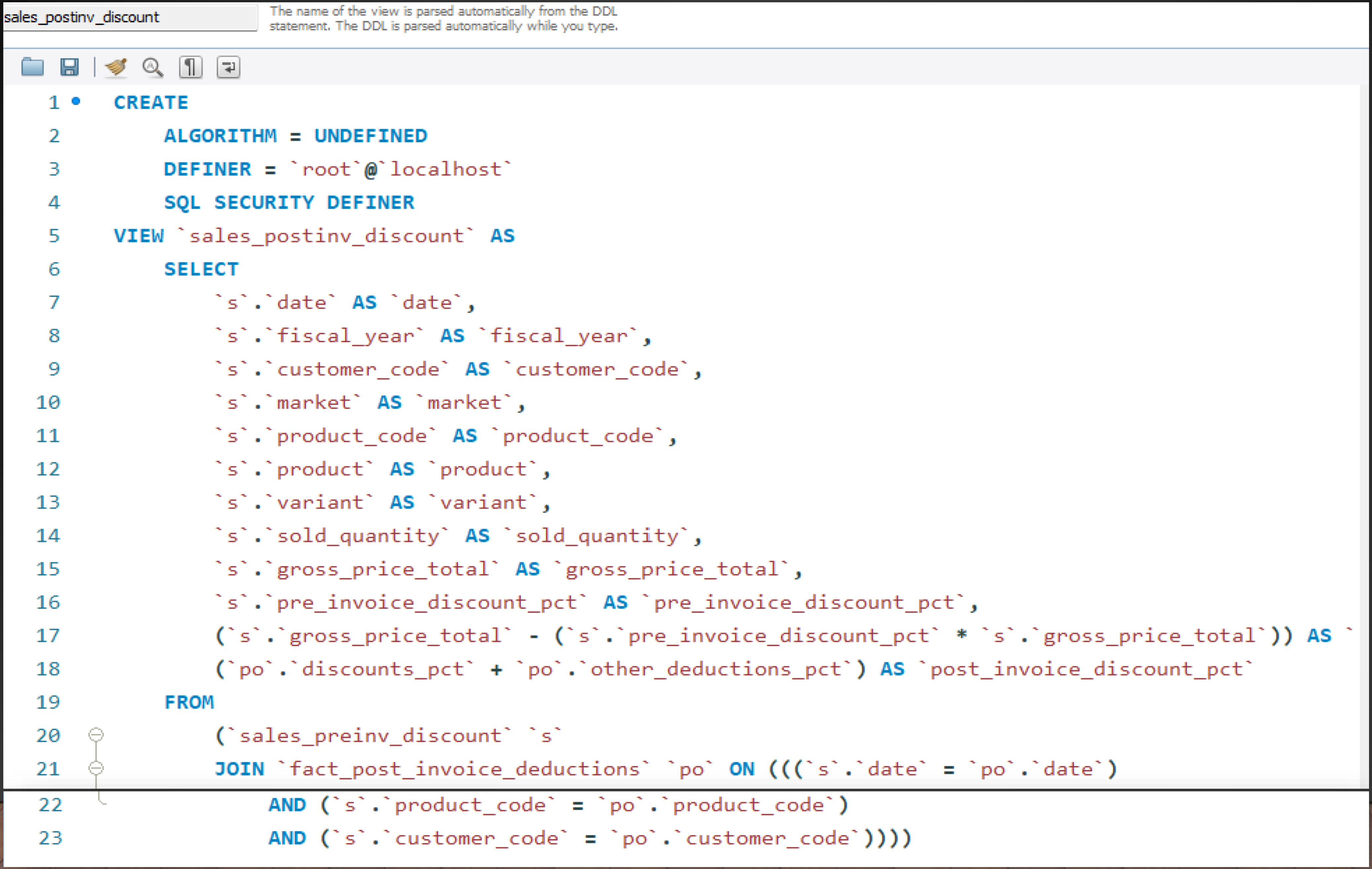


```sql
    FROM
        ((((`fact_sales_monthly` `s`
        JOIN `dim_customer` `c` ON ((`s`.`customer_code` = `c`.`customer_code`)))
        JOIN `dim_product` `p` ON ((`s`.`product_code` = `p`.`product_code`)))
        JOIN `fact_gross_price` `g` ON (((`g`.`product_code` = `s`.`product_code
            AND (`g`.`fiscal_year` = `s`.`fiscal_year`))))
        JOIN `fact_pre_invoice_deductions` `pre` ON (((`pre`.`customer_code` = `
            AND (`pre`.`fiscal_year` = `s`.`fiscal_year`))))
```

❑**Use Case: Database Views: Post Invoice Discount, Net Sales**

Create a view for post invoice deductions: `sales_postinv_discount`

```
 CREATE VIEW `sales_postinv_discount` AS
 SELECT
s.date, s.fiscal_year,
 s.customer_code, s.market,
 s.product_code, s.product, s.variant,
 s.sold_quantity, s.gross_price_total,
 s.pre_invoice_discount_pct,
 (s.gross_price_total-s.pre_invoice_discount_pct*s.gross_price_total) as net_invoice_sales,
 (po.discounts_pct+po.other_deductions_pct) as post_invoice_discount_pct
 FROM sales_preinv_discount s
 JOIN fact_post_invoice_deductions po
 ON po.customer_code = s.customer_code AND po.product_code = s.product_code AND
 po.date = s.date;
```



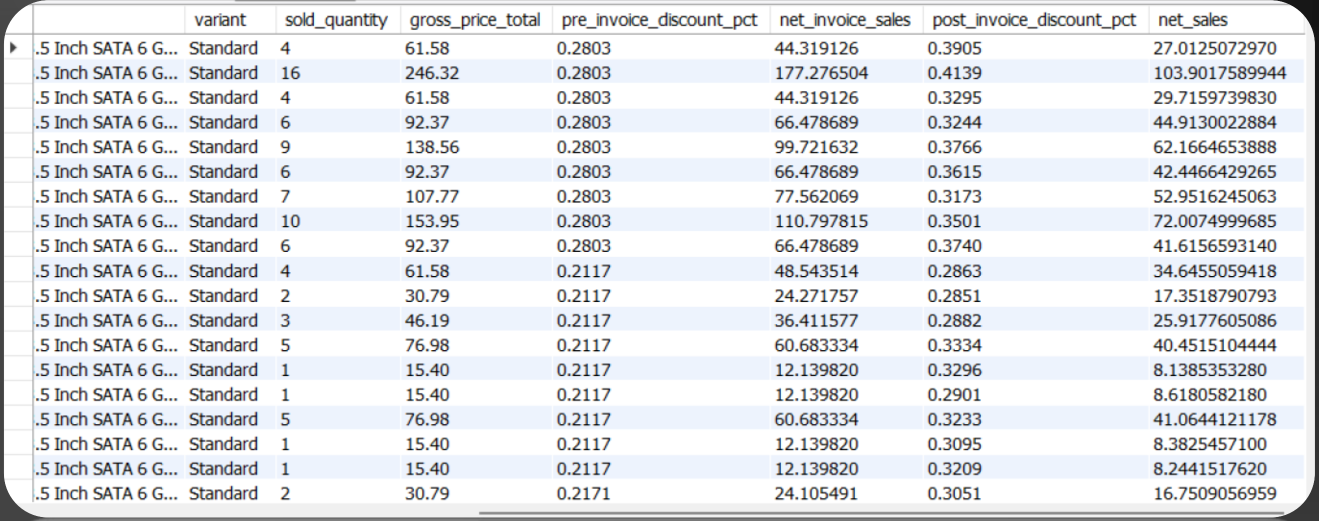
sales_postinv_discount

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

```sql
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `sales_postinv_discount` AS
    SELECT
        `s`.`date` AS `date`,
        `s`.`fiscal_year` AS `fiscal_year`,
        `s`.`customer_code` AS `customer_code`,
        `s`.`market` AS `market`,
        `s`.`product_code` AS `product_code`,
        `s`.`product` AS `product`,
        `s`.`variant` AS `variant`,
        `s`.`sold_quantity` AS `sold_quantity`,
        `s`.`gross_price_total` AS `gross_price_total`,
        `s`.`pre_invoice_discount_pct` AS `pre_invoice_discount_pct`,
        (`s`.`gross_price_total` - (`s`.`pre_invoice_discount_pct` * `s`.`gross_price_total`)) AS `
        (`po`.`discounts_pct` + `po`.`other_deductions_pct`) AS `post_invoice_discount_pct`
    FROM
        (`sales_preinv_discount` `s`
        JOIN `fact_post_invoice_deductions` `po` ON (((`s`.`date` = `po`.`date`)
            AND (`s`.`product_code` = `po`.`product_code`)
            AND (`s`.`customer_code` = `po`.`customer_code`))))
```

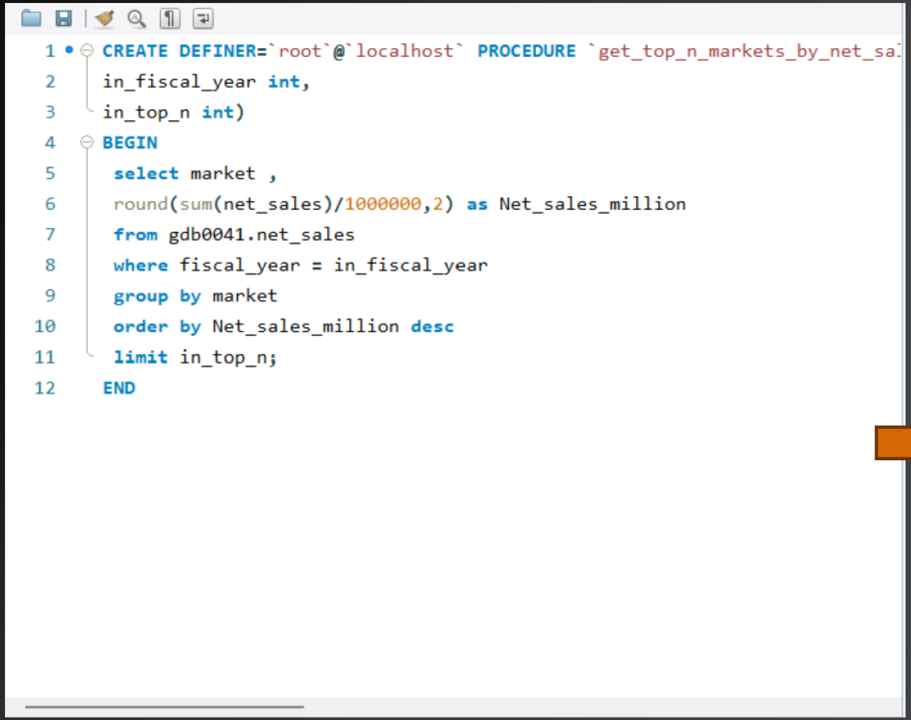
❑Create a report for net sales

```
SELECT
*,
net_invoice_sales*(1-post_invoice_discount_pct) as net_sales
 FROM gdb0041.sales_postinv_discount;


-- Finally creating the view `net_sales` which inbuiltly use/include all the previous created view and gives
the final result
 CREATE VIEW `net_sales` AS
 SELECT
*,
net_invoice_sales*(1-post_invoice_discount_pct)
as net_sales
 FROM gdb0041.sales_postinv_discount;
```

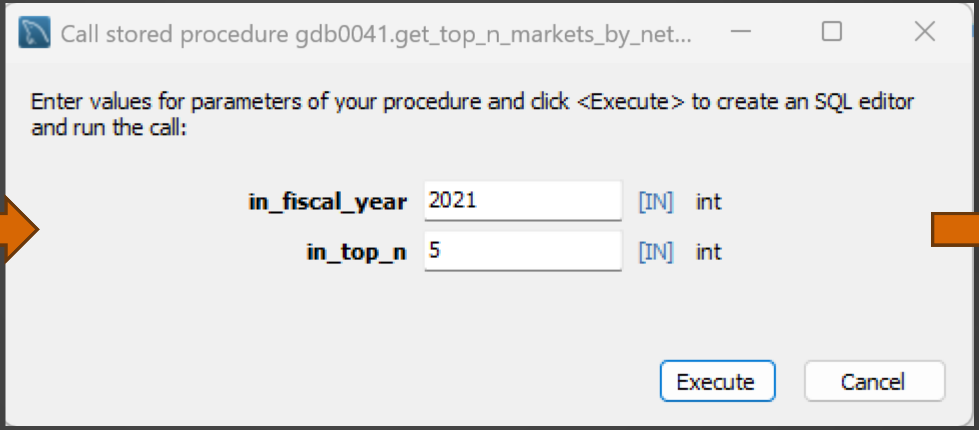


	variant	sold_quantity	gross_price_total	pre_invoice_discount_pct	net_invoice_sales	post_invoice_discount_pct	net_sales
.5 Inch SATA 6 G...	Standard	4	61.58	0.2803	44.319126	0.3905	27.0125072970
.5 Inch SATA 6 G...	Standard	16	246.32	0.2803	177.276504	0.4139	103.9017589944
.5 Inch SATA 6 G...	Standard	4	61.58	0.2803	44.319126	0.3295	29.7159739830
.5 Inch SATA 6 G...	Standard	6	92.37	0.2803	66.478689	0.3244	44.9130022884
.5 Inch SATA 6 G...	Standard	9	138.56	0.2803	99.721632	0.3766	62.1664653888
.5 Inch SATA 6 G...	Standard	6	92.37	0.2803	66.478689	0.3615	42.4466429265
.5 Inch SATA 6 G...	Standard	7	107.77	0.2803	77.562069	0.3173	52.9516245063
.5 Inch SATA 6 G...	Standard	10	153.95	0.2803	110.797815	0.3501	72.0074999685
.5 Inch SATA 6 G...	Standard	6	92.37	0.2803	66.478689	0.3740	41.6156593140
.5 Inch SATA 6 G...	Standard	4	61.58	0.2117	48.543514	0.2863	34.6455059418
.5 Inch SATA 6 G...	Standard	2	30.79	0.2117	24.271757	0.2851	17.3518790793
.5 Inch SATA 6 G...	Standard	3	46.19	0.2117	36.411577	0.2882	25.9177605086
.5 Inch SATA 6 G...	Standard	5	76.98	0.2117	60.683334	0.3334	40.4515104444
.5 Inch SATA 6 G...	Standard	1	15.40	0.2117	12.139820	0.3296	8.1385353280
.5 Inch SATA 6 G...	Standard	1	15.40	0.2117	12.139820	0.2901	8.6180582180
.5 Inch SATA 6 G...	Standard	5	76.98	0.2117	60.683334	0.3233	41.0644121178
.5 Inch SATA 6 G...	Standard	1	15.40	0.2117	12.139820	0.3095	8.3825457100
.5 Inch SATA 6 G...	Standard	1	15.40	0.2117	12.139820	0.3209	8.2441517620
.5 Inch SATA 6 G...	Standard	2	30.79	0.2171	24.105491	0.3051	16.7509056959

➢ Stored proc to get top n markets by net sales for a given year



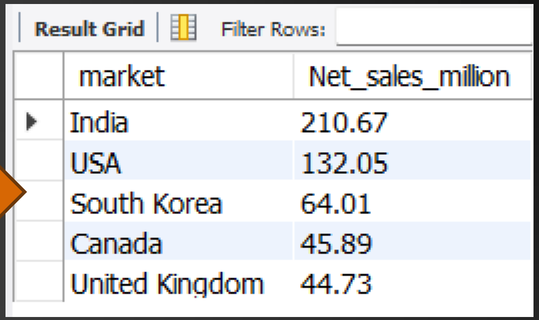
```sql
CREATE DEFINER=`root`@`localhost` PROCEDURE `get_top_n_markets_by_net_sa
in_fiscal_year int,
in_top_n int)
BEGIN
 select market ,
 round(sum(net_sales)/1000000,2) as Net_sales_million
 from gdb0041.net_sales
 where fiscal_year = in_fiscal_year
 group by market
 order by Net_sales_million desc
 limit in_top_n;
END
```



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

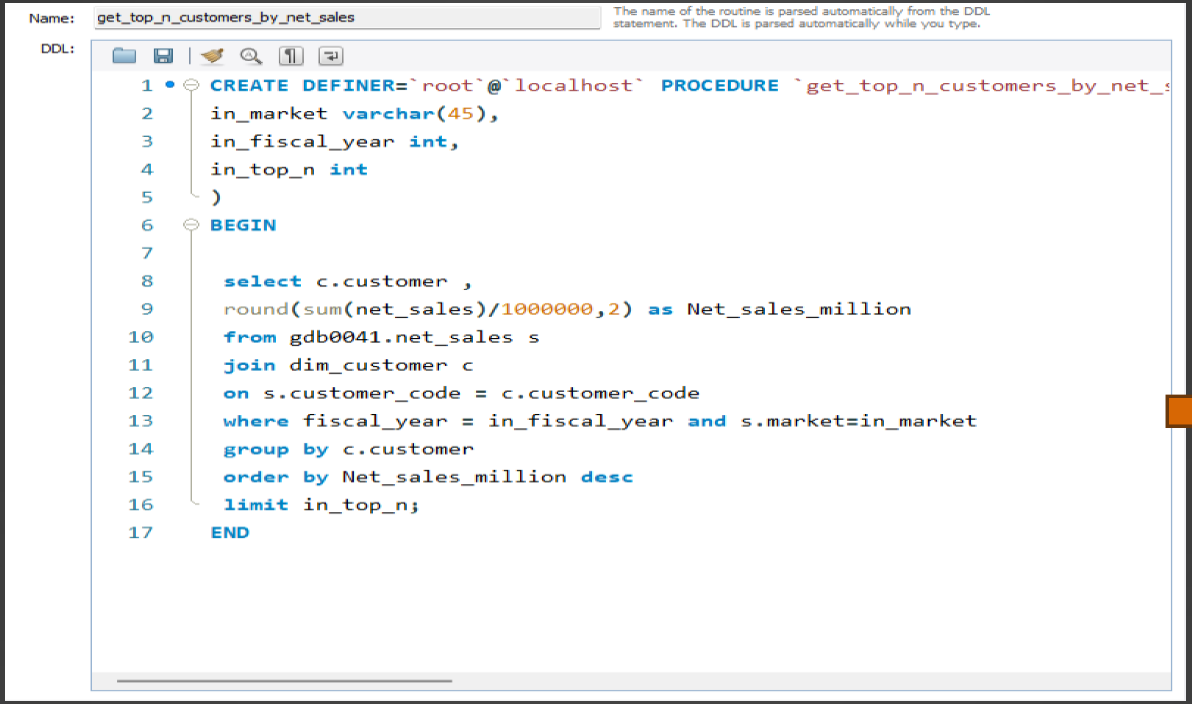
in_top_n 5 [IN] int



market	Net_sales_million
India	210.67
USA	132.05
South Korea	64.01
Canada	45.89
United Kingdom	44.73

- created stored procedure that takes market, fiscal_year and top n as an input and returns **Top n customers by net sales in that given fiscal year and market.**

```
CREATE PROCEDURE `get_top_n_customers_by_net_sales`(
 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 net_sales s
 join dim_customer c
 on s.customer_code=c.customer_code
 where
s.fiscal_year=in_fiscal_year
and s.market=in_market
 group by customer
 order by net_sales_mln desc
 limit in_top_n;
 END
```

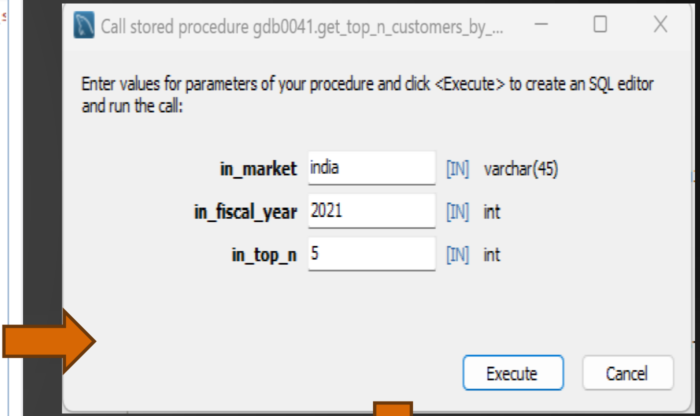


Name: get_top_n_customers_by_net_sales

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

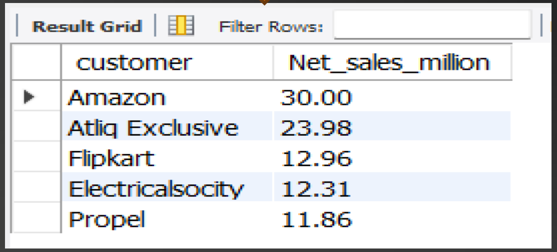
DDL:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `get_top_n_customers_by_net_s
in_market varchar(45),
in_fiscal_year int,
in_top_n int
)
BEGIN

 select c.customer ,
 round(sum(net_sales)/1000000,2) as Net_sales_million
 from gdb0041.net_sales s
 join dim_customer c
 on s.customer_code = c.customer_code
 where fiscal_year = in_fiscal_year and s.market=in_market
 group by c.customer
 order by Net_sales_million desc
 limit in_top_n;
END
```



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

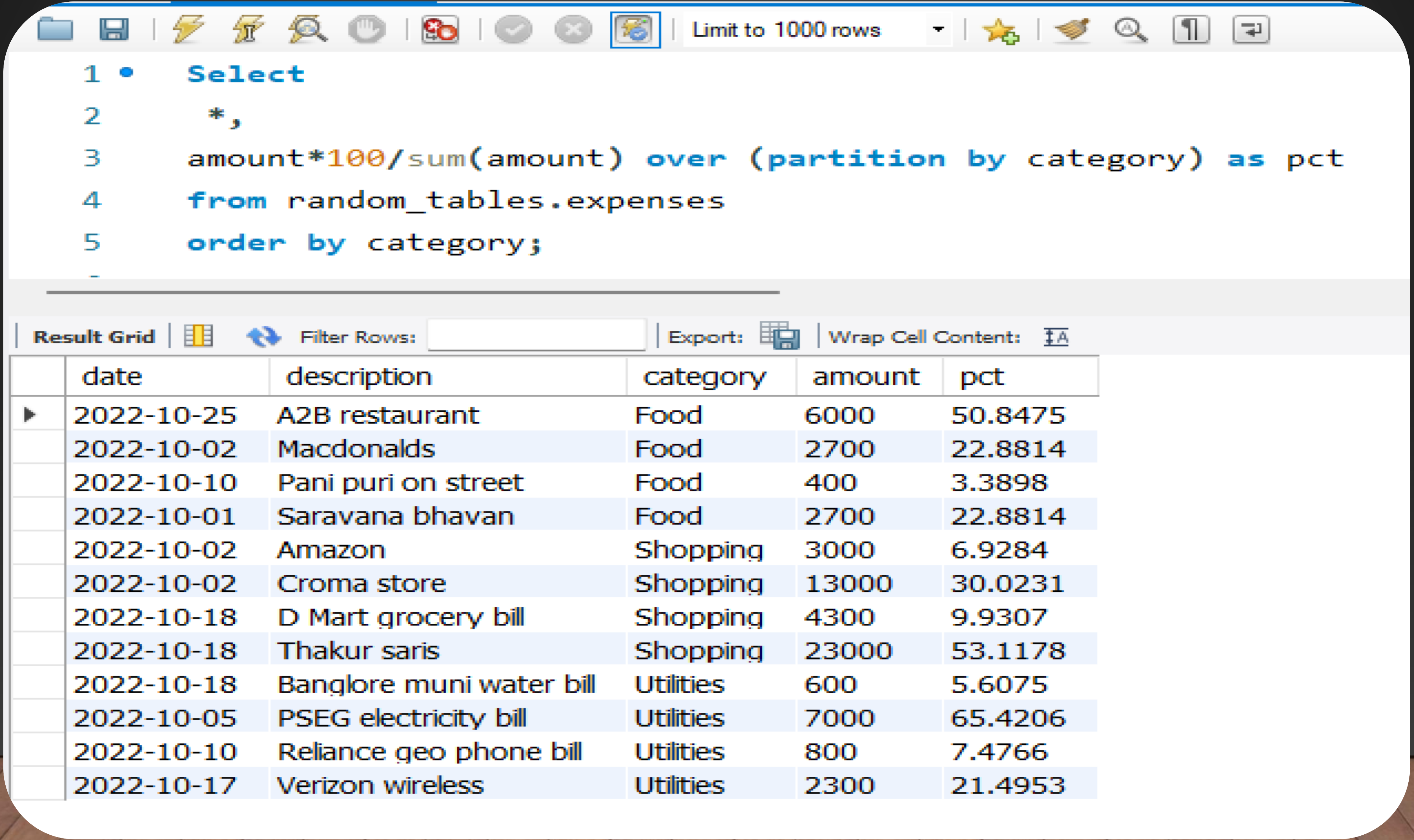
in_market	india	[IN]	varchar(45)
in_fiscal_year	2021	[IN]	int
in_top_n	5	[IN]	int



customer	Net_sales_million
Amazon	30.00
Atliq Exclusive	23.98
Flipkart	12.96
Electricalsocity	12.31
Propel	11.86

Use Case: Window Functions: OVER Clause

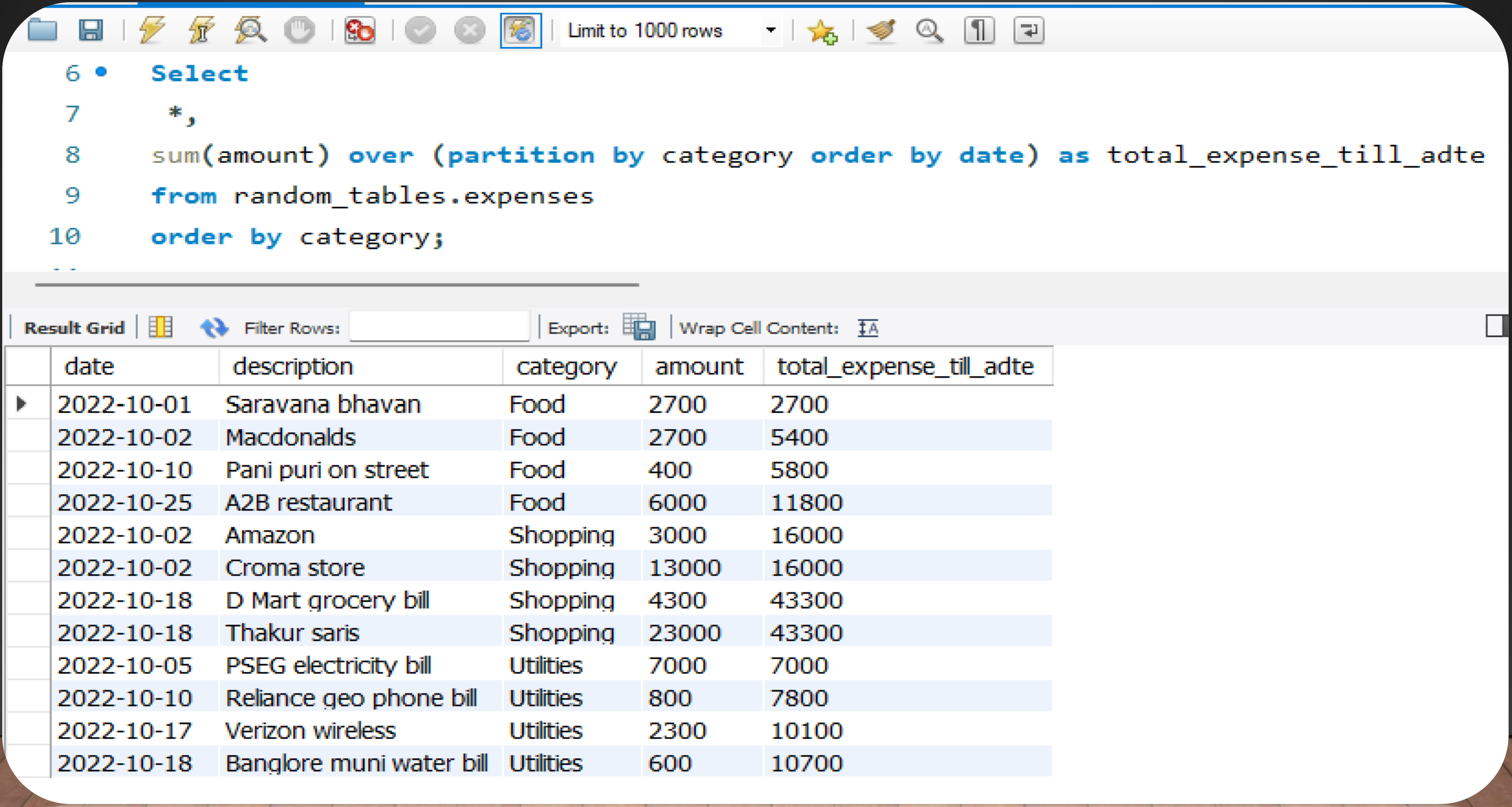
show % of total expense per category



```
Select
 *,
amount*100/sum(amount) over (partition by category) as pct
from random_tables.expenses
order by category;
```

date	description	category	amount	pct
2022-10-25	A2B restaurant	Food	6000	50.8475
2022-10-02	Macdonalds	Food	2700	22.8814
2022-10-10	Pani puri on street	Food	400	3.3898
2022-10-01	Saravana bhavan	Food	2700	22.8814
2022-10-02	Amazon	Shopping	3000	6.9284
2022-10-02	Croma store	Shopping	13000	30.0231
2022-10-18	D Mart grocery bill	Shopping	4300	9.9307
2022-10-18	Thakur saris	Shopping	23000	53.1178
2022-10-18	Banglore muni water bill	Utilities	600	5.6075
2022-10-05	PSEG electricity bill	Utilities	7000	65.4206
2022-10-10	Reliance geo phone bill	Utilities	800	7.4766
2022-10-17	Verizon wireless	Utilities	2300	21.4953

➢ Show expenses per category till date



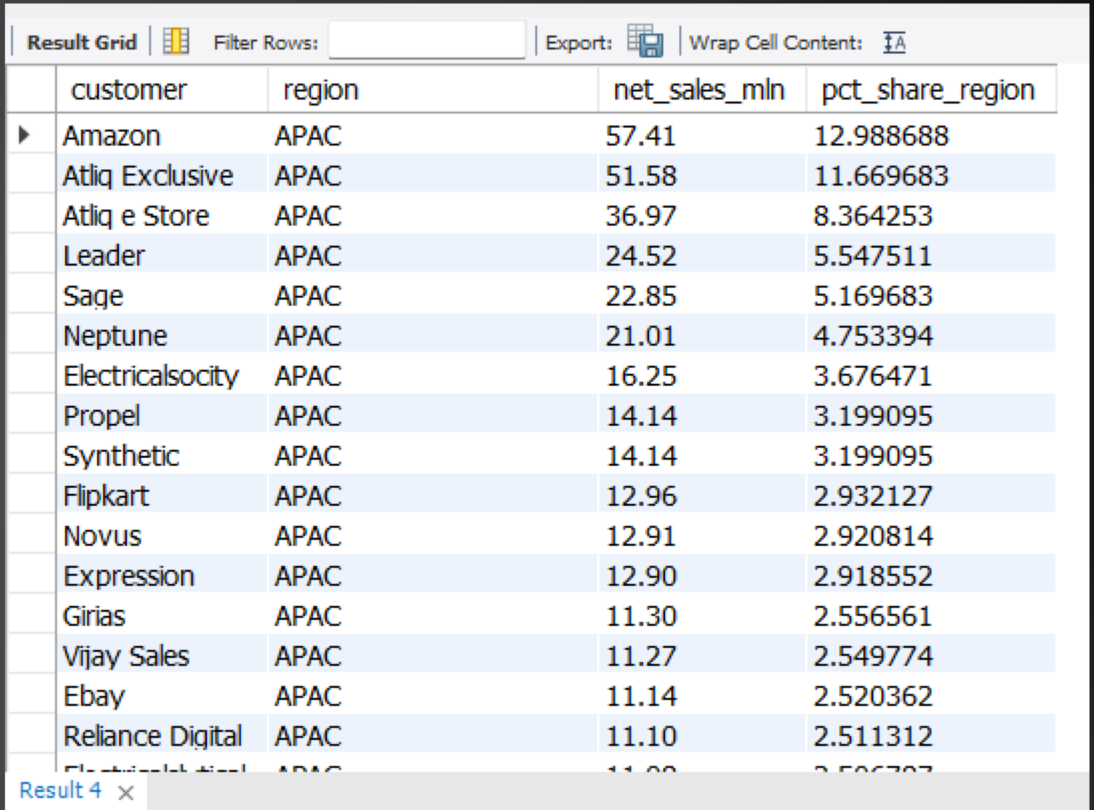
```
Select
 *,
sum(amount) over (partition by category order by date) as total_expense_till_adte
from random_tables.expenses
order by category;
```

date	description	category	amount	total_expense_till_adte
2022-10-01	Saravana bhavan	Food	2700	2700
2022-10-02	Macdonalds	Food	2700	5400
2022-10-10	Pani puri on street	Food	400	5800
2022-10-25	A2B restaurant	Food	6000	11800
2022-10-02	Amazon	Shopping	3000	16000
2022-10-02	Croma store	Shopping	13000	16000
2022-10-18	D Mart grocery bill	Shopping	4300	43300
2022-10-18	Thakur saris	Shopping	23000	43300
2022-10-05	PSEG electricity bill	Utilities	7000	7000
2022-10-10	Reliance geo phone bill	Utilities	800	7800
2022-10-17	Verizon wireless	Utilities	2300	10100
2022-10-18	Banglore muni water bill	Utilities	600	10700

❑ Use Case: Window Functions: OVER Clause

Find customer wise net sales distribution per region for FY 2021.

```
 with cte1 as (
 select
c.customer,
 c.region,
 round(sum(net_sales)/1000000,2) as net_sales_mln
 from gdb0041.net_sales n
 join dim_customer c
 on n.customer_code=c.customer_code
 where fiscal_year=2021
 group by c.customer, c.region)
 select
 *,
 net_sales_mln*100/sum(net_sales_mln) over (partition by region)
as pct_share_region
 from cte1
 order by region, pct_share_region desc
```



customer	region	net_sales_mln	pct_share_region
Amazon	APAC	57.41	12.988688
Atliq Exclusive	APAC	51.58	11.669683
Atliq e Store	APAC	36.97	8.364253
Leader	APAC	24.52	5.547511
Sage	APAC	22.85	5.169683
Neptune	APAC	21.01	4.753394
Electricalsocity	APAC	16.25	3.676471
Propel	APAC	14.14	3.199095
Synthetic	APAC	14.14	3.199095
Flipkart	APAC	12.96	2.932127
Novus	APAC	12.91	2.920814
Expression	APAC	12.90	2.918552
Girias	APAC	11.30	2.556561
Vijay Sales	APAC	11.27	2.549774
Ebay	APAC	11.14	2.520362
Reliance Digital	APAC	11.10	2.511312

❑USE CASE: WINDOW FUNCTIONS: ROW_NUMBER, RANK, DENSE_RANK

FIND OUT TOP 3 PRODUCTS FROM EACH DIVISION BY TOTAL QUANTITY SOLD IN A GIVEN YEAR

```
with cte1 as
 (select
            p.division,
             p.product,
          sum(sold_quantity) as total_qty
             from fact_sales_monthly s
             join dim_product p
                  on p.product_code=s.product_code
             where fiscal_year=2021
             group by p.product),
         cte2 as
        (select
                 *,
                 dense_rank() over (partition by division order by total_qty desc) as drnk
             from cte1)
 select * from cte2 where drnk<=3
```

➢CREATING STORED PROCEDURE FOR THE ABOVE QUERY

```
CREATE PROCEDURE `get_top_n_products_per_division_by_qty_sold`(

        in_fiscal_year INT,

     in_top_n INT

)

 BEGIN

     with cte1 as (

     select

                  p.division,  p.product,

                  sum(sold_quantity) as total_qty

               from fact_sales_monthly s

               join dim_product p

                  on p.product_code=s.product_code

               where fiscal_year=in_fiscal_year

               group by p.product),
cte2 as (
    select    *,
                   dense_rank() over (partition by division
order by total_qty desc) as drnk
                from cte1)
    select * from cte2 where drnk <= in_top_n;
END
```