**SQL Case Study: Texture Tales**

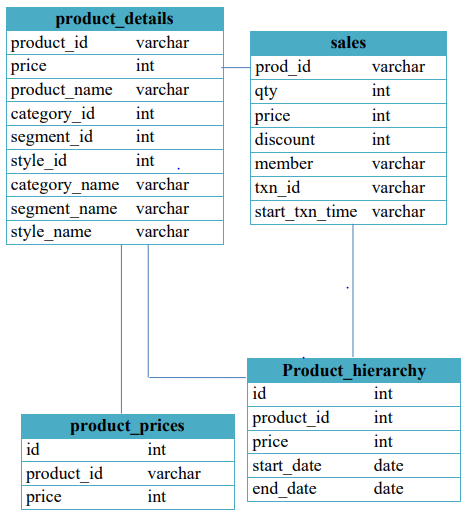
**Done by: S.YAZHINI (DE130)**



**INTRODUCTION:**

* Texture Tales Clothing Company prides themselves on providing an optimized range of clothing and lifestyle wear for the modern adventurer!
* I, the CEO of this trendy fashion company is asking you to assist the team’s merchandising teams analyze their sales performance and generate a basic financial report to share with the wider business

**SCHEMA USED**



# CASE STUDY QUESTIONS

****

1. What was the total quantity sold for all products?
2. What is the total generated revenue for all products before discounts?
3. What was the total discount amount for all products?
4. How many unique transactions were there?
5. What are the average unique products purchased in each transaction?
6. What is the average discount value per transaction?
7. What is the average revenue for member transactions and non- member transactions?
8. What are the top 3 products by total revenue before discount?
9. What are the total quantity, revenue and discount for each segment?
10. What is the top selling product for each segment?
11. What are the total quantity, revenue and discount for each category?
12. What is the top selling product for each category?

**NOTE:**

* First we have to create a Database:

Syntax: Create Database Textile\_CaseStudy\_DB

(Where Textile\_CaseStudy\_DB is the Database Name)

* Tables created:

1. Product\_details
2. Product\_hierarchy
3. Product\_prices
4. sales

* Main SQL File where all solutions will be there:

<https://drive.google.com/file/d/1g0ti6NEh66jhl8ykh7hkb5OIKAtJrrDM/view?usp=drive_link>

------------------------------------------------------------------------------------------------

**1.What was the total quantity sold for all products?**

**Solution:**

This query finds the total quantity sold for each product, ordered by the highest sales:

* JOIN: Combines sales and product\_details tables on matching product IDs.
* SELECT: Retrieves each product's name and total sales quantity (SUM(s.qty)).
* GROUP BY: Groups data by product\_name to calculate total sales per product.
* ORDER BY: Sorts products by total quantity sold in descending order.

SELECT p.product\_name, SUM(s.qty) AS sale\_counts

FROM sales AS s

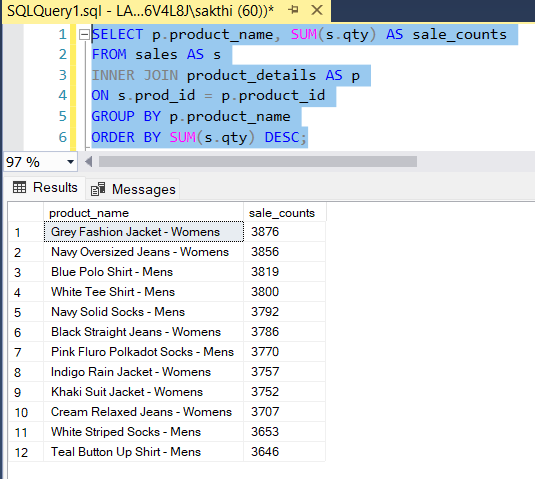
INNER JOIN product\_details AS p

ON s.prod\_id = p.product\_id

GROUP BY p.product\_name

ORDER BY SUM(s.qty) DESC;

**Output:**



**2. What is the total generated revenue for all products before discounts?**

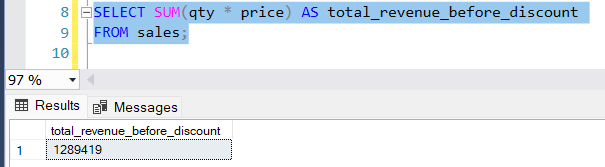
**Solution:**

Total Revenue is the sum of total price of each sale where total price of each sale = qty\*price. So, this query calculates the total revenue generated before discounts for all products.

SELECT SUM(qty \* price) AS total\_revenue\_before\_discount

FROM sales;

**Output:**



**3. What was the total discount amount for all products?**

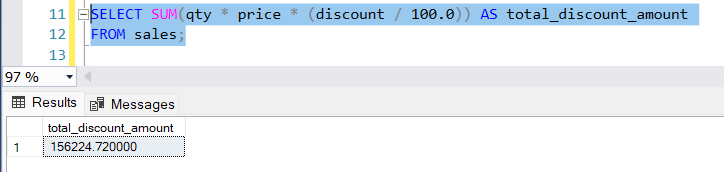
**Solution:**

In this formula used, we have to carefully use the parenthesis. Here discount amount for all products **=** SUM(qty \* price \* (discount / 100.0))

SELECT SUM(qty \* price \* (discount / 100.0)) AS total\_discount\_amount

FROM sales;

**Output**



**4. How many unique transactions were there?**

**Solution:**

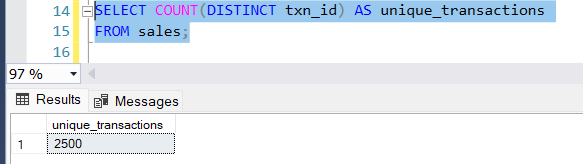
This query counts the number of unique transactions.

* DISTINCT is used to get non-repeating values or data from the dataset table.
* SYNTAX: SELECT DISTINCT column\_name FROM table\_name;

SELECT COUNT(DISTINCT txn\_id) AS unique\_transactions

FROM sales;

**Output:**



**5. What are the average unique products purchased in each transaction?**

**Solution:**

SELECT AVG(product\_count) AS avg\_unique\_products\_per\_transaction

FROM (

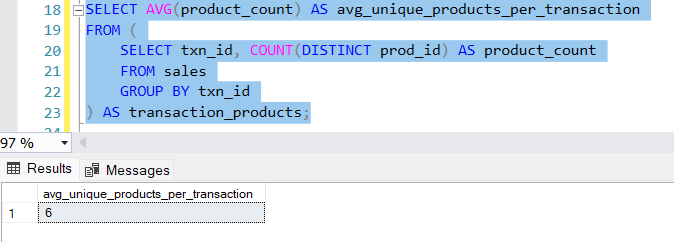
SELECT txn\_id, COUNT(DISTINCT prod\_id) AS product\_count

FROM sales

GROUP BY txn\_id

) AS transaction\_products;

**Output:**



**6. What is the average discount value per transaction?**

**Solution:**

SELECT AVG(total\_discount) AS avg\_discount\_per\_transaction

FROM (

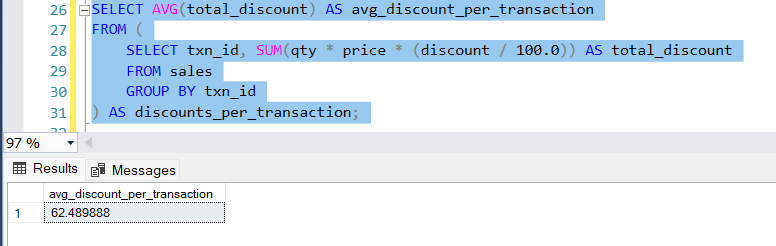
SELECT txn\_id, SUM(qty \* price \* (discount / 100.0)) AS total\_discount

FROM sales

GROUP BY txn\_id

) AS discounts\_per\_transaction;

**Output:**



**7. What is the average revenue for member transactions and non - member transactions?**

**Solution:**

WITH cte\_member\_revenue AS (

SELECT member, txn\_id, SUM(price \* qty) AS revenue

FROM sales

GROUP BY member, txn\_id

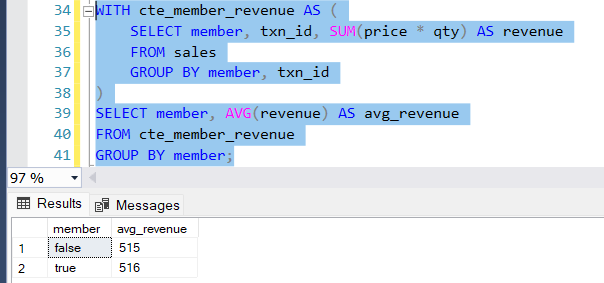
)

SELECT member, AVG(revenue) AS avg\_revenue

FROM cte\_member\_revenue

GROUP BY member;

**Output:**



**8. What are the top 3 products by total revenue before discount ?**

**Solution:**

SELECT TOP 3 pd.product\_name,

SUM(s.qty \* s.price) AS total\_revenue

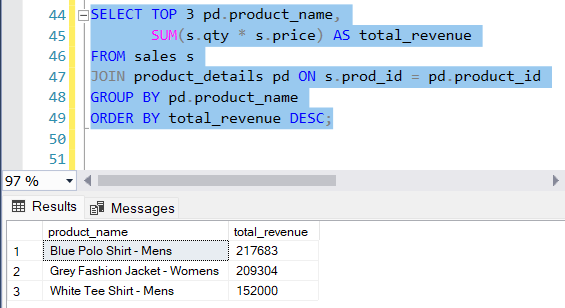
FROM sales s

JOIN product\_details pd ON s.prod\_id = pd.product\_id

GROUP BY pd.product\_name

ORDER BY total\_revenue DESC;

**Output:**



**9. What are the total quantity, revenue and discount for each segment?**

**Solution:**

SELECT pd.segment\_name,

SUM(s.qty) AS total\_quantity,

SUM(s.qty \* s.price) AS total\_revenue,

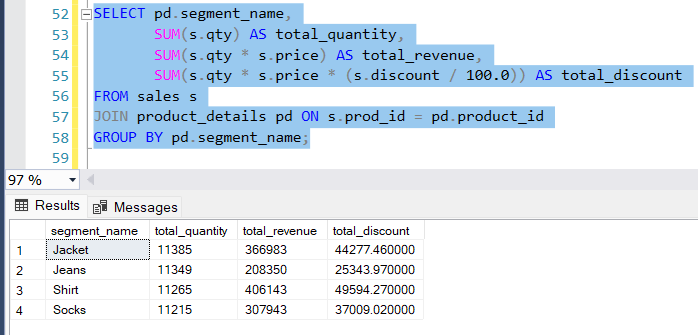
SUM(s.qty \* s.price \* (s.discount / 100.0)) AS total\_discount

FROM sales s

JOIN product\_details pd ON s.prod\_id = pd.product\_id

GROUP BY pd.segment\_name;

**Output:**



**10. What is the top selling product for each segment?**

**Solution:**

SELECT TOP 3 pd.segment\_name,

pd.product\_name,

SUM(s.qty) AS total\_quantity

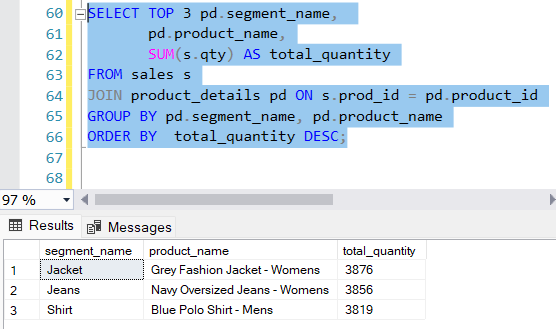
FROM sales s

JOIN product\_details pd ON s.prod\_id = pd.product\_id

GROUP BY pd.segment\_name, pd.product\_name

ORDER BY total\_quantity DESC;

**Output:**



**11. What are the total quantity, revenue and discount for each category**

**Solution:**

SELECT pd.category\_name,

SUM(s.qty) AS total\_quantity,

SUM(s.qty \* s.price) AS total\_revenue,

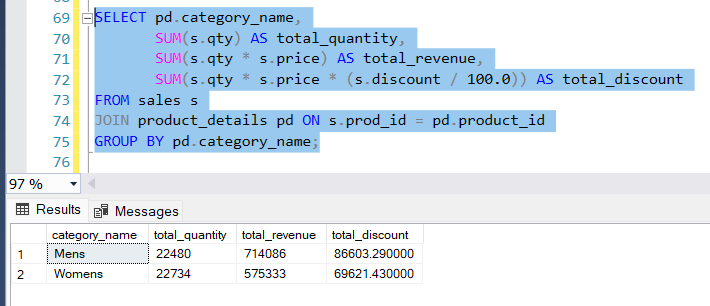
SUM(s.qty \* s.price \* (s.discount / 100.0)) AS total\_discount

FROM sales s

JOIN product\_details pd ON s.prod\_id = pd.product\_id

GROUP BY pd.category\_name;

**Output:**



**12. What is the top selling product for each category?**

**Solution:**

* TOP Keyword is used to limit the output data based on the given condition.
* SYNTAX: SELECT TOP n column\_name from table;

Where n can be any number.

SELECT TOP 5 pd.category\_name,

pd.product\_name,

SUM(s.qty) AS total\_quantity

FROM sales s

JOIN product\_details pd ON s.prod\_id = pd.product\_id

GROUP BY pd.category\_name, pd.product\_name

ORDER BY total\_quantity DESC;

**Output:**

