SQL Queries Used for Atliq Hardware Project

Finance Analytics

 # Finance Analytics

 # Fields required

-- Month

-- Product Name

-- Variant

-- Sold Quantity

-- Gross Price Per Item

-- Gross Price Total

-- Find out Customer Code for Croma India --> '90002002'

If you need data for calendar year 2021

```
SELECT * FROM fact_sales_monthly
WHERE
        customer_code = 90002002 AND
    YEAR(date) = 2021
ORDER By date DESC;
```

-- This gives the fiscal year --> YEAR(DATE_ADD(date, INTERVAL 4 MONTH)) i.e., adding 4 months to calendar date

-- Re-write above query to get transactions for Customer "Croma" for fiscal Year 2021

```
SELECT * FROM fact_sales_monthly
WHERE
        customer_code = 90002002 AND
    YEAR(DATE_ADD(date, INTERVAL 4 MONTH)) = 2021
ORDER By date DESC;
```

-- Re-write the query using user defined function get_fiscal_year created

```
SELECT * FROM fact_sales_monthly
WHERE
```

```
        customer_code = 90002002 AND
    get_fiscal_year(date) = 2021
ORDER By date DESC;
```

 # Exercise

-- Create quarters, to get data on quarter level

```
-- 9, 10, 11 --> Q1
-- 12, 1, 2 --> Q2
-- 3, 4, 5 --> Q3
-- 6, 7, 8 --> Q4

SELECT * FROM fact_sales_monthly
WHERE
        customer_code = 90002002 AND
    get_fiscal_year(date) = 2021 AND
    get_fiscal_quarter(date) = "Q4"
ORDER By date ASC;
```

 # Gross Sales Report: Monthly Product Transactions

-- Retrieve Product name and variant

```
SELECT
        s.date, s.product_code,
    p.product, p.variant, s.sold_quantity
FROM fact_sales_monthly s
JOIN dim_product p
ON p.product_code = s.product_code
WHERE
        customer_code = 90002002 AND
    get_fiscal_year(s.date) = 2021
ORDER By s.date ASC;
```

 # Retrieve Gross price per item and Gross price Total

```
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_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.product_code = s.product_code AND
   g.fiscal_year = get_fiscal_year(s.date)
WHERE
        customer_code = 90002002 AND
   get_fiscal_year(s.date) = 2021
ORDER By s.date ASC
LIMIT 1000000;
```

Performance Analytics

 # Top markets, products, customers for a given financial year

```
EXPLAIN ANALYZE
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_price_total,
   pre.pre_invoice_discount_pct
FROM fact_sales_monthly s
JOIN dim_product p
ON s.product_code = p.product_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)
WHERE
    get_fiscal_year(s.date) = 2021
ORDER By s.date ASC
LIMIT 1000000;
```

 # Performance Improvement 1

 # Create dim_date and join the query with it to derive the same result to improve performance

 # Re-write the above query

```
EXPLAIN ANALYZE
SELECT
        s.date, s.product_code,
    p.product, p.variant, s.sold_quantity,
    g.gross_price as gross_price_per_time,
    ROUND(g.gross_price*s.sold_quantity, 2) as gross_price_total,
    pre.pre_invoice_discount_pct
FROM fact_sales_monthly s
JOIN dim_product p
        ON s.product_code = p.product_code
JOIN dim_date dt
        ON dt.calendar_date=s.date
JOIN fact_gross_price g
ON
        g.product_code = s.product_code AND
    g.fiscal_year = dt.fiscal_year
```

```
JOIN fact_pre_invoice_deductions pre
ON
        pre.customer_code=s.customer_code AND
    pre.fiscal_year=dt.fiscal_year
WHERE
    dt.fiscal_year = 2021
LIMIT 1000000;
```

 # Performance Improvement 2

 # Add fiscal year to the fact_sales_monthly itself to avoid extra join to dim_date table

```
SELECT
        s.date, s.product_code,
    p.product, p.variant, s.sold_quantity,
    g.gross_price as gross_price_per_time,
    ROUND(g.gross_price*s.sold_quantity, 2) as gross_price_total,
    pre.pre_invoice_discount_pct
FROM fact_sales_monthly s
JOIN dim_product p
        ON s.product_code = p.product_code
JOIN fact_gross_price g
ON
        g.product_code = s.product_code AND
    g.fiscal_year = s.fiscal_year
JOIN fact_pre_invoice_deductions pre
ON
        pre.customer_code=s.customer_code AND
    pre.fiscal_year=s.fiscal_year
WHERE
    s.fiscal_year = 2021
LIMIT 1000000;
```

 # Calculate Net Invoice Sales Using CTE

```
WITH cte1 as(
SELECT
        s.date, s.product_code,
   p.product, p.variant, s.sold_quantity,
   g.gross_price as gross_price_per_time,
   ROUND(g.gross_price*s.sold_quantity, 2) as gross_price_total,
   pre.pre_invoice_discount_pct
FROM fact_sales_monthly s
JOIN dim_product p
        ON s.product_code = p.product_code
JOIN fact_gross_price g
ON
        g.product_code = s.product_code AND
   g.fiscal_year = s.fiscal_year
JOIN fact_pre_invoice_deductions pre
ON
        pre.customer_code=s.customer_code AND
   pre.fiscal_year=s.fiscal_year
WHERE
   s.fiscal_year = 2021)
SELECT
        *,
   ROUND((gross_price_total - gross_price_total*pre_invoice_discount_pct),2) as net_invoice_sales
FROM cte1;
```

```
# Creating a virtual table -> Views to store the result of a query
SELECT
        *,
   (1 - pre_invoice_discount_pct)*gross_price_total as net_invoice_sales
FROM sales_preinv_discount;
```

```
# Database Views: Post Invoice Discount, Net Sales
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,
   (1 - pre_invoice_discount_pct)*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
        s.date = po.date AND
   s.product_code = po.product_code AND
   s.customer_code = po.customer_code;



# Top n products in each division by their quantity sold
WITH cte1 AS(SELECT
                p.division, p.product,
                sum(sold_quantity) as total_qty
FROM fact_sales_monthly s
JOIN dim_product p
        ON p.product_code = s.product_code
WHERE s.fiscal_year=2021
GROUP BY p.product),

cte2 AS(SELECT
        *,
   dense_rank() over(partition by division order by total_qty desc) as drnk
```

```
FROM cte1)
SELECT * FROM cte2 where drnk<=3;
```

 # Top n markets in every region by gross sales amount

```
SELECT * FROM `gross sales`;
WITH cte1 AS(SELECT
                    gs.market, c.region,
      round(sum(gross_price_total)/1000000,2) as gross_sales_mln
FROM `gross sales` gs
JOIN dim_customer c
          ON gs.customer_code = c.customer_code
WHERE fiscal_year=2021
GROUP BY gs.market, c.region),

cte2 AS(SELECT
          *,
   dense_rank() over(partition by region order by gross_sales_mln desc) as drnk
FROM cte1)
SELECT * FROM cte2 where drnk<=2;



with cte1 as (
                    select
                              c.market,
                              c.region,
                              round(sum(gross_price_total)/1000000,2) as gross_sales_mln
                              from gross_sales s
                              join dim_customer c
                              on c.customer_code=s.customer_code
                              where fiscal_year=2021
                              group by market
                              order by gross_sales_mln desc
```

```
),
cte2 as (
        select *,
        dense_rank() over(partition by region order by gross_sales_mln desc) as
drnk
        from cte1
)
    select * from cte2 where drnk<=2
```

Supply Chain Analytics

```
# Supply Chain Analytics
SELECT
    s.*,
  f.forecast_quantity
FROM fact_sales_monthly s
JOIN fact_forecast_monthly f
USING (date, product_code, customer_code);

# With Left Join
SELECT
    s.date as date,
  s.fiscal_year as fiscal_year,
  s.product_code as product_code,
  s.customer_code as customer_code,
  s.sold_quantity as sold_quantity,
  f.forecast_quantity as forecast_quantity
FROM
    fact_sales_monthly s
LEFT JOIN fact_forecast_monthly f
USING (date, product_code, customer_code)

# Right Join
```

```
SELECT
        s.date as date,
    s.fiscal_year as fiscal_year,
    s.product_code as product_code,
    s.customer_code as customer_code,
    s.sold_quantity as sold_quantity,
    f.forecast_quantity as forecast_quantity
FROM
        fact_sales_monthly s
RIGHT JOIN fact_forecast_monthly f
USING (date, product_code, customer_code)

# Do full outer join, and assume the value to be 0 if there is no data
CREATE TABLE fact_act_est (
SELECT
        s.date as date,
    s.fiscal_year as fiscal_year,
    s.product_code as product_code,
    s.customer_code as customer_code,
    s.sold_quantity as sold_quantity,
    f.forecast_quantity as forecast_quantity
FROM
        fact_sales_monthly s
LEFT JOIN fact_forecast_monthly f
USING (date, product_code, customer_code)

UNION

SELECT
        f.date as date,
    f.fiscal_year as fiscal_year,
```

```
   f.product_code as product_code,
   f.customer_code as customer_code,
   s.sold_quantity as sold_quantity,
   f.forecast_quantity as forecast_quantity
FROM
        fact_forecast_monthly f
LEFT JOIN fact_sales_monthly s
USING (date, product_code, customer_code)
);
```

 # Update the quantities to 0 if it is null

```
UPDATE fact_act_est
SET sold_quantity = 0
WHERE sold_quantity is null;

UPDATE fact_act_est
SET forecast_quantity = 0
WHERE forecast_quantity is null;

SELECT * FROM fact_act_est;
```

 # Calculate Net Error

```
WITH forecast_err_table as (SELECT
        s.customer_code,
        sum(sold_quantity) as total_sold_qty,
   sum(forecast_quantity) as total_forecast_qty,
        sum(forecast_quantity-sold_quantity) as net_err,
   sum(forecast_quantity-sold_quantity)*100/sum(forecast_quantity) as net_err_pct,
   sum(abs(forecast_quantity-sold_quantity)) as abs_err,
   sum(abs(forecast_quantity-sold_quantity))*100/sum(forecast_quantity) as abs_err_pct
FROM fact_act_est s
```

```
WHERE
        s.fiscal_year=2021
group by customer_code)

SELECT
        e.*,
        c.customer, c.market,
        if(abs_err_pct > 100, 0, 100-abs_err_pct) as forecast_accuracy
FROM forecast_err_table e
JOIN dim_customer c
USING (customer_code)
order by forecast_accuracy desc;
```

 # Calculate Net Error with temporary table

```
create temporary table forecast_err_table
        SELECT
                s.customer_code,
                sum(sold_quantity) as total_sold_qty,
                sum(forecast_quantity) as total_forecast_qty,
                sum(forecast_quantity-sold_quantity) as net_err,
                sum(forecast_quantity-sold_quantity)*100/sum(forecast_quantity) as net_err_pct,
                sum(abs(forecast_quantity-sold_quantity)) as abs_err,
                sum(abs(forecast_quantity-sold_quantity))*100/sum(forecast_quantity) as
abs_err_pct
        FROM fact_act_est s
        WHERE
                s.fiscal_year=2021
        group by customer_code;

SELECT * from forecast_err_table;
```

```
SELECT
	e.*,
	c.customer, c.market,
	if(abs_err_pct > 100, 0, 100-abs_err_pct) as forecast_accuracy
FROM forecast_err_table e
JOIN dim_customer c
USING (customer_code)
order by forecast_accuracy desc;
```

Exercise

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

```
SELECT
	e.*,
	c.customer, c.market,
	if(abs_err_pct > 100, 0, 100-abs_err_pct) as forecast_accuracy
FROM forecast_err_table e
JOIN dim_customer c
USING (customer_code)
order by forecast_accuracy desc;
```