ATLIQ HARDWARE
FINANCE ANALYTICS

ABHIJEET KUMAR
abhijeet.kr7252@gmail.com

Introduction

In today's rapidly changing business environment, the ability to effectively monitor sales and extract accurate insights is crucial for driving sustainable growth and strategic decision-making. AtliQ Hardware, a prominent supplier of computer hardware and peripherals in India, faces such challenges as it manages its operations. With a dynamic marketing landscape and a widespread network of regional offices, the company's sales director is tasked with deciphering intricate data to gain valuable insights into sales performance and devise effective strategies.

Headquartered in Delhi, AtliQ Hardware operates through various regional offices across the country. However, despite its extensive reach, the sales director encounters obstacles in accessing timely and comprehensive updates on sales metrics from regional sales managers. Traditional reporting methods, such as manual data entry and the exchange of numerous Excel files, have proven insufficient in providing the clarity and actionable insights required.

In addressing these challenges, this project aims to emulate a proactive approach to resolving business issues for AtliQ Hardware. By employing data analytics and visualization techniques, the objective is to equip the sales director with a robust platform for seamlessly tracking sales performance across all operations. Through this endeavor, the project seeks to bridge the gap between raw sales data and actionable intelligence, empowering the sales director to make well-informed decisions in a timely manner.

The scenario unfolds with the sales director's pursuit of a more efficient method for monitoring sales performance. Confronted with fragmented and incomplete data from regional managers, the director recognizes the urgency of implementing a centralized and user-friendly system that offers immediate insights into the company's sales landscape. Instead of grappling with raw data or cumbersome spreadsheets, the director envisions a solution that presents sales data in a cohesive and visually appealing format, facilitating swift comprehension and informed decision-making.

In summary, this project acts as a catalyst for transitioning AtliQ Hardware's sales tracking mechanism from reactive to proactive. By harnessing the capabilities of data analytics and visualization, it seeks to empower the sales director with the necessary tools and insights to navigate the complexities of the business environment effectively. Through collaborative efforts and innovative solutions, the project endeavors to establish a more streamlined and informed approach to sales management within AtliQ Hardware.

Analysis 1:

Generate report of individual product sales (aggregated on a monthly basis at the product code level) for Croma India customer for FY=2021. Required fields:

- Month
- Product Name
- Varient
- Sold Quantity
- Gross Price Per Item
- Gross Price Total

```
# Step 1: Get Croma India Customer Code
SELECT * FROM dim_customer
WHERE customer LIKE "%croma%" AND market = "%India%";

-- croma customer code = 90002002

# Step 2: Get sales for Fisical Year 2021
-- As FY 2021 for AtliQ starts in Sep 2020 (2020-09-01) to Aug 2021 (2021-09-01)

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

-- define a function to automatically get fiscal year
SELECT * FROM fact_sales_monthly
WHERE
    get_fiscal_year(date) = 2021 AND
    customer_code = 90002002
ORDER BY date;

 # Step 3: Get Product Name and Varient from Product Table By Using JOIN
 SELECT
     s.date,
     s.product_code,
     p.product,
     p.variant,
     s.sold_quantity
 FROM fact_sales_monthly AS s
 JOIN dim_product AS p
     ON s.product_code = p.product_code
 WHERE
     get_fiscal_year(date) = 2021 AND
     customer_code = 90002002
 ORDER BY date;
```

```sql
# Step 4: Get Gross price
SELECT
    s.date,
    s.product_code,
    g.fiscal_year,
    p.product,
    p.variant,
    s.sold_quantity,
    g.gross_price
FROM fact_sales_monthly AS s
JOIN dim_product AS p
    ON s.product_code = p.product_code
JOIN fact_gross_price AS g
    ON g.product_code = s.product_code AND
    g.fiscal_year = get_fiscal_year(s.date)
WHERE
    get_fiscal_year(date) = 2021 AND
    customer_code = 90002002
ORDER BY date;
```

```sql
# Step 5: Cal Total Gross Price (Qty * gross_price)
SELECT
    MONTHNAME(s.date) AS month,
    p.product,
    p.variant,
    s.sold_quantity,
    g.gross_price AS gross_price_per_item,
    ROUND(s.sold_quantity * g.gross_price, 2) AS total_gross_price
FROM fact_sales_monthly AS s
JOIN dim_product AS p
    ON s.product_code = p.product_code
JOIN fact_gross_price AS g
    ON g.product_code = s.product_code AND
    g.fiscal_year = get_fiscal_year(s.date)
WHERE
    get_fiscal_year(date) = 2021 AND
    customer_code = 90002002
ORDER BY date;
```

Result:

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

month	product	variant	sold_quantity	gross_price_per_item	total_gross_price
September	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 ...	Standard	202	19.0573	3849.57
September	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 ...	Plus	162	21.4565	3475.95
September	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 ...	Premium	193	21.7795	4203.44
September	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 ...	Premium Plus	146	22.9729	3354.04
September	AQ WereWolf NAS Internal Hard Drive HDD – ...	Standard	149	23.6987	3531.11
September	AQ WereWolf NAS Internal Hard Drive HDD – ...	Plus	107	24.7312	2646.24
September	AQ WereWolf NAS Internal Hard Drive HDD – ...	Premium	123	23.6154	2904.69
September	AQ Zion Saga	Standard	146	23.7223	3463.46
September	AQ Zion Saga	Plus	236	27.1027	6396.24
September	AQ Zion Saga	Premium	137	28.0059	3836.81
September	AQ Mforce Gen X	Standard 3	23	19.5235	449.04
September	AQ Mforce Gen X	Plus 1	82	19.9239	1633.76
September	AQ Mforce Gen X	Plus 2	86	20.0766	1726.59
September	AQ Mforce Gen X	Plus 3	48	19.9365	956.95
September	AQ Mforce Gen Y	Standard 1	138	22.3984	3090.98
September	AQ Mforce Gen Y	Standard 2	72	24.9298	1794.95
September	AQ Mforce Gen Y	Standard 3	38	26.5871	1010.31
September	AQ Mforce Gen Y	Plus 1	149	26.1081	3890.11
September	AQ Mforce Gen Y	Plus 2	29	29.7008	861.32
September	AQ Mforce Gen Y	Plus 3	28	31.2439	874.83
September	AQ Mforce Gen Y	Premium 1	171	32.4427	5547.70
September	AQ Mforce Gen Y	Premium 2	118	30.5816	3608.63

Result 25

Output

Action Output

#	Time	Action	Message
29	21:34:47	SELECT MONTHNAME(s.date) AS month, p.product, p.variant, s.sold_quantity, g.gross_p...	3006 row(s) returned

Analysis 2:

Create Stored Procedure to generate monthly sales report of given customer code

Name: monthly_total_gross_sales

DDL:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `monthly_total_gross_sales`(cus_code INT)
BEGIN
    SELECT
        s.date,
        SUM(s.sold_quantity * gp.gross_price) AS total_gross_price
    FROM fact_sales_monthly AS s
    JOIN fact_gross_price AS gp
        ON s.product_code = gp.product_code AND
        get_fiscal_year(s.date) = gp.fiscal_year
    WHERE customer_code = cus_code
    GROUP BY s.date
    ORDER BY date;
END
```

Call Stored Procedure for Flipkart:

Call stored procedure gdb0041.monthly_total_gross_sales

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

cus_code 90002009 [IN] INT

Execute Cancel

Result:

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

date	total_gross_price
2017-09-01	198752.6804
2017-10-01	220761.8754
2017-11-01	327835.3319
2018-01-01	178745.6941
2018-02-01	169139.7248
2018-03-01	199598.6641
2018-05-01	175089.3984

Analysis 3:

Create Stored Procedure to generate monthly sales report of given customer, if given customer has multiple customer code.

Input: list of customer code

Name: monthly_gross_sales

DDL:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `monthly_gross_sales`(cus_code_list TEXT)
BEGIN
    SELECT
        s.date,
        SUM(s.sold_quantity * gp.gross_price) AS total_gross_price
    FROM fact_sales_monthly AS s
    JOIN fact_gross_price AS gp
        ON gp.fiscal_year = get_fiscal_year(s.date) AND
        gp.product_code = s.product_code
    WHERE
        FIND_IN_SET(s.customer_code, cus_code_list) > 0
    GROUP BY s.date
    ORDER BY date;
END
```

Call Stored Procedure:

Call stored procedure gdb0041.monthly_gross_sales

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

cus_code_list 0002008,90002016 [IN] TEXT

Execute Cancel

Result:

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

date	total_gross_price
2017-10-01	438917.7663
2017-11-01	575530.5767
2017-12-01	617049.8852
2018-02-01	358033.1389
2018-03-01	384656.6960

Result 2

Output

Action Output

#	Time	Action	Message
34	22:39:06	call gdb0041.monthly_gross_sales('90002008,90002016')	39 row(s) returned

Analysis 4:

Generate a yearly report for "Atliq e Store" for all market

1. Fiscal Year
2. Total Gross Sales amount in that year

```
# use cte to get customer codes
WITH cte1 AS(
    SELECT customer_code FROM dim_customer
    WHERE customer LIKE "%Atliq e Store%")
-- get yearly sales by using cte table
SELECT
    gp.fiscal_year,
    ROUND(SUM(s.sold_quantity * gp.gross_price), 2) AS total_yearly_gross_price
FROM fact_sales_monthly AS s
JOIN fact_gross_price AS gp
    ON s.product_code=gp.product_code AND
    get_fiscal_year(s.date) = gp.fiscal_year
WHERE
    s.customer_code IN (SELECT * FROM cte1)
GROUP BY gp.fiscal_year;
```

Result:

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

customer_code
70002018
70003182
70004070
70005163
70006158
70007199

dim_customer 32

Output

Action Output

#	Time	Action	Message
40	22:51:10	WITH cte1 AS(SELECT customer_code FROM dim_customer WHERE customer LIKE "%Atliq e St...	5 row(s) returned

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

fiscal_year	total_yearly_gross_price
2018	4516428.06
2019	17238524.65
2020	63207694.55
2021	139849432.55
2022	270228044.47

Result 33

Output

Action Output

#	Time	Action	Message
41	22:52:14	SELECT customer_code FROM dim_customer WHERE customer LIKE "%Atliq e Store%"	24 row(s) returned

Analysis 5:

Stored Procedure for Market Badge
Create a stored proc that can determine the market badge based on the following logic
If total sold quantity > 5 million that market is considered Gold else Silver
Input: market, fiscal year
Output: market badge

```
SELECT
    market,
    CASE
        WHEN SUM(sold_quantity) > 5000000 THEN "Gold"
        ELSE "Silver"
    END AS market_badge
FROM fact_sales_monthly AS s
JOIN dim_customer AS c
    ON s.customer_code = c.customer_code
WHERE c.market = "India"
    AND get_fiscal_year(s.date) =2020
GROUP BY market ;
```

Result:

Result Grid | Filter Rows:

market	market_badge
India	Gold

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

@out_market_badge	@out_total_sold_qty
Gold	5381959

Analysis 6:

Create a view for gross sales.

It should have the following columns:
date, fiscal_year, customer_code, customer, market, product_code, product, variant, sold_quanity, gross_price_per_item, gross_price_total

```sql
CREATE VIEW gross_sales AS
        SELECT
            s.date,
            s.fiscal_year,
            s.customer_code,
            c.customer,
            c.market,
            s.product_code,
            p.product,
            p.variant,
            s.sold_quantity,
            gp.gross_price AS gross_price_per_item,
            (s.sold_quantity * gp.gross_price) AS gross_price_total
        FROM fact_sales_monthly AS s
        JOIN fact_gross_price AS gp
            ON s.product_code = gp.product_code
            AND s.fiscal_year = gp.fiscal_year
        JOIN dim_customer AS c
            ON s.customer_code =c.customer_code
        JOIN dim_product AS p
            ON s.product_code = p.product_code;
```

Analysis 7:

Performance Improvement of SQL Query

Performance Improvement: 1

Create a lookup table which contains date and fiscal year then join with date and get fiscal year

```sql
SELECT
    s.date,
    s.product_code,
    dt.fiscal_year,
    g.fiscal_year,
    p.product,
    p.variant,
    s.sold_quantity,
    g.gross_price,
    ROUND(s.sold_quantity * g.gross_price, 2) AS total_gross_price,
    pre.pre_invoice_discount_pct
FROM fact_sales_monthly AS s
JOIN dim_date_fiscal_year AS dt
    ON s.date = dt.date
JOIN dim_product AS p
    ON s.product_code = p.product_code
JOIN fact_gross_price AS g
    ON g.product_code = s.product_code AND
    g.fiscal_year = dt.fiscal_year
JOIN fact_pre_invoice_deductions AS pre
    ON s.customer_code = pre.customer_code
    AND dt.fiscal_year = pre.fiscal_year
WHERE
    dt.fiscal_year = 2021
ORDER BY s.date;
```

Performance Improvement: 2

Again, Performance can be reduced by adding extra column in fact_sales_monthly for fiscal year.
Add generated column in fact_sales_monthly table to get fiscal year from date.

Table Name: fact_sales_monthly
Schema: gdb0041
Charset/Collation: latin1 latin1_bin
Engine: InnoDB
Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
date	DATE	☑	☑	☐	☐	☐	☐	☐	☐	
fiscal_year	YEAR	☐	☐	☐	☐	☐	☐	☐	☑	year((`date` + interval 4 month))
product_code	VARCHAR(45)	☑	☑	☐	☐	☐	☐	☐	☐	
customer_code	INT	☑	☑	☐	☐	☑	☐	☐	☐	
sold_quantity	INT	☐	☑	☐	☐	☑	☐	☐	☐	
		☐	☐	☐	☐	☐	☐	☐	☐	

```sql
SELECT
    s.date,
    s.product_code,
    s.fiscal_year,
    p.product,
    p.variant,
    s.sold_quantity,
    g.gross_price,
    ROUND(s.sold_quantity * g.gross_price, 2) AS total_gross_price,
    pre.pre_invoice_discount_pct
FROM fact_sales_monthly AS s
JOIN dim_product AS p
    ON s.product_code = p.product_code
JOIN fact_gross_price AS g
    ON g.product_code = s.product_code AND
    g.fiscal_year = s.fiscal_year
JOIN fact_pre_invoice_deductions AS pre
    ON s.customer_code = pre.customer_code
    AND s.fiscal_year = pre.fiscal_year
WHERE
    s.fiscal_year = 2021
ORDER BY s.date;
```

Analysis 8:

Calculate Net Sales Amount

- Net Sales Amount = (Total_Gross_Sales - Pre_Invoice_Deduction - Post_Invoice_Deduction)

```
-- Step: 1 Join pre invoice discounts table
-- Store sales_pre_invoice_discount result as a VIEW
CREATE VIEW sales_pre_invoice_discounts AS
    SELECT
        s.date,
        s.fiscal_year,
        s.customer_code,
        s.product_code,
        c.market,
        p.product,
        p.variant,
        s.sold_quantity,
        g.gross_price AS gross_price_per_item,
        ROUND(s.sold_quantity * g.gross_price, 2) AS total_gross_price,
        pre.pre_invoice_discount_pct
    FROM fact_sales_monthly AS s
    JOIN dim_customer AS c
        ON s.customer_code = c.customer_code
    JOIN dim_product AS p
        ON s.product_code = p.product_code
    JOIN fact_gross_price AS g
        ON g.product_code = s.product_code AND
        g.fiscal_year = s.fiscal_year
    JOIN fact_pre_invoice_deductions AS pre
        ON s.customer_code = pre.customer_code
        AND s.fiscal_year = pre.fiscal_year
    ORDER BY s.date;

-- fetch from view
SELECT * FROM sales_pre_invoice_discounts;

-- Step: 2 Get net invoice sales (total gross price -  pre invoice discounts price)
#net_invoice_sales
SELECT
    *,
    (1- pre_invoice_discount_pct) * total_gross_price AS net_invoice_sale
FROM sales_pre_invoice_discounts;
```

```sql
-- Step: 3 JOIN post_invoice_discount table with it and create view for 'sales_post_invoice_discounts'
CREATE VIEW sales_post_invoice_discounts AS
        SELECT
            pre.date, pre.fiscal_year, pre.customer_code, pre.product_code, pre.product, pre.variant,
            pre.market, pre.sold_quantity, pre.gross_price_per_item,
            pre.total_gross_price,
            pre.pre_invoice_discount_pct,
            (1- pre_invoice_discount_pct) * total_gross_price AS net_invoice_sale,
            (pos.discounts_pct + pos.other_deductions_pct) AS post_invoice_discount_pct
        FROM sales_pre_invoice_discounts AS pre
        JOIN fact_post_invoice_deductions AS pos
            ON pre.date = pos.date
            AND pre.customer_code = pos.customer_code
            AND pre.product_code = pos.product_code;

-- fetch from view
SELECT * FROM sales_post_invoice_discounts;

-- Step: 4 Get Net Sale
SELECT
    *,
    (1- post_invoice_discount_pct) * net_invoice_sale AS net_sale
FROM sales_post_invoice_discounts;

#create VIEW for net_sale
CREATE VIEW net_sales AS
        SELECT
            *,
            (1- post_invoice_discount_pct) * net_invoice_sale AS net_sale
        FROM sales_post_invoice_discounts;
```

Result:

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

date	fiscal_year	customer_code	product_code	product	variant	market	sold_quantity	gross_price_per_item	total_gross_price	pre_invoice_discount_pct	net_invoice_sale	post_invoice_discount_pct	net_sale
2017-09-01	2018	90027207	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb...	Standard	Brazil	4	15.3952	61.58	0.2803	44.319126	0.3905	27.0125072970
2017-11-01	2018	90027207	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb...	Standard	Brazil	16	15.3952	246.32	0.2803	177.276504	0.4139	103.9017589944
2017-12-01	2018	90027207	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb...	Standard	Brazil	4	15.3952	61.58	0.2803	44.319126	0.3295	29.7159739830
2018-01-01	2018	90027207	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb...	Standard	Brazil	6	15.3952	92.37	0.2803	66.478689	0.3244	44.9130022884
2018-03-01	2018	90027207	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb...	Standard	Brazil	9	15.3952	138.56	0.2803	99.721632	0.3766	62.1664653888
2018-04-01	2018	90027207	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb...	Standard	Brazil	6	15.3952	92.37	0.2803	66.478689	0.3615	42.4466429265
2018-05-01	2018	90027207	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb...	Standard	Brazil	7	15.3952	107.77	0.2803	77.562069	0.3173	52.9516245063
2018-07-01	2018	90027207	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb...	Standard	Brazil	10	15.3952	153.95	0.2803	110.797815	0.3501	72.0074999685
2018-08-01	2018	90027207	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb...	Standard	Brazil	6	15.3952	92.37	0.2803	66.478689	0.3740	41.6156593140
2017-09-01	2018	90023030	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb...	Standard	Canada	4	15.3952	61.58	0.2117	48.543514	0.2863	34.6455059418
2017-10-01	2018	90023030	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb...	Standard	Canada	2	15.3952	30.79	0.2117	24.271757	0.2851	17.3518790793
2017-12-01	2018	90023030	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb...	Standard	Canada	3	15.3952	46.19	0.2117	36.411577	0.2882	25.9177605086
2018-01-01	2018	90023030	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb...	Standard	Canada	5	15.3952	76.98	0.2117	60.683334	0.3334	40.4515104444
2018-02-01	2018	90023030	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb...	Standard	Canada	1	15.3952	15.40	0.2117	12.139820	0.3296	8.1385353280
2018-04-01	2018	90023030	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb...	Standard	Canada	1	15.3952	15.40	0.2117	12.139820	0.2901	8.6180582180
2018-05-01	2018	90023030	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb...	Standard	Canada	5	15.3952	76.98	0.2117	60.683334	0.3233	41.0644121178
2018-06-01	2018	90023030	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb...	Standard	Canada	1	15.3952	15.40	0.2117	12.139820	0.3095	8.3825457100
2018-08-01	2018	90023030	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb...	Standard	Canada	1	15.3952	15.40	0.2117	12.139820	0.3209	8.2441517620
2017-09-01	2018	90023029	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb...	Standard	Canada	2	15.3952	30.79	0.2171	24.105491	0.3051	16.7509056959
2017-10-01	2018	90023029	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb...	Standard	Canada	4	15.3952	61.58	0.2171	48.210982	0.3053	33.4921691954
2017-11-01	2018	90023029	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb...	Standard	Canada	3	15.3952	46.19	0.2171	36.162151	0.3608	23.1148469192
2018-01-01	2018	90023029	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb...	Standard	Canada	1	15.3952	15.40	0.2171	12.056660	0.2672	8.8351204480
2018-02-01	2018	90023029	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb...	Standard	Canada	4	15.3952	61.58	0.2171	48.210982	0.2803	34.6974437454
2018-03-01	2018	90023029	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb...	Standard	Canada	2	15.3952	30.79	0.2171	24.105491	0.2958	16.9750867622

Analysis 9:

Top Market and Customer by Net Sales

Create Stored Procedure for top Market for given fiscal year by "net_sale"

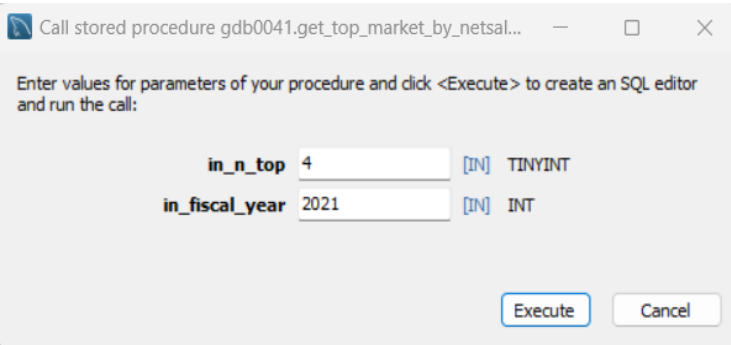
- Input Parameter: fiscal year, top n market
- Get net sales in Million

```
SELECT
    market,
    ROUND(SUM(net_sale) / 1000000, 2) AS net_sale_million
FROM net_sales
WHERE fiscal_year = 2021
GROUP BY market
ORDER BY net_sale_million DESC
LIMIT 5;
```

CREATED STORED PROCEDURE "get_top_market_by_netsales"

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `get_top_market_by_netsales`(
    IN in_n_top TINYINT,
    IN in_fiscal_year INT
    )
BEGIN
    SELECT
        market,
        ROUND(SUM(net_sale) / 1000000, 2) AS net_sale_million
    FROM net_sales
    WHERE fiscal_year = in_fiscal_year
    GROUP BY market
    ORDER BY net_sale_million DESC
    LIMIT in_n_top;
END
```

Result:



market	net_sale_million
India	210.67
USA	132.05
South Korea	64.01
Canada	45.89

Top Customer for given fiscal year by "net_sale" and "Market"

```
SELECT
    c.customer,
    ROUND(SUM(net_sale) / 1000000, 2) AS net_sale_million
FROM net_sales AS s
JOIN dim_customer AS c
    ON s.customer_code =c.customer_code
WHERE fiscal_year = 2021
    AND s.market = "india"
GROUP BY c.customer
ORDER BY net_sale_million DESC
LIMIT 5;
```

Result:

Result Grid | Filter Rows: | Export: | W

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

Analysis 10:

Percentage share of net sales for each customer within their respective region for 2021

```
WITH cte AS(
    SELECT
        c.region,
        c.customer,
        ROUND(SUM(net_sale) / 1000000, 2) AS net_sale_million
    FROM net_sales AS s
    JOIN dim_customer AS c
        ON s.customer_code =c.customer_code
    WHERE fiscal_year = 2021
    GROUP BY c.region,c.customer
    )

SELECT
    *,
    net_sale_million * 100 / SUM(net_sale_million) OVER( PARTITION BY region) AS pct_share_by_region
FROM cte
ORDER BY region, pct_share_by_region DESC;
```

Result:

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

region	customer	net_sale_million	pct_share_by_region
EU	Billa	1.65	0.821631
EU	Unity Stores	1.60	0.796733
EU	Otto	1.57	0.781795
EU	Saturn	1.56	0.776815
EU	Info Stores	1.51	0.751917
EU	Forward Stores	1.48	0.736978
EU	Flawless Stores	1.47	0.731999
EU	Notebillig	1.47	0.731999
EU	Digimarket	1.44	0.717060
EU	Premium Stores	1.39	0.692162
EU	Nova	0.46	0.229061
LATAM	Amazon	1.54	48.734177
LATAM	Atliq e Store	1.09	34.493671
LATAM	Electricalsbea...	0.53	16.772152
NA	Amazon	30.31	17.033832
NA	Atliq Exclusive	14.95	8.401708
NA	walmart	12.63	7.097898
NA	Atliq e Store	12.42	6.979881
NA	Costco	12.19	6.850624
NA	Staples	11.49	6.457233
NA	Flipkart	10.35	5.816567
NA	Path	9.10	5.114083
NA	Ebay	8.74	4.911768
NA	Acclaimed Sto...	8.53	4.793751
NA	Control	8.48	4.765651
NA	BestBuy	8.26	4.642014

Analysis 11:

Get Top n product in each division by their sold quantity in given FY

```
WITH cte AS (
    SELECT
        p.division,
        p.product,
        SUM(s.sold_quantity) AS total_quantity
    FROM fact_sales_monthly AS s
    JOIN dim_product AS p
        ON s.product_code = p.product_code
    WHERE s.fiscal_year = 2021
    GROUP BY p.division, p.product
    )
SELECT * FROM
            (SELECT
                *,
                DENSE_RANK() OVER(PARTITION BY division ORDER BY total_quantity DESC) AS dense_rnk
            FROM cte) AS d_table
WHERE dense_rnk <=3;
```

Create Stored Procedure:

Name: get_top_n_product_per_divison_by_sold_quantity

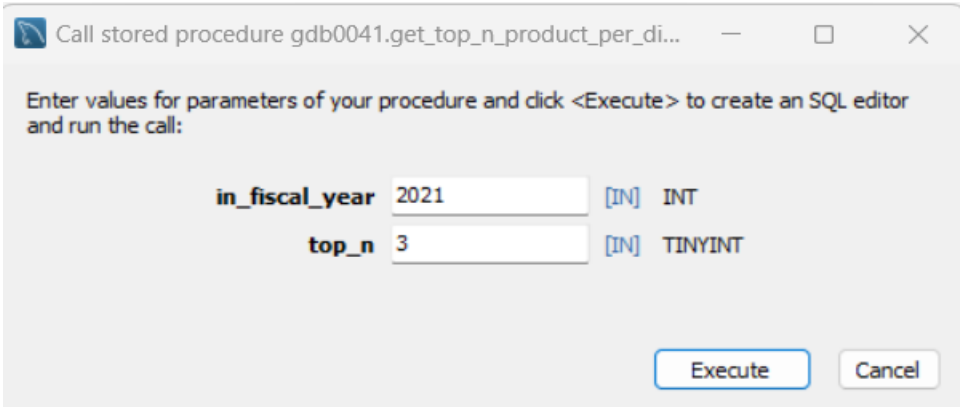
The name of the routine is parsed automatically from the DDL statement. The DDL is parsed automatically while you type.

DDL:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `get_top_n_product_per_divison_by_sold_quantity`(
        IN in_fiscal_year INT,
        IN top_n TINYINT)
BEGIN
        WITH cte1 AS (
                SELECT
                    p.division,
                    p.product,
                    SUM(s.sold_quantity) AS total_quantity
                FROM fact_sales_monthly AS s
                JOIN dim_product AS p
                    ON s.product_code = p.product_code
                WHERE s.fiscal_year = in_fiscal_year
                GROUP BY p.division, p.product
                ),
            cte2 AS (
                SELECT
                    *,
                    DENSE_RANK() OVER(PARTITION BY division ORDER BY total_quantity DESC) AS dense_rnk
                FROM cte1
                )

        SELECT * FROM cte2
        WHERE dense_rnk <= top_n;
END
```

Call Stored Procedure "get_top_n_product_per_divison_by_sold_quantity"



Result:

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

division	product	total_quantity	dense_rnk
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