

Comprehensive Data Analysis for Atliq Hardware

Financial Performance, Supply Chain Optimization,
and Ad Hoc Insights

Name: P. Rahul
Date: 03-01-2025

PROBLEM STATEMENT:

Atliq Hardware, a leading hardware manufacturer producing products like CPUs, mouse, desktops and other components, supplies to major retailers including Croma, Best Buy, Staples, and Flipkart. The company seeks to analyze its financial performance (Profit and Loss Statements), optimize supply chain operations, and conduct Ad Hoc data analysis to uncover actionable insights for better decision-making.

INTRODUCTION

Brief Overview of the Project

- This project focuses on leveraging SQL and advanced analytics techniques to enhance Atliq Hardware's business operations and decision-making capabilities. The analysis spans financial performance, supply chain optimization, and data management processes, providing actionable insights to improve operational efficiency and profitability.

OBJECTIVES

Developing Financial Analysis Tools:

- Creation of Profit and Loss (P&L) statements, including metrics such as net invoice sales, net sales, gross margin, and gross margin percentage.

Optimizing Supply Chain Analytics:

- Calculation of key metrics like net error, net error percentage, absolute net error, absolute net error percentage, and forecast accuracy to identify inefficiencies and improve forecasting.

Defining Data Management Processes:

- Establishing data warehouse architecture and incorporating ETL (Extract, Transform, Load) processes.
- Differentiating and implementing OLAP (Online Analytical Processing) and OLTP (Online Transaction Processing) systems for efficient data handling.

Data Gathering and Preparation:

- Collecting and processing all necessary data from various sources to ensure accurate analysis.

Implementing Advanced SQL Features:

- Developing stored procedures, user-defined functions, views, triggers, and indexes to automate processes and enhance query performance.

DATA MANAGEMENT PROCESSES

Establishing Data Warehouse Architecture and Incorporating ETL Processes

To support Atliq Hardware's analytical and reporting needs, a robust data warehouse architecture was designed and implemented. The data warehouse serves as a centralized repository, aggregating data from various operational systems and external sources.

The **ETL (Extract, Transform, Load)** process was integral to this architecture:

- **Extract:** Data was retrieved from multiple disparate sources, including transactional databases and external vendor systems.
- **Transform:** Data was cleansed, normalized, and structured to ensure consistency and reliability for analysis.
- **Load:** The processed data was loaded into the data warehouse, enabling seamless access for reporting and analytical purposes.

This approach ensures the data is accurate, consistent, and readily available for decision-making.

OLTP (Online Transaction Processing):

- Focused on handling day-to-day transactional operations, OLTP systems were optimized for speed and accuracy. Key features include:
- Managing real-time data updates, such as sales and inventory transactions.
- Ensuring data integrity and reliability for operational tasks.

OLAP (Online Analytical Processing):

- Designed to support complex analytical queries and multidimensional data analysis, OLAP systems were utilized for generating reports and dashboards. Key features include:
- Aggregating historical data for trend analysis.
- Supporting ad hoc queries to facilitate business insights.
- Enabling decision-makers to analyze data across various dimensions, such as product, region, and time.

DATASET DESCRIPTION

Table: **dim_customer**

Columns:

<u>customer_code</u>	int UN PK
customer	varchar(150)
platform	varchar(45)
channel	varchar(45)
market	varchar(45)
sub_zone	varchar(45)
region	varchar(45)

Table: **fact_forecast_monthly**

Columns:

date	date
fiscal_year	year
product_code	varchar(45)
customer_code	int
forecast_quantity	int

Table: **fact_post_invoice_deductions**

Columns:

<u>customer_code</u>	int UN PK
<u>product_code</u>	varchar(45) PK
<u>date</u>	date PK
discounts_pct	decimal(5,4)
other_deductions_pct	decimal(5,4)

Table: **dim_date**

Columns:

<u>calender_date</u>	date PK
fiscal_year	year

Table: **fact_freight_cost**

Columns:

<u>market</u>	varchar(45) PK
<u>fiscal_year</u>	year PK
freight_pct	decimal(5,4) UN
other_cost_pct	decimal(5,4) UN

Table: **fact_pre_invoice_deductions**

Columns:

<u>customer_code</u>	int UN PK
<u>fiscal_year</u>	year PK
pre_invoice_discount_pct	decimal(5,4)

Table: **dim_product**

Columns:

<u>product_code</u>	varchar(45) PK
division	varchar(45)
segment	varchar(45)
category	varchar(45)
product	varchar(200)
variant	varchar(45)

Table: **fact_gross_price**

Columns:

<u>product_code</u>	varchar(45) PK
<u>fiscal_year</u>	year PK
gross_price	decimal(15,4) UN

Table: **fact_sales_monthly**

Columns:

<u>date</u>	date PK
fiscal_year	year
<u>product_code</u>	varchar(45) PK
<u>customer_code</u>	int UN PK
sold_quantity	int UN

Table: **fact_act_est**

Columns:

<u>date</u>	date PK
fiscal_year	year
<u>product_code</u>	varchar(45) PK
<u>customer_code</u>	int PK
sold_quantity	int
forecast_quantity	int

Table: **fact_manufacturing_cost**

Columns:

<u>product_code</u>	varchar(45) PK
<u>cost_year</u>	year PK
manufacturing_cost	decimal(15,4) UN

Origin: The dataset represents real-world data simulated to reflect Atliq Hardware's business activities, including sales, inventory, and forecast metrics.

Volume: The dataset comprises approximately **1.4 million records**, covering multiple fiscal years to ensure a comprehensive analysis.

FINANCIAL METRICS

Profit and Loss (P&L) statements

- **Net Invoice Sales:** Represents the total revenue generated from customer invoices after accounting for discounts and adjustments. In our case the Net Invoice sales is the difference between Gross Price and Pre-Invoice deductions
- **Net Sales:** Total sales revenue after deducting returns, allowances, and discounts. In our case the Net Sales is the difference between Net-Invoice Sales and Post Invoice deductions
- **Gross Margin:** The difference between net sales and the cost of goods sold (COGS), providing insights into profitability before considering operating expenses.
- **Gross Margin Percentage:** A percentage representation of gross margin relative to net sales, indicating the company's efficiency in managing production and sales costs.

SUPPLY CHAIN METRICS

- **Net Error:** The difference between forecasted and actual sold quantities, highlighting forecasting inaccuracies.
- **Net Error Percentage:** The percentage representation of net error relative to forecasted quantities, quantifying overall forecasting deviations.
- **Absolute Net Error:** The total absolute value of deviations between forecasted and sold quantities, ensuring all errors are accounted for irrespective of direction.
- **Absolute Net Error Percentage:** The absolute net error expressed as a percentage of forecasted quantities, providing a clearer view of aggregate forecasting inaccuracies.
- **Forecast Accuracy:** Calculated as $100 - \text{Absolute Net Error Percentage}$, this metric measures the reliability of forecasting processes.

Essential Functions for Analysis in MySQL

Write a function using SQL to get fiscal year and fiscal Quarter as per Atliq financial Year

```
CREATE DEFINER='root'@'localhost' FUNCTION `get_fiscal_quarter`(  
    calender_date date  
) RETURNS char(2) CHARSET utf8mb4  
    DETERMINISTIC  
BEGIN  
    declare m tinyint;  
    declare qtr char(2);  
    set m = month(calender_date);  
    case  
        when m in (9,10,11) then  
            set qtr = "Q1";  
        when m in (12,1,2) then  
            set qtr = "Q2";  
        when m in (3,4,5) then  
            set qtr = "Q3";  
        else  
            set qtr = "Q4";  
    end case;  
    RETURN qtr;  
END
```

```
select gdb0041.get_fiscal_quarter('2017-11-01');
```

Output

	gdb0041.get_fiscal_quarter('2017-11-01')
▶	Q1

```
CREATE DEFINER='root'@'localhost' FUNCTION `get_fiscal_year`(  
    calender_date Date  
) RETURNS int  
    DETERMINISTIC  
BEGIN  
    declare fiscal_year int;  
    set fiscal_year = year(date_add(calender_date, interval 4 month));  
    RETURN fiscal_year;  
END
```

```
select gdb0041.get_fiscal_year('2017-11-01');
```

Output

	gdb0041.get_fiscal_year('2017-11-01')
▶	2018

Croma India product wise sales report for fiscal year 2021

Description:

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.

The report should have the following fields,

Month, Product Name & Variant, Sold Quantity, Gross Price Per Item, Gross Price Total, Variants

```
select f.date,
       f.product_code,
       dp.product,
       dp.variant,
       f.sold_quantity,
       g.gross_price,
       round(f.sold_quantity*g.gross_price,2) as gross_price_total
from fact_sales_monthly f
join dim_product dp
on f.product_code = dp.product_code
join fact_gross_price g
on f.product_code = g.product_code and g.fiscal_year = f.fiscal_year
where f.customer_code = 90002002 and
f.fiscal_year = 2021
order by date asc
```

Output

date	product_code	product	variant	sold_quantity	gross_price	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	AO Zion Saga	Premium	137	28.0059	3836.81

Stored Procedure for dynamic fiscal Year & customer

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `customer&Year wise product sales` (
    in_customer_code int,
    in_market varchar(150),
    in_fiscal_year int
)
BEGIN
    select f.date,
           f.product_code,
           dp.product,
           dp.variant,
           f.sold_quantity,
           g.gross_price,
           round(f.sold_quantity*g.gross_price,2) as gross_price_total
    from fact_sales_monthly f
    join dim_product dp
    on f.product_code = dp.product_code
    join fact_gross_price g
    on f.product_code = g.product_code and g.fiscal_year = f.fiscal_year
    join dim_customer c
    on c.customer_code = f.customer_code
    where f.customer_code = in_customer_code and
    f.fiscal_year = in_fiscal_year and c.market = in_market
    order by date asc;
```

END

Gross monthly total sales report for Croma

Description:

As a product owner, I need an aggregate monthly gross sales report for Croma India customers 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 fields:

Month

Total gross sales amount to Croma India in this month

```
SELECT
    s.date,
    SUM(g.gross_price*s.sold_quantity) as gross_price_total
FROM fact_sales_monthly s
JOIN fact_gross_price g
ON
    g.product_code=s.product_code and
    g.fiscal_year=get_fiscal_year(s.date)
WHERE customer_code=90002002
GROUP BY s.date
ORDER BY s.date asc
```

Output




date	gross_price_total
2017-09-01	122407.5582
2017-10-01	162687.5716
2017-12-01	245673.8042
2018-01-01	127574.7372
2018-02-01	144799.5182
2018-04-01	130643.8976
2018-05-01	139165.0975
2018-06-01	125735.3786
2018-08-01	125409.8801

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

- 1. Fiscal Year
- 2. Total Gross Sales amount In that year from Croma

```
select get_fiscal_year(f.date) as Fiscal_Year,
       sum(round(gross_price*sold_quantity,2)) as total_gross_price
from fact_sales_monthly f
join fact_gross_price g
on f.product_code = g.product_code and
g.fiscal_year = get_fiscal_year(f.date)
where customer_code = 90002002
group by get_fiscal_year(f.date)
order by get_fiscal_year(f.date) asc
```

Output

Result Grid   Filter Rows: <input type="text"/> Export:  Wrap		
	Fiscal_Year	total_gross_price
▶	2018	1324097.48
	2019	3555079.19
	2020	6502182.12
	2021	23216512.73
	2022	44638199.11

Stored procedure for market badge

Description:

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**.

My input will be: Market, Fiscal year , **Output:** Market badge

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `get_market_badge`(  
    IN in_market varchar(45),  
    IN in_fiscal_year YEAR,  
    OUT out_badge varchar(45)  
)  
BEGIN  
  
    declare qty int default 0;  
    # retrieve total qty for a given market+fyear  
    if in_market="" then  
        set in_market = "india";  
    end if;  
  
    select sum(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  
    group by c.market;  
    # determine market badge  
    if qty > 5000000 then  
        set out_badge= "GOLD";  
    else  
  
        set out_badge = "Silver";  
    end if;  
  
END
```

```
set @out_badge = '0';  
call gdb0041.get_market_badge('India', 2021, @out_badge);  
select @out_badge;
```

Output

	@out_badge
►	GOLD

Profit & Loss Statement Analysis

```
WITH gross_price AS (  
    SELECT  
        f.customer_code,  
        f.product_code,  
        f.fiscal_year,  
        ROUND((f.sold_quantity * g.gross_price), 2) AS gross_price  
    FROM fact_sales_monthly f  
    JOIN fact_gross_price g  
        ON f.product_code = g.product_code  
    WHERE f.fiscal_year = 2021  
        AND f.product_code = 'A0118150101'  
),  
net_invoice_sales AS (  
    SELECT  
        g.customer_code,  
        g.product_code,  
        g.fiscal_year,  
        ROUND((1 - fpr.pre_invoice_discount_pct) * g.gross_price, 2) AS net_invoice_sales_amt  
    FROM gross_price g  
    JOIN fact_pre_invoice_deductions fpr  
        ON g.customer_code = fpr.customer_code  
    WHERE g.product_code= 'A0118150101' AND fpr.fiscal_year = 2021  
),  
net_sales AS (  
    SELECT  
        nis.customer_code,  
        nis.product_code,  
        nis.fiscal_year,  
        ROUND((1 - (fpo.discounts_pct + fpo.other_deductions_pct)) * nis.net_invoice_sales_amt, 2) AS net_sales_amt  
    FROM net_invoice_sales nis  
    INNER JOIN fact_post_invoice_deductions fpo  
        ON fpo.customer_code = nis.customer_code  
    WHERE fpo.product_code= 'A0118150101'  
)
```

Output

customer_code	product_code	fiscal_year	gross_price	net_invoice_sales_amt	net_sales_amt
70002017	A0118150101	2021	4726.21	4393.96	2909.24
70002017	A0118150101	2021	4726.21	4393.96	2607.38
70002017	A0118150101	2021	4726.21	4393.96	2607.82
70002017	A0118150101	2021	4726.21	4393.96	2777.86
70002017	A0118150101	2021	4726.21	4393.96	2635.94
70002017	A0118150101	2021	4726.21	4393.96	2793.68
70002017	A0118150101	2021	4726.21	4393.96	3053.36
70002017	A0118150101	2021	4726.21	4393.96	2719.42
70002017	A0118150101	2021	4726.21	4393.96	2613.53
70002017	A0118150101	2021	4726.21	4393.96	3155.74
70002017	A0118150101	2021	4726.21	4393.96	3117.51
70002017	A0118150101	2021	4726.21	4393.96	3004.59
70002017	A0118150101	2021	4726.21	4393.96	3010.74
70002017	A0118150101	2021	4726.21	4393.96	2865.30
70002017	A0118150101	2021	4726.21	4393.96	3057.76
70002017	A0118150101	2021	4726.21	4393.96	3028.32
70002017	A0118150101	2021	4726.21	4393.96	3166.73
70002017	A0118150101	2021	4726.21	4393.96	3020.85
70002017	A0118150101	2021	4726.21	4393.96	2600.35

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.
write a stored procedure for this as we will need this report going forward as well.

- 1)Report for top markets
- 2)Report for top products
- 3) Report for top customers

```
-- Top 5 Markets
SELECT
    dc.market,
    round(SUM(ns.net_sales_amt)/1000000,2) as net_sales_mln
FROM net_sales ns
INNER JOIN fact_sales_monthly fs ON fs.customer_code = ns.customer_code
INNER JOIN dim_customer dc ON dc.customer_code = ns.customer_code
WHERE fs.product_code='A0118150101' AND fs.fiscal_year = 2021
GROUP BY dc.market
order by net_sales_mln desc limit 5;

-- top 5 customers
SELECT
    dc.customer,
    round(SUM(ns.net_sales_amt)/1000000,2) as net_sales_mln
FROM net_sales ns
INNER JOIN fact_sales_monthly fs ON fs.customer_code = ns.customer_code
INNER JOIN dim_customer dc ON dc.customer_code = ns.customer_code
WHERE fs.product_code='A0118150101' AND fs.fiscal_year = 2021
GROUP BY dc.customer
order by net_sales_mln desc limit 5
```



Result Grid			Filter Rows:
	market	net_sales_mln	
▶	India	270.83	
	USA	179.18	
	South Korea	83.80	
	Canada	63.21	
	Philippines	54.82	



Result Grid			Filter Rows:
	customer	net_sales_mln	
▶	Amazon	136.54	
	Atliq Exclusive	99.93	
	Atliq e Store	92.32	
	Sage	38.15	
	Neptune	31.71	

Write a stored procedure to get the top n products by net sales for a given year. Use product name without a variant.

```
CREATE DEFINER='root'@'localhost' PROCEDURE `get_top_n_products`(  
    in_fiscal_year int,  
    in_top_n int  
)  
BEGIN  
    with cte as (  
        select division, product,sum(sold_quantity) as Total_quantity  
        from fact_sales_monthly f  
        join dim_product dp  
        on f.product_code = dp.product_code  
        where fiscal_year = in_fiscal_year  
        group by division, product  
        order by Total_quantity desc),  
        cte2 as (select division, product,Total_quantity, dense_rank() over(partition by division order by Total_quantity desc) as dn  
        from cte)  
        select *  
        from cte2  
        where dn <= in_top_n;  
  
END
```

```
call gdb0041.get_top_n_products(2020, 5);
```

	division	product	Total_quantity	dn
►	N & S	AQ Clx1	935128	1
	N & S	AQ Neuer SSD	924264	2
	N & S	AQ Digit SSD	920105	3
	N & S	AQ Wi Power Dx2	846576	4
	N & S	AQ Wi Power Dx1	844664	5
	P & A	AQ Master wired x1 Ms	1578253	1
	P & A	AQ Gamers Ms	1566445	2
	P & A	AQ Lite Ms	1554000	3

Forecast Accuracy for all customers for a given fiscal year

Description 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 the following fields: Customer Code, Name, Market, Total Sold Quantity, Total Forecast Quantity, Net Error, Absolute Error, Forecast Accuracy %

```
CREATE DEFINER='root'@'localhost' PROCEDURE `get_forecast_accuracy`(  
    in_fiscal_year INT  
)  
BEGIN  
  
    with forecast_err_table as (select  
        s.customer_code as customer_code,  
        c.customer as customer_name,  
        c.market as market,  
        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,  
        round(sum(s.forecast_quantity-s.sold_quantity)*100/sum(s.forecast_quantity),1) as net_error_pct,  
        sum(abs(s.forecast_quantity-s.sold_quantity)) as abs_error,  
        round(sum(abs(s.forecast_quantity-s.sold_quantity))*100/sum(s.forecast_quantity),2) as abs_error_pct  
    from fact_act_est s  
    join dim_customer c  
    on s.customer_code = c.customer_code  
    where s.fiscal_year= in_fiscal_year  
    group by customer_code)  
    select *, if(abs_error_pct > 100, 0, 100-abs_error_pct) as forecast_accuracy  
    from forecast_err_table  
    order by forecast_accuracy desc;
```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:								
	customer_code	customer_name	market	total_sold_qty	total_forecast_qty	net_error	net_error_pct	abs_error	abs_error_pct	forecast_accuracy	
▶	70006158	Atliq e Store	Philiphines	136991	136991	5044	18053	11.6	88917	57.35	42.65
	90010046	Amazon	Bangladesh	55532	58644		3112	5.3	33716	57.49	42.51
	90023030	Amazon	Canada	127854	140248		12394	8.8	82588	58.89	41.11
	70008170	Atliq e Store	Australia	178182	192586		14404	7.5	113708	59.04	40.96
	90023026	Relief	Canada	85944	143041		57097	39.9	85555	59.81	40.19

Write a query for the below scenario.

The supply chain business manager wants to see which customers' forecast accuracy has dropped from 2020 to 2021. Provide a complete report with these columns: customer_code, customer_name, market, forecast_accuracy_2020, forecast_accuracy_2021

```
CREATE TEMPORARY TABLE forecast_err_table AS
SELECT
    s.customer_code AS customer_code,
    s.fiscal_year,
    c.customer AS customer_name,
    c.market AS market,
    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,
    CASE
        WHEN SUM(s.forecast_quantity) > 0
        THEN ROUND(SUM(s.forecast_quantity - s.sold_quantity) * 100 / SUM(s.forecast_quantity), 1)
        ELSE NULL
    END AS net_error_pct,
    SUM(ABS(s.forecast_quantity - s.sold_quantity)) AS abs_error,
    CASE
        WHEN SUM(s.forecast_quantity) > 0
        THEN ROUND(SUM(ABS(s.forecast_quantity - sold_quantity)) * 100 / SUM(s.forecast_quantity), 2)
        ELSE NULL
    END AS abs_error_pct
FROM fact_act_est s
JOIN dim_customer c
    ON s.customer_code = c.customer_code
GROUP BY s.customer_code, s.fiscal_year;

SELECT *,
    CASE
        WHEN abs_error_pct > 100 THEN 0
        ELSE 100 - abs_error_pct
    END AS forecast_accuracy
FROM forecast_err_table
ORDER BY forecast_accuracy DESC;
```

```
select customer_code, customer_name, market,
       sum(case when fiscal_year = 2020 then forecast_accuracy end) as forecast_accuracy_2020,
       sum(case when fiscal_year = 2021 then forecast_accuracy end) as forecast_accuracy_2021
from forecast_analytics
group by customer_code, customer_name, market
having forecast_accuracy_2021 < forecast_accuracy_2020
order by market
```

Result Grid					
Filter Rows: <input type="text"/>					
Export: <input type="button" value=""/>					
Wrap Cell Content: <input type="checkbox"/>					
	customer_code	customer_name	market	forecast_accuracy_2020	forecast_accuracy_2021
▶	70008170	Atliq e Store	Australia	40.96	38.74
	90008164	Digimarket	Australia	37.15	36.01
	90008166	Sound	Australia	38.51	36.79
	70012042	Atliq Exclusive	Germany	24.28	22.88
	90012040	Fnac-Darty	Germany	23.25	21.85
	90012037	Saturn	Germany	25.11	19.16
	90012036	Billa	Germany	26.05	18.29
	90012034	Otto	Germany	28.26	18.37
	90012033	Digimarket	Germany	22.86	22.51

KEY INSIGHTS AND RESULTS

Profit & Loss Statements:

- Successfully generated P&L statements, including metrics such as total gross price, net invoice sales and net sales
- Identified trends in profitability across different markets and customer segments, highlighting areas for improvement in pricing and cost control.

Supply Chain Metrics:

- Calculated critical metrics such as net error, net error percentage, absolute net error, absolute net error percentage, and forecast accuracy.
- Identified inefficiencies in forecasting, with actionable recommendations to improve forecast accuracy and minimize supply chain errors.

Customer and Product Insights:

- Determined the **top 5 customers** contributing the highest revenue and their corresponding products, aiding in targeted marketing strategies.
- Identified at-risk customers based on declining sales trends, enabling proactive customer retention initiatives.

Market Insights:

- Highlighted the **top-performing markets**, providing insights into regional demand patterns and growth opportunities.

Year-over-Year Trends:

- Conducted a year-over-year analysis of forecast quantities, identifying significant increases or drops.
- Provided actionable insights to adjust production and inventory strategies accordingly.

Ad Hoc Analysis:

- Successfully performed ad hoc analyses to answer specific business questions, enhancing decision-making processes.

CHALLENGES

During the project, several challenges were encountered, particularly related to query optimization due to the large dataset, which consisted of approximately 1.4 million records. At times, queries took 10 to 15 minutes to execute. Below are the key challenges I faced when running the queries:

Challenges

- 1.Slow Query Execution:** The large volume of data significantly increased query execution time.
- 2.Resource Utilization:** High memory and processing requirements caused delays in obtaining results.
- 3.Data Aggregation Complexity:** Aggregating large amounts of data for financial and supply chain metrics proved computationally expensive.

SOLUTIONS

Effective Grouping Strategies:

- Reorganized queries to use efficient grouping and aggregation methods, ensuring optimized performance.

Row Limitation:

- Limited the number of rows processed during initial queries to **20,000**, allowing for quicker results during testing and debugging.

Functions and Stored Procedures:

- Created reusable **functions** and **stored procedures** to handle repetitive operations, reducing execution time by avoiding redundant computations.

Indexing:

- Implemented indexing on frequently grouped and filtered columns, which significantly improved query performance by reducing the time required to locate and retrieve data.

Partitioning Large Tables:

- Partitioned the dataset into smaller, manageable chunks based on fiscal year or market, enabling parallel processing and faster data access.

Query Simplification:

- Simplified complex queries by breaking them into smaller subqueries, which were then optimized individually.

Caching Results:

- Used temporary tables to cache intermediate results, avoiding re-execution of subqueries for repetitive tasks.

CONCLUSION

This project was a transformative experience, enhancing my SQL expertise through hands-on implementation of advanced techniques such as stored procedures, indexing, and query optimization. Analyzing over 1.4 million records to generate actionable insights on profit and loss, supply chain metrics, and customer trends demonstrated my ability to handle complex datasets and deliver meaningful business outcomes. Overcoming challenges like long query runtimes sharpened my problem-solving and optimization skills. These learnings have equipped me to confidently apply data-driven strategies in future projects, focusing on scalable solutions and innovative approaches to address real-world analytical challenges.