

SQL Sales & Finance/Supply Chain Analytics

Business Model Explained

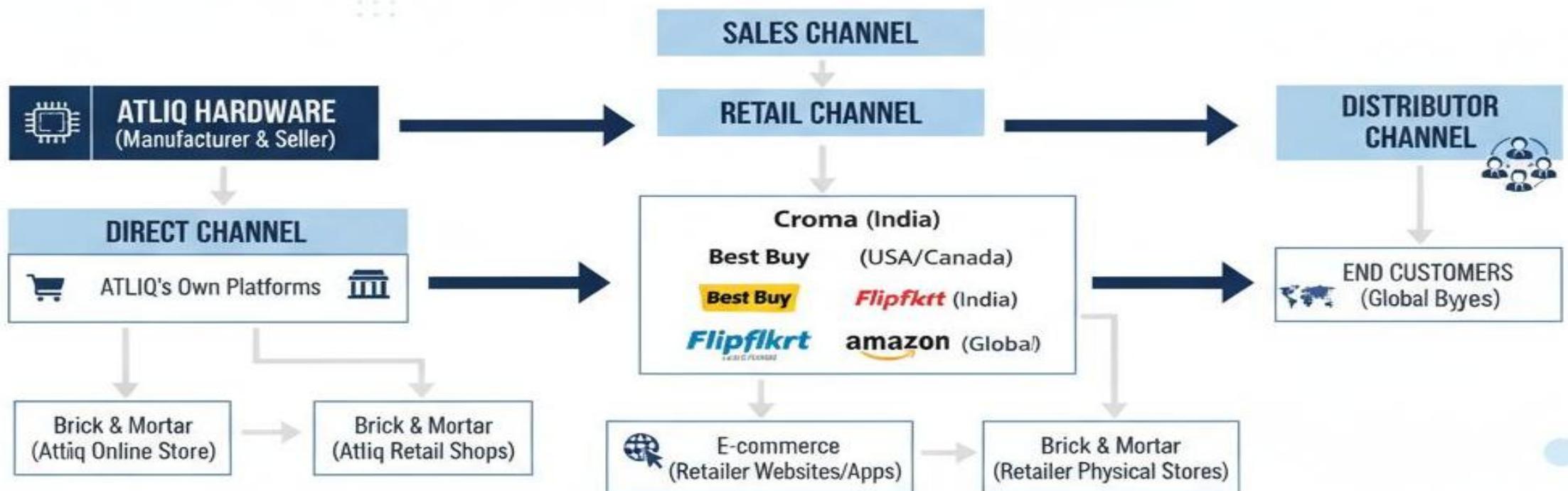
- Atliq Hardware is a leading hardware supplier that caters to multiple customer segments in various countries across **E-commerce and Brick & Mortar platforms.**

Their products are sold through three primary channels:

- **Retail** – Sold to major retail chains such as **Croma, Best Buy, Flipkart, and Amazon** etc.
- **Direct** – Customers can purchase directly from Atliq via online platforms and physical stores.
- **Distributor** – Products are supplied to authorized distributors who sell to secondary markets.

This multi-channel approach ensures **widespread market coverage**, allowing Atliq to serve both **end-consumers** and **B2B partners** across multiple regions and countries.

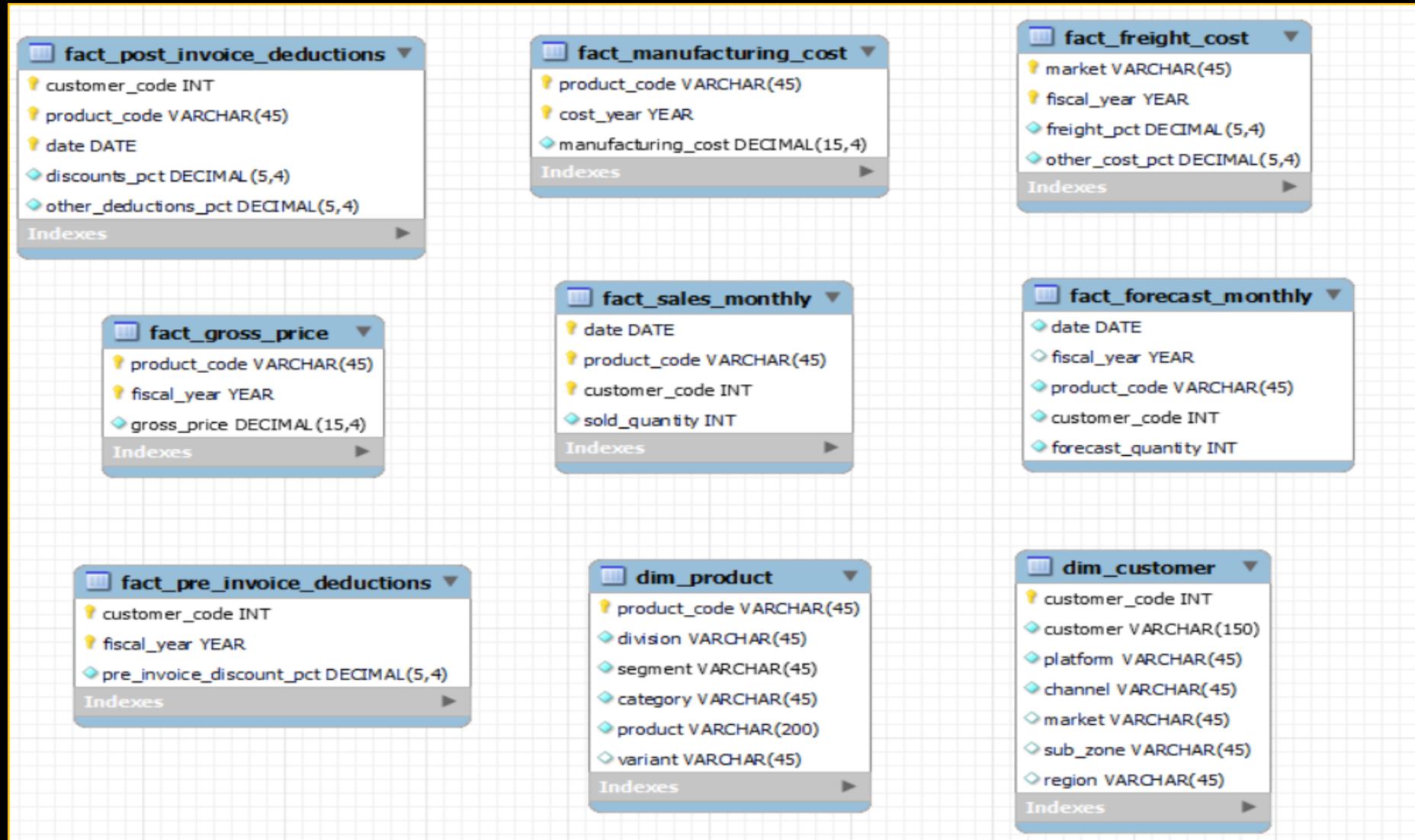
ATLIQ HARDWARE GLOBAL SALES & DISTRIBUTION MODEL



OBJECTIVE!

- Initially, all company sales reports were managed in **Excel**, but as data volume grew across multiple countries, products, and platforms, Excel could no longer handle the scale and complexity **with millions of records in a dataset**. Reports became slow, inconsistent, and hard to update.
- To overcome this, the company **migrated to a SQL-based data system MySQL** enabling faster, more accurate, and automated analysis. Using SQL, detailed insights were generated across **sales trends, product performance, customer segments, and supply chain flow**.

DATASET



1. As a product owner I want to generate a report of individual product sales (aggregated on a monthly basis at the product level) for Croma India customers for FY=2021 so that I can track individual product sales and run further product analytics on it in excel

Croma India Product-wise Sales Report for Fiscal Year 2021

Generate a report that includes the following fields:

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

SELECT

```
f.date,  
p.product_code,  
p.product,  
p.variant,  
f.sold_quantity,  
fg.gross_price,  
ROUND(fg.gross_price * f.sold_quantity, 2) as gross_price_total  
FROM fact_sales_monthly f  
JOIN dim_product p  
    ON f.product_code = p.product_code  
JOIN fact_gross_price fg  
    ON fg.product_code = f.product_code  
    AND fg.fiscal_year = get_fiscal_year(f.date)  
WHERE  
customer_code = 90002002  
AND get_fiscal_year(f.date) = 2021  
ORDER BY f.date;
```

get_fiscal_year() is user defined function

DESIRED OUTPUT

| date | product_code | product | variant | sold_quantity | gross_price | gross_price_total |
|------------|--------------|--------------------|-------------|---------------|-------------|-------------------|
| 2020-09-01 | A0118150101 | AQ Dracula HDD ... | Standard | 202 | 19.0573 | 3849.57 |
| 2020-09-01 | A0118150102 | AQ Dracula HDD ... | Plus | 162 | 21.4565 | 3475.95 |
| 2020-09-01 | A0118150103 | AQ Dracula HDD ... | Premium | 193 | 21.7795 | 4203.44 |
| 2020-09-01 | A0118150104 | AQ Dracula HDD ... | Premium ... | 146 | 22.9729 | 3354.04 |
| 2020-09-01 | A0219150201 | AQ WereWolf NA... | Standard | 149 | 23.6987 | 3531.11 |
| 2020-09-01 | A0219150202 | AQ WereWolf NA... | Plus | 107 | 24.7312 | 2646.24 |
| 2020-09-01 | A0220150203 | AQ WereWolf NA... | 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 |

2. As a product owner I need an aggregate monthly gross sales report for Croma India customer so that I can track how much sales this particular customer is generating for AtliQ and manage our relationships accordingly.

The report should have the following field

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

SELECT

```
f.date,  
ROUND(SUM(fg.gross_price * f.sold_quantity), 2) AS gross_total  
FROM fact_sales_monthly f  
JOIN fact_gross_price fg  
ON f.product_code = fg.product_code  
AND fg.fiscal_year = get_fiscal_year(f.date)  
WHERE customer_code = 90002002  
GROUP BY f.date;
```

DESIRED OUTPUT

The screenshot shows a database query results window with the following details:

Result Grid: A table titled "Result Grid" showing monthly gross sales. The columns are "date" and "gross_total". The data is as follows:

| date | gross_total |
|------------|-------------|
| 2017-09-01 | 122407.56 |
| 2017-10-01 | 162687.57 |
| 2017-12-01 | 245673.80 |
| 2018-01-01 | 127574.74 |
| 2018-02-01 | 144799.52 |
| 2018-04-01 | 130643.90 |
| 2018-05-01 | 139165.10 |
| 2018-06-01 | 125735.38 |
| 2018-08-01 | 125409.88 |
| 2018-09-01 | 343337.17 |
| 2018-10-01 | 440562.08 |

Output: Shows the query executed: "16 13:18:21 SELECT f.product_code,f.date,ROUND(SUM(fg.gross_price*f.sold_quantity),2) as gross_total FROM fact... 8184 row(s) returned".

Execution Plan: Shows the execution plan for the query.

3. Generate a yearly report for Croma India where there are two columns

-Fiscal Year

-Total Gross sales amount in that year from Croma

```
SELECT
```

```
    get_fiscal_year(date) AS fiscal_year,  
    SUM(ROUND(s.sold_quantity * g.gross_price, 2)) AS yearly_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 get_fiscal_year(date)
```

```
ORDER BY fiscal_year;
```

DESIRED OUTPUT

The screenshot shows a database interface with a results grid and a query history panel.

Result Grid:

| fiscal_year | yearly_sales |
|-------------|--------------|
| 2018 | 1324097.48 |
| 2019 | 3555079.19 |
| 2020 | 6502182.12 |
| 2021 | 23216512.73 |
| 2022 | 44638199.11 |

Output:

| # | Time | Action | Message |
|----|----------|--|-------------------|
| 25 | 13:53:05 | SELECT fg.fiscal_year, ROUND(SUM(fg.gross_price * f.sold_quantity), 2) AS gross_total FROM fact_s... | 5 row(s) returned |

Toolbar: Result Grid, Form Editor, Field Types, Query Stats, Read Only.

4. As a DA I want to create a stored proc for monthly gross sales report so that I don't have to manually modify the query every time.

The report should have following columns:

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

Stored proc can be run by other users to (who have limited access to database) and they can generate this report without having to involve data analytics team.

Just need to enter customer_code for which monthly gross sales is needed.

```
CREATE PROCEDURE get_monthly_gross_sales_for_customer (
    IN c_code INT
)
BEGIN
    SELECT
        f.date,
        ROUND(SUM(fg.gross_price * f.sold_quantity), 2) AS gross_total
    FROM fact_sales_monthly f
    JOIN fact_gross_price fg
        ON f.product_code = fg.product_code
        AND fg.fiscal_year = get_fiscal_year(f.date)
    WHERE customer_code = c_code
    GROUP BY f.date;
END;
```

The screenshot shows a database interface with a call dialog and a result grid.

Call Dialog:

- Header: Call stored procedure gdb0041.get_monthly_gross_sales...
- Text: Enter values for parameters of your procedure and click <Execute> to create an SQL editor and run the call:
- Parameter: c_code 90002002 [IN] INT
- Buttons: Execute, Cancel

Result Grid:

| date | gross_total |
|------------|-------------|
| 2017-09-01 | 122407.56 |
| 2017-10-01 | 162687.57 |
| 2017-12-01 | 245673.80 |
| 2018-01-01 | 127574.74 |
| 2018-02-01 | 144799.52 |
| 2018-04-01 | 130643.90 |
| 2018-05-01 | 139165.10 |
| 2018-06-01 | 125735.38 |
| 2018-08-01 | 125409.88 |
| 2018-09-01 | 343337.17 |
| 2018-10-01 | 440562.08 |
| 2018-12-01 | 653944.75 |

Action Output:

- # 29 17:10:51 Time Action
- Apply changes to get_monthly_gross_sales_for_customer

Message: Changes applied

5. Stored procedure for market badge

Create a store proc that can determine the market badge based on the following logic-

If total sold qty>5million that market is considered **Gold** else it is **Silver**

My input will be - Market , fiscal year

Output should be - Market badge

```
CREATE PROCEDURE get_market_badge (
    IN in_market VARCHAR(45),
    IN in_fiscal_year YEAR,
    OUT out_badge VARCHAR(45)
)
BEGIN
    DECLARE qty INT DEFAULT 0;
    IF in_market = '' THEN SET in_market = 'India'; END IF;

    SELECT SUM(sold_quantity) INTO qty
    FROM fact_sales_monthly f
    JOIN dim_customer c ON f.customer_code = c.customer_code
    WHERE get_fiscal_year(f.date) = in_fiscal_year
        AND c.market = in_market
    GROUP BY c.market;

    IF qty > 5000000 THEN
        SET out_badge = 'Gold';
    ELSE
        SET out_badge = 'Silver';
    END IF;
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 | India | [IN] varchar(45) |
| in_fiscal_year | 2021 | [IN] year |
| out_badge | | [OUT] varchar(45) |

Execute Cancel

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

Result Grid | Filter Rows | Export | Wrap Cell Contents | Result 1 | Output | Action Output | Message

| |
|------------|
| @out_badge |
| Gold |

| | | |
|----|----------|--|
| # | Time | Action |
| 40 | 17:57:45 | set @out_badge = '0' |
| 41 | 17:57:45 | call gdb0041.get_market_badge('India', 2021, @out_badge) |
| 42 | 17:58:00 | select @out_badge LIMIT 0, 500 |

0 row(s) affected
1 row(s) affected
1 row(s) returned

6. As a product owner I want a report for top markets , products , customers by net sales for a given financial year so that I can have a holistic view of our financial performance and can take appropriate actions to address any potential issues.

1. Optimized query performance - Used same code from previous query and replaced get_fiscal_year() function by adding a generated column fiscal_year in fact_sales_monthly table.

VIEW-Table -1

2. Created VIEWS to simplify query and in order to reuse same code to get required final report.

sales_preinv_discount

Gross_price – pre_invoice_deduction=Net_Invoice_sales

Net_Invoice_sales-post_invoice_deduction= Net_sales

| date | fiscal_yea | product_code | market | product | varian | sold_ | gross_price | customer_co | gross_price |
|--------|------------|--------------|-----------|-----------|--------|-------|-------------|-------------|-------------|
| 201... | 2018 | A0118150101 | India | AQ Dra... | Sta... | 51 | 15.3952 | 70002017 | 785.16 |
| 201... | 2018 | A0118150101 | India | AQ Dra... | Sta... | 77 | 15.3952 | 70002018 | 1185.43 |
| 201... | 2018 | A0118150101 | Indo... | AQ Dra... | Sta... | 17 | 15.3952 | 70003181 | 261.72 |
| 201... | 2018 | A0118150101 | Indo... | AQ Dra... | Sta... | 6 | 15.3952 | 70003182 | 92.37 |
| 201... | 2018 | A0118150101 | Philip... | AQ Dra... | Sta... | 5 | 15.3952 | 70006157 | 76.98 |
| 201... | 2018 | A0118150101 | Philip... | AQ Dra... | Sta... | 7 | 15.3952 | 70006158 | 107.77 |
| 201... | 2018 | A0118150101 | Sout... | AQ Dra... | Sta... | 29 | 15.3952 | 70007198 | 446.46 |
| 201... | 2018 | A0118150101 | Sout... | AQ Dra... | Sta... | 34 | 15.3952 | 70007199 | 523.44 |
| 201... | 2018 | A0118150101 | Austr... | AQ Dra... | Sta... | 22 | 15.3952 | 70008169 | 338.69 |

```
SELECT
    f.date,
    f.fiscal_year,
    p.product_code,
    c.market,
    p.product,
    p.variant,
    f.sold_quantity,
    fg.gross_price,
    f.customer_code,
    ROUND(fg.gross_price * f.sold_quantity, 2) AS gross_price_total
FROM fact_sales_monthly f
JOIN dim_customer c
    ON f.customer_code = c.customer_code
JOIN dim_product p
    ON f.product_code = p.product_code
JOIN fact_gross_price fg
    ON fg.product_code = f.product_code
    AND fg.fiscal_year = f.fiscal_year
JOIN fact_pre_invoice_deductions pre
    ON f.customer_code = pre.customer_code
    AND pre.fiscal_year = f.fiscal_year;
```

VIEW- Table -2

sales_postinv_discount

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;
```

1 • `SELECT* FROM sales_postinv_discount;`

| date | fiscal_year | customer_code | market | product_code | product | variant | sold_qty |
|------------|-------------|---------------|--------|--------------|----------------------|----------|----------|
| 2017-09-01 | 2018 | 90027207 | Brazil | A0118150101 | AQ Dracula HDD - ... | Standard | 4 |
| 2017-11-01 | 2018 | 90027207 | Brazil | A0118150101 | AQ Dracula HDD - ... | Standard | 16 |
| 2017-12-01 | 2018 | 90027207 | Brazil | A0118150101 | AQ Dracula HDD - ... | Standard | 4 |
| 2018-01-01 | 2018 | 90027207 | Brazil | A0118150101 | AQ Dracula HDD - ... | Standard | 6 |
| 2018-03-01 | 2018 | 90027207 | Brazil | A0118150101 | AQ Dracula HDD - ... | Standard | 9 |
| 2018-04-01 | 2018 | 90027207 | Brazil | A0118150101 | AQ Dracula HDD - ... | Standard | 6 |
| 2018-05-01 | 2018 | 90027207 | Brazil | A0118150101 | AQ Dracula HDD - ... | Standard | 7 |
| 2018-07-01 | 2018 | 90027207 | Brazil | A0118150101 | AQ Dracula HDD - ... | Standard | 10 |
| 2018-08-01 | 2018 | 90027207 | Brazil | A0118150101 | AQ Dracula HDD - ... | Standard | 6 |

sales_postinv_discount: 1000 row(s) returned

| variant | sold_qty | gross_price_total | pre_invoice_discount_pct | net_invoice_sales | post_invoice_discount_pct |
|----------|----------|-------------------|--------------------------|-------------------|---------------------------|
| Standard | 4 | 61.58 | 0.2803 | 44.319126 | 0.3905 |
| Standard | 16 | 246.32 | 0.2803 | 177.276504 | 0.4139 |
| Standard | 4 | 61.58 | 0.2803 | 44.319126 | 0.3295 |
| Standard | 6 | 92.37 | 0.2803 | 66.478689 | 0.3244 |
| Standard | 9 | 138.56 | 0.2803 | 99.721632 | 0.3766 |
| Standard | 6 | 92.37 | 0.2803 | 66.478689 | 0.3615 |
| Standard | 7 | 107.77 | 0.2803 | 77.562069 | 0.3173 |
| Standard | 10 | 153.95 | 0.2803 | 110.797815 | 0.3501 |
| Standard | 6 | 92.37 | 0.2803 | 66.478689 | 0.3740 |

We have two views — `sales_preinv_discount` and `sales_postinv_discount` now we can easily calculate the **Net Sales**, and create a consolidated view to represent it.

The **Net Sales** column has now been derived. We'll convert this query into a **view**, enabling us to generate reports that identify the **top markets, products, and customers by net sales** for a given fiscal year.

```
1 •  SELECT*,  
2   net_invoice_sales*(1-post_invoice_discount_pct) as net_sales  
3   FROM sales_postinv_discount;
```

The screenshot shows a database query results interface. At the top, there is a code editor window containing the SQL query. Below it is a result grid table with the following data:

| # | gross_price_total | pre_invoice_discount_pct | net_invoice_sales | post_invoice_discount_pct | net_sales |
|---|-------------------|--------------------------|-------------------|---------------------------|----------------|
| 1 | 61.58 | 0.2803 | 44.319126 | 0.3905 | 27.0125072970 |
| 2 | 246.32 | 0.2803 | 177.276504 | 0.4139 | 103.9017589944 |
| 3 | 61.58 | 0.2803 | 44.319126 | 0.3295 | 29.7159739830 |
| 4 | 92.37 | 0.2803 | 66.478689 | 0.3244 | 44.9130022884 |
| 5 | 138.56 | 0.2803 | 99.721632 | 0.3766 | 62.1664653888 |
| 6 | 92.37 | 0.2803 | 66.478689 | 0.3615 | 42.4466429265 |
| 7 | 107.77 | 0.2803 | 77.562069 | 0.3173 | 52.9516245063 |
| 8 | 153.95 | 0.2803 | 110.797815 | 0.3501 | 72.0074999685 |
| 9 | 92.37 | 0.2803 | 66.478689 | 0.3740 | 41.6156593140 |

On the right side of the interface, there is a sidebar with several tabs: Result Grid (which is selected), Form Editor, Field Types, Query Stats, and Execution Plan.

At the bottom left, there is a "Result 6" tab and an "Output" section. The output section shows two rows of log information:

| # | Time | Action | Message | Duration / Fetch |
|---|----------|--|----------------------|-----------------------|
| 7 | 12:51:43 | SELECT*. (net_invoice_sales*net_invoice_sales*post_invoice_discount_pct) as net_sales FROM sales_... | 1000 row(s) returned | 0.015 sec / 0.047 sec |
| 8 | 12:54:55 | SELECT*. net_invoice_sales*(1-post_invoice_discount_pct) as net_sales FROM sales_postinv_discoun... | 1000 row(s) returned | 0.000 sec / 0.062 sec |

Net Sales view is now ready!

Navigator: fact_sales_monthly net_sales

SCHEMAS: Filter objects

gdb0041

- Tables: dim_customer, dim_product, fact_forecast_monthly, fact_freight_cost, fact_gross_price, fact_manufacturing_cost, fact_post_invoice_deductions, fact_pre_invoice_deductions, fact_sales_monthly
- Views: net_sales, sales_postinv_discount, sales_preinv_discount
- Stored Procedures: get_market_badge, get_monthly_gross_sales_for_all
- Functions: f0 get_fiscal_quarter, f0 get fiscal year

Administration Schemas Information

View: net_sales

Columns:

| date | fiscal_y | customer_cc | market | product_c | product | varia | sold_quantity | gross_price_total | pre_invoic |
|--------|----------|-------------|--------|-----------|-----------|-------|---------------|-------------------|------------|
| 201... | 2018 | 90027207 | Brazil | A01181... | AQ Dra... | St... | 4 | 61.58 | 0.2803 |
| 201... | 2018 | 90027207 | Brazil | A01181... | AQ Dra... | St... | 16 | 246.32 | 0.2803 |
| 201... | 2018 | 90027207 | Brazil | A01181... | AQ Dra... | St... | 4 | 61.58 | 0.2803 |
| 201... | 2018 | 90027207 | Brazil | A01181... | AQ Dra... | St... | 6 | 92.37 | 0.2803 |
| 201... | 2018 | 90027207 | Brazil | A01181... | AQ Dra... | St... | 9 | 138.56 | 0.2803 |
| 201... | 2018 | 90027207 | Brazil | A01181... | AQ Dra... | St... | 6 | 92.37 | 0.2803 |
| 201... | 2018 | 90027207 | Brazil | A01181... | AQ Dra... | St... | 7 | 107.77 | 0.2803 |
| 201... | 2018 | 90027207 | Brazil | A01181... | AQ Dra... | St... | 10 | 153.95 | 0.2803 |
| 201... | 2018 | 90027207 | Brazil | A01181... | AQ Dra... | St... | 6 | 92.37 | 0.2803 |

net_sales 4 x

Output: Action Output # Time Action

1 13:10:27 SELECT * FROM gdb0041.net_sales LIMIT 0, 50

Message: 50 row(s) returned Duration / Fetch: 0.000 sec / 0.000 sec

Object Info Session

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

| sold_quantity | gross_price_total | pre_invoice_discount_pct | net_invoice_sales | post_invoice_discount_pct | net_sales |
|---------------|-------------------|--------------------------|-------------------|---------------------------|-----------|
| 61.58 | 0.2803 | 44.319126 | 0.3905 | 27.0125072970 | |
| 246.32 | 0.2803 | 177.276504 | 0.4139 | 103.9017589944 | |
| 61.58 | 0.2803 | 44.319126 | 0.3295 | 29.7159739830 | |
| 92.37 | 0.2803 | 66.478689 | 0.3244 | 44.9130022884 | |
| 138.56 | 0.2803 | 99.721632 | 0.3766 | 62.1664653888 | |
| 92.37 | 0.2803 | 66.478689 | 0.3615 | 42.4466429265 | |
| 107.77 | 0.2803 | 77.562069 | 0.3173 | 52.9516245063 | |
| 153.95 | 0.2803 | 110.797815 | 0.3501 | 72.0074999685 | |
| 92.37 | 0.2803 | 66.478689 | 0.3740 | 41.6156593140 | |

net_sales 4 x

Output: Action Output # Time Action

1 13:10:27 SELECT * FROM gdb0041.net_sales LIMIT 0, 50

Message: 50 row(s) returned Duration / Fetch: 0.000 sec / 0.000 sec

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

TOP 5 MARKETS BY NET SALES(in millions) IN FISCAL YEAR 2021

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

The screenshot shows a database query results interface. At the top, there are several buttons: 'Result Grid' (highlighted in blue), 'Filter Rows...', 'Export...', 'Wrap Cell Content...', 'Fetch rows...', and a 'Read Only' button. To the right of the grid, there is a vertical toolbar with icons for 'Result Grid', 'Form Editor', 'Field Types', 'Query Stats', and a dropdown menu.

| market | Net_sales_in_mln |
|------------|------------------|
| India | 210.67 |
| USA | 132.05 |
| South ... | 64.01 |
| Canada | 45.89 |
| United ... | 44.73 |

Below the table, there is a message bar: 'Result 8 x' and 'Output'. Under 'Output', there is a dropdown labeled 'Action Output' and a table with one row:

| # | Time | Action | Message | Duration / Fetch |
|---|----------|--|-------------------|-----------------------|
| 1 | 16:11:29 | SELECT market, ROUND(SUM(net_sales/1000000),2) as Net_sales_in_mln FROM net_sales WHERE f... | 5 row(s) returned | 7.059 sec / 0.000 sec |

TOP 5 CUSTOMERS BY NET SALES(in millions) IN FISCAL YEAR 2021

```
SELECT
    customer,
    ROUND(SUM(net_sales / 1000000), 2) AS Net_sales_in_mln
FROM net_sales n
JOIN dim_customer c
    ON n.customer_code = c.customer_code
WHERE fiscal_year = 2021
GROUP BY customer
ORDER BY Net_sales_in_mln DESC
LIMIT 5;
```

The screenshot shows a database query results interface. At the top, there are several buttons: 'Result Grid' (highlighted in blue), 'Filter Rows:', 'Export:', 'Wrap Cell Content:', 'Fetch rows:', and a refresh icon. On the right side, there is a vertical toolbar with icons for 'Result Grid' (highlighted in blue), 'Form Editor', 'Field Types', and 'Query Stats'. Below the toolbar, the results are displayed in a table titled 'Result Grid'. The table has two columns: 'customer' and 'Net_sales_in_mln'. The data is as follows:

| customer | Net_sales_in_mln |
|-----------------|------------------|
| Amazon | 109.03 |
| Atliq Exclusive | 79.92 |
| Atliq e Store | 70.31 |
| Sage | 27.07 |
| Flipkart | 25.25 |

Below the table, there is a message bar that says 'Result 10 x' and 'Output'. Underneath the message bar, there is an 'Action Output' section with a table showing the execution details. The table has columns '#', 'Time', 'Action', 'Message', and 'Duration / Fetch'. The log entry is:

| # | Time | Action | Message | Duration / Fetch |
|---|----------|---|-------------------|-----------------------|
| 1 | 16:51:22 | SELECT customer, ROUND(SUM(net_sales/1000000),2) as Net_sales_in_mln FROM net_sales n JOIN dim_customer c ON n.customer_code = c.customer_code WHERE fiscal_year = 2021 GROUP BY customer ORDER BY Net_sales_in_mln DESC LIMIT 5; | 5 row(s) returned | 8.094 sec / 0.000 sec |

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

TOP 5 PRODUCTS BY NET SALES(in millions) IN FISCAL YEAR 2021

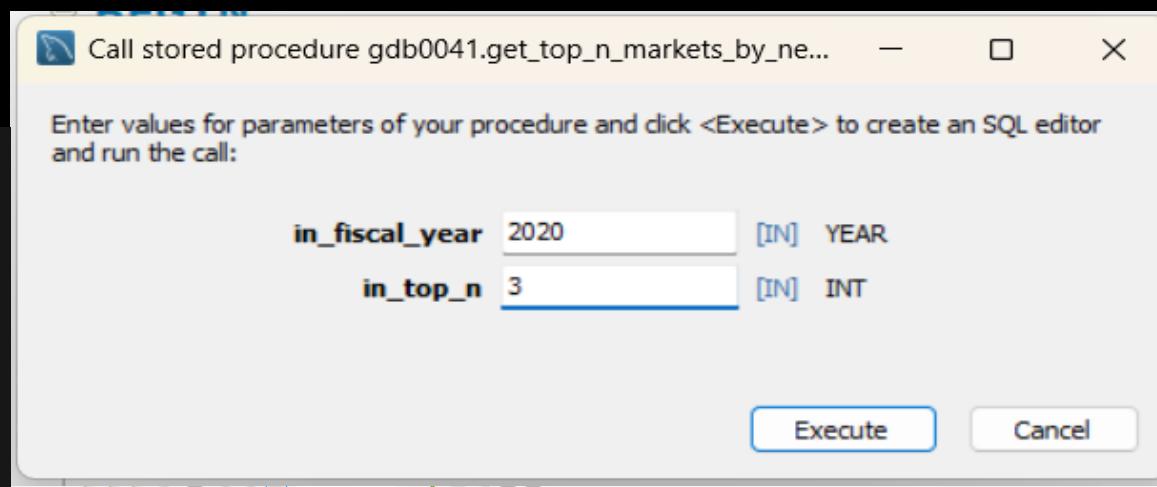
The screenshot shows a database query results interface. At the top, there are several buttons: 'Result Grid' (highlighted in blue), 'Filter Rows', 'Export', 'Wrap Cell Content', 'Fetch rows', and a 'Result Grid' icon on the right. Below this is a table with the following data:

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

Below the table, a message bar says 'Result 14 x'. Underneath the table, there's an 'Output' section with a dropdown set to 'Action Output'. A log table shows one entry: '# Time Action Message Duration / Fetch' with row 1 at '17:28:35' and the query details. The bottom right corner has a 'Read Only' button.

7. Automated report - Stored procedure for Top N markets by Net sales in any fiscal year

```
CREATE PROCEDURE `get_top_n_markets_by_net_sales` (
    IN in_fiscal_year INT,
    IN in_top_n INT
)
BEGIN
    SELECT
        market,
        ROUND(SUM(net_sales / 1000000), 2) AS Net_sales_in_mln
    FROM net_sales
    WHERE fiscal_year = in_fiscal_year
    GROUP BY market
    ORDER BY Net_sales_in_mln DESC
    LIMIT in_top_n;
END;
```



```
1 • call gdb0041.get_top_n_markets_by_net_sales(2020, 3);
2 |
```

| market | Net_sales_in_mln |
|-----------|------------------|
| India | 64.73 |
| USA | 46.35 |
| South ... | 22.38 |

| Result 2 | | |
|---------------|----------|--|
| Output | | |
| Action Output | | |
| # | Time | Action |
| 4 | 16:28:51 | Apply changes to get_top_n_markets_by_net_sales |
| 5 | 16:29:04 | call gdb0041.get_top_n_markets_by_net_sales(2020, 3) |
| | | Changes applied |
| | | 3 row(s) returned |
| | | Duration / Fetch |
| | | 6.500 sec / 0.000 sec |

8. Automated report - Stored procedure for Top N Customers by Net sales in given fiscal year

```
CREATE PROCEDURE `get_top_n_customers_by_net_sales` (
    IN in_fiscal_year INT,
    IN in_top_n INT
)
BEGIN
    SELECT
        customer,
        ROUND(SUM(net_sales / 1000000), 2) AS Net_sales_in_mln
    FROM net_sales n
    JOIN dim_customer c
        ON n.customer_code = c.customer_code
    WHERE fiscal_year = in_fiscal_year
    GROUP BY customer
    ORDER BY Net_sales_in_mln DESC
    LIMIT in_top_n;
END;
```

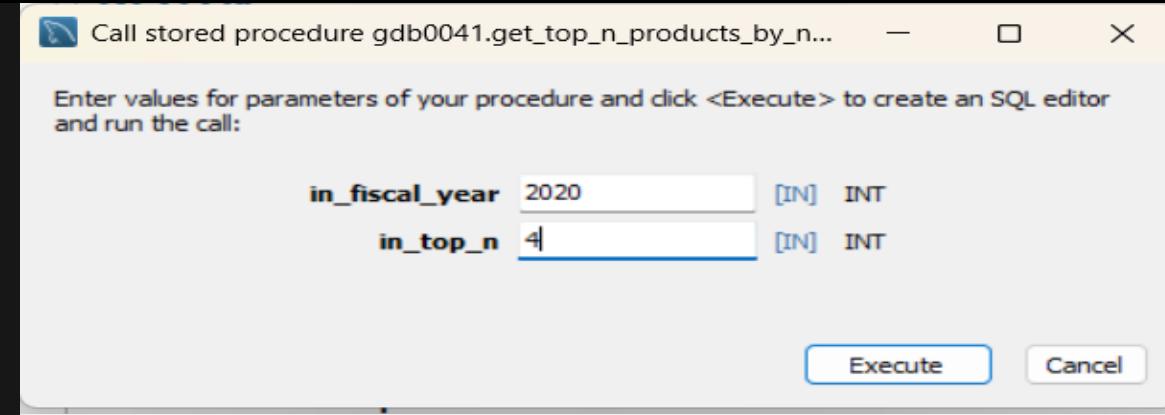
The screenshot shows a database interface with a modal dialog for calling a stored procedure. The dialog title is "Call stored procedure gdb0041.get_top_n_customers_by_net_sales". It contains two input fields: "in_fiscal_year" set to 2020 and "in_top_n" set to 5. Below the inputs is a SQL query: "call gdb0041.get_top_n_customers_by_net_sales(2020, 5);". There are "Execute" and "Cancel" buttons. The main window shows the results of the query in a grid:

| customer | Net_sales_in_mln |
|-----------------|------------------|
| Amazon | 49.77 |
| Atliq e Store | 31.74 |
| Atliq Exclusive | 22.97 |
| Flipkart | 10.92 |
| Sage | 8.31 |

The status bar at the bottom indicates the operation was successful: "Changes applied" and "5 rows returned".

9. Automated report - Stored procedure for Top N products by Net sales in any fiscal year

```
CREATE PROCEDURE `get_top_n_products_by_net_sales` (
    IN in_fiscal_year INT,
    IN in_top_n INT
)
BEGIN
    SELECT
        product,
        ROUND(SUM(net_sales / 1000000), 2) AS Net_sales_in_mln
    FROM net_sales
    WHERE fiscal_year = in_fiscal_year
    GROUP BY product
    ORDER BY Net_sales_in_mln DESC
    LIMIT in_top_n;
END;
```



The screenshot shows the results of the stored procedure execution. The "Result Grid" displays the following data:

| product | Net_sales_in_mln |
|-----------------|------------------|
| AQ Wi Power Dx2 | 14.37 |
| AQ BZ Gen Y | 12.09 |
| AQ Wi Power Dx1 | 11.84 |
| AQ Lite | 11.55 |

The "Result 1" tab shows the SQL command: "call gdb0041.get_top_n_products_by_net_sales(2020, 4);". The "Action Output" tab shows the log entries:

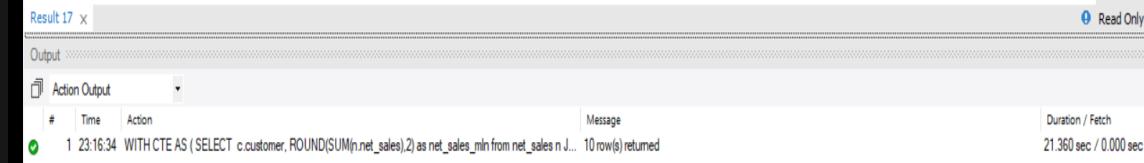
| # | Time | Action | Message | Duration / Fetch |
|---|----------|---|-------------------|------------------------|
| 6 | 17:56:24 | Apply changes to get_top_n_products_by_net_sales | Changes applied | |
| 7 | 17:57:42 | call gdb0041.get_top_n_products_by_net_sales(2020, 4) | 4 row(s) returned | 16.328 sec / 0.000 sec |

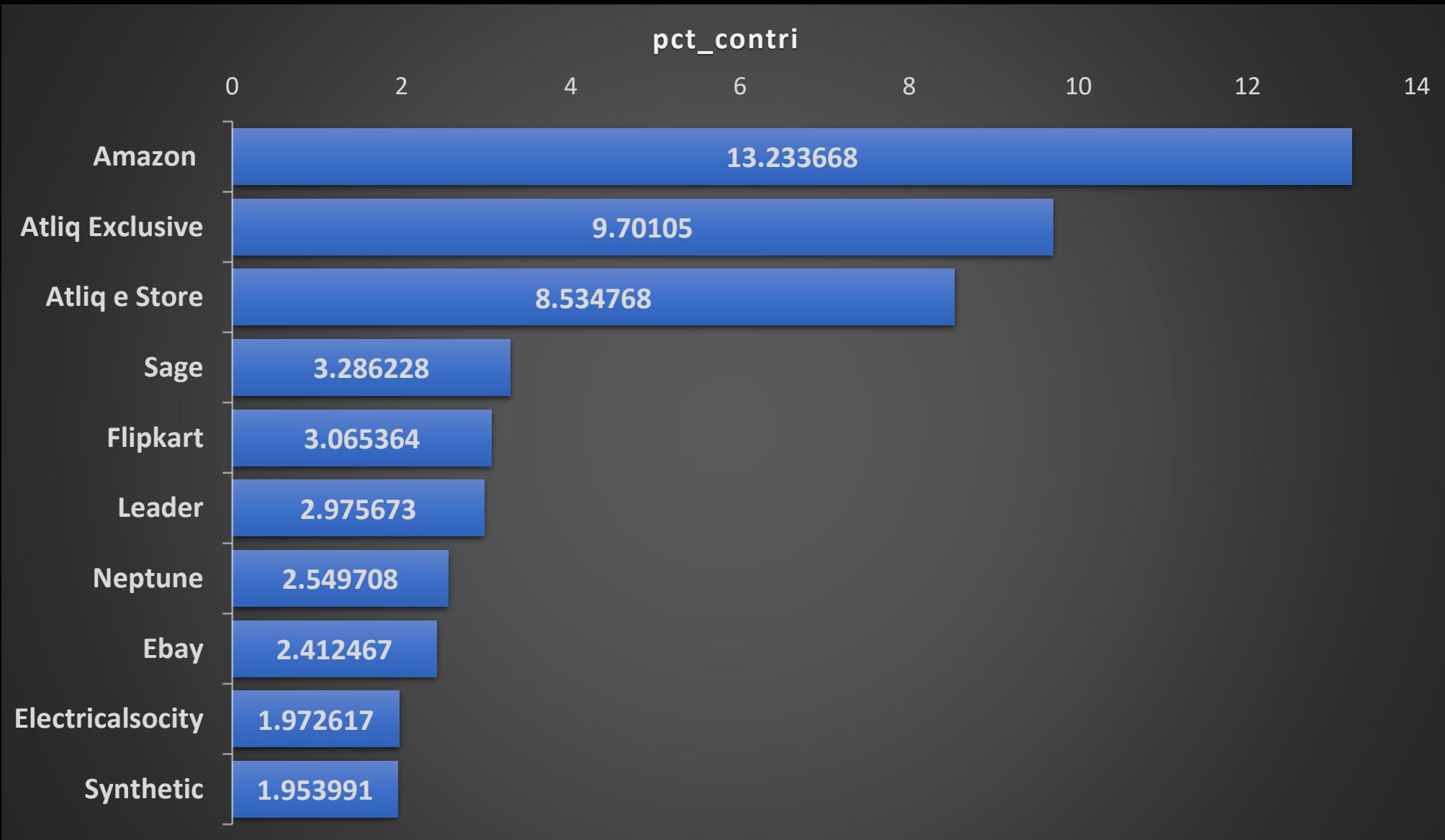
10. As a product owner, I want to see a bar chart that shows the top 10 markets with the highest percentage of net sales for the financial year 2021. The chart should display each market and its percentage share of total net sales.

```
WITH CTE AS (
    SELECT
        c.customer,
        ROUND(SUM(n.net_sales), 2) AS net_sales_mln
    FROM net_sales n
    JOIN dim_customer c
        ON n.customer_code = c.customer_code
    WHERE fiscal_year = 2021
    GROUP BY customer
    ORDER BY net_sales_mln DESC
)
SELECT *,
    (net_sales_mln * 100) / SUM(net_sales_mln) OVER() AS pct_contri
FROM CTE
LIMIT 10;
```

Will export this file to excel to create a bar chart

| customer | net_sales_mln | pct_contri |
|------------------|---------------|------------|
| Amazon | 109025241.97 | 13.233668 |
| Atliq Exclusive | 79921855.88 | 9.701050 |
| Atliq e Store | 70313473.20 | 8.534768 |
| Sage | 27073502.27 | 3.286228 |
| Flipkart | 25253923.33 | 3.065364 |
| Leader | 24515002.60 | 2.975673 |
| Neptune | 21005706.42 | 2.549708 |
| Ebay | 19875050.71 | 2.412467 |
| Electricalsocity | 16251353.05 | 1.972617 |
| Synthetic | 16097907.62 | 1.953991 |



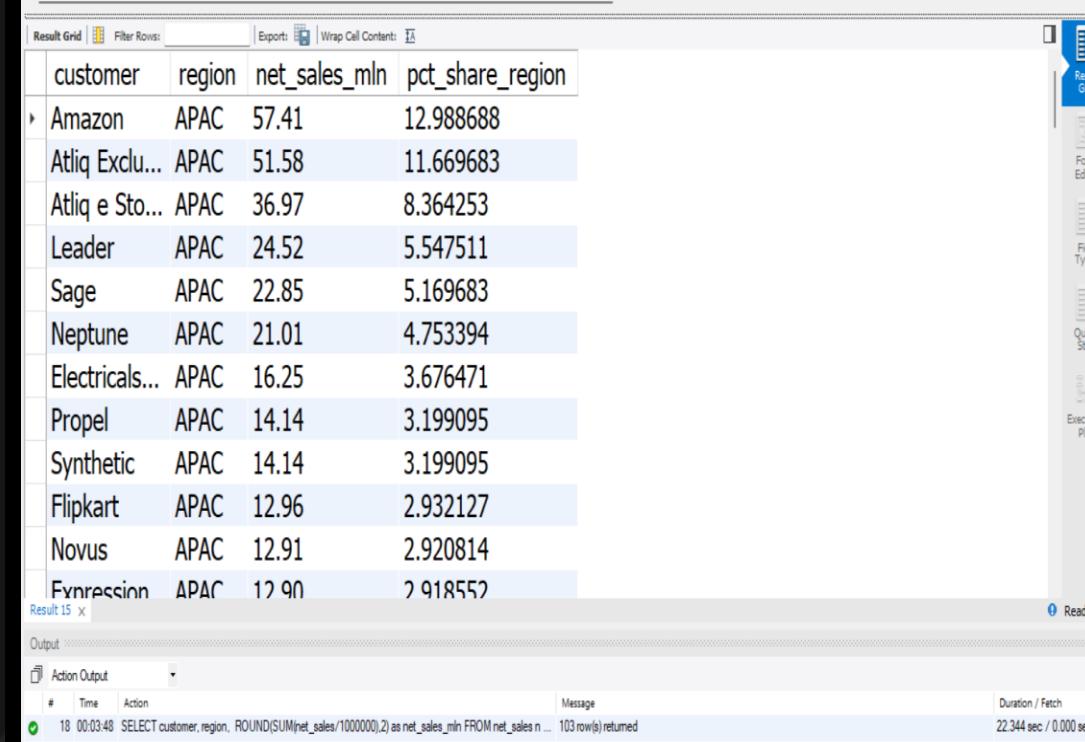


11. As a product owner, I want to see region wise (APAC, EU, LATAM etc) % net sales breakdown by customers in a respective region so that I can perform my regional analysis on financial performance of the company. The end results should be charts in pie format for FY =2021 .

Build a reusable asset that we can use to conduct analysis for any financial year.

```
WITH CTE AS (
    SELECT
        customer,
        region,
        ROUND(SUM(net_sales / 1000000), 2) AS net_sales_mln
    FROM net_sales n
    JOIN dim_customer c
        ON c.customer_code = n.customer_code
    WHERE fiscal_year = 2021
    GROUP BY customer, region
    ORDER BY net_sales_mln DESC
)
SELECT *,
    net_sales_mln * 100 / SUM(net_sales_mln) OVER(PARTITION BY region) AS pct_share_region
FROM CTE
ORDER BY region, net_sales_mln DESC;
```

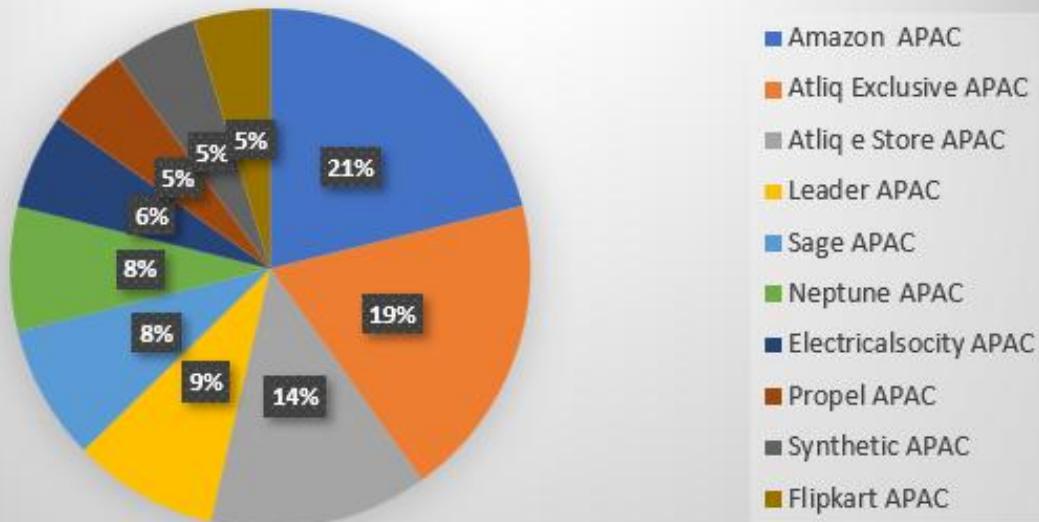
Will export this file to excel to create a pie chart



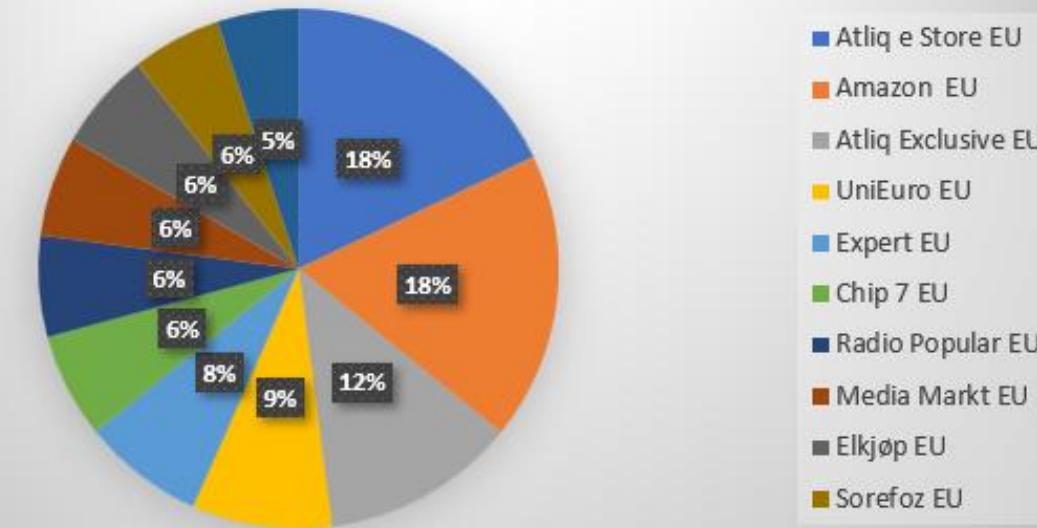
The screenshot shows a database query results grid with the following data:

| customer | region | net_sales_mln | pct_share_region |
|----------------|--------|---------------|------------------|
| Amazon | APAC | 57.41 | 12.988688 |
| Atliq Exclu... | APAC | 51.58 | 11.669683 |
| Atliq e Sto... | APAC | 36.97 | 8.364253 |
| Leader | APAC | 24.52 | 5.547511 |
| Sage | APAC | 22.85 | 5.169683 |
| Neptune | APAC | 21.01 | 4.753394 |
| Electricals... | 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 |
| Fynession | APAC | 12.90 | 2.918552 |

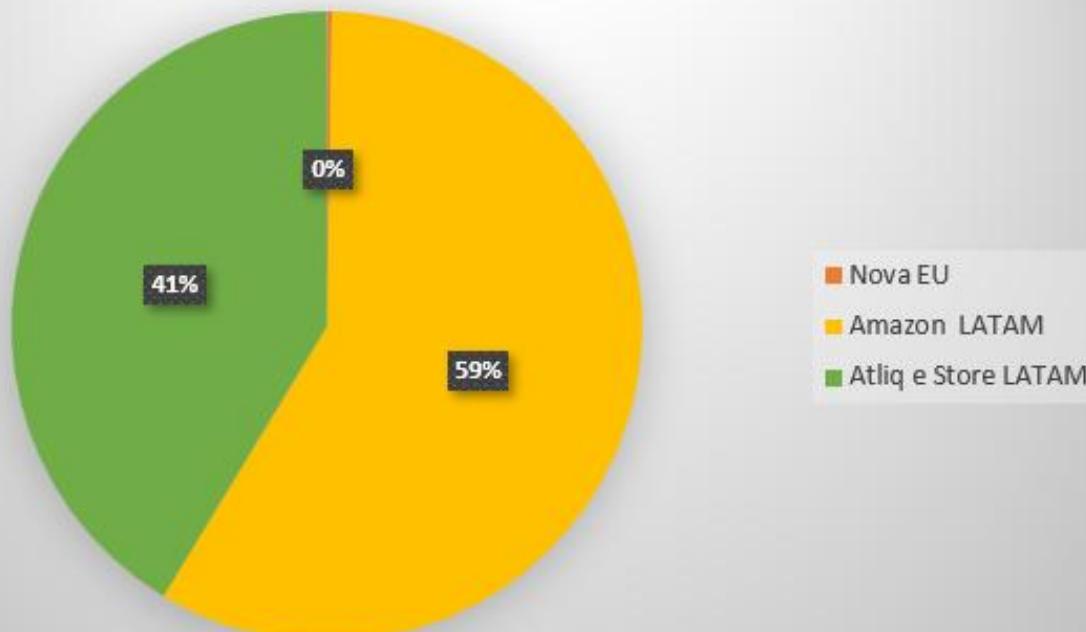
APAC Market Share : Net Sales



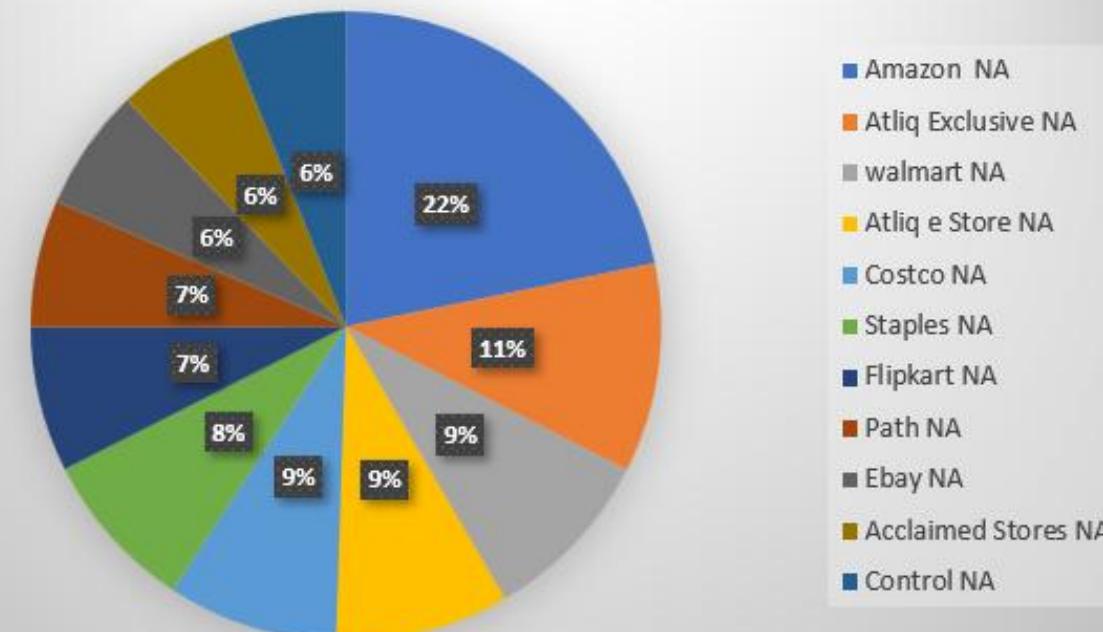
EU Market Share: Net Sales



LATAM Market Share : Net Sales



NA Market Share : Net Sales



12. Retrieve the top 2 markets in every region by their gross sales amount in FY=2021.

```
WITH CTE AS (
    SELECT
        c.market,
        c.region,
        ROUND(SUM(g.gross_price_total / 1000000), 2) AS gross_sales_mln
    FROM gross_sales g
    JOIN dim_customer c
        ON c.customer_code = g.customer_code
    WHERE g.fiscal_year = 2021
    GROUP BY c.market, c.region
    ORDER BY gross_sales_mln DESC
),
CTE1 AS (
    SELECT *,
        DENSE_RANK() OVER (PARTITION BY region ORDER BY gross_sales_mln DESC) AS rnk
    FROM CTE
)
SELECT *
FROM CTE1
WHERE rnk < 3;
```

The screenshot shows a database query results interface. At the top, there are buttons for 'Result Grid', 'Filter Rows', 'Export', and 'Wrap Cell Content'. On the right side, there is a vertical toolbar with icons for 'Result Grid', 'Form Editor', 'Field Types', 'Query Stats', 'Execution Plan', and a 'Read Only' button. The main area displays a result grid with the following data:

| market | region | gross_sales_mln | rnk |
|------------|--------|-----------------|-----|
| India | APAC | 455.05 | 1 |
| South ... | APAC | 131.86 | 2 |
| United ... | EU | 78.11 | 1 |
| France | EU | 67.62 | 2 |
| Mexico | LATAM | 2.30 | 1 |
| Brazil | LATAM | 2.14 | 2 |
| USA | NA | 264.46 | 1 |
| Canada | NA | 89.78 | 2 |

Below the grid, there is a message bar indicating the query was executed successfully: '1 13:37:08 WITH CTE AS (SELECT c.market,c.region,ROUND(SUM(g.gross_price_total/1000000),2)as gross_s... 8 row(s) returned'. The bottom right corner shows a 'Duration / Fetch' of '2.031 sec / 0.000 sec'.

14. As a product owner , I need an aggregate forecast accuracy report for all the customers for a given fiscal year so that I can track the accuracy of the forecast we make for these customers.

The report should have fields-

- 1. Customer code , name market**
- 2. Total Sold Quantity**
- 3. Total Forecast Quantity**
- 4. Net Error**
- 5. Absolute Error**
- 6. Forecast Accuracy%**

| customer_code | customer | market | total_sold_qty | total_forecast_qty | net_error | net_error_pct | abs_error | abs_error_pct | forecast_accuracy |
|---------------|----------------|------------|----------------|--------------------|-----------|---------------|-----------|---------------|-------------------|
| 70014142 | Atliq Exclu... | Netherl... | 81324 | 44740 | -36584 | -81.7702 | 47940 | 107.1524 | 0.0000 |
| 70014143 | Atliq e Sto... | Netherl... | 81942 | 45345 | -36597 | -80.7079 | 48575 | 107.1232 | 0.0000 |
| 90014135 | Electricals... | Netherl... | 82731 | 46718 | -36013 | -77.0859 | 48449 | 103.7052 | 0.0000 |
| 90014136 | Reliance ... | Netherl... | 84808 | 48442 | -36366 | -75.0712 | 48472 | 100.0619 | 0.0000 |
| 90014137 | Media Ma... | Netherl... | 77931 | 44802 | -33129 | -73.9454 | 45767 | 102.1539 | 0.0000 |
| 90014138 | Mbit | Netherl... | 79249 | 43779 | -35470 | -81.0206 | 45846 | 104.7214 | 0.0000 |
| 90014139 | Elkjøp | Netherl... | 83198 | 46240 | -36958 | -79.9265 | 48450 | 104.7794 | 0.0000 |
| 90014140 | Radio Pop... | Netherl... | 85581 | 47577 | -38004 | -79.8789 | 49774 | 104.6178 | 0.0000 |
| 90014141 | Amazon | Netherl... | 77462 | 43905 | -33557 | -76.4309 | 45033 | 102.5692 | 0.0000 |
| 90007197 | Amazon | South ... | 344240 | 226857 | -117383 | -51.7432 | 191047 | 84.2147 | 15.7853 |
| 90019202 | Argos (Sai... | Sweden | 26581 | 18218 | -8363 | -45.9051 | 15273 | 83.8347 | 16.1653 |

```
WITH forecast_err_table AS (
    SELECT
        s.customer_code,
        SUM(s.sold_quantity) AS total_sold_qty,
        SUM(s.forecast_quantity) AS total_forecast_qty,
        SUM(s.forecast_quantity - s.sold_quantity) AS net_error,
        SUM(s.forecast_quantity - s.sold_quantity) * 100 / SUM(s.forecast_quantity) AS net_error_pct,
        SUM(ABS(s.forecast_quantity - s.sold_quantity)) AS abs_error,
        SUM(ABS(s.forecast_quantity - s.sold_quantity)) * 100 / SUM(ABS(s.forecast_quantity)) AS abs_error_pct
    FROM fact_act_est s
    JOIN dim_customer c USING(customer_code)
    WHERE s.fiscal_year = 2021
    GROUP BY s.customer_code
)
SELECT
    e.customer_code,
    c.customer,
    c.market,
    e.total_sold_qty,
    e.total_forecast_qty,
    e.net_error,
    e.net_error_pct,
    e.abs_error,
    e.abs_error_pct,
    IF(e.abs_error_pct > 100, 0, 100 - e.abs_error_pct) AS forecast_accuracy
FROM forecast_err_table e
JOIN dim_customer c USING(customer_code)
ORDER BY forecast_accuracy;
```

THANK YOU!