SQL Case Study – 1

Problem Statement:
You are a database administrator. You want to use the data to answer a few questions about your customers, especially about the sales and profit coming from different states, money spent in marketing and various other factors such as COGS (Cost of Goods Sold), budget profit etc. You plan on using these insights to help find out which items are being sold the most. You have been provided with the sample of the overall customer data due to privacy issues. But you hope that these samples are enough for you to write fully functioning SQL queries to help answer the questions.

Dataset:
The 3 key datasets for this case study:

a. **FactTable:** The Fact Table has 14 columns mentioned below and 4200 rows. Date, ProductID, Profit, Sales, Margin, COGS, Total Expenses, Marketing, Inventory, Budget Profit, Budget COGS, Budget Margin, Budget Sales, and Area Code
Note: COGS stands for Cost of Goods Sold

b. **ProductTable:** The ProductTable has four columns named Product Type, Product, ProductID, and Type. It has 13 rows which can be broken down into further details to retrieve the information mentioned in theFactTable.

c. **LocationTable:** Finally, the LocationTable has 156 rows and follows a similar approach to ProductTable. It has four columns named Area Code, State, Market, and Market Size.

```
create database Case_study
use Case_study
select * from fact
select * from  Location
select * from Product
```

```sql
-- 1. Display the number of states present in the LocationTable.
select * from  Location

SELECT COUNT(DISTINCT State) AS num_states FROM Location;

-- 2. How many products are of regular type?

select * from Product
SELECT COUNT( type) Total_regular_products FROM Product
WHERE Type = 'regular'

--3. How much spending has been done on marketing of product ID 1?

SELECT COUNT( ProductId) Total_regular_products FROM fact
    --total no of product 1 item
WHERE ProductId = '1'


select * from fact

SELECT ProductID, sum(Marketing) MARKETING_COST FROM Fact

-- sum of total money used on product 1
WHERE ProductID = 1
group by ProductId

--4. What is the minimum sales of a product?
select * from fact
SELECT MIN(Sales) AS minimum_sales
FROM Fact;

--5. Display the max Cost of Good Sold (COGS).
select * from fact
SELECT MAX(COGS) AS max_cogs
FROM Fact;

--6. Display the details of the product where product type is coffee.

select * from Product

SELECT ProductID,Product_Type, Type, Product  FROM Product
WHERE Product_Type = 'coffee';

--7. Display the details where total expenses are greater than 40.
select * from fact

SELECT * FROM Fact
WHERE Total_Expenses > 40;

--8. What is the average sales in area code 719?
```

```sql
SELECT * FROM Fact
SELECT Area_Code, AVG(Sales) AS avg_sales  FROM Fact
WHERE Area_Code = '719'
group by Area_Code

--9. Find out the total profit generated by Colorado state
SELECT * FROM Fact

select * from fact
select * from  Location
select * from Product

select state, area_code from location
where State = 'Colorado'

SELECT SUM(ft.Profit) TotalProfit , lt.Area_Code,lt.State
FROM Fact ft
 JOIN Location lt
ON ft.Area_Code = lt.Area_Code
WHERE lt.State = 'Colorado'
group by lt.Area_Code , lt.State


--10. Display the average inventory for each product ID.

select * from fact

SELECT ProductID, AVG(Inventory) AS avg_inventory
FROM Fact
GROUP BY ProductID
order by avg_inventory desc


--11. Display state in a sequential order in a Location Table.

SELECT State
FROM Location
ORDER BY State desc

SELECT Area_Code, State
FROM Location
ORDER BY State asc

--12. Display the average budget of the Product where the average budget margin should be greater
than 100.

select * from fact
select * from Product

SELECT p.ProductId,p.Product, AVG(f.Budget_Sales) AS AvgBudgetSales
FROM Fact f
```

```sql
JOIN Product p
ON f.ProductID = p.ProductID
GROUP BY p.Product, p.ProductId
HAVING AVG(f.Budget_Margin) > 100
order by AvgBudgetSales desc


--13. What is the total sales done on date 2010-01-01?
select * from fact

SELECT SUM(Sales) AS TotalSales
FROM Fact
WHERE Date = '2010-01-01';


SELECT ProductID, Area_Code, SUM(Sales) AS total_sales
FROM Fact
WHERE Date = '2010-01-01'
GROUP BY ProductID, Area_Code
order by total_sales desc

--14. Display the average total expense of each product ID on an individual date

SELECT ProductID, Date, AVG(Total_Expenses) AS avg_total_expenses
FROM Fact
GROUP BY ProductID, Date
order by avg_total_expenses desc

--15. Display the table with the following attributes such as date, productID, product_type, product,
sales, profit, state, area_code.

select * from fact
select * from Product

SELECT ft.Date, ft.ProductID,ft.Profit,ft.Area_Code,ft.Sales, pt.Product_Type, pt.Product,lt.State

FROM Fact ft
   JOIN Product pt
          ON ft.ProductID = pt.ProductID
   JOIN Location lt
          ON ft.Area_Code = lt.Area_Code;

--16. Display the rank without any gap to show the sales wise rank.
select * from fact
select * from  Location
select * from Product

SELECT ProductID, Sales, DENSE_RANK() OVER (ORDER BY Sales DESC)  sales_rank
FROM Fact
ORDER BY sales_rank;
```

```sql
--17. Find the state wise profit and sales.
select * from fact
select * from  Location

SELECT l.State, SUM(f.Profit) AS total_profit, SUM(f.Sales) total_sales
FROM Fact f
JOIN Location l
ON f.Area_Code = l.Area_Code
GROUP BY l.State;

--18. Find the state wise profit and sales along with the product name.
select * from fact
select * from  Location
select * from Product

SELECT l.State, p.Product, SUM(f.Profit)total_profit, SUM(f.Sales)  total_sales
FROM Fact f
inner JOIN Product p
ON f.ProductID = p.ProductID

inner JOIN Location l
ON f.Area_Code = l.Area_Code
GROUP BY l.State, p.Product
order by total_profit desc


--19. If there is an increase in sales of 5%, calculate the increasedsales.

SELECT ProductID, Sales, Sales * 1.05 AS increased_sales
FROM Fact;

--20. Find the maximum profit along with the product ID and producttype.
select * from Product
select * from fact

SELECT  f.ProductID, p.Product_Type, MAX(f.Profit) max_profit
FROM Fact f
JOIN Product p
ON f.ProductID = p.ProductID
GROUP BY f.ProductID, p.Product_Type
ORDER BY max_profit DESC


SELECT top 1 f.ProductID, p.Product_Type, MAX(f.Profit) max_profit
FROM Fact f
JOIN Product p
ON f.ProductID = p.ProductID
GROUP BY f.ProductID, p.Product_Type
ORDER BY max_profit DESC
```

```
--21. Create a stored procedure to fetch the result according to the product typefrom Product Table.

select * from Product

CREATE PROCEDURE GetProductsByType
   @ProductType VARCHAR(255)
AS
BEGIN
   SELECT ProductID, Product, Type
   FROM Product
   WHERE Product_Type = @ProductType;
END
GO
--call the stored procedure
EXEC GetProductsByType 'coffee';

--22. Write a query by creating a condition in which if the total expenses is less than 60 then it is a
profit or else loss.

select * from fact
SELECT ProductID, Sales, Total_Expenses ,
      CASE
       WHEN Total_Expenses < 60 THEN 'Profit'
       ELSE 'Loss'
      END AS ProfitOrLoss
FROM Fact
order by Total_Expenses asc



--23. Give the total weekly sales value with the date and product ID details. Use roll-up to pull the data
in hierarchical order.
select * from fact

SELECT ProductID, DATEPART(WEEK, Date)WeekNumber, DATEPART(YEAR, Date)Year,
SUM(Sales)TotalWeeklySales
FROM Fact
GROUP BY ROLLUP(DATEPART(WEEK, Date), DATEPART(YEAR, Date), ProductID)

--ORDER BY TotalWeeklySales desc /
ORDER BY Year, WeekNumber, ProductID;



--24. Apply union and intersection operator on the tables which consist of attribute area code.
SELECT Area_Code
FROM Fact
UNION
SELECT Area_Code
FROM Location;
```

```sql
SELECT Area_Code
FROM Fact
INTERSECT
SELECT Area_Code
FROM Location;
```

--25. Create a user-defined function for the product table to fetch a particular product type based upon the user's preference.

```sql
select * from fact
select * from Product

CREATE FUNCTION user_Get_Products_By_Type ( @ProductType VARCHAR(255) )
RETURNS TABLE
AS
RETURN
(SELECT ProductID, Product, Type
   FROM Product
   WHERE Product_Type = @ProductType );
          -- call the udf
          SELECT * FROM user_Get_Products_By_Type('Herbal tea');
```

--26. Change the product type from coffee to tea where product ID is 1 and undo it.

```sql
select * from Product

-- Update the product type to 'tea':
UPDATE Product
SET Product_Type = 'tea'
WHERE ProductID = 1;

 -- Verify the change:
SELECT ProductID, Product, Product_Type, Type
FROM Product
WHERE ProductID = 1;


-- Undo the change (rollback):
BEGIN TRANSACTION;
UPDATE Product
SET Product_Type = 'coffee'
WHERE ProductID = 1;
COMMIT TRANSACTION;

--Verify the rollback:
SELECT ProductID, Product,Product_Type, Type
FROM Product
WHERE ProductID = 1;
```

```sql
--27. Display the date, product ID and sales where total expenses are between 100 to 200.
select * from fact
SELECT Date, ProductID, Sales
FROM Fact
WHERE Total_Expenses BETWEEN 100 AND 200
order by Sales desc

--28. Delete the records in the Product Table for regular type and undo it
select * from Product

--Create a backup table:
CREATE TABLE ProductTableBackup ( ProductID INT, Product VARCHAR(255),Product_Type
VARCHAR(255), Type VARCHAR(50));

--Insert records to be deleted into the backup table:
INSERT INTO ProductTableBackup
SELECT * FROM Product
WHERE Type = 'regular';

--Delete the records from the ProductTable
DELETE FROM Product
WHERE Type = 'regular';

--Verify the deletion:
SELECT * FROM Product
WHERE Type = 'regular';

--Undo the deletion (rollback):
INSERT INTO Product
SELECT * FROM ProductTableBackup;

--Verify the rollback
SELECT * FROM Product
WHERE Type = 'regular';

DROP TABLE ProductTableBackup;


--29. Display the ASCII value of the fifth character from the columnProduct.
select * from Product
SELECT Product, ASCII(SUBSTRING(Product, 5, 1)) AS fifth_char_ascii  FROM Product;

/*The SUBSTRING(Product, 5, 1) function extracts a substring of length 1 (a single character)
from the Product column, starting at the fifth position (index 5).*/
```