Understand AtliQ Hardware health

(Data Analysis with SQL)

Project Overview

AtliQ Hardware is growing rapidly in the recent years, and they have decided to implement OLAP (Online Analytical Processing) for data analytics using SQL in their company for the first time to surpass their competitors in the market and to make data driven decisions. This project is hoped to give answers to the questions of stakeholder in terms all the aspects like finance, sales, marketing and supply chain.

Company's back ground

AltiQ hardware is a company which has grown vastly in the recent years, and opened business all over the globe. It is a company which sells, computer and computer accessories through three mediums/channel

- Retailers
- Direct
- Distributors

Recently the company has faced a unforeseen loss by opening store in America based on the surveys, intuition and some excel analysis and also the company's competitors has handful of analytics team to perform analysis and make data driven decision. So, the AltiQ hardware has no other option other than building their analytics team for data driven insights and decisions in the future to survive better in the industry.

Dataset Understanding

Understanding what data is available will be more helpful while doing analysis. before jumping on to the analysis get good understanding of what are data available.

Dimension table: It will have the static data like details of customer and products

Fact table: It will have the data about the transactions

- gdb041:
    - dim_customer
        - 27 distinct markets (ex India, USA, spain)
        - 75 distinct customers thorough out the market
        - 2 types of platforms
            - Brick & Motors - Physical/offline store
            - E-commerce - Online Store (Amazon, flipkart)
        - Three channels
            - Retailer
            - Direct
            - Distributors
    - dim_market
        - 27 distinct markets (ex India, USA, spain)
        - 7 sub-zones
        - 4 regions
            - APAC
            - EU

        - Nan
        - LATAM
    - dim_product
        - Divisions
        - P & A
            - Peripherals
            - Accessories
        - PC
            - Notebook
            - Desktop
        - N & S
            - Networking
            - Storage
        - There are 14 different categories, Like Internal HDD, keyboard
        - There are different variants available for the same product
    - fact_forecast_monthly
        - This table is used to forecast the customer's need in advance, which can help in
            - Higher customer satisfaction
            - Reduced cost in warehouses for storage purpose
        - The table is denormalized by data engineering team, as it is a data warehouse which is aimed to be used for analytical work.
        - All the date of the month will be replaced by the start date of the month
        - It will have all the column names and in the end it will have the forecast quantity need of the customer
    - fact_sales_monthly
        - This table is more or less is same as fact_forecase_monthly table, but the last column has the value of sold quantity instead of forecast value.
- gdb056
    - freight_cost
        - This table has details of travel cost and other cost for each market with fiscal year
    - gross_price
        - Has the details of gross prices with product code
    - manufacturing_cost
        - Has the details of manufacturing cost with product code with year
    - Pre_invoice_dedutions
        - Has the details of pre invoice deductions percentage for each cutomer with year
    - Post_invoice_deductions
        - Post invoice deductions and other deductions details

Finance Analytics

User-Defined SQL Functions

a. first grab customer codes for Croma india

SELECT * FROM dim_customer WHERE customer like "%croma%" AND market="india";

customer_code	customer	platform	channel	market	sub_zone	region
90002002	Croma	Brick & Mortar	Retailer	India	India	APAC
NULL	NULL	NULL	NULL	NULL	NULL	NULL

b. Get all the sales transaction data from fact_sales_monthly table for that customer(croma: 90002002) in the fiscal_year 2021

```
SELECT * FROM fact_sales_monthly
        WHERE
       customer_code=90002002 AND
       YEAR(DATE_ADD(date, INTERVAL 4 MONTH))=2021
        ORDER BY date asc
        LIMIT 100000;
```

date	fiscal_year	product_code	customer_code	sold_quantity
2020-09-01	2021	A0118150101	90002002	202
2020-09-01	2021	A0118150102	90002002	162
2020-09-01	2021	A0118150103	90002002	193
2020-09-01	2021	A0118150104	90002002	146
2020-09-01	2021	A0219150201	90002002	149
2020-09-01	2021	A0219150202	90002002	107

c. create a function 'get_fiscal_year' to get fiscal year by passing the date

```
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
```

d. Replacing the function created in the step:b

```
SELECT * FROM fact_sales_monthly
        WHERE
       customer_code=90002002 AND
       get_fiscal_year(date)=2021
        ORDER BY date asc
        LIMIT 100000;
```

date	fiscal_year	product_code	customer_code	sold_quantity
2020-09-01	2021	A0118150101	90002002	202
2020-09-01	2021	A0118150102	90002002	162
2020-09-01	2021	A0118150103	90002002	193
2020-09-01	2021	A0118150104	90002002	146
2020-09-01	2021	A0219150201	90002002	149

Gross Sales Report: Monthly Product Transactions

a. Perform joins to pull product information

```
SELECT s.date, s.product_code, p.product, p.variant, s.sold_quantity
        FROM fact_sales_monthly s
        JOIN dim_product p
    ON s.product_code=p.product_code
```

```
  WHERE
customer_code=90002002 AND
    get_fiscal_year(date)=2021
  LIMIT 1000000;
```

date	product_code	product	variant	sold_quantity
2020-09-01	A0118150101	AQ Dracula HDD ...	Standard	202
2020-09-01	A0118150102	AQ Dracula HDD ...	Plus	162
2020-09-01	A0118150103	AQ Dracula HDD ...	Premium	193
2020-09-01	A0118150104	AQ Dracula HDD ...	Premium Plus	146
2020-09-01	A0219150201	AQ WereWolf N...	Standard	149
2020-09-01	A0219150202	AQ WereWolf N...	Plus	107
2020-09-01	A0220150203	AQ WereWolf N...	Premium	123
2020-09-01	A0320150301	AQ Zion Saga	Standard	146
2020-09-01	A0321150302	AQ Zion Saga	Plus	236
2020-09-01	A0321150303	AQ Zion Saga	Premium	137
2020-09-01	A0418150103	AQ Mforce Gen X	Standard 3	23
2020-09-01	A0418150104	AQ Mforce Gen X	Plus 1	82
2020-09-01	A0418150105	AQ Mforce Gen X	Plus 2	86
2020-09-01	A0418150106	AQ Mforce Gen X	Plus 3	48

b. Performing join with 'fact_gross_price' table with the above query and generating required fields

```
SELECT s.date, s.product_code, p.product, p.variant, s.sold_quantity, g.gross_price,
      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;
```

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 Plus	146	22.9729	3354.04
2020-09-01	A0219150201	AQ WereWolf N...	Standard	149	23.6987	3531.11
2020-09-01	A0219150202	AQ WereWolf N...	Plus	107	24.7312	2646.24
2020-09-01	A0220150203	AQ WereWolf N...	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

Gross Sales Report: Total Sales Amount

a. Generate monthly gross sales report for Croma India for all the years

```
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;
```

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

Stored Procedures: Monthly Gross Sales Report

a. Generate monthly gross sales report for any customer using stored procedure

```
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
```

Stored Procedure: Market Badge

a. Write a stored proc that can retrieve market badge. i.e. if total sold quantity > 5 million that market is considered "Gold" else "Silver"

```
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
```

Market Analytics (Top Customers, Products, Markets)

Problem Statement and Pre-Invoice Discount Report

a. Include pre-invoice deductions in Croma detailed report

```
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,
       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.fiscal_year=get_fiscal_year(s.date)
```

```
    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=get_fiscal_year(s.date)
  WHERE
    s.customer_code=90002002 AND
    get_fiscal_year(s.date)=2021
  LIMIT 1000000;
```

date	product_code	product	variant	sold_quantity	gross_price_per_item	gross_price_total	pre_invoice_discount_pct
2020-09-01	A0118150101	AQ Dracula HDD ...	Standard	202	19.0573	3849.57	0.3025
2020-09-01	A0118150102	AQ Dracula HDD ...	Plus	162	21.4565	3475.95	0.3025
2020-09-01	A0118150103	AQ Dracula HDD ...	Premium	193	21.7795	4203.44	0.3025
2020-09-01	A0118150104	AQ Dracula HDD ...	Premium Plus	146	22.9729	3354.04	0.3025
2020-09-01	A0219150201	AQ WereWolf N...	Standard	149	23.6987	3531.11	0.3025
2020-09-01	A0219150202	AQ WereWolf N...	Plus	107	24.7312	2646.24	0.3025
2020-09-01	A0220150203	AQ WereWolf N...	Premium	123	23.6154	2904.69	0.3025
2020-09-01	A032015	Q Zion Saga	Standard	146	23.7223	3463.46	0.3025
2020-09-01	A0321150302	AQ Zion Saga	Plus	236	27.1027	6396.24	0.3025
2020-09-01	A0321150303	AQ Zion Saga	Premium	137	28.0059	3836.81	0.3025
2020-09-01	A0418150103	AQ Mforce Gen X	Standard 3	23	19.5235	449.04	0.3025
2020-09-01	A0418150104	AQ Mforce Gen X	Plus 1	82	19.9239	1633.76	0.3025

27.1027

b. Same report but all the customers

```
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,
      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.fiscal_year=get_fiscal_year(s.date)
          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=get_fiscal_year(s.date)
        WHERE
          get_fiscal_year(s.date)=2021
        LIMIT 1000000;
```

date	product_code	product	variant	sold_quantity	gross_price_per_item	gross_price_total	pre_invoice_discount_pct
2020-09-01	A0118150101	AQ Dracula HDD ...	Standard	248	19.0573	4726.21	0.0703
2020-09-01	A0118150101	AQ Dracula HDD ...	Standard	240	19.0573	4573.75	0.2061
2020-09-01	A0118150101	AQ Dracula HDD ...	Standard	31	19.0573	590.78	0.0974
2020-09-01	A0118150101	AQ Dracula HDD ...	Standard	37	19.0573	705.12	0.2065
2020-09-01	A0118150101	AQ Dracula HDD ...	Standard	7	19.0573	133.40	0.1068
2020-09-01	A0118150101	AQ Dracula HDD ...	Standard	12	19.0573	228.69	0.2612
2020-09-01	A0118150101	AQ Dracula HDD ...	Standard	17	19.0573	323.97	0.2471
2020-09-01	A0118150101	AQ Dracula HDD ...	Standard	60	19.0573	1143.44	0.0858
2020-09-01	A0118150101	AQ Dracula HDD ...	Standard	34	19.0573	647.95	0.2450
2020-09-01	A0118150101	AQ Dracula HDD ...	Standard	24	19.0573	457.38	0.0736
2020-09-01	A0118150101	AQ Dracula HDD ...	Standard	88	19.0573	1677.04	0.2105
2020-09-01	A0118150101	AQ Dracula HDD ...	Standard	49	19.0573	933.81	0.0793
2020-09-01	A0118150101	AQ Dracula HDD ...	Standard	60	19.0573	1143.44	0.1817

Performance Improvement # 1

a. creating dim_date and joining with this table and avoid using the function 'get_fiscal_year()' to reduce the amount of time taking to run the query

```
SELECT s.date, s.customer_code, 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_date dt
         ON dt.calendar_date = s.date
         JOIN dim_product p
         ON s.product_code=p.product_code
         JOIN fact_gross_price g
                ON g.fiscal_year=dt.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=dt.fiscal_year
         WHERE
                dt.fiscal_year=2021
         LIMIT 1500000;
```

date	product_code	product	variant	sold_quantity	gross_price_per_item	gross_price_total	pre_invoice_discount_pct
2020-09-01	A0118150101	AQ Dracula HDD ...	Standard	248	19.0573	4726.21	0.0703
2020-09-01	A0118150101	AQ Dracula HDD ...	Standard	240	19.0573	4573.75	0.2061
2020-09-01	A0118150101	AQ Dracula HDD ...	Standard	31	19.0573	590.78	0.0974
2020-09-01	A0118150101	AQ Dracula HDD ...	Standard	37	19.0573	705.12	0.2065
2020-09-01	A0118150101	AQ Dracula HDD ...	Standard	7	19.0573	133.40	0.1068
2020-09-01	A0118150101	AQ Dracula HDD ...	Standard	12	19.0573	228.69	0.2612
2020-09-01	A0118150101	AQ Dracula HDD ...	Standard	17	19.0573	323.97	0.2471
2020-09-01	A0118150101	AQ Dracula HDD ...	Standard	60	19.0573	1143.44	0.0858
2020-09-01	A0118150101	AQ Dracula HDD ...	Standard	34	19.0573	647.95	0.2450
2020-09-01	A0118150101	AQ Dracula HDD ...	Standard	24	19.0573	457.38	0.0736
2020-09-01	A0118150101	AQ Dracula HDD ...	Standard	88	19.0573	1677.04	0.2105
2020-09-01	A0118150101	AQ Dracula HDD ...	Standard	49	19.0573	933.81	0.0793
2020-09-01	A0118150101	AQ Dracula HDD ...	Standard	60	19.0573	1143.44	0.1817

Performance Improvement # 2

a. Added the fiscal year in the fact_sales_monthly table itself

```
SELECT s.date, s.customer_code, 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_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
WHERE s.fiscal_year=2021
LIMIT 1500000;
```

date	customer_code	product_code	product	variant	sold_quantity	gross_price_per_item	gross_price_total	pre_invoice_discount_pct
2020-09-01	70002017	A0118150101	AQ Dracula HDD ...	Standard	248	19.0573	4726.21	0.0703
2020-09-01	70002018	A0118150101	AQ Dracula HDD ...	Standard	240	19.0573	4573.75	0.2061
2020-09-01	70003181	A0118150101	AQ Dracula HDD ...	Standard	31	19.0573	590.78	0.0974
2020-09-01	70003182	A0118150101	AQ Dracula HDD ...	Standard	37	19.0573	705.12	0.2065
2020-09-01	70004069	A0118150101	AQ Dracula HDD ...	Standard	7	19.0573	133.40	0.1068
2020-09-01	70004070	A0118150101	AQ Dracula HDD ...	Standard	12	19.0573	228.69	0.2612
2020-09-01	70005163	A0118150101	AQ Dracula HDD ...	Standard	17	19.0573	323.97	0.2471
2020-09-01	70006157	A0118150101	AQ Dracula HDD ...	Standard	60	19.0573	1143.44	0.0858
2020-09-01	70006158	A0118150101	AQ Dracula HDD ...	Standard	34	19.0573	647.95	0.2450
2020-09-01	70007198	A0118150101	AQ Dracula HDD ...	Standard	24	19.0573	457.38	0.0736
2020-09-01	70007199	A0118150101	AQ Dracula HDD ...	Standard	88	19.0573	1677.04	0.2105

Database Views: Introduction

a. Get the net_invoice_sales amount using the CTE's

```
WITH cte1 AS (
        SELECT s.date, s.customer_code, 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_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
        WHERE
                s.fiscal_year=2021)
SELECT
   *,
   (gross_price_total-pre_invoice_discount_pct*gross_price_total) as net_invoice_sales
FROM cte1
LIMIT 1500000;
```

date	customer_code	product_code	product	variant	sold_quantity	gross_price_per_item	gross_price_total	pre_invoice_discount_pct	net_invoice_sales
2020-09-01	70002017	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 R...	Standard	248	19.0573	4726.21	0.0703	4393.957437
2020-09-01	70002018	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 R...	Standard	240	19.0573	4573.75	0.2061	3631.100125
2020-09-01	70003181	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 R...	Standard	31	19.0573	590.78	0.0974	533.238028
2020-09-01	70003182	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 R...	Standard	37	19.0573	705.12	0.2065	559.512720
2020-09-01	70004069	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 R...	Standard	7	19.0573	133.40	0.1068	119.152880
2020-09-01	70004070	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 R...	Standard	12	19.0573	228.69	0.2612	168.956172
2020-09-01	70005163	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 R...	Standard	17	19.0573	323.97	0.2471	243.917013
2020-09-01	70006157	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 R...	Standard	60	19.0573	1143.44	0.0858	1045.332848
2020-09-01	70006158	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 R...	Standard	34	19.0573	647.95	0.2450	489.202250
2020-09-01	70007198	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 R...	Standard	24	19.0573	457.38	0.0736	423.716832
2020-09-01	70007199	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 R...	Standard	88	19.0573	1677.04	0.2105	1324.023080
2020-09-01	70008169	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 R...	Standard	49	19.0573	933.81	0.0793	859.758867
2020-09-01	70008170	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 R...	Standard	60	19.0573	1143.44	0.1817	935.676952

b. Creating the view `sales_preinv_discount` and store all the data in like a virtual table

```
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
```

c. Now generate net_invoice_sales using the above created view "sales_preinv_discount"

```
SELECT *, (gross_price_total-pre_invoice_discount_pct*gross_price_total) as net_invoice_sales
FROM gdb0041.sales_preinv_discount
```

Database Views: Post Invoice Discount, Net Sales

a. Create a view for post invoice deductions: `sales_postinv_discount`

```
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;
```

b. Create a report for net sales

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

c. Finally creating the view `net_sales` which inbuiltly use/include all the previous created view and gives the final result

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

Top Markets and Customers

a. Get top 5 market by net sales in fiscal year 2021

```
SELECT market, round(sum(net_sales)/1000000,2) as net_sales_mln
        FROM gdb0041.net_sales
        where fiscal_year=2021
        group by market
        order by net_sales_mln desc
        limit 5
```

market	net_sales_mln
India	210.65
USA	132.04
South Korea	64.01
Canada	45.89
United Kingdom	44.73

b. Stored proc to get top n markets by net sales for a given year

```
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
```

c. 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

```
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
```

Window Functions: OVER Clause

a. show % of total expense

```
select *, amount*100/sum(amount) over() as pct
from random_tables.expenses
order by category;
```

date	description	category	amount	pct
2022-10-25	A2B restaurant	Food	6000	9.1185
2022-10-02	Macdonalds	Food	2700	4.1033
2022-10-10	Pani puri on street	Food	400	0.6079
2022-10-01	Saravana bhavan	Food	2700	4.1033
2022-10-02	Amazon	Shopping	3000	4.5593
2022-10-02	Croma store	Shopping	13000	19.7568
2022-10-18	D Mart grocery bill	Shopping	4300	6.5350
2022-10-18	Thakur saris	Shopping	23000	34.9544
2022-10-18	Banglore muni water bill	Utilities	600	0.9119
2022-10-05	PSEG electricity bill	Utilities	7000	10.6383
2022-10-10	Reliance geo phone bill	Utilities	800	1.2158
2022-10-17	Verizon wireless	Utilities	2300	3.4954

b. show % of total expense per category

```
select *, amount*100/sum(amount) over(partition by category) as pct
from random_tables.expenses
order by category,  pct desc;
```

date	description	category	amount	pct
2022-10-25	A2B restaurant	Food	6000	50.8475
2022-10-02	Macdonalds	Food	2700	22.8814
2022-10-01	Saravana bhavan	Food	2700	22.8814
2022-10-10	Pani puri on street	Food	400	3.3898
2022-10-18	Thakur saris	Shopping	23000	53.1178
2022-10-02	Croma store	Shopping	13000	30.0231
2022-10-18	D Mart grocery bill	Shopping	4300	9.9307
2022-10-02	Amazon	Shopping	3000	6.9284
2022-10-05	PSEG electricity bill	Utilities	7000	65.4206
2022-10-17	Verizon wireless	Utilities	2300	21.4953
2022-10-10	Reliance geo pho...	Utilities	800	7.4766
2022-10-18	Banglore muni wa...	Utilities	600	5.6075

c. Show expenses per category till date

```
select *, sum(amount) over(partition by category order by date) as expenses_till_date
from random_tables.expenses;
```

	date	description	category	amount	expenses_till_date
▶	2022-10-01	Saravana bhavan	Food	2700	2700
	2022-10-02	Macdonalds	Food	2700	5400
	2022-10-10	Pani puri on street	Food	400	5800
	2022-10-25	A2B restaurant	Food	6000	11800
	2022-10-02	Amazon	Shopping	3000	16000
	2022-10-02	Croma store	Shopping	13000	16000
	2022-10-18	D Mart grocery bill	Shopping	4300	43300
	2022-10-18	Thakur saris	Shopping	23000	43300
	2022-10-05	PSEG electricity bill	Utilities	7000	7000
	2022-10-10	Reliance geo phone bill	Utilities	800	7800
	2022-10-17	Verizon wireless	Utilities	2300	10100
	2022-10-18	Banglore muni water bill	Utilities	600	10700

Window Functions: Using it In a Task

a. find out customer wise net sales percentage contribution

```
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
```

Window Functions: OVER Clause

a. Find customer wise net sales distibution per region for FY 2021

```
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
```

Window Functions: ROW_NUMBER, RANK, DENSE_RANK

a. Find out top 3 products from each division by total quantity sold in a given year

```
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 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
```

b. Creating stored procedure for the above query

```
CREATE PROCEDURE `get_top_n_products_per_division_by_qty_sold`(
    in_fiscal_year INT, in_top_n INT )
    BEGIN
        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 fiscal_year=in_fiscal_year
        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 <= in_top_n;
    END
```

Supply Chain Analytics

Create a Helper Table

a. Create fact_act_est table

```
drop table if exists fact_act_est;

    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, customer_code, product_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, customer_code, product_code)
);

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;
```

Database Triggers

a. create the trigger to automatically insert record in fact_act_est table whenever insertion happens in fact_sales_monthly

```
DELIMITER //

CREATE DEFINER=CURRENT_USER TRIGGER `fact_sales_monthly_AFTER_INSERT`
AFTER INSERT ON `fact_sales_monthly`
FOR EACH ROW
BEGIN
  INSERT INTO fact_act_est
    (date, product_code, customer_code, sold_quantity)
  VALUES
    (NEW.date, NEW.product_code, NEW.customer_code, NEW.sold_quantity)
  ON DUPLICATE KEY UPDATE
    sold_quantity = NEW.sold_quantity;
END;

//

DELIMITER ;
```

b. create the trigger to automatically insert record in fact_act_est table whenever insertion happens in fact_forecast_monthly

```
DELIMITER //
```

```
CREATE DEFINER=CURRENT_USER TRIGGER `fact_forecast_monthly_AFTER_INSERT` AFTER INSERT ON
`fact_forecast_monthly` FOR EACH ROW
        BEGIN
                insert into fact_act_est
                        (date, product_code, customer_code, forecast_quantity)
                values (
                        NEW.date,
                        NEW.product_code,
                        NEW.customer_code,
                        NEW.forecast_quantity
                 )
                on duplicate key update
                        forecast_quantity = values(forecast_quantity);
        END;

//

DELIMITER ;
```

c. To see all the Triggers

```
show triggers;
```

Trigger	Event	Table	Statement	Timing	Created	sql_mode	Definer	character_set_client
fact_forecast_monthly_AFTER_INSERT	INSERT	fact_forecast_monthly	BEGIN insert into fact_act_est ...	AFTER	2024-06-11 22:21:06.65	ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLE...	root@localhost	utf8mb4
fact_sales_monthly_AFTER_INSERT	INSERT	fact_sales_monthly	BEGIN INSERT INTO fact_act_est (date...	AFTER	2024-06-11 22:19:32.30	ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLE...	root@localhost	utf8mb4

d. Insert the records in the fact_sales_monthly and fact_forecast_monthly tables and check whether records inserted in fact_act_est table

```
insert into fact_sales_monthly
        (date, product_code, customer_code, sold_quantity)
        values
            ("2030-09-01", "HAHA", 99, 89);

        insert into fact_forecast_monthly
       (date, product_code, customer_code, forecast_quantity)
        values
            ("2030-09-01", "HAHA", 99, 43);

select * from fact_act_est where customer_code = 99;
```

Temporary Tables & Forecast Accuracy Report

a. Forecast accuracy report using cte (It exists at the scope of statements)

```
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-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=2021
        group by customer_code
         )
         select *,
        if (abs_error_pct > 100, 0, 100.0 - abs_error_pct) as forecast_accuracy
         from forecast_err_table
    order by forecast_accuracy desc;
```

b. Write a stored proc for the same

```
CREATE 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-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, 100.0 - abs_error_pct) as forecast_accuracy
             from forecast_err_table
         order by forecast_accuracy desc;
        END
```

c. Forecast accuracy report using temporary table (It exists for the entire session)

```
drop table if exists forecast_err_table;
          create temporary table forecast_err_table
         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-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=2021
        group by customer_code;

         select
        *,
        if (abs_error_pct > 100, 0, 100.0 - abs_error_pct) as forecast_accuracy
         from forecast_err_table
     order by forecast_accuracy desc;
```