Atliq Hardwares

# SQL AD-HOC FINANCIAL ANALYTICS

Created By: Avinash Turkar

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.

Atliq Hardwares

## Company's Market

| NA | LATAM | EU | APAC |
| --- | --- | --- | --- |
| North America | Latin America | European Union | 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:**

```sql
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
```

## 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:**

```sql
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 1000000;
```

## 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:**

```sql
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;
```

- **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:

```sql
CREATE PROCEDURE `get_monthly_gross_sales_for_customer`(
    in_customer_codes TEXT
)
BEGIN
    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
        FIND_IN_SET(s.customer_code, in_customer_codes) > 0
    GROUP BY s.date
    ORDER BY s.date DESC;
END
```

- **Query Result**

```
1  CREATE DEFINER=`root`@`localhost` PROCEDURE `get_monthly_gross_sales_for_customer`(
2              in_customer_codes TEXT
3      )
4  BEGIN
5              SELECT
6                      s.date,
7                                                                          ales
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** 9002002,90002008 [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

## 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:**

```sql
CREATE PROCEDURE `get_market_badge`(
    IN in_market VARCHAR(45),
    IN in_fiscal_year YEAR,
    OUT out_level VARCHAR(45)
)
BEGIN
    DECLARE qty INT DEFAULT 0;

    # Default market is India
    IF in_market = "" THEN
        SET in_market="India";
    END IF;

    # Retrieve total sold quantity for a given market in a given year
    SELECT
        SUM(s.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;

    # Determine Gold vs Silver status
    IF qty > 5000000 THEN
        SET out_level = 'Gold';
    ELSE
        SET out_level = 'Silver';
    END IF;
END
```

## Query Result

```
7       DECLARE qty INT DEFAULT 0;
8
9       # Default market is India
10      IF in_market = "" THEN
11
12      EN
13
14      #
15      SE
16
17      FR
18      JO
19      ON
20      WH
21
22
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
```

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:**

```sql
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 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;
```

## 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:**

```sql
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;
```

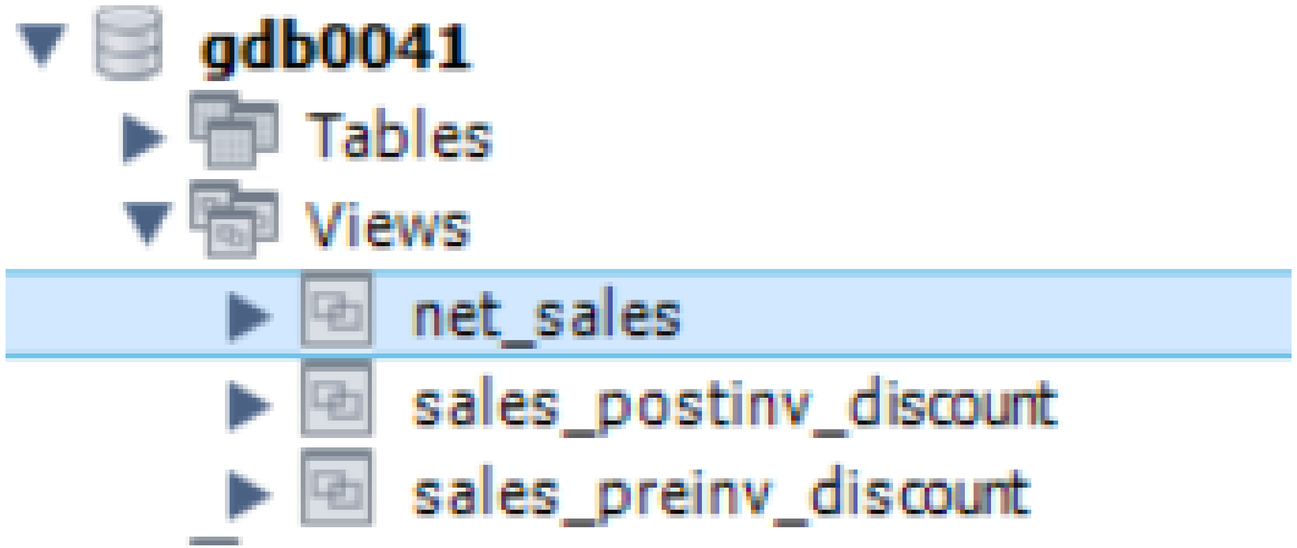
## 5.3. Creating the net_sales View:

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:**

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

- **Query Result**



## 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:**

new_procedure

The name of the routine is parsed automati
statement. The DDL is parsed automaticall

```sql
CREATE PROCEDURE `get_top_n_markets_by_net_sales`(
            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
```

## 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:

| 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:**

```sql
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
```

## Query Result

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_sales`(
            in_market VARCHAR(45),
            in_fiscal_year INT,
            in_top_n INT
    )
BEGIN

        order by net_sales_mln desc
        limit in_top_n;
    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 | india | [IN] | VARCHAR(45) |
| in_fiscal_year | 2021 | [IN] | INT |
| in_top_n | 5 | [IN] | INT |

Execute Cancel

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

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

| customer | net_sales_mln |
|---|---|
| Amazon | 30.00 |
| Atliq Exclusive | 23.98 |
| Flipkart | 12.96 |
| Electricalsocity | 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:**

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


CREATE PROCEDURE get_top_n_products_by_net_sales(
          in_fiscal_year int,
          in_top_n int
)
BEGIN
        select
             product,
             round(sum(net_sales)/1000000,2) as net_sales_mln
        from gdb0041.net_sales
        where fiscal_year=in_fiscal_year
        group by product
        order by net_sales_mln desc
        limit in_top_n;
END

```

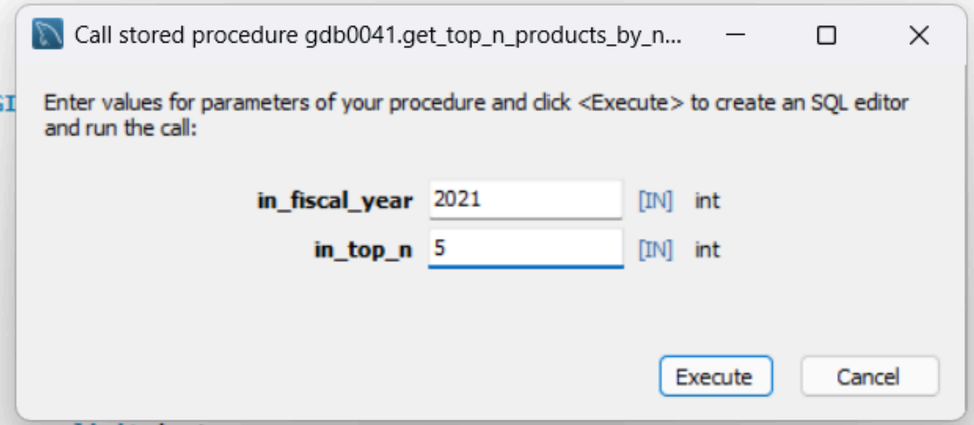
## Query Result

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


CREATE PROCEDURE get_top_n_products_by_net_sales(
        in_fiscal_year int,

)
BEGI

        limit in_top_n;
END
```




```sql
call gdb0041.get_top_n_products_by_net_sales(2021, 5);
```

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

| product | net_sales_mln |
|---|---|
| AQ BZ Allin1 | 33.75 |
| AQ Qwerty | 27.84 |
| AQ Trigger | 26.95 |
| AQ Gen Y | 23.58 |
| AQ Maxima | 22.32 |

## 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:**

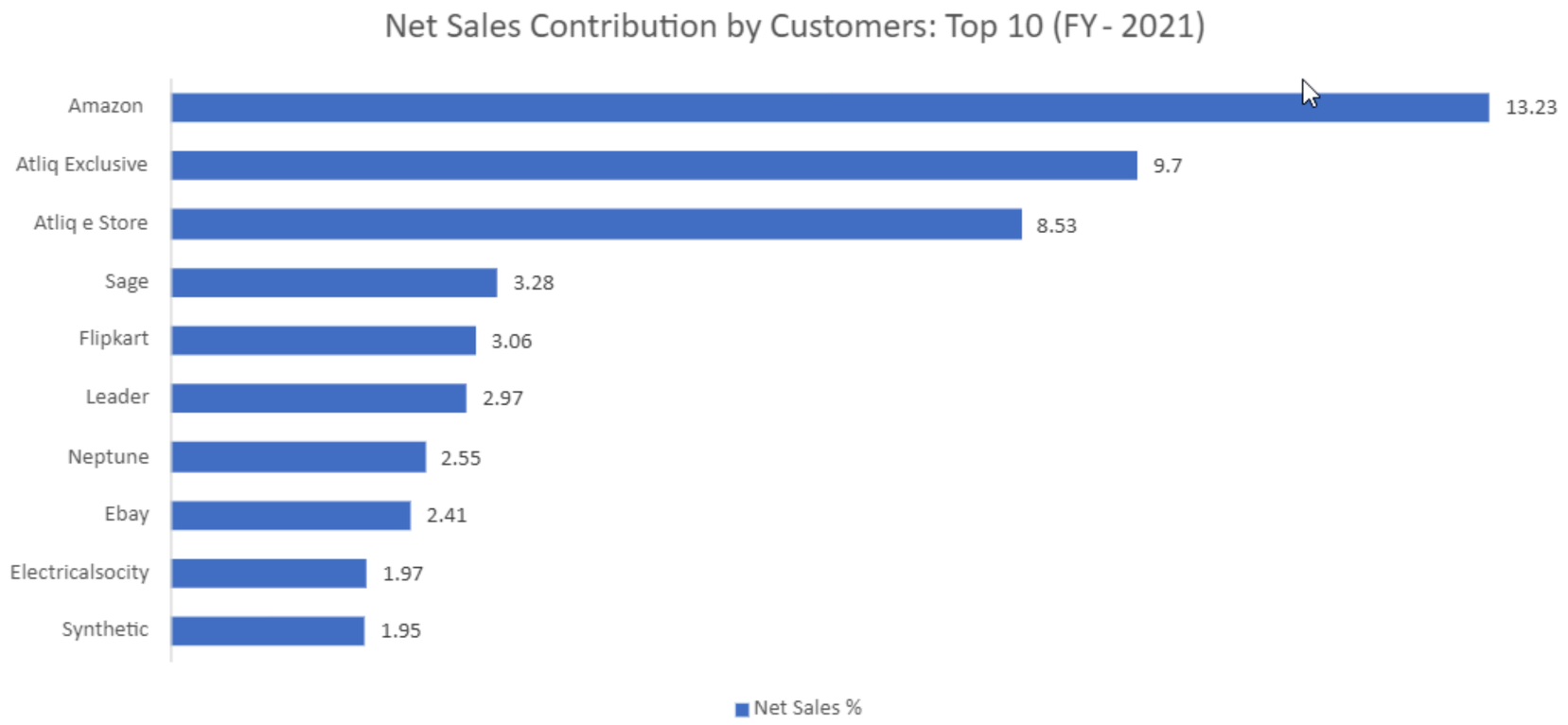
```sql
with cte1 as (
    select
        customer,
        round(sum(net_sales)/1000000,2) as net_sales_mln
    from net_sales s
    join dim_customer c
        on s.customer_code=c.customer_code
    where s.fiscal_year=2021
    group by customer)
select
    *,
    net_sales_mln*100/sum(net_sales_mln)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

- **Query Result**

Result Grid | Filter Rows: | Expo

| 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.

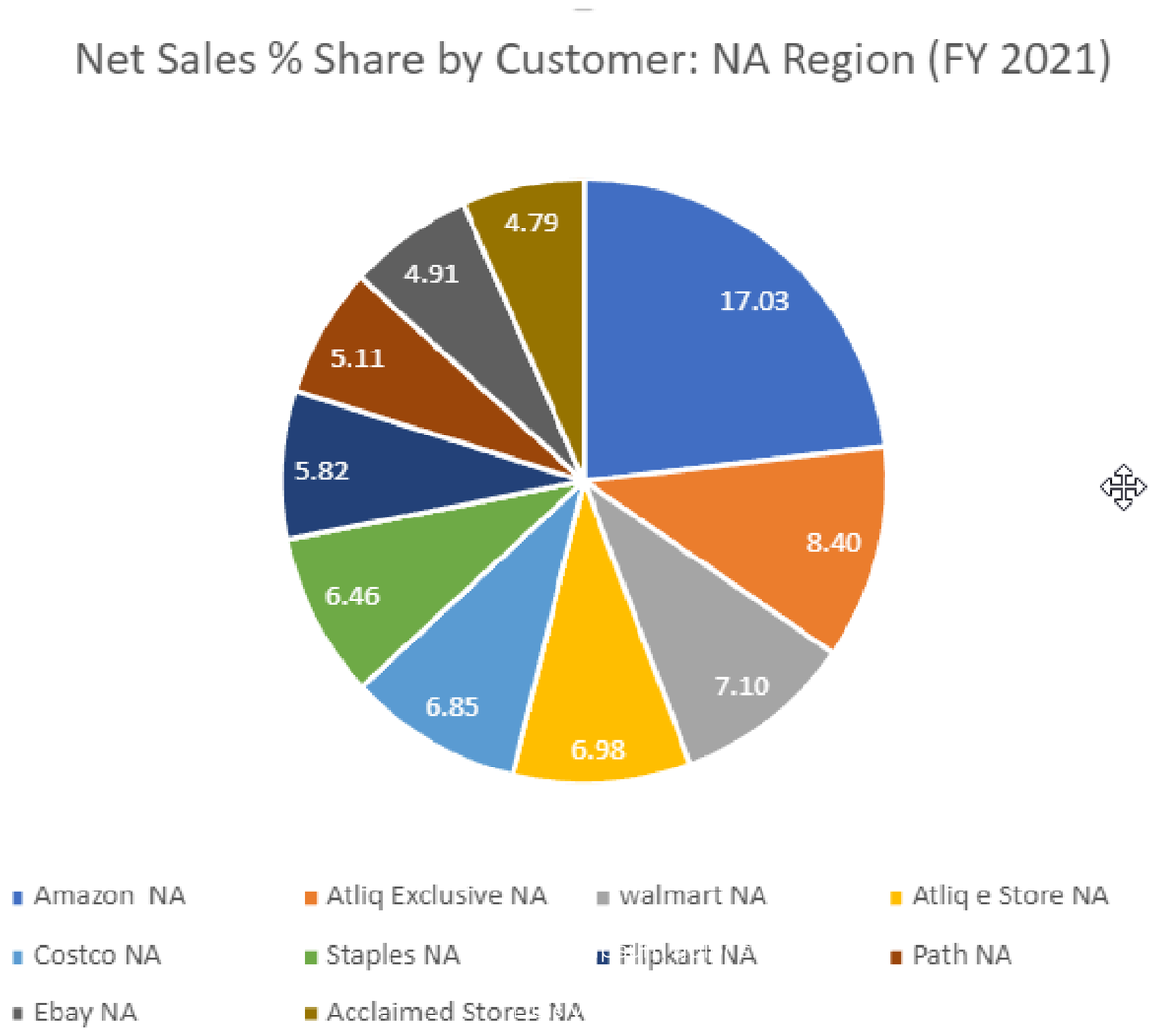
- **SQL Query:**

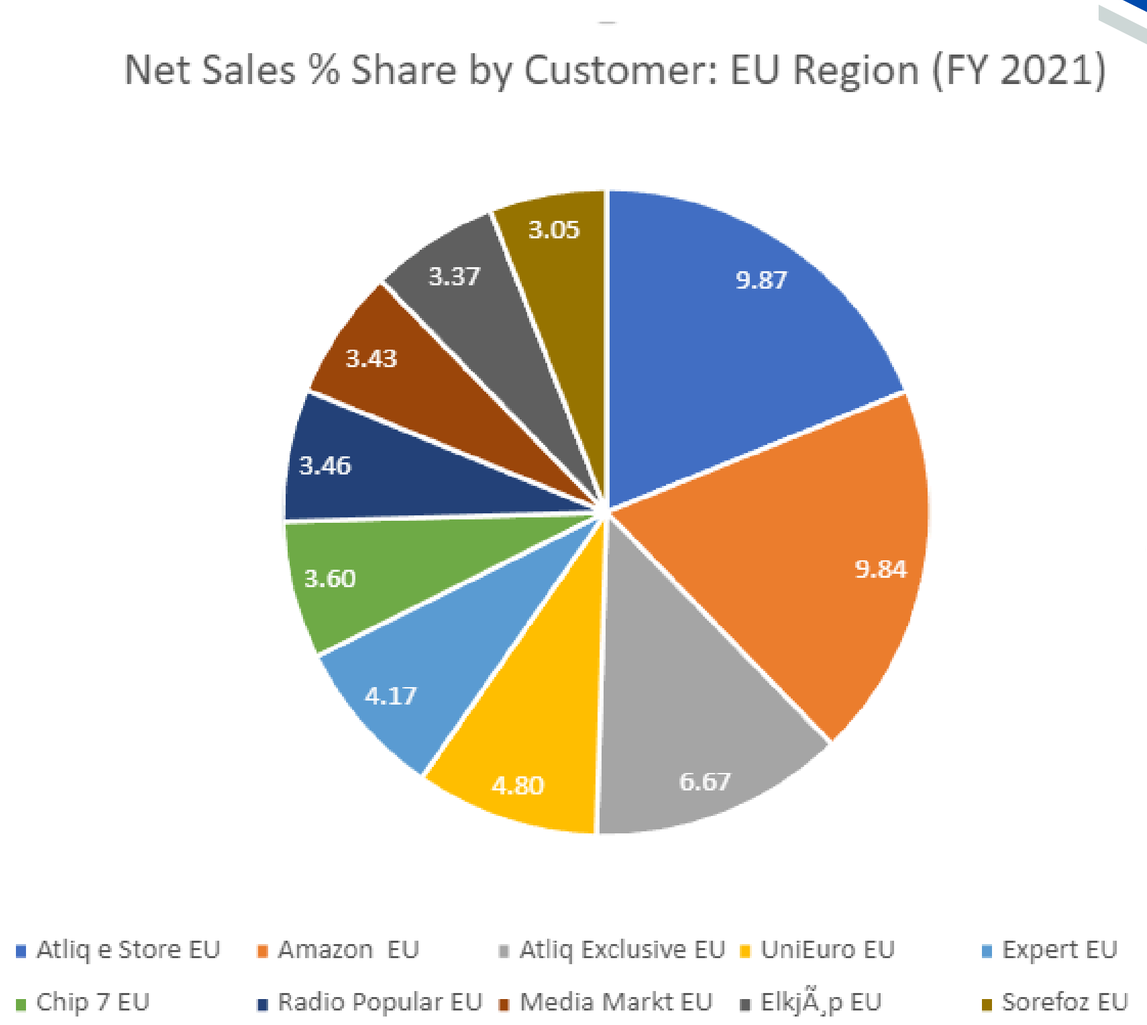
```sql
with cte1 as (
    select
        c.customer,
        c.region,
        round(sum(net_sales)/1000000,2) as net_sales_mln
    from gdb0041.net_sales n
    join dim_customer c
        on n.customer_code=c.customer_code
    where fiscal_year=2021
    group by c.customer, c.region)
select
    *,
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

- **Query Result**

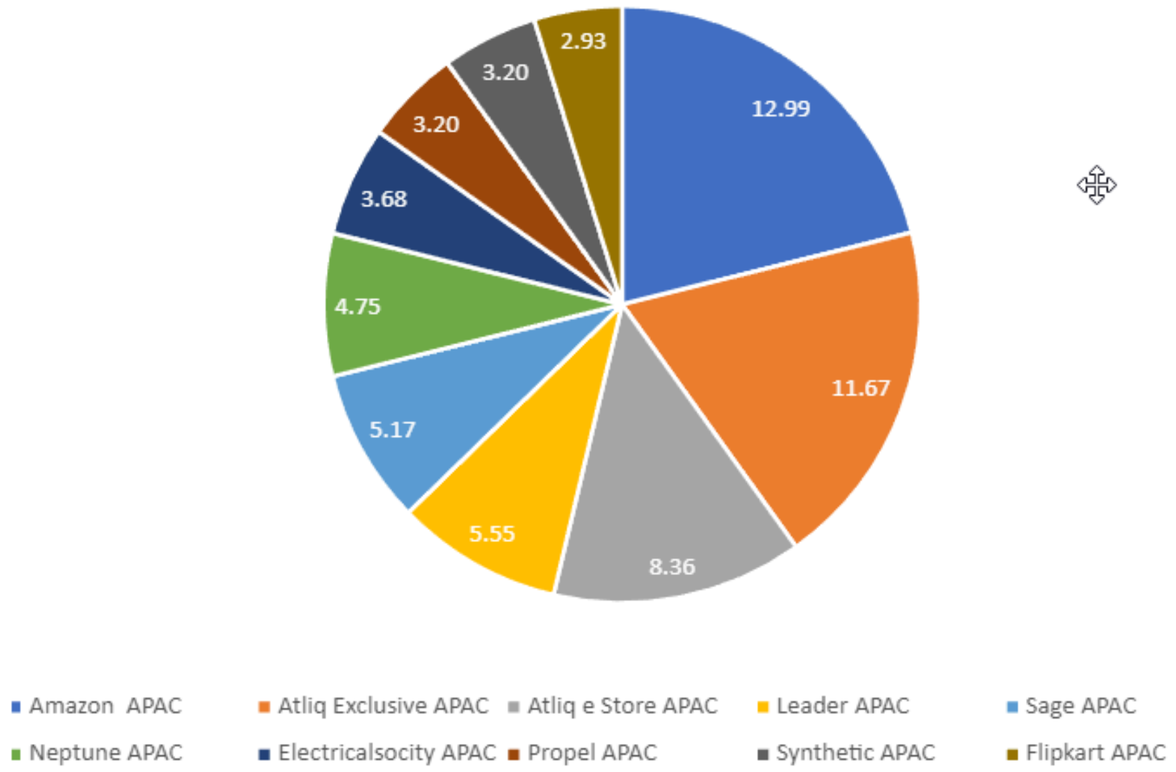
| customer | region | net_sales_mln | pct_share_region |
|---|---|---|---|
| Amazon | APAC | 57.41 | 12.988688 |
| Atliq Exclusive | APAC | 51.58 | 11.669683 |
| Atliq e Store | APAC | 36.97 | 8.364253 |
| Leader | APAC | 24.52 | 5.547511 |
| Sage | APAC | 22.85 | 5.169683 |
| Neptune | APAC | 21.01 | 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 |

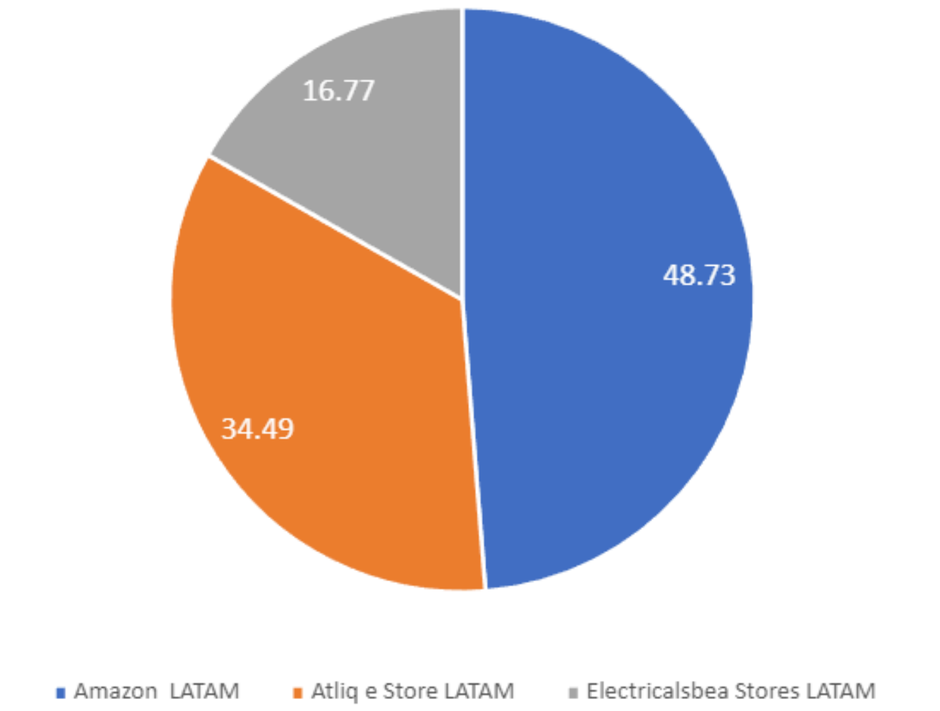
- FY2021: Regional Customer Sales Breakdown (Top 10)



- FY2021: Regional Customer Sales Breakdown (Top 10)



## 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 partitoning 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