

# SQL AD-HOC FINANCIAL ANALYTICS



AtliQ  
Hardwares

## About the Company

AtliQ Hardware is a prominent player in the **computer hardware** industry, offering a diverse range of products such as PCs, storage devices, peripherals, and networking equipment. With a global reach, the company serves customers through various channels, including **retailers** like Croma and Amazon, its own **direct** platforms like the AtliQ e-store and exclusive outlets, and **distributors** such as Neptune. Whether you're shopping in physical stores or online, AtliQ Hardware ensures its products are easily accessible through both **Brick&Mortar** and **E-Commerce** platforms.



## Company's Market

NA

North America

LATAM

Latin America

EU

European Union

APAC


Asia-Pacific



## Problem Statement:

The current sales reporting processes at AtliQ Hardware are inefficient and require significant manual effort from the data analytics team. The complex SQL queries used for analysis are difficult to maintain and execute, limiting the ability of business users to access timely and relevant sales information. The database schema requires optimization to improve query performance and enable scalable reporting

## Project Objectives:

- To determine the top-performing markets, products, and customers based on net sales.
  - To efficiently address specific, **Ad-Hoc** data requests from stakeholders.
  - To build reusable SQL components to simplify sales reporting and analysis.
  - To create stored procedures that can be executed by users with limited database access.
- 

## Data Understanding:

This project utilizes a relational database stored in a **MySQL** (gdb0041) server and **Microsoft Excel** was selected for data visualization.

The database contains several tables. The key tables used in this analysis are:

- **dim\_customer**: Customer information (customer\_code, customer, market, region, platform, channel, sub\_zone).
- **dim\_date**: Date related information (calendar\_date, fiscal\_year).
- **dim\_product**: Product information (product\_code, product, variant, division, segment, category, product\_varchar).
- **fact\_act\_est**: Forecast quantity and actual sales quantity (date, fiscal\_year, product\_code, customer\_code, sold\_quantity, forecast\_quantity).
- **fact\_forecast\_monthly**: Forecasted sales quantity (date, product\_code, customer\_code, forecast\_quantity).
- **fact\_freight\_cost**: Freight cost data (market, fiscal\_year, freight\_pct, other\_cost\_pct).
- **fact\_gross\_price**: Gross price of each product per fiscal year (product\_code, fiscal\_year, gross\_price).
- **fact\_manufacturing\_cost**: Manufacturing cost data (product\_code, cost\_year, manufacturing\_cost).
- **fact\_post\_invoice\_deductions**: Post-invoice discounts (customer\_code, product\_code, date, discounts\_pct, other\_deductions\_pct).
- **fact\_pre\_invoice\_deductions**: Pre-invoice discounts per customer and fiscal year (customer\_code, fiscal\_year, pre\_invoice\_discount\_pct).
- **fact\_sales\_monthly**: Core sales transaction data (date, product\_code, fiscal\_year, customer\_code, sold\_quantity).

# Addressing Ad-Hoc Requests

## Prerequisite - The get\_fiscal\_year() Function

Many of the SQL queries in this project rely on a user-defined function called get\_fiscal\_year(). To ensure accurate and consistent reporting, this project utilizes get\_fiscal\_year() function. This function calculates the fiscal year from a given calendar date, based on AtliqQ's specific fiscal year definition

- SQL Query:

```
1 • CREATE FUNCTION `get_fiscal_year` (calendar_date DATE)
2   RETURNS int
3   DETERMINISTIC
4   BEGIN
5       DECLARE fiscal_year INT;
6       SET fiscal_year = YEAR(DATE_ADD(calendar_date, INTERVAL 4 MONTH));
7       RETURN fiscal_YEAR;
8   END
```

# 1. Croma India Product Wise Sales Report

**Ad-Hoc Request:** Generate a report of individual product sales (aggregated on a monthly basis at the product code level) for Croma India customer for FY-2021 to track individual product sales and run further product analytics on it as well.

The report should have the following fields:

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

◦ SQL Query:

```
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
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
customer_code=90002002 AND
get_fiscal_year(s.date)=2021
LIMIT 100000;
```

- Query Result

date	product_code	product	variant	sold_quantity	gross_price_per_item	gross_price_total
2020-09-01	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 R...	Standard	202	19.0573	3849.57
2020-09-01	A0118150102	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 R...	Plus	162	21.4565	3475.95
2020-09-01	A0118150103	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 R...	Premium	193	21.7795	4203.44
2020-09-01	A0118150104	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 R...	Premium Plus	146	22.9729	3354.04
2020-09-01	A0219150201	AQ WereWolf NAS Internal Hard Drive HDD – 8....	Standard	149	23.6987	3531.11
2020-09-01	A0219150202	AQ WereWolf NAS Internal Hard Drive HDD – 8....	Plus	107	24.7312	2646.24
2020-09-01	A0220150203	AQ WereWolf NAS Internal Hard Drive HDD – 8....	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
2020-09-01	A0519150201	AQ Mforce Gen Y	Standard 1	138	22.3984	3090.98
2020-09-01	A0519150202	AQ Mforce Gen Y	Standard 2	72	24.9298	1794.95



## 2. Gross Monthly Total Sales Report for Croma:

### Ad-Hoc Request:

Need an aggregate monthly gross sales report for Croma India customer to track how much sales this particular customer is generating for AtliqQ and manage our relationships accordingly.

The report should have the following fields:

- Month
- Total gross sales amount to Croma India in this month

### SQL Query:

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

◦ Query Result

	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	379549.74
	2019-05-01	340152.29
	2019-06-01	343792.08

### 3. Stored Procedure for Monthly Gross Sales Report:

#### Ad-Hoc Request:

Create a stored proc for monthly gross sales report so that a user doesn't have to manually modify the query every time. Stored proc can be run by other users too who have limited access to database and they can generate this report without needing to involve the data analytics team.

The report should have the following columns:

- Month
- Total gross sales in that month from a given customer.

#### SQL Query:

DDL:

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

- Query Result

```
1 CREATE DEFINER='root'@'localhost' PROCEDURE `get_monthly_gross_sales_for_customer`(  
2     in_customer_codes TEXT  
3 )  
4 BEGIN  
5  
6     SELECT  
7         s.date,  
8  
9  
10  
11  
12  
13  
14  
15     ORDER BY s.date DESC;  
16 END
```

Call stored procedure gdb0041.get\_monthly\_gross\_sales\_...

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

**in\_customer\_codes**  [IN] TEXT

Execute Cancel

Limit to 1000 rows

```
1 call gdb0041.get_monthly_gross_sales_for_customer('9002002,90002008');  
2
```

Result Grid Filter Rows: Export: Wrap Cell Content:

	date	monthly_sales
▶	2021-12-01	31235101.21
	2021-11-01	30191765.50
	2021-10-01	22735992.10
	2021-08-01	3645597.75
	2021-07-01	3862788.49
	2021-06-01	3728275.41
	2021-04-01	3856602.45
	2021-03-01	3606840.15
	2021-02-01	3614565.35
	2020-12-01	6421539.33
	2020-11-01	6392234.29
	2020-10-01	4942029.15
	2020-08-01	1288334.26
	2020-07-01	2784738.73
	2020-06-01	2486977.98

Result 1 x

## 4. Stored Procedure for Market Badge:

### Ad-Hoc Request:

- Create 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.

### SQL Query:

```
1 CREATE PROCEDURE `get_market_badge` (  
2     IN in_market VARCHAR(45),  
3     IN in_fiscal_year YEAR,  
4     OUT out_level VARCHAR(45)  
5 )  
6 BEGIN  
7     DECLARE qty INT DEFAULT 0;  
8  
9     # Default market is India  
10    IF in_market = "" THEN  
11        SET in_market="India";  
12    END IF;  
13  
14    # Retrieve total sold quantity for a given market in a given year  
15    SELECT  
16        SUM(s.sold_quantity) INTO qty  
17    FROM fact_sales_monthly s  
18    JOIN dim_customer c  
19    ON s.customer_code=c.customer_code  
20    WHERE  
21        get_fiscal_year(s.date)=in_fiscal_year AND  
22        c.market=in_market;  
23  
24    # Determine Gold vs Silver status  
25    IF qty > 5000000 THEN  
26        SET out_level = 'Gold';  
27    ELSE  
28        SET out_level = 'Silver';  
29    END IF;  
30 END
```

## Query Result

```
7 DECLARE qty INT DEFAULT 0;
8
9 # Default market is India
10 IF in_market = "" THEN
11     SET in_market = 'India';
12 END IF;
13
14 # Determine Gold vs Silver status
15 IF qty > 5000000 THEN
16     SET out_level = 'Gold';
17 ELSE
18     SET out_level = 'Silver';
19 END IF;
20
21 END
```

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	usa	[IN]	VARCHAR(45)
in_fiscal_year	2021	[IN]	YEAR
out_level		[OUT]	VARCHAR(45)

Execute Cancel

```
1 • set @out_level = '0';
2 • call gdb0041.get_market_badge('usa', 2021, @out_level);
3 • select @out_level;
4
```

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

	@out_level
▶	Gold



## 5. Improving Data Structure for Analysis

To streamline queries, promote code reuse, and ensure accurate calculations, a series of database views were created. These views encapsulate the logic for calculating net sales by incorporating pre- and post-invoice discounts.

### 5.1. Creating the sales\_preinv\_discount View:

To simplify calculations and create a reusable object, a view is created that encapsulates the logic for calculating gross sales and applying pre-invoice discounts.

This view will be used to create further required views

- SQL Query:

```
1 • CREATE VIEW `sales_preinv_discount` AS
2 SELECT
3     s.date,
4     s.fiscal_year,
5     s.customer_code,
6     c.market,
7     s.product_code,
8     p.product,
9     p.variant,
10    s.sold_quantity,
11    g.gross_price as gross_price_per_item,
12    ROUND(s.sold_quantity*g.gross_price,2) as gross_price_total,
13    pre.pre_invoice_discount_pct
14 FROM fact_sales_monthly s
15 JOIN dim_customer c
16     ON s.customer_code = c.customer_code
17 JOIN dim_product p
18     ON s.product_code=p.product_code
19 JOIN fact_gross_price g
20     ON g.fiscal_year=s.fiscal_year
21     AND g.product_code=s.product_code
22 JOIN fact_pre_invoice_deductions as pre
23     ON pre.customer_code = s.customer_code AND
24     pre.fiscal_year=s.fiscal_year;
```

## 5.2. Creating the sales\_postinv\_discount View:

To extend the previous view to include post-invoice discounts and calculate net invoice sales, a view has been created.

This view will be used to create final view of net\_sales

### ◦ SQL Query:

```
1 • CREATE VIEW `sales_postinv_discount` AS
2 SELECT
3     s.date, s.fiscal_year,
4     s.customer_code, s.market,
5     s.product_code, s.product, s.variant,
6     s.sold_quantity, s.gross_price_total,
7     s.pre_invoice_discount_pct,
8     (s.gross_price_total-s.pre_invoice_discount_pct*s.gross_price_total) as net_invoice_sales,
9     (po.discounts_pct+po.other_deductions_pct) as post_invoice_discount_pct
10 FROM sales_preinv_discount s
11 JOIN fact_post_invoice_deductions po
12     ON po.customer_code = s.customer_code AND
13     po.product_code = s.product_code AND
14     po.date = s.date;
```




### 5.3. Creating the net\_sales View:

A final view needs to be created that incorporates all discounts (pre- and post-invoice) to calculate the final net sales amount for use in subsequent reporting queries.


- SQL Query:

```
1 • CREATE VIEW `net_sales` AS
2 SELECT
3     *,
4     net_invoice_sales*(1-post_invoice_discount_pct) as net_sales
5 FROM gdb0041.sales_postinv_discount;
```


▼

 **gdb0041**


▶

 Tables


▼

 Views


▶

 net\_sales

▶

 sales\_postinv\_discount

▶

 sales\_preinv\_discount

## 6. Top Markets, Products, Customers for a Given Financial Year:

### Ad-Hoc Request:

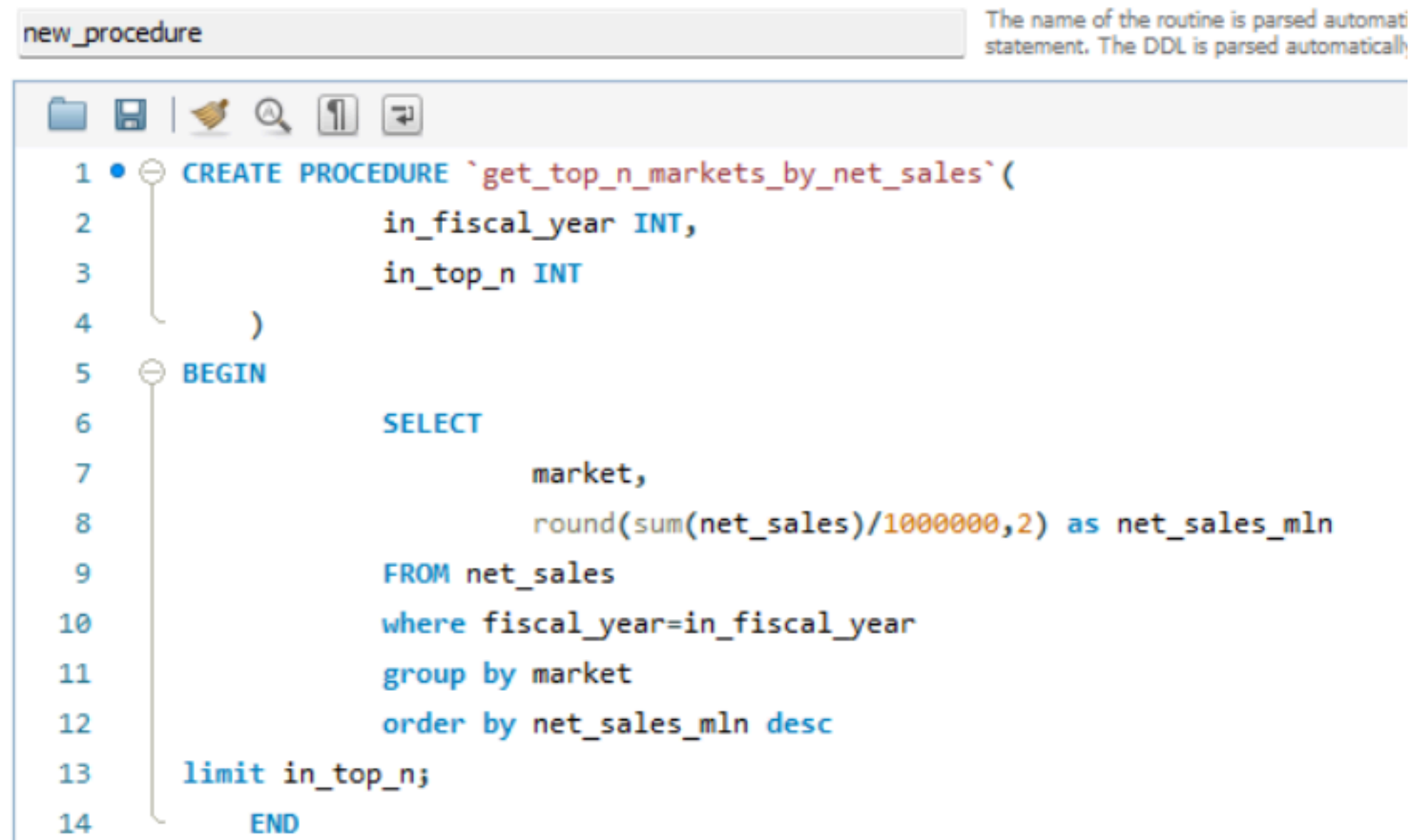
A report is needed for top markets, products, and customers by net sales for a given financial year so that a user can have a holistic view of our financial performance and can take appropriate actions to address any potential issues.

To facilitate reuse and simplify report generation, create stored procedures for each of these reports.

- Report for top markets.
- Report for top products.
- Report for top customers.

- 6.1: Creating Store Procedure to get top n markets by net sales for a given year

- SQL Query:



The screenshot shows a SQL IDE window titled "new\_procedure". The main text area contains the following SQL code:

```
1 CREATE PROCEDURE `get_top_n_markets_by_net_sales`(  
2     in_fiscal_year INT,  
3     in_top_n INT  
4 )  
5 BEGIN  
6     SELECT  
7         market,  
8         round(sum(net_sales)/1000000,2) as net_sales_mln  
9     FROM net_sales  
10    where fiscal_year=in_fiscal_year  
11    group by market  
12    order by net_sales_mln desc  
13    limit in_top_n;  
14 END
```

On the right side of the IDE, a tooltip states: "The name of the routine is parsed automatically. The DDL is parsed automatically."

- Query Result

```
1 CREATE DEFINER='root'@'localhost' PROCEDURE `get_top_n_markets_by_net_sales`(  
2     in_fiscal_year INT,  
3     in_top_n INT  
4 )  
5 BEGIN  
6  
7  
8  
9  
10  
11  
12  
13 limit in  
14 END
```

Call stored procedure gdb0041.get\_top\_n\_markets\_by\_ne... — □ ×

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

Execute Cancel

Limit to 1000 rows

```
1 call gdb0041.get_top_n_markets_by_net_sales(2021, 5);  
2
```

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

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

Result 1 ×

## 6. Top Markets, Products, Customers for a Given Financial Year:

### Ad-Hoc Request:

A report is needed for top markets, products, and customers by net sales for a given financial year so that a user can have a holistic view of our financial performance and can take appropriate actions to address any potential issues.

To facilitate reuse and simplify report generation, create stored procedures for each of these reports.

- Report for top markets.
- Report for top products.
- Report for top customers.

6.2: Create 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

#### ◦ SQL Query:

```
1  CREATE PROCEDURE `get_top_n_customers_by_net_sales` (  
2      in_market VARCHAR(45),  
3      in_fiscal_year INT,  
4      in_top_n INT  
5  )  
6  BEGIN  
7      select  
8          customer,  
9          round(sum(net_sales)/1000000,2) as net_sales_mln  
10     FROM net_sales s  
11     join dim_customer c  
12         on s.customer_code=c.customer_code  
13     where  
14         s.fiscal_year=in_fiscal_year  
15         and s.market=in_market  
16     group by customer  
17     order by net_sales_mln desc  
18     limit in_top_n;  
19  END
```

- Query Result

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

DDL:

```
1 CREATE DEFINER='root'@'localhost' PROCEDURE `get_top_n_customers_by_net_sales`(  
2     in_market VARCHAR(45),  
3     in_fiscal_year INT,  
4     in_top_n INT  
5 )  
6 BEGIN  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17     order by net_sales_mln desc  
18     limit in_top_n;  
19 END
```

Call stored procedure gdb0041.get\_top\_n\_customers\_by\_...

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

in_market	<input type="text" value="india"/>	[IN]	VARCHAR(45)
in_fiscal_year	<input type="text" value="2021"/>	[IN]	INT
in_top_n	<input type="text" value="5"/>	[IN]	INT

Execute Cancel

```
1 call gdb0041.get_top_n_customers_by_net_sales('india', 2021, 5);  
2
```

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

	customer	net_sales_mln
▶	Amazon	30.00
	Atiq Exclusive	23.98
	Flipkart	12.96
	Electricalsociety	12.31
	Propel	11.86

## 6. Top Markets, Products, Customers for a Given Financial Year:

### Ad-Hoc Request:

A report is needed for top markets, products, and customers by net sales for a given financial year so that a user can have a holistic view of our financial performance and can take appropriate actions to address any potential issues.

To facilitate reuse and simplify report generation, create stored procedures for each of these reports.

- Report for top markets.
- Report for top products.
- Report for top customers.

### 6.3: Create Stored Procedure to get top n products by net sales for a given year (product name without variant)

#### ◦ SQL Query:

```
1  -- Get top n products by net sales for a given year
2
3
4  CREATE PROCEDURE get_top_n_products_by_net_sales(
5      in_fiscal_year int,
6      in_top_n int
7  )
8  BEGIN
9      select
10         product,
11         round(sum(net_sales)/1000000,2) as net_sales_mln
12     from gdb0041.net_sales
13     where fiscal_year=in_fiscal_year
14     group by product
15     order by net_sales_mln desc
16     limit in_top_n;
17  END
18
```



- Query Result

```
1 • -- Get top n products by net sales for a given year
```

```
2
```

```
3
```

```
4 CREATE PROCEDURE get_top_n_products_by_net_sales(  
5     in_fiscal_year int,  
6  
7 )  
8 BEGIN  
9  
10  
11  
12  
13  
14  
15  
16     limit in_top_n;  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000  
1001  
1002  
1003  
1004  
1005  
1006  
1007  
1008  
1009  
1010  
1011  
1012  
1013  
1014  
1015  
1016  
1017  
1018  
1019  
1020  
1021  
1022  
1023  
1024  
1025  
1026  
1027  
1028  
1029  
1030  
1031  
1032  
1033  
1034  
1035  
1036  
1037  
1038  
1039  
1040  
1041  
1042  
1043  
1044  
1045  
1046  
1047  
1048  
1049  
1050  
1051  
1052  
1053  
1054  
1055  
1056  
1057  
1058  
1059  
1060  
1061  
1062  
1063  
1064  
1065  
1066  
1067  
1068  
1069  
1070  
1071  
1072  
1073  
1074  
1075  
1076  
1077  
1078  
1079  
1080  
1081  
1082  
1083  
1084  
1085  
1086  
1087  
1088  
1089  
1090  
1091  
1092  
1093  
1094  
1095  
1096  
1097  
1098  
1099  
1100  
1101  
1102  
1103  
1104  
1105  
1106  
1107  
1108  
1109  
1110  
1111  
1112  
1113  
1114  
1115  
1116  
1117  
1118  
1119  
1120  
1121  
1122  
1123  
1124  
1125  
1126  
1127  
1128  
1129  
1130  
1131  
1132  
1133  
1134  
1135  
1136  
1137  
1138  
1139  
1140  
1141  
1142  
1143  
1144  
1145  
1146  
1147  
1148  
1149  
1150  
1151  
1152  
1153  
1154  
1155  
1156  
1157  
1158  
1159  
1160  
1161  
1162  
1163  
1164  
1165  
1166  
1167  
1168  
1169  
1170  
1171  
1172  
1173  
1174  
1175  
1176  
1177  
1178  
1179  
1180  
1181  
1182  
1183  
1184  
1185  
1186  
1187  
1188  
1189  
1190  
1191  
1192  
1193  
1194  
1195  
1196  
1197  
1198  
1199  
1200  
1201  
1202  
1203  
1204  
1205  
1206  
1207  
1208  
1209  
1210  
1211  
1212  
1213  
1214  
1215  
1216  
1217  
1218  
1219  
1220  
1221  
1222  
1223  
1224  
1225  
1226  
1227  
1228  
1229  
1230  
1231  
1232  
1233  
1234  
1235  
1236  
1237  
1238  
1239  
1240  
1241  
1242  
1243  
1244  
1245  
1246  
1247  
1248  
1249  
1250  
1251  
1252  
1253  
1254  
1255  
1256  
1257  
1258  
1259  
1260  
1261  
1262  
1263  
1264  
1265  
1266  
1267  
1268  
1269  
1270  
1271  
1272  
1273  
1274  
1275  
1276  
1277  
1278  
1279  
1280  
1281  
1282  
1283  
1284  
1285  
1286  
1287  
1288  
1289  
1290  
1291  
1292  
1293  
1294  
1295  
1296  
1297  
1298  
1299  
1300  
1301  
1302  
1303  
1304  
1305  
1306  
1307  
1308  
1309  
1310  
1311  
1312  
1313  
1314  
1315  
1316  
1317  
1318  
1319  
1320  
1321  
1322  
1323  
1324  
1325  
1326  
1327  
1328  
1329  
1330  
1331  
1332  
1333  
1334  
1335  
1336  
1337  
1338  
1339  
1340  
1341  
1342  
1343  
1344  
1345  
1346  
1347  
1348  
1349  
1350  
1351  
1352  
1353  
1354  
1355  
1356  
1357  
1358  
1359  
1360  
1361  
1362  
1363  
1364  
1365  
1366  
1367  
1368  
1369  
1370  
1371  
1372  
1373  
1374  
1375  
1376  
1377  
1378  
1379  
1380  
1381  
1382  
1383  
1384  
1385  
1386  
1387  
1388  
1389  
1390  
1391  
1392  
1393  
1394  
1395  
1396  
1397  
1398  
1399  
1400  
1401  
1402  
1403  
1404  
1405  
1406  
1407  
1408  
1409  
1410  
1411  
1412  
1413  
1414  
1415  
1416  
1417  
1418  
1419  
1420  
1421  
1422  
1423  
1424  
1425  
1426  
1427  
1428  
1429  
1430  
1431  
1432  
1433  
1434  
1435  
1436  
1437  
1438  
1439  
1440  
1441  
1442  
1443  
1444  
1445  
1446  
1447  
1448  
1449  
1450  
1451  
1452  
1453  
1454  
1455  
1456  
1457  
1458  
1459  
1460  
1461  
1462  
1463  
1464  
1465  
1466  
1467  
1468  
1469  
1470  
1471  
1472  
1473  
1474  
1475  
1476  
1477  
1478  
1479  
1480  
1481  
1482  
1483  
1484  
1485  
1486  
1487  
1488  
1489  
1490  
1491  
1492  
1493  
1494  
1495  
1496  
1497  
1498  
1499  
1500  
1501  
1502  
1503  
1504  
1505  
1506  
1507  
1508  
1509  
1510  
1511  
1512  
1513  
1514  
1515  
1516  
1517  
1518  
1519  
1520  
1521  
1522  
1523  
1524  
1525  
1526  
1527  
1528  
1529  
1530  
1531  
1532  
1533  
1534  
1535  
1536  
1537  
1538  
1539  
1540  
1541  
1542  
1543  
1544  
1545  
1546  
1547  
1548  
1549  
1550  
1551  
1552  
1553  
1554  
1555  
1556  
1557  
1558  
1559  
1560  
1561  
1562  
1563  
1564  
1565  
1566  
1567  
1568  
1569  
1570  
1571  
1572  
1573  
1574  
1575  
1576  
1577  
1578  
1579  
1580  
1581  
1582  
1583  
1584  
1585  
1586  
1587  
1588  
1589  
1590  
1591  
1592  
1593  
1594  
1595  
1596  
1597  
1598  
1599  
1600  
1601  
1602  
1603  
1604  
1605  
1606  
1607  
1608  
1609  
1610  
1611  
1612  
1613  
1614  
1615  
1616  
1617  
1618  
1619  
1620  
1621  
1622  
1623  
1624  
1625  
1626  
1627  
1628  
1629  
1630  
1631  
1632  
1633  
1634  
1635  
1636  
1637  
1638  
1639  
1640  
1641  
1642  
1643  
1644  
1645  
1646  
1647  
1648  
1649  
1650  
1651  
1652  
1653  
1654  
1655  
1656  
1657  
1658  
1659  
1660  
1661  
1662  
1663  
1664  
1665  
1666  
1667  
1668  
1669  
1670  
1671  
1672  
1673  
1674  
1675  
1676  
1677  
1678  
1679  
1680  
1681  
1682  
1683  
1684  
1685  
1686  
1687  
1688  
1689  
1690  
1691  
1692  
1693  
1694  
1695  
1696  
1697  
1698  
1699  
1700  
1701  
1702  
1703  
1704  
1705  
1706  
1707  
1708  
1709  
1710  
1711  
1712  
1713  
1714  
1715  
1716  
1717  
1718  
1719  
1720  
1721  
1722  
1723  
1724  
1725  
1726  
1727  
1728  
1729  
1730  
1731  
1732  
1733  
1734  
1735  
1736  
1737  
1738  
1739  
1740  
1741  
1742  
1743  
1744  
1745  
1746  
1747  
1748  
1749  
1750  
1751  
1752  
1753  
1754  
1755  
1756  
1757  
1758  
1759  
1760  
1761  
1762  
1763  
1764  
1765  
1766  
1767  
1768  
1769  
1770  
1771  
1772  
1773  
1774  
1775  
1776  
1777  
1778  
1779  
1780  
1781  
1782  
1783  
1784  
1785  
1786  
1787  
1788  
1789  
1790  
1791  
1792  
1793  
1794  
1795  
1796  
1797  
1798  
1799  
1800  
1801  
1802  
1803  
1804  
1805  
1806  
1807  
1808  
1809  
1810  
1811  
1812  
1813  
1814  
1815  
1816  
1817  
1818  
1819  
1820  
1821  
1822  
1823  
1824  
1825  
1826  
1827  
1828  
1829  
1830  
1831  
1832  
1833  
1834  
1835  
1836  
1837  
1838  
1839  
1840  
1841  
1842  
1843  
1844  
1845  
1846  
1847  
1848  
1849  
1850  
1851  
1852  
1853  
1854  
1855  
1856  
1857  
1858  
1859  
1860  
1861  
1862  
1863  
1864  
1865  
1866  
1867  
1868  
1869  
1870  
1871  
1872  
1873  
1874  
1875  
1876  
1877  
1878  
1879  
1880  
1881  
1882  
1883  
1884  
1885  
1886  
1887  
1888  
1889  
1890  
1891  
1892  
1893  
1894  
1895  
1896  
1897  
1898  
1899  
1900  
1901  
1902  
1903  
1904  
1905  
1906  
1907  
1908  
1909  
1910  
1911  
1912  
1913  
1914  
1915  
1916  
1917  
1918  
1919  
1920  
1921  
1922  
1923  
1924  
1925  
1926  
1927  
1928  
1929  
1930  
1931  
1932  
1933  
1934  
1935  
1936  
1937  
1938  
1939  
1940  
1941  
1942  
1943  
1944  
1945  
1946  
1947  
1948  
1949  
1950  
1951  
1952  
1953  
1954  
1955  
1956  
1957  
1958  
1959  
1960  
1961  
1962  
1963  
1964  
1965  
1966  
1967  
1968  
1969  
1970  
1971  
1972  
1973  
1974  
1975  
1976  
1977  
1978  
1979  
1980  
1981  
1982  
1983  
1984  
1985  
1986  
1987  
1988  
1989  
1990  
1991  
1992  
1993  
1994  
1995  
1996  
1997  
1998  
1999  
2000  
2001  
2002  
2003  
2004  
2005  
2006  
2007  
2008  
2009  
2010  
2011  
2012  
2013  
2014  
2015  
2016  
2017  
2018  
2019  
2020  
2021  
2022  
2023  
2024  
2025  
2026  
2027  
2028  
2029  
2030  
2031  
2032  
2033  
2034  
2035  
2036  
2037  
2038  
2039  
2040  
2041  
2042  
2043  
2044  
2045  
2046  
2047  
2048  
2049  
2050  
2051  
2052  
2053  
2054  
2055  
2056  
2057  
2058  
2059  
2060  
2061  
2062  
2063  
2064  
2065  
2066  
2067  
2068  
2069  
2070  
2071  
2072  
2073  
2074  
2075  
2076  
2077  
2078  
2079  
2080  
2081  
2082  
2083  
2084  
2085  
2086  
2087  
2088  
2089  
2090  
2091  
2092  
2093  
2094  
2095  
2096  
2097  
2098  
2099  
2100  
2101  
2102  
2103  
2104  
2105  
2106  
2107  
2108  
2109  
2110  
2111  
2112  
2113  
2114  
2115  
2116  
2117  
2118  
2119  
2120  
2121  
2122  
2123  
2124  
2125  
2126  
2127  
2128  
2129  
2130  
2131  
2132  
2133  
2134  
2135  
2136  
2137  
2138  
2139  
2140  
2141  
2142  
2143  
2144  
2145  
2146  
2147  
2148  
2149  
2150  
2151  
2152  
2153  
2154  
2155  
2156  
2157  
2158  
2159  
2160  
2161  
2162  
2163  
2164  
2165  
2166  
2167  
2168  
2169  
2170  
2171  
2172  
2173  
2174  
2175  
2176  
2177  
2178  
2179  
2180  
2181  
2182  
2183  
2184  
2185  
2186  
2187  
2188  
2189  
2190  
2191  
2192  
2193  
2194  
2195  
2196  
2197  
2198  
2199  
2200  
2201  
2202  
2203  
2204  
2205  
2206  
2207  
2208  
2209  
2210  
2211  
2212  
2213  
2214  
2215  
2216  
22
```



## 7. Net Sales % Share Global:

### Ad-Hoc Request:

Develop a bar chart report to display the top 10 markets in FY-2021, ranked by their percentage contribution to total net sales.

#### SQL Query:

```
1  • ⊖ with cte1 as (  
2      select  
3          customer,  
4          round(sum(net_sales)/1000000,2) as net_sales_mln  
5      from net_sales s  
6      join dim_customer c  
7          on s.customer_code=c.customer_code  
8      where s.fiscal_year=2021  
9      group by customer)  
10 select  
11     *,  
12     net_sales_mln*100/sum(net_sales_mln) over() as pct_net_sales  
13 from cte1  
14 order by net_sales_mln desc
```

Query Result

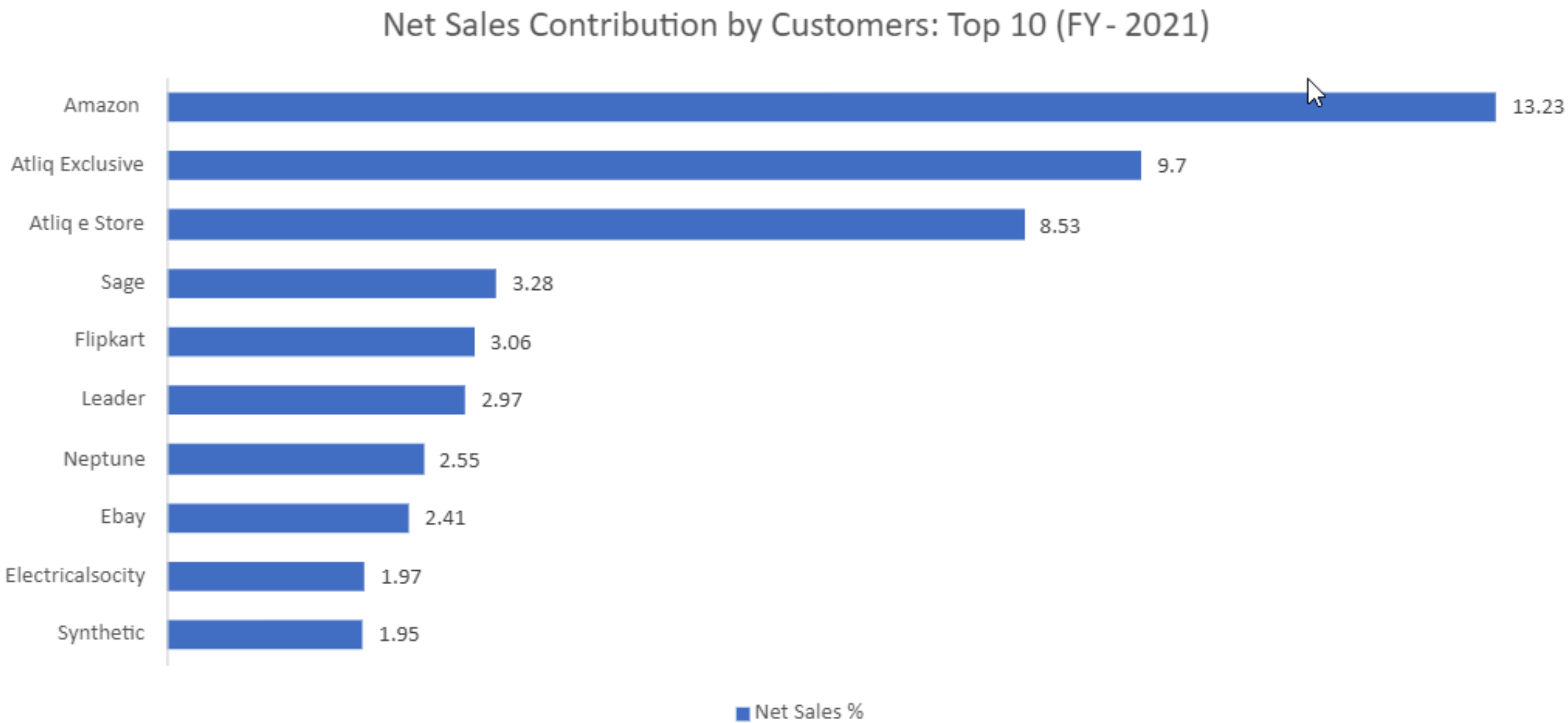
Result Grid

Filter Rows:

Export

	customer	net_sales_mln	pct_net_sales
▶	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
	Synthetic	16.10	1.954121
	Electricalslytical	15.64	1.898289
	Acclaimed Sto...	14.32	1.738075
	Propel	14.14	1.716228
	Novus	12.91	1.566938
	Expression	12.90	1.565724
	Reliance Digital	12.75	1.547518
	walmart	12.63	1.532953
	Costco	12.10	1.470548

Result 2



## 8. Net Sales % Share by Region:

### Ad-Hoc Request:

Develop a set of pie charts showing the percentage breakdown of net sales by the top 10 customers within each region (APAC, EU, LTAM, etc.) for FY-2021.

This will enable regional analysis of the company's financial performance, focusing on the key contributors to sales in each region.

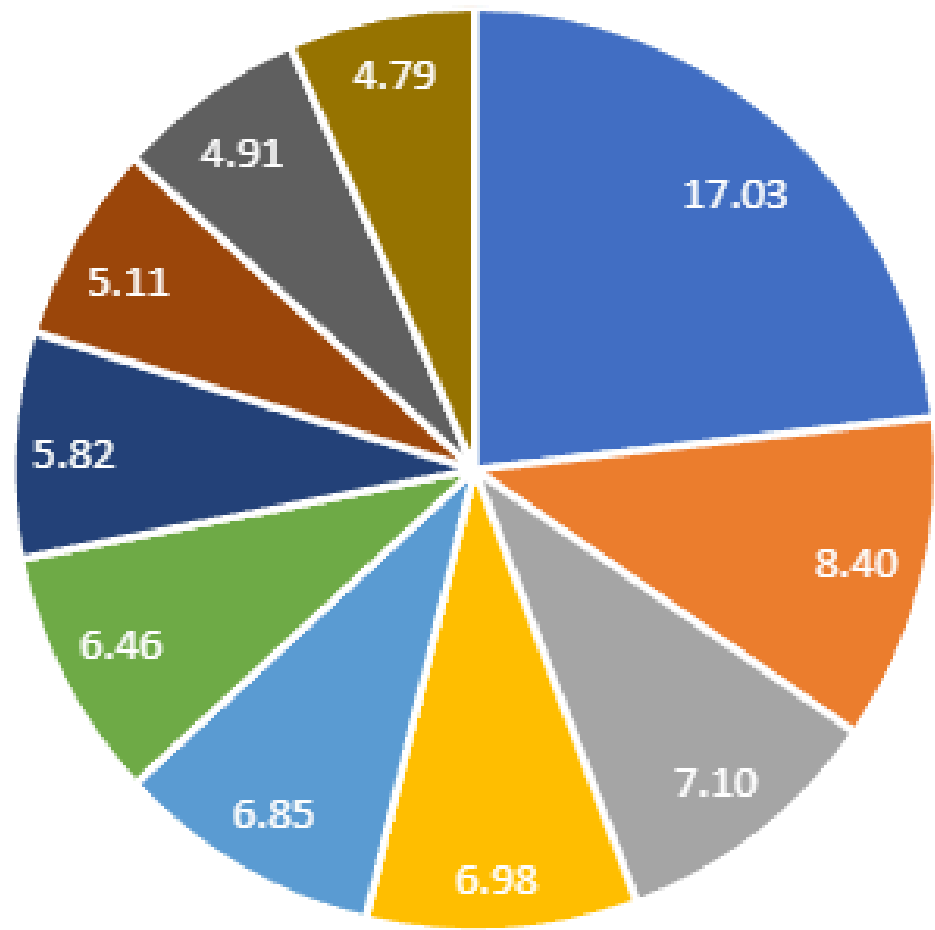
### SQL Query:

```
1 • with cte1 as (  
2     select  
3         c.customer,  
4         c.region,  
5         round(sum(net_sales)/1000000,2) as net_sales_mln  
6     from gdb0041.net_sales n  
7     join dim_customer c  
8         on n.customer_code=c.customer_code  
9     where fiscal_year=2021  
10    group by c.customer, c.region)  
11    select  
12        *,  
13        net_sales_mln*100/sum(net_sales_mln) over (partition by region) as pct_share_region  
14    from cte1  
15    order by region, pct_share_region desc
```

- Query Result

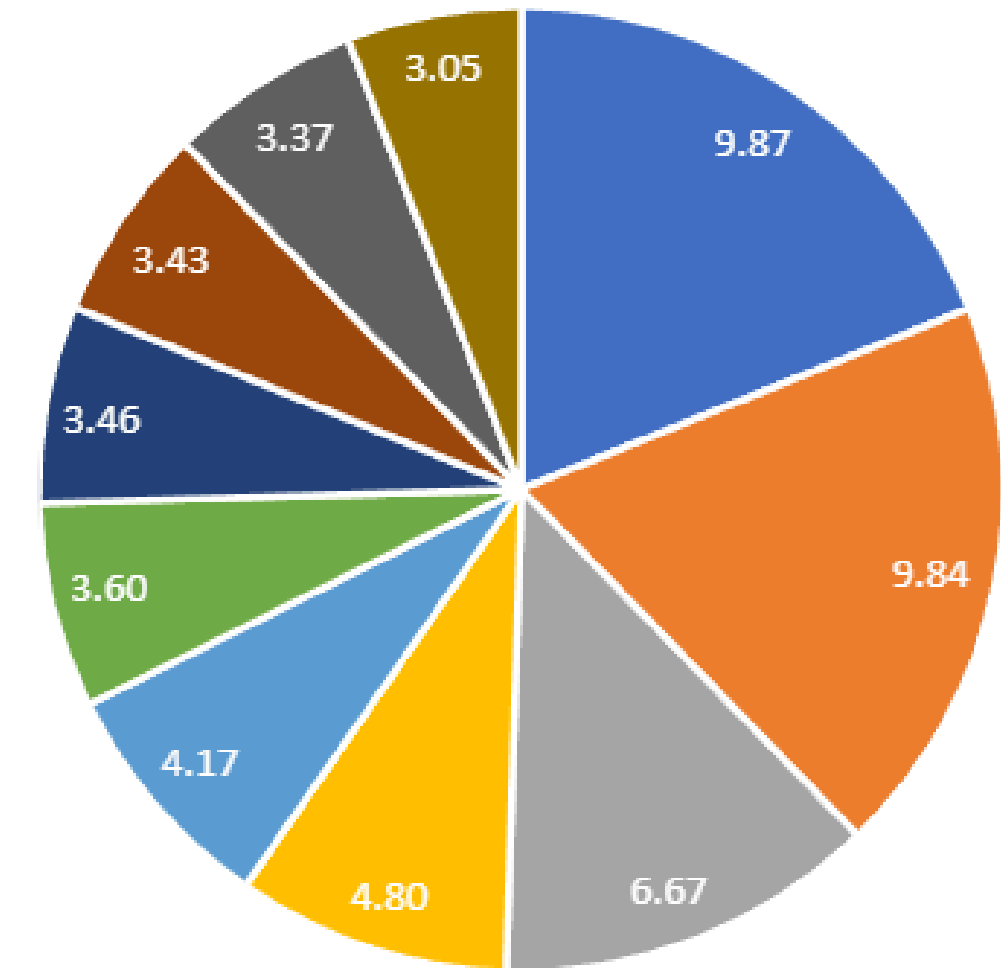
	customer	region	net_sales_mln	pct_share_region
▶	Amazon	APAC	57.41	12.988688
	Atliq Exclusive	APAC	51.58	11.669683
	Atliq e Store	APAC	36.97	8.364253
	Leader	APAC	24.52	5.547511
	Sage	APAC	22.85	5.169683
	Neptune	APAC	21.01	4.753394
	Electricalsocity	APAC	16.25	3.676471
	Propel	APAC	14.14	3.199095
	Synthetic	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

Net Sales % Share by Customer: NA Region (FY 2021)



■ Amazon NA      ■ Atliq Exclusive NA      ■ walmart NA      ■ Atliq e Store NA  
 ■ Costco NA      ■ Staples NA      ■ Flipkart NA      ■ Path NA  
 ■ Ebay NA      ■ Acclaimed Stores NA

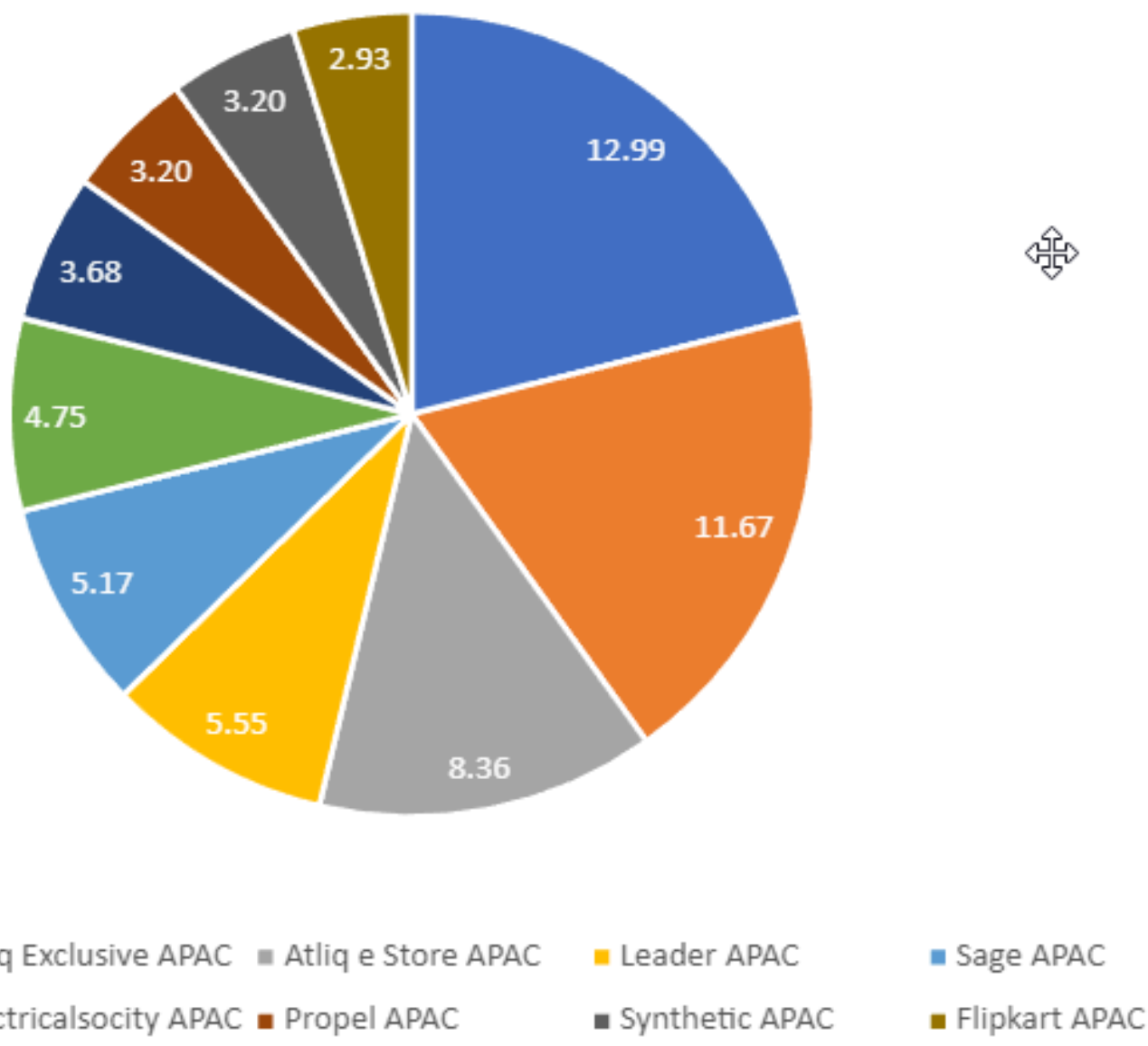
Net Sales % Share by Customer: EU Region (FY 2021)



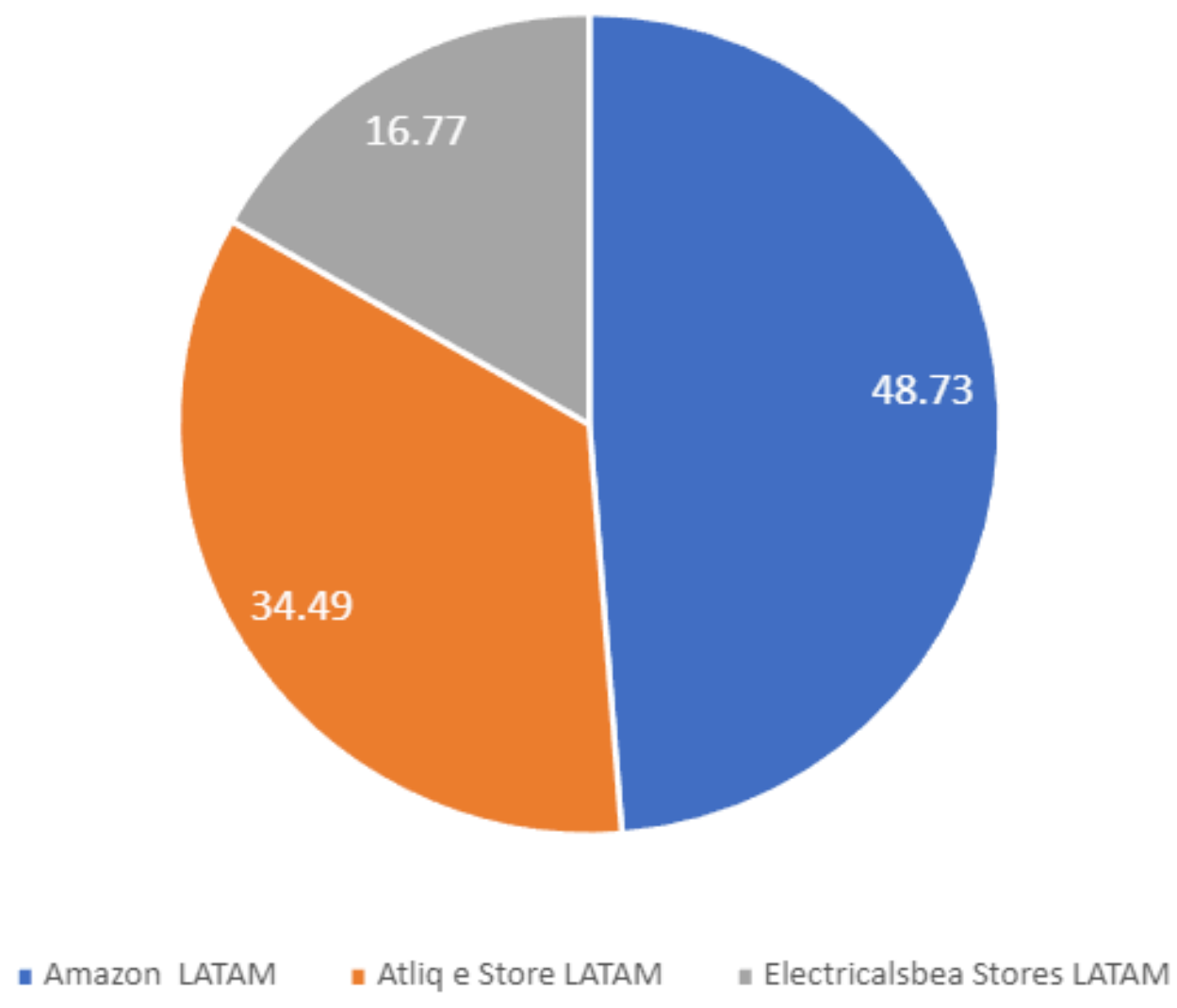
■ Atliq e Store EU      ■ Amazon EU      ■ Atliq Exclusive EU      ■ UniEuro EU      ■ Expert EU  
 ■ Chip 7 EU      ■ Radio Popular EU      ■ Media Markt EU      ■ Elkj p EU      ■ Sorefoz EU

◦ FY2021: Regional Customer Sales Breakdown (Top 10)

Net Sales % Share by Customer: APAC Region (FY 2021)



Net Sales % Share by Customer: LATAM Region (FY 2021)



## Key Insights:

### 1. Top Markets, Products, and Customers (from Stored Procedure Results):

- Market Concentration: The "get\_top\_n\_markets\_by\_net\_sales" result reveals that India and the USA are significantly larger markets for AtliQ than other regions. A substantial portion of revenue is generated from these two areas.
- Customer Dependency: From the get\_top\_n\_customers\_by\_net\_sales screenshot, Amazon and Atliq Exclusive are key customers of atliq across the market.
- Dominant Products: The "get\_top\_n\_products\_by\_net\_sales" result clearly identifies "AQ BZ Allin1" and "AQ Qwerty" as the top-selling products.

### 2. Overall Customer Contribution (Bar Chart - "Net Sales Contribution by Customers: Top 10 (FY-2021)")

- Top-Heavy Revenue Distribution: The bar chart visually confirms that AtliQ Hardware's revenue is heavily weighted towards a few top customers, with Amazon and Atliq Exclusive contributing a substantial portion of the total.
- Atliq Exclusive Channel : The results for Net Sales contribution by Customers show that Atliq Exclusive contribute 9.7% net sales.

## Key Insights:

### 3. Regional Customer Breakdown (Pie Charts - "Net Sales % Share by Customer: NA, EU, APAC, LATAM Regions (FY 2021)")

- Varying Regional Dynamics: The pie charts demonstrate that the distribution of sales across customers differs considerably from region to region.
- Some observations:
  - LATAM sales heavily concentrated with Amazon and Atliq E Store.
  - North America shows a more diversified customer base compared to LATAM, but still relies on Amazon, Atliq Exclusive, Walmart and Atliq e Store.
  - EU has wider revenue sources compared to north america and latam.
  - APAC is dependent on Amazon and Atliq Exclusive, but it does not seem as bad as latam.



## Recommendations:

Based on the findings of this analysis, I recommend that AtliQ Hardware take the following actions to improve its sales performance and strategic decision-making:

- **Prioritize Top Customer Relationships:** Implement a program to proactively manage and nurture relationships with top customers, such as Croma India, Amazon, and AtliQ Exclusive.
- **This can involve:**
  - Dedicated personnel responsible for these accounts.
  - Regular communication and collaboration.
  - Tailored product offerings and marketing campaigns.
  - Monitoring customer satisfaction and loyalty.
- **Pursue Customer Diversification:** Actively seek to expand the customer base beyond the top accounts by targeting new segments and markets.
- **This could include:**
  - Expanding into new geographic regions.
  - Establishing partnerships with distributors and resellers.

## Recommendations:

- **Customize Regional Sales Strategies:** Develop region-specific sales and marketing plans that reflect the unique customer characteristics of each market.
- **Automate Performance Monitoring:** Regularly monitor automated reports using the created views and stored procedures. This enables AtliQ Hardware to:
  - Identify trends and anomalies quickly.
  - Make decisions based on current, readily available data.
- **Refine Pricing Strategies:** Re-evaluate its pricing and discount approach to make better data driven decisions.

## Impact:

This data analytics project has the potential to deliver significant value to AtliQ Hardware by providing actionable insights, improving reporting efficiency, and enabling data-driven decision-making across the organization. Specifically, the implementation of the project's findings and solutions could lead to the following positive impacts:

- **Improved Strategic Decision-Making:** By identifying top-performing markets, products, and customers, AtliQ Hardware can make more informed decisions about where to focus its efforts and investments.
- **Enhanced Customer Relationships:** Understanding regional customer dynamics and individual customer contributions enables AtliQ to better tailor its sales and marketing activities to different customer groups.
- **Optimized Pricing Strategies:** By quantifying the impact of pre- and post-invoice discounts on net sales, AtliQ can refine its pricing strategies to maximize profitability.
- **Increased Reporting Efficiency:** The creation of reusable views and stored procedures streamlines the sales reporting process, reducing the time and effort required to generate key reports.
- **Democratized Data Access:** By enabling users with limited database access to execute stored procedures and generate reports, AtliQ can empower business users to make data-driven decisions more easily.
- **Enhanced Scalability:** By adding the `fiscal_year` column to `fact_sales_monthly`, it increases data access. Scalability ensures it can support business without performance impact.

## Demonstrated SQL Skills:

This project demonstrates proficiency in a range of SQL skills relevant to data analysis, including:

- Data Definition Language (DDL): Created database objects using CREATE FUNCTION, and CREATE VIEW statements.
- Data Querying:
  - Used SELECT statements with WHERE, GROUP BY, ORDER BY, and LIMIT clauses.
  - Performed data aggregations and transformations, having written functions such as
    - Used JOIN Operations: INNER JOIN
    - Using Aggregate Functions: SUM(), AVG(), and ROUND().
    - Created the User-Defined Function (UDF): get\_fiscal\_year().
    - Used functions such as FIND\_IN\_SET statements.
    - Leveraged **Views** table to easily perform analysis.

## Demonstrated SQL Skills:

- Stored Procedures:
  - Created stored procedures to encapsulate complex logic and automate tasks using CREATE PROCEDURE.
  - Utilized input and output parameters.
  - Used control flow statements such as IF and ELSE inside procedure to make action.
- Also,
  - Utilized Window Functions by partitioning with OVER clause .
  - Used String manipulation techniques, such as creating a LIKE statement.
  - Calculated date with operations such as DATE\_ADD.
  - Utilized Date and Time Functions such as MONTH.
  - Improved performance by leveraging WITH clause.

# Thank You!!



[Linkedin](#)



Mail

avinashturkar922@gmail.com