ATLIQ HARDWARE SQL

CHALLENGE SOLUTIONS & EXPLANATION

**Requests:**

1. Provide the list of markets in which customer "Atliq Exclusive" operates its business in the APAC region.

**Solution**

**SELECT DISTINCT market**

**FROM gdb023.dim\_customer**

**WHERE customer = 'Atliq Exclusive'**

**AND region = 'APAC';**

**Explanation**

In this SQL solution, we are querying the 'gdb023.dim\_customer' table to obtain the list of markets where the customer 'Atliq Exclusive' operates in the APAC (Asia Pacific) region.

Breaking down the query:

1. \*\*SELECT DISTINCT market:\*\* We are selecting unique market values from the 'dim\_customer' table. The DISTINCT keyword ensures that we retrieve only distinct (unique) values.

2. \*\*FROM gdb023.dim\_customer:\*\* Specifies the source table for the data retrieval, which is the 'dim\_customer' table in the 'gdb023' database.

3. \*\*WHERE customer = 'Atliq Exclusive' AND region = 'APAC';:\*\* This part of the query filters the results. It ensures that we only consider records where the customer is 'Atliq Exclusive' and the region is 'APAC'.

In summary, the SQL query retrieves and displays distinct market values for the customer 'Atliq Exclusive' in the APAC region. This information helps identify the specific markets where Atliq Exclusive conducts its business in the Asia Pacific region.

2. What is the percentage of unique product increase in 2021 vs. 2020? The final output contains these fields,

unique\_products\_2020

unique\_products\_2021

percentage\_chg

**Solution**

SELECT

COUNT(DISTINCT CASE WHEN fiscal\_year = 2020 THEN product\_code END) AS unique\_products\_2020,

COUNT(DISTINCT CASE WHEN fiscal\_year = 2021 THEN product\_code END) AS unique\_products\_2021,

((COUNT(DISTINCT CASE WHEN fiscal\_year = 2021 THEN product\_code END) -

COUNT(DISTINCT CASE WHEN fiscal\_year = 2020 THEN product\_code END)) /

COUNT(DISTINCT CASE WHEN fiscal\_year = 2020 THEN product\_code END)) \* 100 AS percentage\_chg

FROM

gdb023.fact\_sales\_monthly

WHERE

fiscal\_year IN (2020, 2021);

**Explanation:**  This SQL solution calculates the percentage increase in the number of unique products between the fiscal years 2020 and 2021. It involves counting the distinct product codes for each year and then computing the percentage change.

***Breaking down the query:***

**1. SELECT:**

- COUNT(DISTINCT CASE WHEN fiscal\_year = 2020 THEN product\_code END) AS unique\_products\_2020: Counts the distinct product codes for the fiscal year 2020 and assigns the result to the alias 'unique\_products\_2020'.

- COUNT(DISTINCT CASE WHEN fiscal\_year = 2021 THEN product\_code END) AS unique\_products\_2021: Counts the distinct product codes for the fiscal year 2021 and assigns the result to the alias 'unique\_products\_2021'.

- ((COUNT(DISTINCT CASE WHEN fiscal\_year = 2021 THEN product\_code END) - COUNT(DISTINCT CASE WHEN fiscal\_year = 2020 THEN product\_code END)) / COUNT(DISTINCT CASE WHEN fiscal\_year = 2020 THEN product\_code END)) \* 100 AS percentage\_chg:

Calculates the percentage change by taking the difference between unique products in 2021 and 2020, dividing by the unique products in 2020, and then multiplying by 100. The result is assigned to the alias 'percentage\_chg'.

**2. FROM gdb023.fact\_sales\_monthly:**

Specifies the source table for the data retrieval, which is the 'fact\_sales\_monthly' table in the 'gdb023' database.

**3. WHERE fiscal\_year IN (2020, 2021):**

Filters the results to include only records from the fiscal years 2020 and 2021.

In summary, this SQL query provides a concise way to quantify the growth in unique products from 2020 to 2021, presenting the counts for each year and the percentage change.

3. Provide a report with all the unique product counts for each segment and sort them in descending order of product counts. The final output contains 2 fields,

segment

product\_count

**Solution**

SELECT segment, COUNT(DISTINCT product\_code) AS product\_count

FROM gdb023.dim\_product

GROUP BY segment

ORDER BY product\_count DESC;

**Explanation:**

This SQL solution generates a report presenting the unique product counts for each segment, sorted in descending order based on the product counts.

***Breaking down the query:***

**1. SELECT segment, COUNT(DISTINCT product\_code) AS product\_count:**

- segment: The 'segment' column is selected to be included in the result.

- COUNT(DISTINCT product\_code) AS product\_count: Counts the distinct product codes within each segment and labels the result as 'product\_count'.

**2. FROM gdb023.dim\_product:**

- Specifies the source table for data retrieval, which is the 'dim\_product' table in the 'gdb023' database.

**3.GROUP BY segment:**

- Groups the data by the 'segment' column, so the subsequent aggregate function is applied to each unique segment.

**4. ORDER BY product\_count DESC:**

- Orders the result set in descending order based on the 'product\_count' column, ensuring the segments with the highest product counts appear first.

In summary, this SQL query efficiently summarizes the unique product counts for each segment and arranges the results in a descending order of product counts, providing a clear overview of the distribution of products across different segments.

4. Follow-up: Which segment had the most increase in unique products in 2021 vs 2020? The final output contains these fields, segment, product\_count\_2020, product\_count\_2021, difference

**Solution**

WITH product\_counts AS (

SELECT

segment,

COUNT(DISTINCT CASE WHEN fiscal\_year = 2020 THEN dim\_product.product\_code END) AS product\_count\_2020,

COUNT(DISTINCT CASE WHEN fiscal\_year = 2021 THEN dim\_product.product\_code END) AS product\_count\_2021

FROM

gdb023.dim\_product

INNER JOIN

gdb023.fact\_sales\_monthly ON gdb023.dim\_product.product\_code = gdb023.fact\_sales\_monthly.product\_code

WHERE

fiscal\_year IN (2020, 2021)

GROUP BY

segment

)

SELECT

segment,

product\_count\_2020,

product\_count\_2021,

(product\_count\_2021 - product\_count\_2020) AS difference

FROM

product\_counts

ORDER BY

difference DESC

;

**Explanation:**

This SQL solution identifies the segment with the most increase in unique products in 2021 compared to 2020. It provides a clear breakdown of the product counts for each segment in both years, along with the calculated difference.

***Breaking down the query:***

**1. WITH product\_counts AS (...):**

- Defines a Common Table Expression (CTE) named 'product\_counts' to calculate the product counts for each segment in both 2020 and 2021.

- Uses a CASE WHEN statement to count distinct product codes based on the fiscal year.

**2. SELECT segment, product\_count\_2020, product\_count\_2021, (product\_count\_2021 - product\_count\_2020) AS difference:**

- Retrieves the segment, product counts for 2020 and 2021, and calculates the difference between the counts.

- The SELECT statement references the 'product\_counts' CTE.

**3. FROM product\_counts:**

- Specifies the source of data as the 'product\_counts' CTE, ensuring that the aggregated product counts are used for further analysis.

**4. ORDER BY difference DESC:**

- Orders the result set based on the calculated difference in descending order, so the segment with the highest increase appears first.

In summary, this SQL query efficiently compares unique product counts for each segment in 2020 and 2021, highlighting the segment with the most significant increase in unique products. The 'difference' column provides a clear indicator of the change in product counts, aiding in strategic decision-making for Atliq Hardwares.

5. Get the products that have the highest and lowest manufacturing costs. The final output should contain these fields,

product\_code, product, manufacturing\_cost

**Solution**

-- Highest Manufacturing Cost

SELECT

dp.product\_code,

dp.product,

fmc.manufacturing\_cost

FROM

gdb023.dim\_product dp

INNER JOIN

gdb023.fact\_manufacturing\_cost fmc ON dp.product\_code = fmc.product\_code

ORDER BY

fmc.manufacturing\_cost DESC

LIMIT 1;

-- Lowest Manufacturing Cost

SELECT

dp.product\_code,

dp.product,

fmc.manufacturing\_cost

FROM

gdb023.dim\_product dp

INNER JOIN

gdb023.fact\_manufacturing\_cost fmc ON dp.product\_code = fmc.product\_code

ORDER BY

fmc.manufacturing\_cost ASC

LIMIT 1;

**Explanation:**

This SQL solution retrieves the products with the highest and lowest manufacturing costs from the given database. It presents two separate queries, one for the highest manufacturing cost and the other for the lowest.

**Highest Manufacturing Cost:**

- SELECT dp.product\_code, dp.product, fmc.manufacturing\_cost:

- Specifies the columns to be retrieved: product\_code, product, and manufacturing\_cost.

- FROM gdb023.dim\_product dp INNER JOIN gdb023.fact\_manufacturing\_cost fmc ON dp.product\_code = fmc.product\_code:

- Joins the 'dim\_product' and 'fact\_manufacturing\_cost' tables based on the product\_code to get relevant information.

- ORDER BY fmc.manufacturing\_cost DESC:

-Orders the result set in descending order based on manufacturing cost, ensuring the product with the highest cost appears first.

- **LIMIT 1:**

- Restricts the result set to only one row, representing the product with the highest manufacturing cost.

**Lowest Manufacturing Cost:**

- SELECT dp.product\_code, dp.product, fmc.manufacturing\_cost:

- Similar to the highest cost query, it selects the same columns.

- FROM gdb023.dim\_product dp INNER JOIN gdb023.fact\_manufacturing\_cost fmc ON dp.product\_code = fmc.product\_code:

- Joins the relevant tables.

- ORDER BY fmc.manufacturing\_cost ASC:

- Orders the result set in ascending order, placing the product with the lowest manufacturing cost first.

**- LIMIT 1:**

- Limits the result set to one row.

In summary, these queries efficiently retrieve the product information with the highest and lowest manufacturing costs separately, allowing Atliq Hardwares to identify both ends of the cost spectrum in their product lineup.

6. Generate a report which contains the top 5 customers who received an average high pre\_invoice\_discount\_pct for the fiscal year 2021 and in the Indian market. The final output contains these fields, customer\_code, customer, average\_discount\_percentage

**Solution**

SELECT

gdb023.dim\_customer.customer\_code,

customer,

AVG(pre\_invoice\_discount\_pct) AS average\_discount\_percentage

FROM

gdb023.dim\_customer

INNER JOIN

gdb023.fact\_pre\_invoice\_deductions ON gdb023.dim\_customer.customer\_code = gdb023.fact\_pre\_invoice\_deductions.customer\_code

WHERE

gdb023.fact\_pre\_invoice\_deductions.fiscal\_year = 2021

AND gdb023.dim\_customer.market = 'India'

GROUP BY

customer\_code, customer

ORDER BY

average\_discount\_percentage DESC

LIMIT 5;

**Explanation:**

This SQL solution generates a report to identify the top 5 customers who received the highest average pre-invoice discount percentage in the fiscal year 2021 within the Indian market.

- SELECT gdb023.dim\_customer.customer\_code, customer, AVG(pre\_invoice\_discount\_pct) AS average\_discount\_percentage:

- Specifies the columns to be retrieved: customer\_code, customer, and the average pre\_invoice\_discount\_pct.

- FROM gdb023.dim\_customer INNER JOIN gdb023.fact\_pre\_invoice\_deductions ON gdb023.dim\_customer.customer\_code = gdb023.fact\_pre\_invoice\_deductions.customer\_code:

- Joins the 'dim\_customer' and 'fact\_pre\_invoice\_deductions' tables based on the customer\_code to get relevant information.

- WHERE gdb023.fact\_pre\_invoice\_deductions.fiscal\_year = 2021 AND gdb023.dim\_customer.market = 'India':

- Filters the results to include only records for the fiscal year 2021 and within the Indian market.

- GROUP BY customer\_code, customer:

- Groups the results by customer\_code and customer to calculate the average discount for each customer.

- ORDER BY average\_discount\_percentage DESC:

- Orders the result set in descending order based on the average pre-invoice discount percentage.

- LIMIT 5:

- Restricts the result set to only the top 5 customers with the highest average pre-invoice discount percentage.

This solution provides Atliq Hardwares with insights into the customers who received the most significant average pre-invoice discounts in 2021 within the Indian market.

7. Get the complete report of the Gross sales amount for the customer “Atliq Exclusive” for each month . This analysis helps to get an idea of low and high-performing months and take strategic decisions. The final report contains these columns:

Month, Year, Gross sales Amount

**Solution**

SELECT

MONTH(fsm.date) AS Month,

MONTHNAME(MAX(fsm.date)) AS MonthName,

YEAR(MAX(fsm.date)) AS Year,

SUM(fgp.gross\_price \* fsm.sold\_quantity) AS Gross\_Sales\_Amount

FROM

gdb023.fact\_sales\_monthly fsm

INNER JOIN

gdb023.dim\_customer dc ON fsm.customer\_code = dc.customer\_code

INNER JOIN

gdb023.fact\_gross\_price fgp ON fsm.product\_code = fgp.product\_code

WHERE

dc.customer = 'Atliq Exclusive'

GROUP BY

MONTH(fsm.date), YEAR(fsm.date)

ORDER BY

YEAR(fsm.date), MONTH(fsm.date);

**Explanation:**

This SQL solution generates a report of the gross sales amount for the customer "Atliq Exclusive" for each month. The analysis helps to understand the performance trends, identifying low and high-performing months, and facilitating strategic decision-making.

- SELECT MONTH(fsm.date) AS Month, MONTHNAME(MAX(fsm.date)) AS MonthName, YEAR(MAX(fsm.date)) AS Year, SUM(fgp.gross\_price \* fsm.sold\_quantity) AS Gross\_Sales\_Amount:

- Specifies the columns to be retrieved:

- Month: Extracts the month from the date.

- MonthName: Retrieves the name of the month.

- Year: Extracts the year from the date.

- Gross\_Sales\_Amount: Calculates the total gross sales amount by multiplying the gross\_price with sold\_quantity and summing the results.

- FROM gdb023.fact\_sales\_monthly fsm INNER JOIN gdb023.dim\_customer dc ON fsm.customer\_code = dc.customer\_code INNER JOIN gdb023.fact\_gross\_price fgp ON fsm.product\_code = fgp.product\_code:

- Joins the 'fact\_sales\_monthly,' 'dim\_customer,' and 'fact\_gross\_price' tables based on the respective keys to get the required information.

- WHERE dc.customer = 'Atliq Exclusive':

- Filters the results to include only records related to the customer 'Atliq Exclusive.'

- GROUP BY MONTH(fsm.date), YEAR(fsm.date):

- Groups the results by month and year to calculate the total gross sales amount for each month.

- ORDER BY YEAR(fsm.date), MONTH(fsm.date):

- Orders the result set in chronological order based on the year and month.

This solution provides a detailed report to Atliq Hardwares, highlighting the gross sales amount for each month, enabling them to identify patterns and make informed decisions based on monthly performance.

8. In which quarter of 2020, got the maximum total\_sold\_quantity? The final output contains these fields sorted by the total\_sold\_quantity, Quarter, total\_sold\_quantity

**Solution**

SELECT

QUARTER(fsm.date) AS Quarter,

SUM(fsm.sold\_quantity) AS total\_sold\_quantity

FROM

gdb023.fact\_sales\_monthly fsm

WHERE

fsm.fiscal\_year = 2020

GROUP BY

Quarter

ORDER BY

total\_sold\_quantity DESC

;

**Explanation:** This SQL solution determines in which quarter of 2020 the maximum total\_sold\_quantity was achieved. The result set includes the quarter and the corresponding total\_sold\_quantity, sorted in descending order of total\_sold\_quantity.

- SELECT QUARTER(fsm.date) AS Quarter, SUM(fsm.sold\_quantity) AS total\_sold\_quantity:

- Specifies the columns to be retrieved:

- Quarter: Extracts the quarter from the date.

- total\_sold\_quantity: Calculates the sum of sold\_quantity for each quarter.

- FROM gdb023.fact\_sales\_monthly fsm:

- Specifies the 'fact\_sales\_monthly' table to retrieve data from.

- WHERE fsm.fiscal\_year = 2020:

- Filters the results to include only records from the fiscal year 2020.

- GROUP BY Quarter:

- Groups the results by quarter to calculate the total sold quantity for each quarter.

- ORDER BY total\_sold\_quantity DESC:

- Orders the result set in descending order based on the total\_sold\_quantity.

This solution provides insights into the sales performance for each quarter of the year 2020, helping Atliq Hardwares identify the quarter with the highest total\_sold\_quantity and make data-informed decisions.

9. Which channel helped to bring more gross sales in the fiscal year 2021 and the percentage of contribution? The final output contains these fields,

Channel, gross\_sales\_mln, percentage

**Solution**

WITH ChannelSales AS (

SELECT

dc.channel,

SUM(fgp.gross\_price \* fsm.sold\_quantity) AS gross\_sales\_mln

FROM

gdb023.fact\_sales\_monthly fsm

INNER JOIN

gdb023.dim\_customer dc ON fsm.customer\_code = dc.customer\_code

INNER JOIN

gdb023.fact\_gross\_price fgp ON fsm.product\_code = fgp.product\_code

WHERE

fsm.fiscal\_year = 2021

GROUP BY

dc.channel

)

SELECT

cs.channel,

cs.gross\_sales\_mln,

(cs.gross\_sales\_mln / total.total\_sales) \* 100 AS percentage

FROM

ChannelSales cs,

(SELECT SUM(fgp.gross\_price \* fsm.sold\_quantity) AS total\_sales FROM gdb023.fact\_sales\_monthly fsm INNER JOIN gdb023.fact\_gross\_price fgp ON fsm.product\_code = fgp.product\_code WHERE fsm.fiscal\_year = 2021) total

ORDER BY

cs.gross\_sales\_mln DESC;

**Explanation:**

This SQL solution assesses the contribution of different sales channels to the gross sales of Atliq Hardwares in the fiscal year 2021. It employs a Common Table Expression (CTE) named 'ChannelSales' to calculate the gross sales (in million) for each channel. The main query then calculates the percentage contribution of each channel by dividing its gross sales by the total sales for the fiscal year 2021.

- WITH ChannelSales AS:

- Creates a CTE to compute the gross sales (in million) for each channel by joining relevant tables and filtering for the fiscal year 2021.

- SELECT cs.channel, cs.gross\_sales\_mln, (cs.gross\_sales\_mln / total.total\_sales) \* 100 AS percentage:

- Retrieves the channel, gross sales in million, and the percentage contribution.

- The percentage contribution is calculated by dividing the channel's gross sales by the total sales for fiscal year 2021 and multiplying by 100.

- FROM ChannelSales cs, (SELECT SUM... WHERE fsm.fiscal\_year = 2021) total:

- Combines the CTE with a subquery to obtain the total sales for the fiscal year 2021.

- ORDER BY cs.gross\_sales\_mln DESC:

- Orders the result set in descending order based on gross sales.

This analysis offers insights into the performance of sales channels, identifying which channels contributed the most to gross sales in the fiscal year 2021. This information is crucial for strategic decision-making at Atliq Hardwares.

10. Get the Top 3 products in each division that have a high total\_sold\_quantity in the fiscal\_year 2021? The final output contains these fields,

Division, product\_code

**Solution**

WITH RankedProducts AS (

SELECT

dp.division,

dp.product\_code,

dp.product,

SUM(fsm.sold\_quantity) AS total\_sold\_quantity,

RANK() OVER (PARTITION BY dp.division ORDER BY SUM(fsm.sold\_quantity) DESC) AS rank\_order

FROM

gdb023.dim\_product dp

INNER JOIN

gdb023.fact\_sales\_monthly fsm ON dp.product\_code = fsm.product\_code

WHERE

fsm.fiscal\_year = 2021

GROUP BY

dp.division, dp.product\_code, dp.product

)

SELECT

division,

product\_code,

product,

total\_sold\_quantity,

rank\_order

FROM

RankedProducts

WHERE

rank\_order <= 3

ORDER BY

division, rank\_order;

**Explanation:**

This SQL solution aims to identify the top 3 products with the highest total\_sold\_quantity in each division for the fiscal year 2021. It utilizes a Common Table Expression (CTE) named 'RankedProducts' to calculate the total sold quantity and assign a rank to each product within its division based on the sold quantity. The main query then filters for products with a rank order of 3 or below, providing the top 3 products for each division.

- WITH RankedProducts AS:

- Creates a CTE to compute the total sold quantity and rank order for each product within its division.

- Uses the RANK() window function to assign a rank based on the sold quantity in descending order.

- SELECT division, product\_code, product, total\_sold\_quantity, rank\_order:

- Retrieves the division, product\_code, product, total sold quantity, and rank order.

- FROM RankedProducts WHERE rank\_order <= 3:

- Filters the results to include only products with a rank order of 3 or below.

- ORDER BY division, rank\_order:

- Orders the result set first by division and then by rank order.

This solution provides valuable insights into the best-performing products within each division for the fiscal year 2021, assisting Atliq Hardwares in focusing on and optimizing its product portfolio.