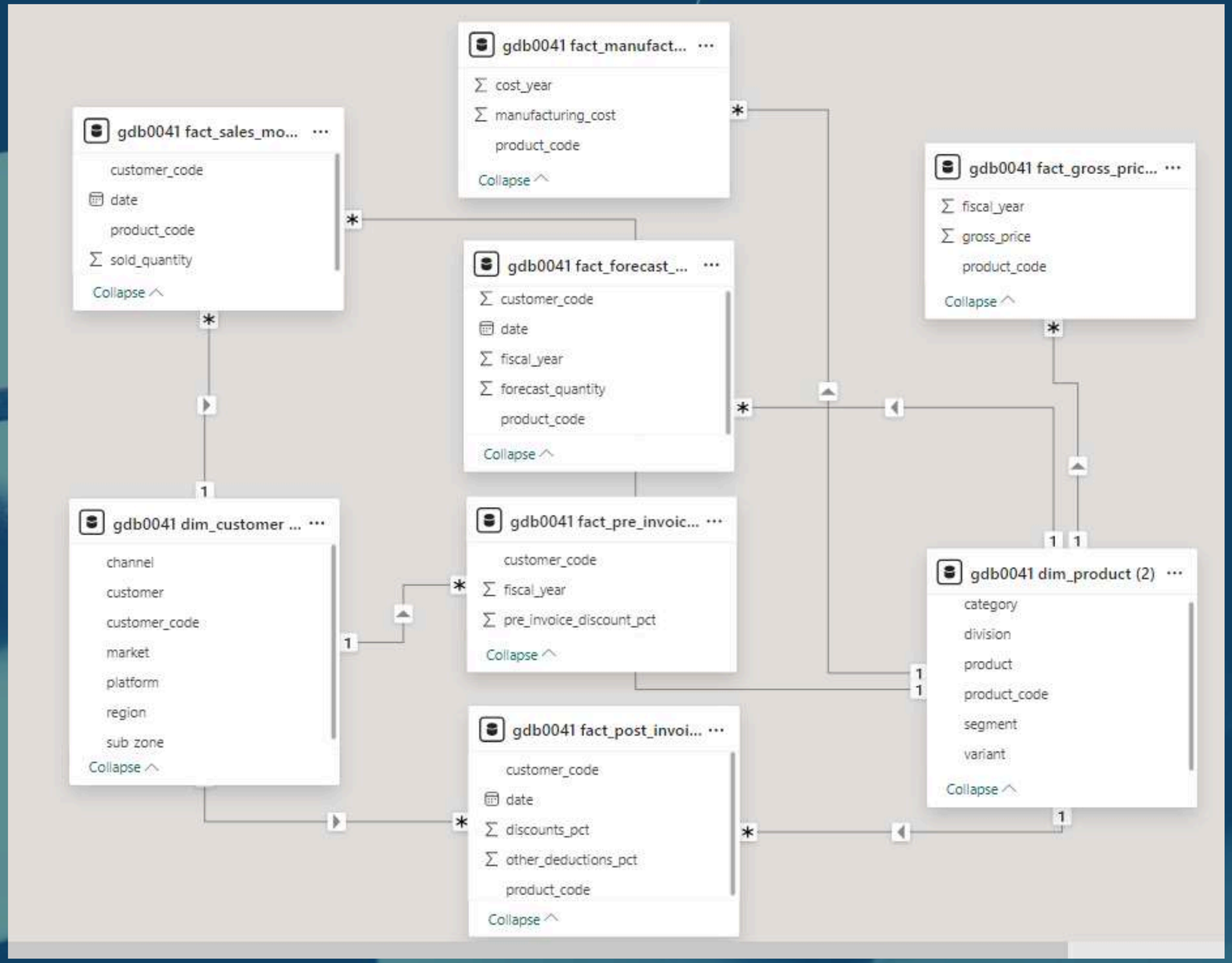
Atliq Financial and Sales Trends: An Analytical Approach Using SQL

IN THIS PROJECT, THE SALES AND FINANCIAL PERFORMANCE OF ATLIQ HARDWARE, A COMPUTER HARDWARE COMPANY, WAS ANALYZED USING SQL. VARIOUS SQL QUERIES WERE EMPLOYED TO EXTRACT INSIGHTS, SUCH AS REVENUE TRENDS, YEAR-OVER-YEAR GROWTH, PROFIT MARGINS, AND SALES PERFORMANCE ACROSS DIFFERENT PRODUCT CATEGORIES. THE ANALYSIS PROVIDED A DETAILED BREAKDOWN OF KEY FINANCIAL METRICS, ENABLING A COMPREHENSIVE UNDERSTANDING OF ATLIQ'S BUSINESS PERFORMANCE. THIS PROJECT DEMONSTRATES THE POWER OF SQL IN UNCOVERING BUSINESS INSIGHTS BY QUERYING AND ANALYZING LARGE DATASETS.

Schema Description



Created Stored Procedures of Sales before Pre-Invoice sales and Post-Invoice sales to reduce network traffic by executing multiple SQL statements in a single call and for better analysis of discounts given to different sellers

Pre-Invoice discount

```
USE `gdb0041`;
CREATE  OR REPLACE VIEW `sales_Prev_InvDiscount` AS
select s.date, s.product_code, c.market, p.product, p.variant, s.sold_quantity, g.gros
round(g.gross_price*s.sold_quantity,2) as Gross_Total_Price, pre.pre_invoice_discou
from fact_sales_monthly s
join dim_product p
using(product_code)
join dim_customer c
using(customer_code)
join fact_gross_price g
on g.product_code = s.product_code
and g.fiscal_year = get_fiscal_year(s.date)
join fact_pre_invoice_deductions pre
on pre.customer_code = s.customer_code
and pre.fiscal_year = get_fiscal_year(s.date)
order by date asc;
```

Post-Invoice discount

```
USE `gdb0041`;
CREATE  OR REPLACE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `post_invoice_deduction` AS
    SELECT
        `sales_prev_invdiscount`.`date` AS `date`,
        `sales_prev_invdiscount`.`customer_code` AS `customer_code`,
        `sales_prev_invdiscount`.`product_code` AS `product_code`,
        `sales_prev_invdiscount`.`variant` AS `variant`,
        `sales_prev_invdiscount`.`sold_quantity` AS `sold_quantity`,
        `sales_prev_invdiscount`.`Gross_Total_Price` AS `gross_total_price`,
        `sales_prev_invdiscount`.`pre_invoice_discount_pct` AS `pre_invoice_discount_pct`,
        ((1 - `sales_prev_invdiscount`.`pre_invoice_discount_pct`) * `sales_prev_invdiscount`.`Gross_Total_Price`
        (`po`.`discounts_pct` + `po`.`other_deductions_pct`) AS `post_invoice_discount`
    FROM
        (`sales_prev_invdiscount`
        JOIN `fact_post_invoice_deductions` `po` ON (((`sales_prev_invdiscount`.`date` = `po`.`date`)
            AND (`sales_prev_invdiscount`.`customer_code` = `po`.`customer_code`)
```

Created Stored Procedures of Net sales after all discounts to reduce network traffic by executing multiple SQL statements in a single call and for better reusability of the queries and analyze further sales trends with queries which are simple and more readable.

```
3        ALGORITHM = UNDEFINED
4        DEFINER = `root`@`localhost`
5        SQL SECURITY DEFINER
6      VIEW `net_sales` AS
7        SELECT
8          `post_invoice_dedudction`.`date` AS `date`,
9          `post_invoice_dedudction`.`customer_code` AS `customer_code`,
10         `post_invoice_dedudction`.`product_code` AS `product_code`,
11         `post_invoice_dedudction`.`variant` AS `variant`,
12         `post_invoice_dedudction`.`sold_quantity` AS `sold_quantity`,
13         `post_invoice_dedudction`.`gross_total_price` AS `gross_total_price`,
14         `post_invoice_dedudction`.`pre_invoice_discount_pct` AS `pre_invoice_discount_pct`
15         `post_invoice_dedudction`.`Net_Invoice_sales` AS `Net_Invoice_sales`,
16         `post_invoice_dedudction`.`post_invoice_discount` AS `post_invoice_discount`,
17         ((1 - `post_invoice_dedudction`.`post_invoice_discount`) * `post_invoice_dedudction`
18       FROM
19         `post_invoice_dedudction`;
```

Generate a report of individual product sales(for Croma India) aggregated monthly at the product code level.

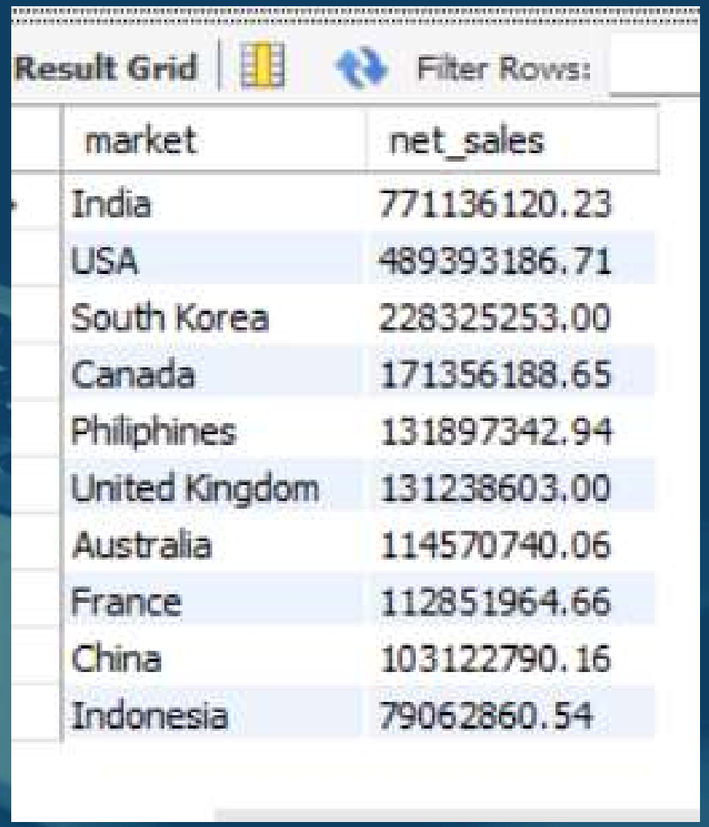
```sql
SELECT
    s.date,
    s.product_code,
    p.product,
    p.variant,
    s.sold_quantity,
    g.gross_price,
    ROUND(g.gross_price * s.sold_quantity, 2) AS Gross_Total_Price
FROM
    fact_sales_monthly s
        JOIN
    dim_product p USING (product_code)
        JOIN
    fact_gross_price g ON g.product_code = s.product_code
        AND g.fiscal_year = GET_FISCAL_YEAR(s.date)
WHERE
    customer_code = 90002002
        AND GET_FISCAL_YEAR(date) = 2021
ORDER BY date ASC
LIMIT 1000000
```

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

date	product_code	product	variant	sold_quantity	gross_price	Gross_Total_Price
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
2020-09-01	A0519150203	AQ Mforce Gen Y	Standard 3	38	26.5871	1010.31
2020-09-01	A0519150204	AQ Mforce Gen Y	Plus 1	149	26.1081	3890.11
2020-09-01	A0519150205	AQ Mforce Gen Y	Plus 2	29	29.7008	861.32
2020-09-01	A0519150206	AQ Mforce Gen Y	Plus 3	28	31.2439	874.83
2020-09-01	A0519150207	AQ Mforce Gen Y	Premium 1	171	32.4427	5547.70
2020-09-01	A0519150208	AQ Mforce Gen Y	Premium 2	118	30.5816	3608.63

Write a query about the top 10 best-performing markets.

```sql
SELECT
    market, ROUND(SUM(net_sales), 2) AS net_sales
FROM
    net_sales
GROUP BY market
ORDER BY net_sales DESC
LIMIT 10
```



Result Grid | Filter Rows:

market	net_sales
India	771136120.23
USA	489393186.71
South Korea	228325253.00
Canada	171356188.65
Philiphines	131897342.94
United Kingdom	131238603.00
Australia	114570740.06
France	112851964.66
China	103122790.16
Indonesia	79062860.54

Write a query about the top 10 best-performing customers.

```sql
SELECT
    c.customer, ROUND(SUM(net_sales), 2) AS net_sales
FROM
    net_sales s
    join dim_customer c
    using(customer_code)
GROUP BY c.customer
ORDER BY net_sales DESC
LIMIT 10
```

Result Grid | Filter Rows:

customer	net_sales
Amazon	396424220.51
Atliq Exclusive	277120647.44
Atliq e Store	246527219.36
Flipkart	98396397.16
Sage	95296603.37
Leader	88670550.16
Neptune	72733263.99
Ebay	70790566.65
Electricalsocity	59515014.33
Electricalslytical	51939083.99

Write a query about the top 10 most-selling products.

```sql
SELECT
    p.product, ROUND(SUM(net_sales), 2) AS net_sales
FROM
    net_sales s
        JOIN
    dim_product p USING (product_code)
GROUP BY p.product
ORDER BY net_sales DESC
LIMIT 10
```

Result Grid | Filter Rows: | Expo

product	net_sales
AQ BZ Allin1	92952219.55
AQ BZ Compact	92574401.87
AQ Smash 1	91205119.50
AQ Smash 2	88390190.67
AQ Gamer 2	87523942.82
AQ BZ Allin1 Gen 2	84631154.03
AQ HOME Allin1 Gen 2	84351361.81
AQ BZ Gen Y	83845307.60
AQ BZ Gen Z	82319828.39
AQ Electron 3 3600 Desktop Processor	80603164.83

Write a query to give the net sales in the last 5 years and the percentage of sales concerning total sales in the previous 5 years.

```sql
WITH CTE1 AS (
    SELECT get_fiscal_year(date) AS fiscal_year,
           ROUND(SUM(net_sales) / 1000000, 2) AS net_sales_mln
    FROM net_sales
    GROUP BY get_fiscal_year(date)
)
SELECT fiscal_year,
       net_sales_mln,
       (net_sales_mln / SUM(net_sales_mln) OVER ()) * 100 AS sales_pct_by_yr
FROM CTE1;
```

Result Grid | Filter Rows:

fiscal_year	net_sales_mln	sales_pct_by_yr
2018	29.11	1.012448
2019	111.37	3.873456
2020	267.98	9.320363
2021	823.85	28.653559
2022	1642.90	57.140174


Thanks You