PROJECT REPORT

CONTENTS:

PROBLEM STATEMENT:

To analyze and seek insights for different scenarios from the composite data of a business organization,confined to the 'sales and delivery' domain is given for the period of the last decade.

- DATASET: SALES AND DELIVERY
- DATA DESCRIPTION

List Of Tables:

1) **Cust_dimen:**

Column Name	**Data type**	**description**
Customer_name	Varchar	Name of the customer
Province	Varchar	ID for Province
Region	Varchar	ID for Region
Customer_Segment	Varchar	Types of the customer segments
Cust_id	Varchar	Id to the customers

Field	Type	Null	Key	Default	Extra
Customer_Name	text	YES		NULL	
Province	text	YES		NULL	
Region	text	YES		NULL	
Customer_Segment	text	YES		NULL	
Cust_id	text	YES		NULL	

There are 5 columns in the table and all of them can contain null values.

2) **Market Fact:**

Column Name	**Data Types**	**description**
Ord id	Varchar	Id for the Order
Prod id	Varchar	Id for the product
Ship id	Varchar	ID for the shipping
Cust id	Varchar	ID for the customer
Sales	Float	The sales price for the product
Discount	Float	Discount for the product
Order Quantity	Float	Number of products have been ordered
Profit	Float	Profit that has been gained from the product
Shipping cost	Float	Shipping cost for the product
Product Base Margin	Float	Base margin value for the product

Field	Type	Null	Key	Default	Extra
Ord_id	text	YES		NULL	
Prod_id	text	YES		NULL	
Ship_id	text	YES		NULL	
Cust_id	text	YES		NULL	
Sales	double	YES		NULL	
Discount	double	YES		NULL	
Order_Quantity	int	YES		NULL	
Profit	double	YES		NULL	
Shipping_Cost	double	YES		NULL	
Product_Base_Margin	double	YES		NULL	

This table contains 10 columns. The table contains columns with datatypes varchar, float, and int.

3) Orders_Dimen:

Column Name	Data type	Description
Order id	integer	Id for the order
Order Date	Varchar	The order date for that order has been ordered
Order Priority	varchar	Priority for the orders
Ord id	varchar	Order id as a varchar

Field	Type	Null	Key	Default	Extra
Order_ID	int	YES		NULL	
Order_Date	text	YES		NULL	
Order_Priority	text	YES		NULL	
Ord_id	text	YES		NULL	

This table consists of 4 columns. Three of the columns have text datatype and one column has integer datatype. All the columns can contain null values.

4) Prod_Dimen:

Column Name	Data type	Description
Product_Category	Varchar	Type of the product
Product_Sub_category	Varchar	Name of the sub-category
Prod_id	Varchar	Id of Product

Field	Type	Null	Key	Default	Extra
Product_Category	text	YES		NULL	
Product_Sub_Category	text	YES		NULL	
Prod_id	text	YES		NULL	

This table contains 3 columns. All three have char datatype and contain null values.

5) **Shipping Dimen:**

Column Name	Data Type	Description
Order_ID	Varchar	Id for the orders
Ship_Mode	Varchar	Type of the shipping
Ship_Date	Varchar	Shipping date
Ship_ID	Varchar	ID for the shipping

Field	Type	Null	Key	Default	Extra
Order_ID	int	YES		NULL	
Ship_Mode	text	YES		NULL	
Ship_Date	text	YES		NULL	
Ship_id	text	YES		NULL	

This table contains 4 columns. Three out of four columns have text values and one column has integer datatype.

Changes made in the tables:

- In the table order_dimen, Changed the datatype of column, order_date from text to date.
- In the table order_dimen, Changed the datatype of column, order_date from text to date.

```
ALTER TABLE orders_dimen MODIFY Order_Date DATE;
ALTER TABLE shipping_dimen MODIFY Ship_Date DATE;
```

- Created a new table called 'Delivery' and populated it with data from the orders_dimen and shipping_dimen. In this table, we created a new column, 'DaysTakenForDelivery' that contains the date difference between Order_Date and Ship_Date.

```
CREATE TABLE Delivery AS (SELECT od.order_id,
    od.order_date,
    od.ord_id,
    sd.ship_id,
    sd.ship_date FROM
    orders_dimen od
        INNER JOIN
    shipping_dimen sd ON od.ORDER_ID = sd.ORDER_ID);

ALTER TABLE Delivery ADD COLUMN DaysTakenForDelivery INT AS (DATEDIFF(ship_date,order_date)) ;
```

1) Objective:

To find the top 3 customers who have the maximum number of orders.

Query:

```
SELECT
    cd.customer_name, COUNT(mf.ord_id) AS number_of_orders
FROM
    cust_dimen cd
        INNER JOIN
    market_fact mf ON cd.cust_id = mf.cust_id
GROUP BY cd.cust_id
ORDER BY number_of_orders DESC
LIMIT 3;
```

Output:

cust_id	customer_name	number_of_orders
Cust_1140	PATRICK JONES	30
Cust_572	LENA CREIGHTON	21
Cust_444	BILL DONATELLI	21

Explanation:

In this query, we used an inner join between the customer dimension table and the market fact table on the common field of the customer ID. The COUNT function is used to count the number of order IDs in the market_fact table that match each customer ID. GROUP BY groups the resultset by the customer ID to ensure that (COUNT) is applied to each customer separately.

Inference:

The top three customers with maximum number of orders are Patrick, Lena, and Bill with 30,21,and 21 orders respectively.

This SQL query is helpful in providing valuable insights for sales and marketing strategies.

2) Objective:

Find the customer whose order took the maximum time to get delivered.

Query:

```
SELECT
    c.Cust_id,
    c.Customer_Name,
    d.order_id,
    d.DaysTakenForDelivery
FROM
    delivery d
        INNER JOIN
    market_fact m ON d.ord_id = m.ord_id
        INNER JOIN
    cust_dimen c ON m.Cust_id = c.Cust_id
ORDER BY DaysTakenForDelivery DESC
LIMIT 1;
```

Output:

	Cust_id	Customer_Name	order_id	DaysTakenForDelivery
▶	Cust_1460	DEAN PERCER	353	92

Explanation:

We used an inner join between the delivery table and the market fact table on the common field of the order ID (ord_id). It then uses another inner join with the customer dimension table. The ORDER BY clause sorts the result set in descending order of the number of days taken for delivery. 'LIMIT' clause limits the result set to only the first row.

Inference:

The customer whose order took the maximum time to get delivered is Dean Percer, who had to wait for 92 days to get his order delivered. This analysis can be used for further analysis to identify any issues in the delivery process and improve customer satisfaction.

3) Objective**:**

To retrieve total sales made by each product from the data.

Query**:**

```
select distinct prod_id,
 sum(sales) OVER(partition by prod_id ) as total_sales
 from market_fact order by total_sales DESC;
```

Output:

Number of rows: 17

prod_id	total_sales
Prod_17	2168697.14
Prod_4	1889313.8
Prod_11	1786776.75
Prod_15	1652823
Prod_14	1130361.3
Prod_1	1028240.76
Prod_3	1022957.59
Prod_10	814425.9
Prod_8	795875.94
Prod_2	736991.54
Prod_5	698093.81
Prod_6	446452.86
Prod_9	174085.8
Prod_13	167107.22
Prod_16	80996.31
Prod_12	38981.55
Prod_7	15006.63

Explanation:

Here, DISTINCT keyword is used to retrieve only unique product IDs from the market fact table. SUM is used to calculate the total sales for each product ID by partitioning the result set by product ID using the OVER clause. This creates a separate group of rows for each product ID, and the SUM function is applied to each group to calculate the total sales.

Inference:

- From the output the product with product id 17 has made the maximum sales of more than 21 lakhs. And product id 7 made least sales of only 15 thousand.
- This query helps in analyzing the sales performance of different products and identifying the products with the highest total sales.

4) Objective**:**

To retrieve the total profit made from each product from the data.

Query:

```
SELECT DISTINCT prod_id,
 ROUND(SUM(profit) OVER(PARTITION BY prod_id),2) AS total_product_profit
FROM market_fact order by total_product_profit desc;
```

Output:

Number of rows: 17

prod_id	total_product_profit
Prod_4	316951.62
Prod_17	307712.93
Prod_3	307413.39
Prod_14	167361.49
Prod_15	122738.07
Prod_5	100427.93
Prod_2	97158.06
Prod_8	94287.48
Prod_9	48182.6
Prod_6	45263.2
Prod_12	13677.17
Prod_1	13599.49
Prod_13	7564.78
Prod_7	-102.67
Prod_16	-7799.25
Prod_10	-33729.09
Prod_11	-113468.18

Explanation:

We used SUM with the OVER clause to calculate the total profit for each product ID by partitioning the result set by product ID. This creates a separate group of rows for each product ID, and the SUM function is applied to each group to calculate the total profit.

Inference:

It can be seen that product 4 did the highest profit of more than 3 lakhs whereas product 7, product 16, product 10, and product 11 led to losses (product 11 was seen to have the highest loss of more than 1 lakh. It can also be observed that product 10 has fairly high sales but that is not turning into profits.

We should discuss this with the organization and try to find the reasons for it.

5) Objective**:**

To find the total number of unique customers in January and how many of them came back every month over the entire year in 2011.

Query**:**

```
CREATE VIEW company_data as(
SELECT  cd.customer_name,cd.cust_id,mt.ord_id,od.order_date
 FROM
    cust_dimen cd
       INNER JOIN
    market_fact mt ON cd.cust_id = mt.cust_id
       INNER JOIN
    orders_dimen od ON mt.ord_id = od.ord_id);

SELECT distinct year(order_date), month(order_date),
count(cust_id) OVER(PARTITION BY month(order_date) order by year(order_date)) AS total_unique_customers
FROM company_data
WHERE year(order_date)=2011 AND cust_id
IN (SELECT DISTINCT cust_id
FROM company_data
WHERE month(order_date)=1
AND year(order_date)=2011);
```

Output:

Year(order_date)	Month(order_date)	Total_Unique_Customers
2011	1	154
2011	2	13
2011	3	14
2011	4	7
2011	5	5
2011	6	7
2011	7	8
2011	8	5
2011	9	6
2011	10	14
2011	11	14
2011	12	11

Explanation:

The first query creates a **view** called **combined table** which combines data from three tables:
1. cust_dimen

2. market_fact, and

3. orders_dimen.

It joins cust_dimen and market_fact on the cust_id column and market_fact and orders_dimen on the ord_id column.

The second query retrieves the **total number of unique customers for each month** in the year 2011, using a window function and a subquery.

The SELECT statement selects distinct Year and Month values from the order_date column of the combined_table and calculates the Total_Unique_Customers for each month. The count(cust_id) OVER(PARTITION BY month(order_date) order by month(order_date)) calculates the count of unique cust_id values for each month using a window function, and assigns the result to the Total_Unique_Customers column.

The WHERE clause filters the result to only include records from the year 2011 and where the cust_id values are present in the subquery.

Takeaway and Conclusion:

It's seen that a total of 154 customers were in January. Out of these customers, only minimum of 5 customers visit every month.

- CONCLUSION:

From the above analysis, the important insights which can be derived are:

- The top three customers in the business organization were recognized. These customers must be retained by maintaining the same service and product quality.
- Maximum time taken for the delivery was observed to be 92 days. The organization needs to work on its delivery services. Also, they must try to give the required benefits to such customers who experience delays in delivery to compensate them.
- Certain products have high sales but still lead to losses. Since this could be due to various reasons, the organization must investigate them.
- According to the data, there are 154 customers at the start of the year, that is, in the month of January. But, only a few of them visit every month or even for half of the year. We should try to find out ways to retain our customers and make sure they utilize our services more often.