Scenario 1: Want to generate report of individual product sales(aggregated on monthly basis at product code level)

#for Croma India Customer FY=2021. So that i can track Individual product sales & run further product analytics on excel

Fields: Month, Product name, variant, Sold Qty, Gross price per item, Gross price total.

we want data for year 2021 but dates here in table is calendar dates. Need to convert it into Fiscal year

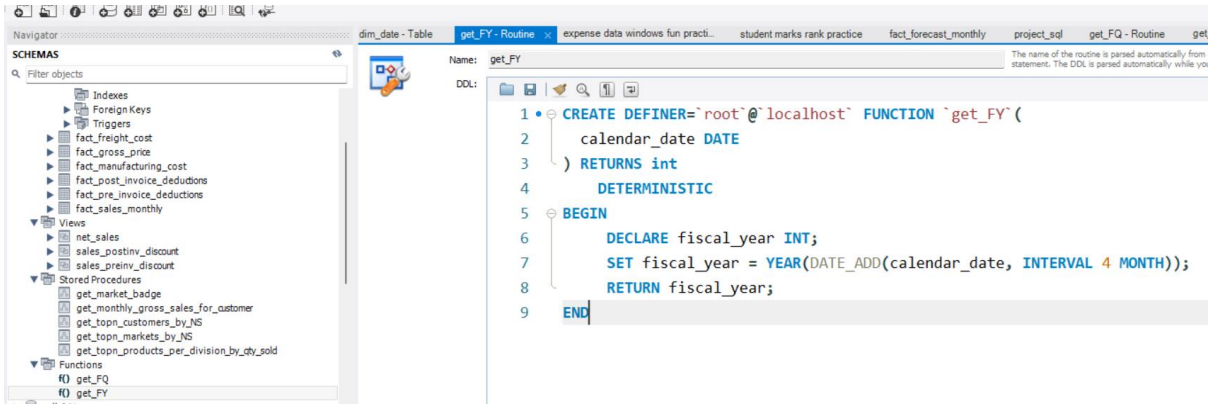
YEAR: select DATE_ADD("2020-09-01", INTERVAL 4 MONTH)

Quarter: SELECT MONTH("2019-08-01")

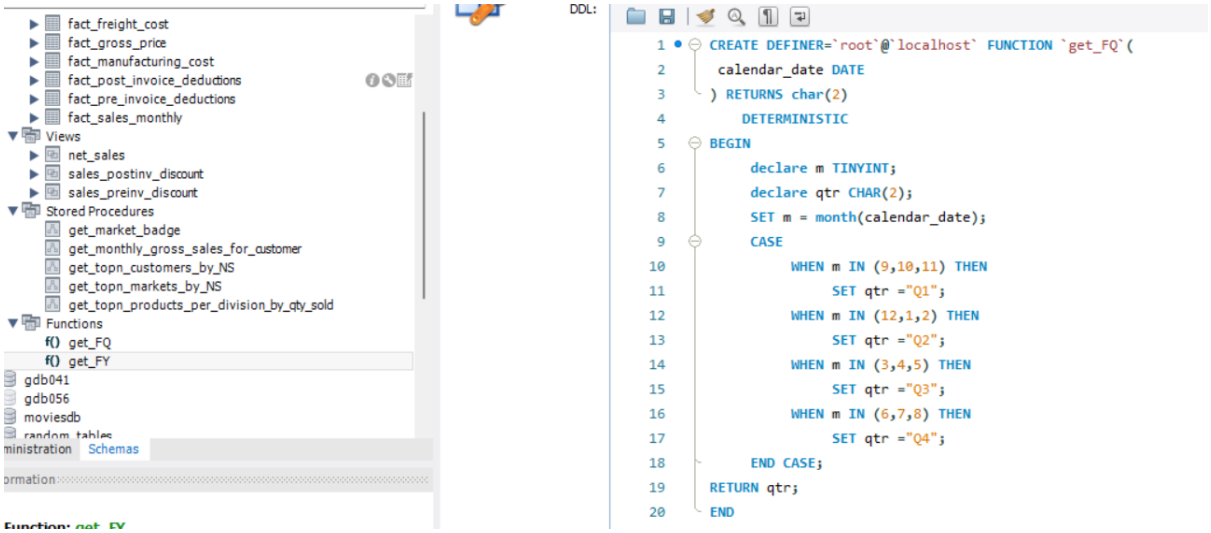
Both the above formula we are going to use everywhere- hence created user defined **functions**.

Screenshot attached

YEAR



Quarter



Using **JOINS** generated the output:

```
select s.date, s.product_code,
      p.product, p.variant, s.sold_quantity,
      g.gross_price,
      ROUND(g.gross_price * s.sold_quantity, 2) as total_grossprice
from fact_sales_monthly s
JOIN dim_product p
ON p.product_code = s.product_code
join fact_gross_price g
on g.product_code = s.product_code and
  g.fiscal_year = get_FY(s.date)
where
    customer_code =90002002 and
    get_fY(date)=2021
ORDER BY date
limit 1000000
```

Output:

date	product_code	product	variant	sold_quantity	gross_price	total_grossprice
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

Senario 2: Gross monthly total sales report for CROMA

As PO, need an aggregate monthly gross sales report for CROMA India customer so that I can track how much

sales this particular cust is generating for AtliQ & manage our relnship acc.

Fields - Month, Total gross sales amt to CROMA India in this month

```
Select
    s.date,
    #g.gross_price * s.sold_quantity as gross_price_total
    SUM(ROUND(g.gross_price * s.sold_quantity, 2)) as monthly_sales
 from fact_sales_monthly s
JOIN fact_gross_price g
on g.product_code = s.product_code and
  g.fiscal_year = get_FY(s.date)
Where customer_code =90002002
GROUP BY s.date # when u do GROUP BY , u need to apply aggreagtion func
ORDER BY s.date
```

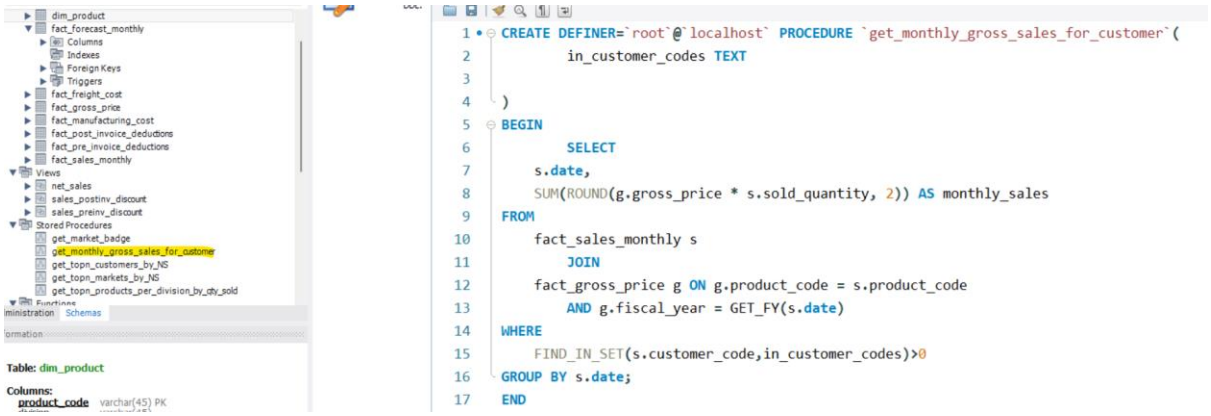
date	monthly_sales
2017-09-01	122407.57
2017-10-01	162687.56
2017-12-01	245673.84
2018-01-01	127574.73
2018-02-01	144799.54
2018-04-01	130643.92
2018-05-01	139165.06
2018-06-01	125735.36
2018-08-01	125409.90
2018-09-01	343337.14

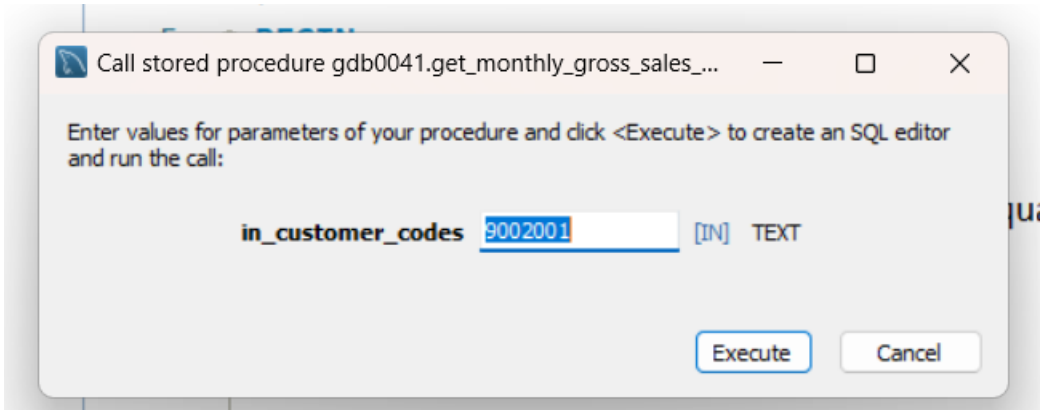
I created a Gross Monthly Total Sales report initially for CROMA. However, since we handle multiple products, the Product Owner frequently requested the same report for different products.

Instead of writing a new query every time, I developed a parameterized **Stored Procedure** in SQL. This stored procedure accepts the product name (or product ID) as an input parameter and dynamically generates the monthly gross sales report for that specific product.

I also provided limited execution access to the Product Owner, so they can generate the report themselves whenever required, without depending on the data team.

This helped reduce repetitive work, improved efficiency, ensured consistent calculations across products, and enabled self-service reporting.





scenario : our customer_code are such that they can be duplicated. eg:amazon(dim_customer)

in sql query,simply we can write like this and get the output

customer_code in(90002008,90002016)

but how will do same in Stored procedure:

SELECT FIND_IN_SET(90002008, "90002008,90002016")(screenshot above)

Scenario 3: Create Stored proc for market badge based on below logic:

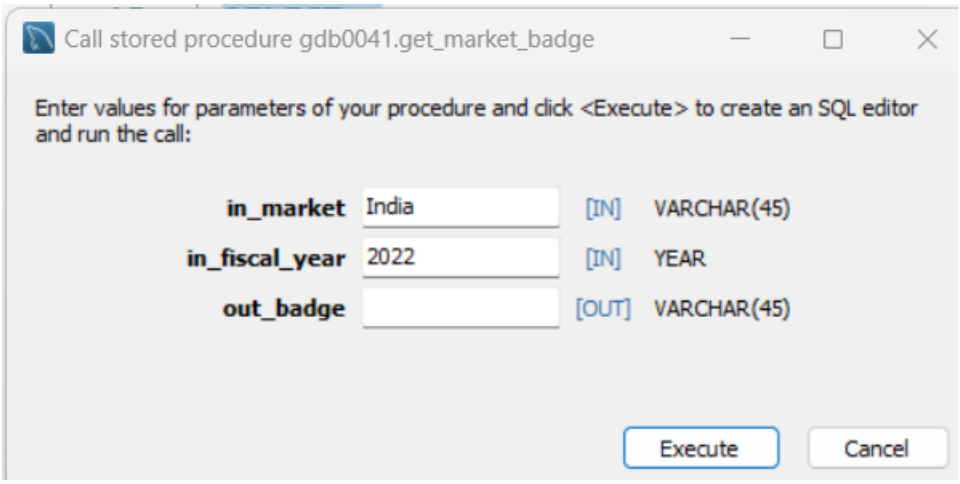
If total sold qty > 5 million that market is considered Global else it is Silver.

i/p: market, fiscal year

o/p: market badge

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `get_market_badge`(
    IN in_market VARCHAR(45),
    IN in_fiscal_year YEAR,
```

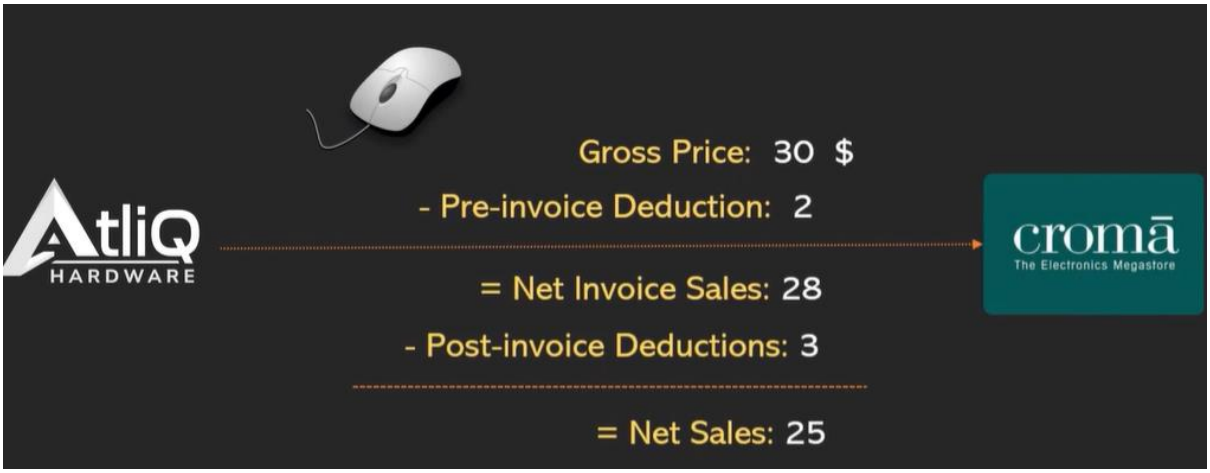
```
    OUT out_badge VARCHAR(45)
)
BEGIN
        DECLARE qty INT default 0;

    # set default market India
    if in_market="" then
      set in_market="india";
        end if;

# retrieve total qty for market + 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_FY(date)=in_fiscal_year and c.market=in_market
GROUP BY c.market;

 # determine market badge
   if qty > 5000000 then
     set out_badge = "Gold";
        else
            set out_badge = "Silver";
        end if;
END
```



Scenario 3: Top markets ,products, customers for given FY

Field:Rank, Market Net sales(in mlns)

Profit & Loss concept:



Step 1: Initially wrote query to calculate gross and pre-invoice discount

```
SELECT
    s.date,
    s.product_code,
    s.sold_quantity,
    p.product,
    p.variant,
    g.gross_price AS grossprice_peritem,
    ROUND(g.gross_price * s.sold_quantity, 2) AS total_grossprice,
    pre.pre_invoice_discount_pct,
```

#(total_grossprice - total_grossprice*pre.pre_invoice_discount_pct) as net_invoice_sales

 # cannot use derived field in same query. Either subquery or CTE

```
FROM fact_sales_monthly s
JOIN dim_product p
    ON p.product_code = s.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 = s.customer_code
    AND pre.fiscal_year = s.fiscal_year
WHERE s.fiscal_year = 2021;
```

Step 2: Used CTE to calculate net_invoice_sales cleanly

```
WITH cte1 AS (
```

```
    SELECT
        s.date,
        s.product_code,
        s.sold_quantity,
        p.product,
        p.variant,
        g.gross_price AS grossprice_peritem,
        ROUND(g.gross_price * s.sold_quantity, 2) AS total_grossprice,
        pre.pre_invoice_discount_pct
    FROM fact_sales_monthly s
    JOIN dim_product p
        ON p.product_code = s.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 = s.customer_code
        AND pre.fiscal_year = s.fiscal_year
    WHERE s.fiscal_year = 2021
)

SELECT *,
      (total_grossprice
       - total_grossprice * pre_invoice_discount_pct)
       AS net_invoice_sales
FROM cte1;
```

Step 3: to get net sales I have to Add now post-invoice discount so will do 1 more join with fact_post_invoice_deductions and use one more CTE on top of it

This will make Query became bigger and logic reusable in multiple reports

-- Instead of creating physical table (to avoid duplication & storage),

I created a **VIEW** for sales_preinv_discount

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

-- Now I simply use the view and join post-invoice table

```
select *,
 (1 - pre_invoice_discount_pct)*total_grossprice as net_invoice_sales
 from sales_preinv_discount
```

Likewise I created view for sales_postinv_discount ,net_sales as they will be used for further calculations:

POST_INVOICE:

```
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.total_grossprice AS total_grossprice,
        s.pre_invoice_discount_pct AS pre_invoice_discount_pct,
        ((1 - s.pre_invoice_discount_pct) * s.total_grossprice) AS net_invoice_sales,
        (po.discounts_pct + po.other_deductions_pct) AS post_inv_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);
```

NET_SALES:

```
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = root@localhost
    SQL SECURITY DEFINER
VIEW net_sales AS
    SELECT
        sales_postinv_discount.date AS date,
        sales_postinv_discount.fiscal_year AS fiscal_year,
        sales_postinv_discount.customer_code AS customer_code,
        sales_postinv_discount.market AS market,
        sales_postinv_discount.product_code AS product_code,
        sales_postinv_discount.product AS product,
        sales_postinv_discount.variant AS variant,
        sales_postinv_discount.sold_quantity AS sold_quantity,
        sales_postinv_discount.total_grossprice AS total_grossprice,
        sales_postinv_discount.pre_invoice_discount_pct AS pre_invoice_discount_pct,
        sales_postinv_discount.net_invoice_sales AS net_invoice_sales,
        sales_postinv_discount.post_inv_discount_pct AS post_inv_discount_pct,
        ((1 - sales_postinv_discount.post_inv_discount_pct)
            * sales_postinv_disc*



SELECT *,
    (1-post_inv_discount_pct)*net_invoice_sales as net_sales
    FROM sales_postinv_discount;
```

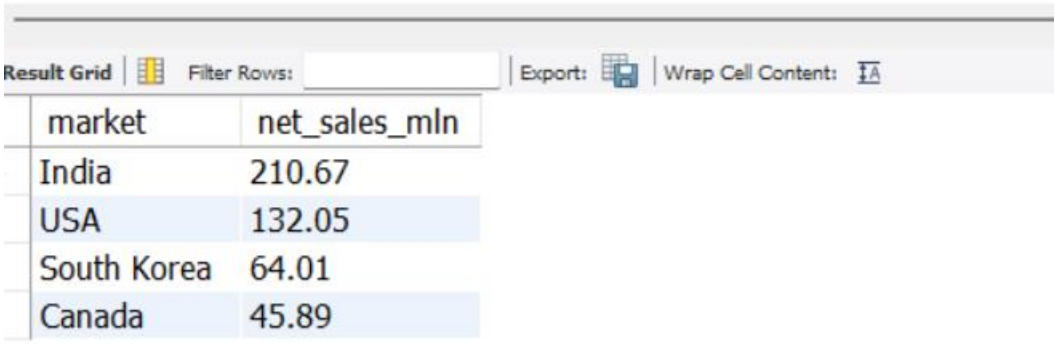
variant	sold_quantity	total_grossprice	pre_invoice_discount_pct	net_invoice_sales	post_inv_discount_pct	net_sales
Standard	4	61.58	0.2803	44.319126	0.3905	27.0125072970
Standard	16	246.32	0.2803	177.276504	0.4139	103.9017589944
Standard	4	61.58	0.2803	44.319126	0.3295	29.7159739830
Standard	6	92.37	0.2803	66.478689	0.3244	44.9130022884
Standard	9	138.56	0.2803	99.721632	0.3766	62.1664653888
Standard	6	92.37	0.2803	66.478689	0.3615	42.4466429265
Standard	7	107.77	0.2803	77.562069	0.3173	52.9516245063
Standard	10	153.95	0.2803	110.797815	0.3501	72.0074999685

Also, for top n market and products, created stored procedure so PO can retrieve the result according to their requirement.

Market:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `get_topn_markets_by_NS`(
            in_fiscal_year INT,
    in_top_n INT
)
BEGIN
 select market ,
    ROUND(sum(net_sales)/1000000,2) as net_sales_mln
    from net_sales
    where fiscal_year=in_fiscal_year
    group by market
    order by net_sales_mln desc
    limit in_top_n;
END
```

```
1 •  call gdb0041.get_topn_markets_by_NS(2021, 4);
2
```



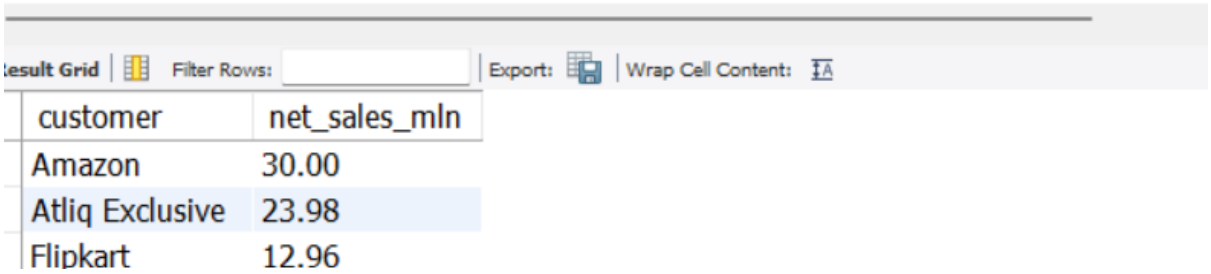
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

market	net_sales_mln
India	210.67
USA	132.05
South Korea	64.01
Canada	45.89

Products:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `get_topn_customers_by_NS`(
        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 n
   join dim_customer c
   on n.customer_code=c.customer_code and n.market=in_market
   where fiscal_year=in_fiscal_year
   group by c.customer
   order by net_sales_mln desc
   limit in_top_n;
END
```

```
1 •  call gdb0041.get_topn_customers_by_NS('India', 2021, 3);
2
```



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

customer	net_sales_mln
Amazon	30.00
Atliq Exclusive	23.98
Flipkart	12.96

Scenario 5: Used windows function **OVER()** to find customer wise net sales % contribution

```
with cte1 as(
        select c.customer ,
   ROUND(sum(net_sales)/1000000,2) as net_sales_mln
   from net_sales n
   join dim_customer c
   on n.customer_code=c.customer_code
   where fiscal_year=2021
   group by c.customer)

 select
  * ,
   net_sales_mln*100/sum(net_sales_mln) OVER() as pct
   from cte1
   order by net_sales_mln desc
```

customer	net_sales_mln	pct
Amazon	109.03	13.233402
Atliq Exclusive	79.92	9.700206
Atliq e Store	70.31	8.533803
Sage	27.07	3.285593
Flipkart	25.25	3.064692
Leader	24.52	2.976089
Neptune	21.01	2.550067
Ebay	19.88	2.412914
Electricalsocity	16.25	1.972327

Scenario 6: Find customer wise Net sales distribution per region for 2021

```
with cte1 as
 (select c.customer , c.region,
    ROUND(sum(net_sales)/1000000,2) as net_sales_mln
    from net_sales n
    join dim_customer c
    on n.customer_code=c.customer_code
    where n.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,net_sales_mln 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

customer	region	net_sales_mln	pct_share_region
Notebillig	EU	1.47	0.731999
Digimarket	EU	1.44	0.717060
Premium Sto...	EU	1.39	0.692162
Nova	EU	0.46	0.229061
Amazon	LATAM	1.54	48.734177
Atliq e Store	LATAM	1.09	34.493671
Electricalsbea...	LATAM	0.53	16.772152
Amazon	NA	30.31	17.033832
Atliq Exclusive	NA	14.95	8.401708

Scenario 6: To Find top n products in each division by their quantity sold

```
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)
```

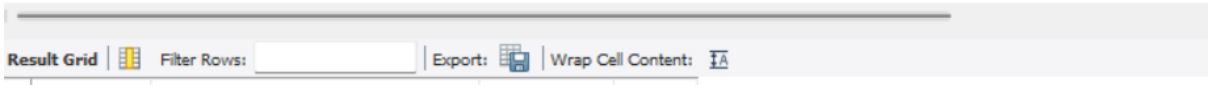
select * from cte2 where drnk<=3

division	product	total_qty	drnk
N & S	AQ Pen Drive DRC	2034569	1
N & S	AQ Digit SSD	1240149	2
N & S	AQ Clx1	1238683	3
P & A	AQ Gamers Ms	2477098	1
P & A	AQ Maxima Ms	2461991	2
P & A	AQ Master wireless x1 Ms	2448784	3
PC	AQ Digit	135092	1
PC	AQ Gen Y	135031	2
PC	AQ Elite	134431	3

Created Stored procedure for same too

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `get_topn_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
```

```
1 •  call gdb0041.get_topn_products_per_division_by_qty_sold(2020, 4);
2
```



division	product	total_qty	drnk
N & S	AQ Clx1	935128	1
N & S	AQ Neuer SSD	924264	2
N & S	AQ Digit SSD	920105	3
N & S	AQ Wi Power Dx2	846576	4
P & A	AQ Master wired x1 Ms	1578253	1
P & A	AQ Gamers Ms	1566445	2
P & A	AQ Lite Ms	1564099	3
P & A	AQ Master wireless x1 Ms	1563844	4
PC	AQ Digit	68862	1
PC	AQ Elite	67841	2
PC	AQ Aspiron	59516	3
PC	AQ BZ Compact	52380	4