8WeekSQLCHALLENGE

BEHIND THE SALES TAG: AN ANALYTICAL REPORT ON BALANCED TREE CLOTHING TRANSACTIONS

Fatih Sahin

https://8weeksqlchallenge.com/case-study-7/

OCTOBER 2023

CONTENTS

Introduction

This report illuminates the findings from the "Balanced Tree Clothing Co." case study, an integral segment of the '8WeekSQLChallenge' orchestrated by Danny Ma. This specific case study presents an analytical expedition into the world of Balanced Tree, a distinguished fashion brand celebrated for its optimized clothing range tailored for modern adventurers. Embracing this challenge granted me the opportunity to emulate the role of a data analyst at Balanced Tree, deciphering insights into sales performance, product popularity, and customer buying behaviors.

Venturing into the "Balanced Tree Clothing Challenge" felt like setting sail into the vast ocean of retail analytics, charting through significant datasets that echoed genuine sales transactions, product hierarchies, and customer preferences. Those enthusiastic about grasping the nuances of this challenge can further explore at https://8weeksqlchallenge.com/case-study-7/

Such endeavors elevate beyond routine SQL tasks; they mirror authentic business scenarios, prompting analysts to derive significant, actionable intelligence from a multitude of data.

For this analytical endeavor, I utilized the MS SQL Server Management Studio, an adaptable tool designed for accurate and intricate data querying. Every revelation and insight presented in this report is the fruit of diligent analysis, bolstered by the expertise I've cultivated on my SQL voyage.

The dual objective of this report is to spotlight the salient discoveries from the Balanced Tree dataset and to emphasize the profound potential of SQL in deriving strategic business insights. I ardently hope that this case study not only imparts invaluable understandings but also exemplifies the indispensable role of data analytics in sculpting the retail and fashion industries of the contemporary era.

Problem Statement

Clique Bait, an innovative player in the digital realm, finds itself at a transformative crossroads. Guided by the principles of engaging user experiences and robust digital campaigns, the platform aims to deeply understand its **user's behavior**, **product interactions**, and the **effectiveness of its campaigns**. However, before embarking on this analytical journey, there was an imperative step of refining and preparing the data:

- **Data Consistency**: Ensured that all event types and identifiers adhered to a uniform categorization system, facilitating a seamless event-based analysis.
- **Temporal Metrics Augmentation:** Enriched the dataset with calculated metrics like visit durations, session lengths, and interactivity times, enhancing the granularity of user interactions.
- **User Segmentation**: Introduced new columns that categorized users based on their engagement levels, preferences, and patterns, aiming for a nuanced user behavior analysis.
- **User Journey Metrics**: Calculated metrics like average interactions per visit and bounce rates, offering a deeper perspective into user journey dynamics.

With the data in its optimal state, the investigative challenges for the business took shape:

- **User Interaction Analysis**: Delve into how users navigate and interact with the platform, identifying potential areas of improvement in UI/UX.
- **Product Preference Assessment**: Understand which products captivate user attention the most, guiding inventory and marketing decisions.
- **Campaign Effectiveness**: Evaluate the performance of various campaigns in terms of impressions, clicks, and conversions, aiming to refine future campaign strategies.
- **Platform Dynamics**: Understand user behavior across different devices and platforms, offering insights into platform-specific optimizations.
- **User Conversion Insights**: Analyze the funnel from views to cart additions and eventually to purchases, identifying bottlenecks and opportunities for increased conversions.

Creating Schema and Tables

All datasets exist within the `balanced_tree` database schema.

```sql
CREATE SCHEMA balanced_tree;

CREATE TABLE balanced_tree.product_hierarchy (
  "id" INTEGER,
  "parent_id" INTEGER,
  "level_text" VARCHAR(19),
  "level_name" VARCHAR(8)
);

INSERT INTO balanced_tree.product_hierarchy
  ("id", "parent_id", "level_text", "level_name")
VALUES
  ('1', NULL, 'Womens', 'Category'),
  ('2', NULL, 'Mens', 'Category'),
  ('3', '1', 'Jeans', 'Segment'),
  ('4', '1', 'Jacket', 'Segment'),
  ('5', '2', 'Shirt', 'Segment'),
  ('6', '2', 'Socks', 'Segment'),
  ('7', '3', 'Navy Oversized', 'Style'),
  ('8', '3', 'Black Straight', 'Style'),
  ('9', '3', 'Cream Relaxed', 'Style'),
  ('10', '4', 'Khaki Suit', 'Style'),
  ('11', '4', 'Indigo Rain', 'Style'),
  ('12', '4', 'Grey Fashion', 'Style'),
  ('13', '5', 'White Tee', 'Style'),
  ('14', '5', 'Teal Button Up', 'Style'),
  ('15', '5', 'Blue Polo', 'Style'),
  ('16', '6', 'Navy Solid', 'Style'),
  ('17', '6', 'White Striped', 'Style'),
  ('18', '6', 'Pink Fluro Polkadot', 'Style');

CREATE TABLE balanced_tree.product_prices (
  "id" INTEGER,
  "product_id" VARCHAR(6),
  "price" INTEGER
);

INSERT INTO balanced_tree.product_prices
  ("id", "product_id", "price")
VALUES
  ('7', 'c4a632', '13'),
  ('8', 'e83aa3', '32'),
  ('9', 'e31d39', '10'),
  ('10', 'd5e9a6', '23'),
  ('11', '72f5d4', '19'),
  ('12', '9ec847', '54'),
  ('13', '5d267b', '40'),
  ('14', 'c8d436', '10'),
  ('15', '2a2353', '57'),
  ('16', 'f084eb', '36'),
  ('17', 'b9a74d', '17'),
  ('18', '2feb6b', '29');

CREATE TABLE balanced_tree.product_details (
  "product_id" VARCHAR(6),
  "price" INTEGER,
  "product_name" VARCHAR(32),
  "category_id" INTEGER,
  "segment_id" INTEGER,
  "style_id" INTEGER,
  "category_name" VARCHAR(6),
  "segment_name" VARCHAR(6),
  "style_name" VARCHAR(19)
);

INSERT INTO balanced_tree.product_details
  ("product_id", "price", "product_name", "category_id", "segment_id", "style_id", "category_name", "segment_name", "style_name")
VALUES
  ('c4a632', '13', 'Navy Oversized Jeans - Womens', '1', '3', '7', 'Womens', 'Jeans', 'Navy Oversized'),
  ('e83aa3', '32', 'Black Straight Jeans - Womens', '1', '3', '8', 'Womens', 'Jeans', 'Black Straight'),
  ('e31d39', '10', 'Cream Relaxed Jeans - Womens', '1', '3', '9', 'Womens', 'Jeans', 'Cream Relaxed'),
  ('d5e9a6', '23', 'Khaki Suit Jacket - Womens', '1', '4', '10', 'Womens', 'Jacket', 'Khaki Suit'),
  ('72f5d4', '19', 'Indigo Rain Jacket - Womens', '1', '4', '11', 'Womens', 'Jacket', 'Indigo Rain'),
  ('9ec847', '54', 'Grey Fashion Jacket - Womens', '1', '4', '12', 'Womens', 'Jacket', 'Grey Fashion'),
  ('5d267b', '40', 'White Tee Shirt - Mens', '2', '5', '13', 'Mens', 'Shirt', 'White Tee'),
  ('c8d436', '10', 'Teal Button Up Shirt - Mens', '2', '5', '14', 'Mens', 'Shirt', 'Teal Button Up'),
  ('2a2353', '57', 'Blue Polo Shirt - Mens', '2', '5', '15', 'Mens', 'Shirt', 'Blue Polo'),
  ('f084eb', '36', 'Navy Solid Socks - Mens', '2', '6', '16', 'Mens', 'Socks', 'Navy Solid'),
  ('b9a74d', '17', 'White Striped Socks - Mens', '2', '6', '17', 'Mens', 'Socks', 'White Striped'),
  ('2feb6b', '29', 'Pink Fluro Polkadot Socks - Mens', '2', '6', '18', 'Mens', 'Socks', 'Pink Fluro Polkadot');
```

```sql
CREATE TABLE balanced_tree.sales (
  "prod_id" VARCHAR(6),
  "qty" INTEGER,
  "price" INTEGER,
  "discount" INTEGER,
  "member" BOOLEAN,
  "txn_id" VARCHAR(6),
  "start_txn_time" TIMESTAMP
);

INSERT INTO balanced_tree.sales
  ("prod_id", "qty", "price", "discount", "member", "txn_id", "start_txn_time")
VALUES
  ('c4a632', '4', '13', '17', 't', '54f307', '2021-02-13 01:59:43.296'),
  ('5d267b', '4', '40', '17', 't', '54f307', '2021-02-13 01:59:43.296'),
  ('b9a74d', '4', '17', '17', 't', '54f307', '2021-02-13 01:59:43.296'),
  ('2feb6b', '2', '29', '17', 't', '54f307', '2021-02-13 01:59:43.296'),
  ('c4a632', '5', '13', '21', 't', '26cc98', '2021-01-19 01:39:00.3456'),
  ('e31d39', '2', '10', '21', 't', '26cc98', '2021-01-19 01:39:00.3456'),
  ('72f5d4', '3', '19', '21', 't', '26cc98', '2021-01-19 01:39:00.3456'),
  ('2a2353', '3', '57', '21', 't', '26cc98', '2021-01-19 01:39:00.3456'),
  ('f084eb', '3', '36', '21', 't', '26cc98', '2021-01-19 01:39:00.3456'),
  ('c4a632', '1', '13', '21', 'f', 'ef648d', '2021-01-27 02:18:17.1648'),
  ('e83aa3', '5', '32', '21', 'f', 'ef648d', '2021-01-27 02:18:17.1648'),
  ('d5e9a6', '1', '23', '21', 'f', 'ef648d', '2021-01-27 02:18:17.1648'),
  ('72f5d4', '1', '19', '21', 'f', 'ef648d', '2021-01-27 02:18:17.1648'),
  ('5d267b', '3', '40', '21', 'f', 'ef648d', '2021-01-27 02:18:17.1648'),
  ('f084eb', '4', '36', '21', 'f', 'ef648d', '2021-01-27 02:18:17.1648'),
  ('b9a74d', '4', '17', '21', 'f', 'ef648d', '2021-01-27 02:18:17.1648'),
  ('c4a632', '2', '13', '23', 't', 'fba96f', '2021-03-03 00:32:56.0544'),
  ('e31d39', '5', '10', '23', 't', 'fba96f', '2021-03-03 00:32:56.0544'),
  ('9ec847', '3', '54', '23', 't', 'fba96f', '2021-03-03 00:32:56.0544'),
  ('f084eb', '4', '36', '23', 't', 'fba96f', '2021-03-03 00:32:56.0544'),
  ('2feb6b', '2', '29', '23', 't', 'fba96f', '2021-03-03 00:32:56.0544'),
  ('c4a632', '5', '13', '11', 't', '4e9268', '2021-01-23 14:18:54.0576'),
  ('e31d39', '4', '10', '11', 't', '4e9268', '2021-01-23 14:18:54.0576'),
  ('72f5d4', '5', '19', '11', 't', '4e9268', '2021-01-23 14:18:54.0576'),
  ('f084eb', '2', '36', '11', 't', '4e9268', '2021-01-23 14:18:54.0576'),
  ('b9a74d', '2', '17', '11', 't', '4e9268', '2021-01-23 14:18:54.0576'),
  ('e83aa3', '4', '32', '4', 't', '9717d4', '2021-01-29 07:22:13.2672'),
  ('d5e9a6', '2', '23', '4', 't', '9717d4', '2021-01-29 07:22:13.2672'),
  ('c8d436', '4', '10', '4', 't', '9717d4', '2021-01-29 07:22:13.2672'),
  ('2a2353', '1', '57', '4', 't', '9717d4', '2021-01-29 07:22:13.2672'),
  ('f084eb', '1', '36', '4', 't', '9717d4', '2021-01-29 07:22:13.2672'),
  ('72f5d4', '3', '19', '14', 'f', 'e9a1dd', '2021-03-28 20:01:43.1328'),
  ('9ec847', '3', '54', '14', 'f', 'e9a1dd', '2021-03-28 20:01:43.1328'),
  ('2a2353', '4', '57', '14', 'f', 'e9a1dd', '2021-03-28 20:01:43.1328'),
  ('f084eb', '2', '36', '14', 'f', 'e9a1dd', '2021-03-28 20:01:43.1328'),
  ('c4a632', '3', '13', '6', 'f', '003ea6', '2021-01-20 14:21:00.9792'),
  ('e31d39', '3', '10', '6', 'f', '003ea6', '2021-01-20 14:21:00.9792'),
  ('d5e9a6', '3', '23', '6', 'f', '003ea6', '2021-01-20 14:21:00.9792'),
  ('9ec847', '3', '54', '6', 'f', '003ea6', '2021-01-20 14:21:00.9792'),
  ('e31d39', '5', '10', '20', 'f', '5d749c', '2021-03-28 22:24:25.8048'),
  ('d5e9a6', '4', '23', '20', 'f', '5d749c', '2021-03-28 22:24:25.8048'),
  ('5d267b', '5', '40', '20', 'f', '5d749c', '2021-03-28 22:24:25.8048'),
  ('2a2353', '4', '57', '20', 'f', '5d749c', '2021-03-28 22:24:25.8048'),
  ('f084eb', '2', '36', '20', 'f', '5d749c', '2021-03-28 22:24:25.8048'),
  ('d5e9a6', '5', '23', '14', 'f', 'cf6517', '2021-02-21 21:45:33.3504'),
  ('72f5d4', '1', '19', '14', 'f', 'cf6517', '2021-02-21 21:45:33.3504'),
  ('5d267b', '5', '40', '14', 'f', 'cf6517', '2021-02-21 21:45:33.3504'),
  ('b9a74d', '3', '17', '14', 'f', 'cf6517', '2021-02-21 21:45:33.3504'),
  ('c4a632', '5', '13', '19', 't', '48b9d7', '2021-02-26 06:49:17.3856'),
  ('e83aa3', '2', '32', '19', 't', '48b9d7', '2021-02-26 06:49:17.3856'),
  ('e31d39', '1', '10', '19', 't', '48b9d7', '2021-02-26 06:49:17.3856'),
  ('d5e9a6', '2', '23', '19', 't', '48b9d7', '2021-02-26 06:49:17.3856'),
  ('5d267b', '5', '40', '19', 't', '48b9d7', '2021-02-26 06:49:17.3856'),
  ('c8d436', '1', '10', '19', 't', '48b9d7', '2021-02-26 06:49:17.3856'),
  ('2a2353', '3', '57', '19', 't', '48b9d7', '2021-02-26 06:49:17.3856'),
  ('b9a74d', '1', '17', '19', 't', '48b9d7', '2021-02-26 06:49:17.3856'),
  ('2feb6b', '5', '29', '19', 't', '48b9d7', '2021-02-26 06:49:17.3856'),
  ('72f5d4', '3', '19', '5', 't', '47251d', '2021-01-27 22:53:10.7808'),
  ('5d267b', '5', '40', '5', 't', '47251d', '2021-01-27 22:53:10.7808'),
  ('2a2353', '5', '57', '5', 't', '47251d', '2021-01-27 22:53:10.7808'),
  ('f084eb', '2', '36', '5', 't', '47251d', '2021-01-27 22:53:10.7808'),
  ('e83aa3', '4', '32', '15', 'f', 'c75ea6', '2021-03-03 04:17:19.8528'),
  ('e31d39', '5', '10', '15', 'f', 'c75ea6', '2021-03-03 04:17:19.8528'),
  ('72f5d4', '2', '19', '15', 'f', 'c75ea6', '2021-03-03 04:17:19.8528'),
  ('9ec847', '4', '54', '15', 'f', 'c75ea6', '2021-03-03 04:17:19.8528'),
  ('5d267b', '1', '40', '15', 'f', 'c75ea6', '2021-03-03 04:17:19.8528'),
  ('f084eb', '5', '36', '15', 'f', 'c75ea6', '2021-03-03 04:17:19.8528'),
  ('b9a74d', '4', '17', '15', 'f', 'c75ea6', '2021-03-03 04:17:19.8528'),
  ('2feb6b', '5', '29', '15', 'f', 'c75ea6', '2021-03-03 04:17:19.8528'),
  ('c4a632', '1', '13', '10', 't', '84a0f1', '2021-03-19 23:33:38.2752'),
  ('e31d39', '4', '10', '10', 't', '84a0f1', '2021-03-19 23:33:38.2752'),
  ('9ec847', '5', '54', '10', 't', '84a0f1', '2021-03-19 23:33:38.2752'),
  ('b9a74d', '2', '17', '10', 't', '84a0f1', '2021-03-19 23:33:38.2752'),
  ('2feb6b', '1', '29', '10', 't', '84a0f1', '2021-03-19 23:33:38.2752'),
  ('c4a632', '4', '13', '9', 'f', '786c60', '2021-01-13 08:41:16.4544'),
  ('e83aa3', '1', '32', '9', 'f', '786c60', '2021-01-13 08:41:16.4544'),
  ('e31d39', '4', '10', '9', 'f', '786c60', '2021-01-13 08:41:16.4544'),
```

Entity Relationship Diagram

Data Structure

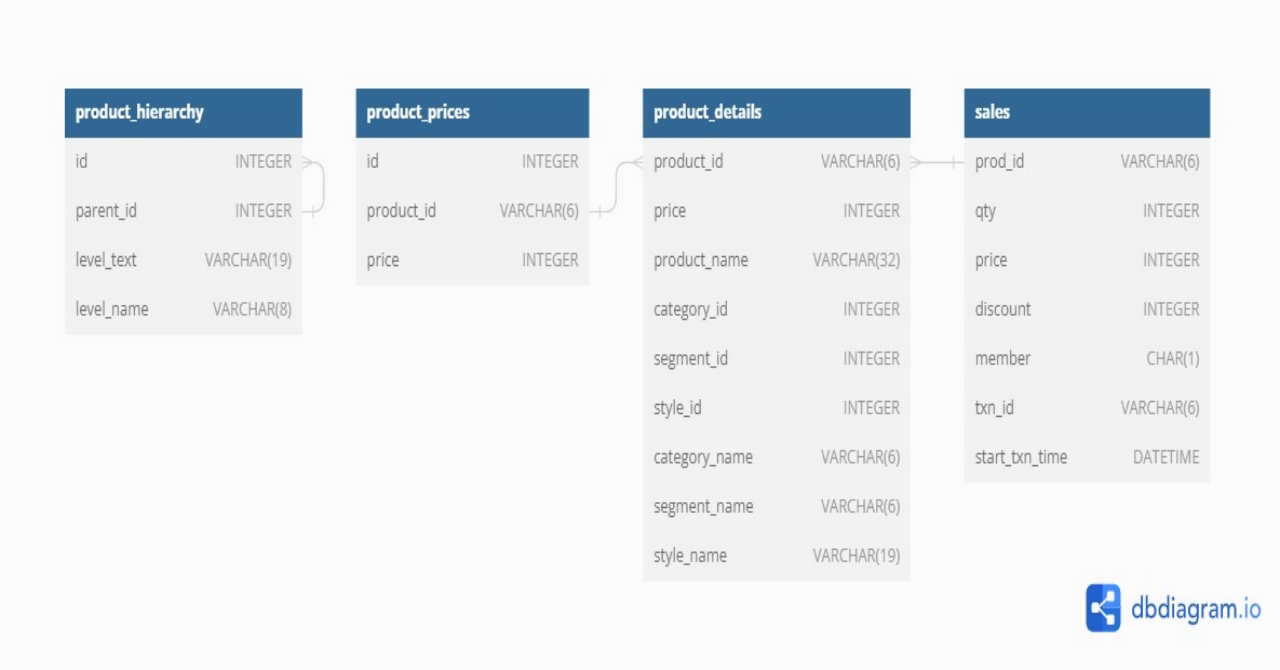
Balanced Tree Clothing Company employs an intricate data system to manage and analyze its broad range of clothing and lifestyle wear sales. The heart of this system consists of tables that record product details, sales transactions, product hierarchies, and product pricing. Here's a detailed breakdown of these tables:

1. **balanced_tree.product_details**: This table stands central to the company's product offerings. It contains essential product details such as:
    - **product_id**: A unique identifier for each product.
    - **product_name**: The name or title of the product.
    - **price**: The base price of the product.
    - **category_name**, **segment_name**, **style_name**: Various classifications of the product.
2. **balanced_tree.sales**: Capturing the essence of customer transactions, this table includes:
    - **txn_id**: A unique identifier for each transaction.
    - **prod_id**: Identifier for the specific product in the transaction.
    - **qty**: Quantity of the product sold in the transaction.
    - **price**: The selling price of the product in the transaction.
    - **discount**: Discounts applied to the product during the sale.
    - **member**: Indicates if the sale was made to a member ('t' for true and 'f' for false).
    - **start_txn_time**: Timestamp indicating when the transaction occurred.
3. **balanced_tree.product_hierarchy:** This table elucidates the hierarchical structure of products, enabling a layered understanding:
    - **id**: A unique identifier for each hierarchy level.
    - **parent_id**: Identifier pointing to the parent level in the hierarchy.
    - **level_name**: Name of the hierarchy level, e.g., "Category" or "Segment".

4. **balanced_tree.product_prices**: While the core product details table provides base prices, this auxiliary table offers insights into specific product pricing:
    - **id**: A unique identifier for each pricing entry.
    - **product_id**: Reference to the specific product.
    - **price**: The price associated with that product.

Through the combined power of these tables, especially the connection between **balanced_tree.sales** and **balanced_tree.product_details**, Balanced Tree Clothing Company can harness a comprehensive view of its sales performance. The structured relationship of these tables not only offers clarity on sales and product trends but also primes the ground for in-depth financial analyses, segmentation, and sales forecasting.

Entity Relationship Diagram



Case Study Questions

A.HIGH LEVEL SALES ANALYSIS

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

```
SELECT
    SUM(qty) AS total_qty
FROM
    balanced_tree.sales

ORDER BY total_qty
```

	total_qty
1	45216

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

```
SELECT
     SUM(qty*price) as total_revenue_bd
 FROM
     balanced_tree.sales
 ORDER BY total_revenue_bd
```

	total_revenue_bd
1	1289453

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

```
SELECT pd.product_name,
       ROUND(SUM((s.price * s.qty) * (CAST(s.discount AS NUMERIC) / 100)), 2 ) AS total_item_discounts
 FROM balanced_tree.sales  s
 JOIN balanced_tree.product_details  pd
 ON pd.product_id = s.prod_id
 GROUP BY pd.product_name
```

	product_name	total_item_discounts
1	Black Straight Jeans - Womens	14744.960000
2	Blue Polo Shirt - Mens	26819.070000
3	Cream Relaxed Jeans - Womens	4463.400000
4	Grey Fashion Jacket - Womens	25391.880000
5	Indigo Rain Jacket - Womens	8642.530000
6	Khaki Suit Jacket - Womens	10243.050000
7	Navy Oversized Jeans - Womens	6135.610000
8	Navy Solid Socks - Mens	16650.360000
9	Pink Fluro Polkadot Socks - Mens	12952.270000
10	Teal Button Up Shirt - Mens	4397.600000
11	White Striped Socks - Mens	7410.810000
12	White Tee Shirt - Mens	18377.600000

B. TRANSACTION ANALYSIS

1. How many unique transactions were there?

```
SELECT
     COUNT(DISTINCT txn_id) AS unique_txns
 FROM balanced_tree.sales
```

unique_txns
2500

2. What is the average unique products purchased in each transaction?

```
WITH TransactionProductCounts AS (
    SELECT
        s.txn_id,
        COUNT(DISTINCT s.prod_id) AS unique_product_count
    FROM balanced_tree.sales AS s
    GROUP BY s.txn_id
)
SELECT
   ROUND(AVG(CAST(unique_product_count AS FLOAT)),1) AS avg_unique_products_per_transaction
FROM TransactionProductCounts;
```

	avg_unique_products_per_transaction
1	6

3. What are the 25th, 50th and 75th percentile values for the revenue per transaction?

```
WITH TransactionRevenue AS (
    SELECT
        s.txn_id,
        SUM(s.price * s.qty * (1 - (CAST(s.discount AS NUMERIC)/100))) AS revenue
    FROM balanced_tree.sales AS s
    GROUP BY s.txn_id
)

, Percentiles AS (
    SELECT
        txn_id,
        revenue,
        PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY revenue) OVER () AS P25,
        PERCENTILE_CONT(0.50) WITHIN GROUP (ORDER BY revenue) OVER () AS P50,
        PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY revenue) OVER () AS P75
    FROM TransactionRevenue
)

SELECT
    DISTINCT
    P25 AS "25th Percentile",
    P50 AS "50th Percentile",
    P75 AS "75th Percentile"
FROM Percentiles;
```

	25th Percentile	50th Percentile	75th Percentile
1	326.405	441.225	572.7625

4. What is the average discount value per transaction?

```
WITH get_discounts AS(
SELECT
    txn_id,
    SUM((price * qty) * (CAST(discount AS NUMERIC) / 100)) AS discounts
FROM balanced_tree.sales
GROUP BY txn_id
)
SELECT ROUND(AVG(discounts),2) AS avg_discount
FROM get_discounts
```

	avg_discount
1	62.490000

5. What is the percentage split of all transactions for members vs non-members?

```
WITH MemberTransactionCounts AS (
    SELECT
        member,
        COUNT(DISTINCT txn_id) AS transaction_count
    FROM balanced_tree.sales
    GROUP BY member
)

SELECT
    member,
    transaction_count,
    (transaction_count * 100.0 / SUM(transaction_count) OVER()) AS percentage
FROM MemberTransactionCounts;
```

	member	transaction_count	percentage
1	f	995	39.800000000000
2	t	1505	60.200000000000

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

```
WITH TransactionRevenue AS (
    SELECT
        s.txn_id,
        s.member,
        SUM(s.price * s.qty * (1 - CAST(s.discount AS NUMERIC) / 100)) AS revenue
    FROM balanced_tree.sales AS s
    GROUP BY s.txn_id, s.member
)

SELECT
    member,
    AVG(revenue) AS average_revenue
FROM TransactionRevenue
GROUP BY member;
```

	member	average_revenue
1	f	452.007768
2	t	454.136963

C.PRODUCT ANALYSIS

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

```
SELECT TOP 3
    pd.product_name,
    SUM(s.price * s.qty) AS revenue
FROM balanced_tree.sales s
JOIN balanced_tree.product_details pd ON pd.product_id=s.prod_id
GROUP BY product_name
ORDER BY revenue DESC;
```

	product_name	revenue
1	Blue Polo Shirt - Mens	217683
2	Grey Fashion Jacket - Womens	209304
3	White Tee Shirt - Mens	152000

2. What is the total quantity, revenue and discount for each segment?

```
SELECT
    pd.segment_id,
    pd.segment_name,
    SUM(s.qty) AS total_quantity,
    SUM(s.price * s.qty) AS total_revenue,
    ROUND(SUM((s.price * s.qty) * (CAST(s.discount AS NUMERIC) / 100)), 2 ) AS total_item_discounts
FROM balanced_tree.sales s
JOIN balanced_tree.product_details pd ON pd.product_id=s.prod_id
GROUP BY segment_id,segment_name
ORDER BY total_quantity DESC;
```

	segment_id	segment_name	total_quantity	total_revenue	total_item_discounts
1	4	Jacket	11385	366983	44277.460000
2	3	Jeans	11349	208350	25343.970000
3	5	Shirt	11265	406143	49594.270000
4	6	Socks	11217	307977	37013.440000

3. What is the top selling product for each segment?

```sql
WITH get_segments_revenue AS (
    SELECT
        pd.segment_name,
        pd.product_name,
        SUM(s.qty) AS total_qty
    FROM balanced_tree.sales s
    JOIN balanced_tree.product_details pd ON pd.product_id = s.prod_id
    GROUP BY pd.segment_name, pd.product_name
)

, RankedProducts AS (
    SELECT
        segment_name,
        product_name,
        total_qty,
        ROW_NUMBER() OVER(PARTITION BY segment_name ORDER BY total_qty DESC) AS product_rank
    FROM get_segments_revenue
)

SELECT
    segment_name,
    product_name,
    total_qty
FROM RankedProducts
WHERE product_rank = 1;
```

	segment_name	product_name	total_qty
1	Jacket	Grey Fashion Jacket - Womens	3876
2	Jeans	Navy Oversized Jeans - Womens	3856
3	Shirt	Blue Polo Shirt - Mens	3819
4	Socks	Navy Solid Socks - Mens	3792

4. What is the total quantity, revenue and discount for each category?

```sql
SELECT
    pd.category_name,
    SUM(s.qty) AS total_qty,
    SUM(s.price * s.qty * (1 - CAST(s.discount AS NUMERIC) / 100)) AS total_revenue
FROM balanced_tree.sales s
JOIN balanced_tree.product_details pd ON pd.product_id = s.prod_id
GROUP BY category_name
```

	category_name	total_qty	total_revenue
1	Mens	22482	627512.290000
2	Womens	22734	505711.570000

5. What is the top selling product for each category?

```
WITH get_category_revenue AS (
SELECT
    pd.category_name,
    pd.product_name,
    SUM(s.qty) AS total_qty
FROM balanced_tree.sales s
JOIN balanced_tree.product_details pd ON pd.product_id = s.prod_id
GROUP BY category_name,product_name
),

ranked_categories AS (
SELECT
    category_name,
    product_name,
    total_qty,
    ROW_NUMBER() OVER(PARTITION BY category_name ORDER BY total_qty DESC) AS  category_rank
FROM get_category_revenue
)
SELECT
    category_name,
    product_name,
    total_qty
FROM ranked_categories
WHERE category_rank = 1;
```

	category_name	product_name	total_qty
1	Mens	Blue Polo Shirt - Mens	3819
2	Womens	Grey Fashion Jacket - Womens	3876

6. What is the percentage split of revenue by product for each segment?

```
WITH ProductSegmentRevenue AS (
    SELECT
        pd.segment_name,
        pd.product_name,
        SUM(s.price * s.qty * (1 - CAST(s.discount AS NUMERIC) / 100)) AS product_revenue
    FROM balanced_tree.sales AS s
    JOIN balanced_tree.product_details AS pd ON pd.product_id = s.prod_id
    GROUP BY pd.segment_name, pd.product_name
)

, SegmentTotalRevenue AS (
    SELECT
        segment_name,
        SUM(product_revenue) AS total_segment_revenue
    FROM ProductSegmentRevenue
    GROUP BY segment_name
)

SELECT
    psr.segment_name,
    psr.product_name,
    psr.product_revenue,
    (psr.product_revenue * 100.0 / str.total_segment_revenue) AS revenue_percentage
FROM ProductSegmentRevenue AS psr
JOIN SegmentTotalRevenue AS str ON psr.segment_name = str.segment_name
ORDER BY psr.segment_name, revenue_percentage DESC;
```

	segment_name	product_name	product_revenue	revenue_percentage
1	Jacket	Grey Fashion Jacket - Womens	183912.120000	56.990691
2	Jacket	Khaki Suit Jacket - Womens	76052.950000	23.567289
3	Jacket	Indigo Rain Jacket - Womens	62740.470000	19.442018
4	Jeans	Black Straight Jeans - Womens	106407.040000	58.144007
5	Jeans	Navy Oversized Jeans - Womens	43992.390000	24.038765
6	Jeans	Cream Relaxed Jeans - Womens	32606.600000	17.817227
7	Shirt	Blue Polo Shirt - Mens	190863.930000	53.530952
8	Shirt	White Tee Shirt - Mens	133622.400000	37.476616
9	Shirt	Teal Button Up Shirt - Mens	32062.400000	8.992431
10	Socks	Navy Solid Socks - Mens	119861.640000	44.235335
11	Socks	Pink Fluro Polkadot Socks - Mens	96377.730000	35.568520
12	Socks	White Striped Socks - Mens	54724.190000	20.196143

7. What is the percentage split of revenue by segment for each category?

```
WITH SegmentCategoryRevenue AS (
    SELECT
        pd.category_name,
        pd.segment_name,
        SUM(s.price * s.qty * (1 - CAST(s.discount AS NUMERIC) / 100)) AS segment_revenue
    FROM balanced_tree.sales AS s
    JOIN balanced_tree.product_details AS pd ON pd.product_id = s.prod_id
    GROUP BY pd.category_name, pd.segment_name
)

, CategoryTotalRevenue AS (
    SELECT
        category_name,
        SUM(segment_revenue) AS total_category_revenue
    FROM SegmentCategoryRevenue
    GROUP BY category_name
)

SELECT
    scr.category_name,
    scr.segment_name,
    scr.segment_revenue,
    (scr.segment_revenue * 100.0 / ctr.total_category_revenue) AS revenue_percentage
FROM SegmentCategoryRevenue AS scr
JOIN CategoryTotalRevenue AS ctr ON scr.category_name = ctr.category_name
ORDER BY scr.category_name, revenue_percentage DESC;
```

	category_name	segment_name	segment_revenue	revenue_percentage
1	Mens	Shirt	356548.730000	56.819401
2	Mens	Socks	270963.560000	43.180598
3	Womens	Jacket	322705.540000	63.812172
4	Womens	Jeans	183006.030000	36.187827

8. What is the percentage split of total revenue by category?

```
WITH CategoryRevenue AS (
    SELECT
        pd.category_name,
        SUM(s.price * s.qty * (1 - CAST(s.discount AS NUMERIC) / 100)) AS category_revenue
    FROM balanced_tree.sales AS s
    JOIN balanced_tree.product_details AS pd ON pd.product_id = s.prod_id
    GROUP BY pd.category_name
)

, TotalRevenue AS (
    SELECT
        SUM(category_revenue) AS total_revenue
    FROM CategoryRevenue
)

SELECT
    cr.category_name,
    cr.category_revenue,
    (cr.category_revenue * 100.0 / tr.total_revenue) AS revenue_percentage
FROM CategoryRevenue AS cr
CROSS JOIN TotalRevenue AS tr
ORDER BY revenue_percentage DESC;
```

	category_name	category_revenue	revenue_percentage
1	Mens	627512.290000	55.374080
2	Womens	505711.570000	44.625919

9. What is the total transaction “penetration” for each product?

(hint: penetration = number of transactions where at least 1 quantity of a product was purchased divided by total number of transactions)

```
WITH ProductTransactions AS (
    SELECT
        s.prod_id,
        COUNT(DISTINCT s.txn_id) AS product_txn_count
    FROM balanced_tree.sales AS s
    GROUP BY s.prod_id
)

, TotalTransactions AS (
    SELECT
        COUNT(DISTINCT txn_id) AS total_txn_count
    FROM balanced_tree.sales
)

SELECT
    pt.prod_id,
    pd.product_name,
    pt.product_txn_count,
    (pt.product_txn_count * 100.0 / tt.total_txn_count) AS penetration_percentage
FROM ProductTransactions AS pt
JOIN balanced_tree.product_details AS pd ON pd.product_id = pt.prod_id
CROSS JOIN TotalTransactions AS tt
ORDER BY penetration_percentage DESC;
```

	prod_id	product_name	product_txn_count	penetration_percentage
1	f084eb	Navy Solid Socks - Mens	1281	51.240000000000
2	9ec847	Grey Fashion Jacket - Womens	1275	51.000000000000
3	c4a632	Navy Oversized Jeans - Womens	1274	50.960000000000
4	5d267b	White Tee Shirt - Mens	1268	50.720000000000
5	2a2353	Blue Polo Shirt - Mens	1268	50.720000000000
6	2feb6b	Pink Fluro Polkadot Socks - Mens	1258	50.320000000000
7	72f5d4	Indigo Rain Jacket - Womens	1250	50.000000000000
8	d5e9a6	Khaki Suit Jacket - Womens	1247	49.880000000000
9	e83aa3	Black Straight Jeans - Womens	1246	49.840000000000
10	e31d39	Cream Relaxed Jeans - Womens	1243	49.720000000000
11	b9a74d	White Striped Socks - Mens	1243	49.720000000000
12	c8d436	Teal Button Up Shirt - Mens	1242	49.680000000000

10. What is the most common combination of at least 1 quantity of any 3 products in a 1 single transaction?

```
WITH TripleProductCombinations AS (
    SELECT
        s1.txn_id,
        s1.prod_id AS product1,
        s2.prod_id AS product2,
        s3.prod_id AS product3
    FROM balanced_tree.sales AS s1
    JOIN balanced_tree.sales AS s2 ON s1.txn_id = s2.txn_id AND s1.prod_id < s2.prod_id
    JOIN balanced_tree.sales AS s3 ON s1.txn_id = s3.txn_id AND s2.prod_id < s3.prod_id
)

, CombinationCounts AS (
    SELECT
        product1,
        product2,
        product3,
        COUNT(*) AS combination_count
    FROM TripleProductCombinations
    GROUP BY product1, product2, product3
)

SELECT TOP 1
    product1,
    product2,
    product3,
    combination_count
FROM CombinationCounts
ORDER BY combination_count DESC
```

	product1	product2	product3	combination_count
1	5d267b	9ec847	c8d436	352

Key Findings and Observations

1. **Sales Overview:**

   - **Volume**: A substantial volume of products was sold, with a total quantity of **45,216 items**. This underscores the brand's popularity and market penetration.

   - **Revenue**: Before considering discounts, the company generated an impressive **revenue of $1,289,453**. This is a testament to the brand's considerable sales capabilities and product appeal.

   - **Discounts**: The products had varying discount amounts, with notable mentions being the "**Black Straight Jeans - Womens**" and the "**Blue Polo Shirt - Mens**" which had discounts amounting to $14,744.96 and $26,819.07 respectively. This indicates strategic pricing and promotional tactics employed by the company.

2. **Transaction Analysis:**

   - **Volume & Diversity**: There were **2,500 unique transactions**, with an average o**f 6 unique products in each transaction**. This suggests a diverse shopping pattern among customers.

   - **Revenue Distribution**: The median (50th percentile) revenue per transaction stood at $441.225, with the 25th and 75th percentiles at $326.405 and $572.7625 respectively. This highlights a significant spread in transaction values.

   - **Membership Insights**: Members contributed to **60.2% of all transactions**, with an average revenue slightly higher ($454.13) compared to **non-members** ($452.01). This implies the importance of loyalty programs and member engagement strategies.

3. **Product Analysis:**

   - **Top Performers**: The "**Blue Polo Shirt - Mens"**, "**Grey Fashion Jacket - Womens**", and "**White Tee Shirt - Mens**" emerged as revenue leaders, collectively contributing to a significant portion of the company's sales.

- **Segment Insights**: **Jackets** led in the segments with a total revenue of $366,983, followed closely by **Shirts** and **Jeans**. This suggests **seasonal or fashion trends** influencing purchasing patterns.
- **Category Differentiation**: The **Mens category outperformed the Womens** category in revenue, making up 55.37% of the total revenue. This indicates a potential market segment that could be further targeted for growth.
- **Product Penetration**: The product "**Navy Solid Socks - Mens**" had the **highest penetration** with 51.24% of transactions including at least one purchase of this product. Such products can be deemed as popular or essential items in the product range.

4. **Unique Combinations**: The **most common product combination in a single transaction** was "**White Tee Shirt - Mens**", "**Grey Fashion Jacket - Womens**", and "**Teal Button Up Shirt - Mens**", with 352 occurrences. This insight can be pivotal for bundling strategies or marketing campaigns.

CONCLUSION

This case study delves into the world of fashion retail, shedding light on sales patterns, product preferences, and customer behavior.

Through the use of Microsoft SQL Server Management Studio, I was able to sift through vast amounts of data to uncover meaningful insights. SQL not only made the data more accessible but also transformed it into actionable information.

The insights gathered provide a clearer picture of consumer choices and preferences. Such knowledge is invaluable for businesses aiming to tailor their offerings or enhance their marketing strategies. This analysis, inspired by Danny's real-world experiences, underlines the significance of sales analytics in today's retail landscape. It showcases the immense potential of SQL in drawing out patterns and stories hidden within numbers, making it an essential tool for modern businesses.