Destiny Corley
07/03/25
Monthly Data Analysis Report

This report answers the following business questions using available sales and product data during the two months of operation –

1. Which product categories generate the highest revenue?
2. What are the top-selling products?
3. Which sales channels performed the best?

Each question is answered using structured data tables, explained below.

Detailed Table

Create Table
A new table called detailed_table was created by combining data from the products and sales tables. This merged table includes the following columns:

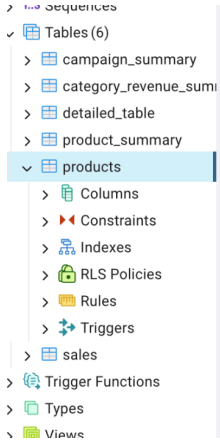
1. ***Product*** (product_id, product_name, category)
2. ***Sales*** (quantity, item_total, sales_date, channel, channel_campaigns)

```
CREATE TABLE detailed_table(
        product_id SMALLINT,
        product_name TEXT,
        product_category TEXT,
        quantity SMALLINT,
        item_total NUMERIC,
        sales_date DATE,
        channel TEXT,
        channel_campaigns TEXT
);
```

Populate Detailed Table Function
A reusable function was created to populate the detailed_table by joining the products and sales tables on product_id. This function inserts the combined data into the detailed_table, ensuring consistency and efficiency in preparing the data for analysis.

The summary tables are derived from the detailed_table rather than the raw data tables, allowing for cleaner aggregation and streamlined reporting. This dataset provides all the necessary information to generate insights through our three summary tables, each designed to address the key business questions.

```
CREATE OR REPLACE FUNCTION populate_detailed()
        RETURNS VOID
        LANGUAGE plpgsql
        AS
        $$
        BEGIN
                INSERT INTO detailed_table(
                product_id, product_name, product_category, quantity, item_total,
                sales_date, channel, channel_campaigns
        )
        SELECT
                p.product_id,
                p.product_name,
                p.category,
                s.quantity,
                s.item_total,
                s.sale_date,
                s.channel,
                s.channel_campaigns

        FROM sales s
        JOIN products p ON s.product_id = p.product_id
        ORDER BY s.sale_date ASC;
        EXCEPTION
        WHEN others THEN
        RAISE NOTICE 'Error in populate_detailed(): %', SQLERRM;
        END;
        $$;
```



	product_id smallint	product_name text	product_category text	quantity smallint	item_total numeric	sales_date date	channel text	channel_campaigns text
1	169	Vintage Linen Set	Sleepwear	5	164.7	2025-04-04	App Mobile	App Mobile
2	191	Polished Satin Dress	Dresses	2	121.04	2025-04-04	App Mobile	App Mobile
3	311	Essential Cotton Dress	Dresses	1	43.38	2025-04-04	E-comme…	Website Banner
4	407	Elegant Sleeveless Tee	T-Shirts	3	170.52	2025-04-04	App Mobile	App Mobile
5	239	Polished Wrap Trousers	Pants	5	334.95	2025-04-04	App Mobile	App Mobile
6	287	Relaxed Ribbed Set	Sleepwear	1	60.34	2025-04-04	App Mobile	App Mobile
7	487	Bold High-Waist Shoes	Shoes	2	99.68	2025-04-04	E-comme…	Website Banner
8	219	Polished Silk Shoes	Shoes	3	149.28	2025-04-04	E-comme…	Website Banner
9	35	Modern Ribbed Dress	Dresses	1	61.61	2025-04-04	App Mobile	App Mobile
10	421	Elegant Sleeveless Set	Sleepwear	3	152.67	2025-04-04	E-comme…	Website Banner
11	336	Tailored Ribbed Tee	T-Shirts	3	153.66	2025-04-04	E-comme…	Website Banner
12	100	Classic Crew Dress	Dresses	2	112.88	2025-04-04	E-comme…	Website Banner
13	206	Vintage Silk Trousers	Pants	3	189.09	2025-04-04	E-comme…	Website Banner
14	32	Soft Silk Tee	T-Shirts	1	54.83	2025-04-04	E-comme…	Website Banner

Product Category Revenue Summary Table

The category_revenue_summary table is designed to answer the first business question:
“Which product categories generate the highest revenue?”
To do this, we extract key metrics from the detailed_table into a summary format that includes:

Category –
The distinct product categories found in the data

Quantity –
The total number of items sold within each category

Total –
The total revenue generated from each category

This summarized view makes it easy to compare performance across product categories and identify the most profitable ones.

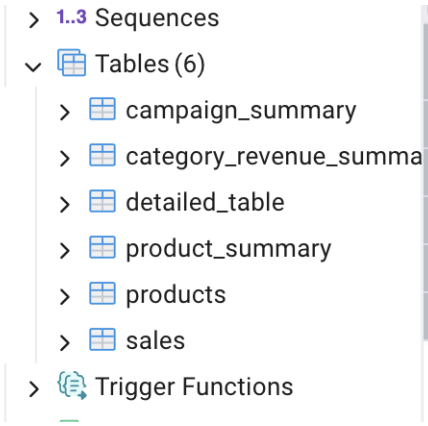
Creating Category Revenue Table

```
CREATE TABLE category_revenue_summary(
    category TEXT PRIMARY KEY,
    quantity SMALLINT,
    total NUMERIC
    );
```

Populate Category Revenue Table Function

```
CREATE OR REPLACE FUNCTION populate_category_summary()
    RETURNS VOID
    LANGUAGE plpgsql
    AS
    $$
    BEGIN
    TRUNCATE category_revenue_summary;

    INSERT INTO category_revenue_summary(
        category,
        quantity,
        total
    )
    SELECT
        product_category,
        SUM(quantity),
        SUM(item_total) AS Total
    FROM detailed_table
    GROUP BY product_category
```

```
ORDER BY Total DESC;
EXCEPTION
WHEN others THEN
RAISE NOTICE 'Error in populate_category_summary(): %', SQLERRM;
END;
$$;
```



	category [PK] text	quantity smallint	total numeric
1	Shoes	1400	70074.00
2	T-Shirts	1439	69692.78
3	Dresses	1451	68390.64
4	Sleepwear	1362	62276.58
5	Pants	1063	53802.66

Product Summary Table

The product_summary table addresses the second business question: “What are the top-selling products?” To answer this, we summarize relevant columns from the detailed_table:

Product ID –
A unique identifier assigned to each product

Product Name –
The name of the product for easy identification

Quantity –
The total number of units sold for each product

Total –
The total revenue generated by each product

This summary allows us to evaluate which products performed the best in terms of both sales volume and revenue, helping the business identify its strongest offerings.

Creating Product Summary Table

```
CREATE TABLE product_summary(
        product_id SMALLINT,
        product_name TEXT,
        quantity INTEGER,
        total NUMERIC,
        PRIMARY KEY (product_id)
);
```

Populate Product Summary Table

```
CREATE OR REPLACE FUNCTION populate_product_summary()
        RETURNS VOID
        LANGUAGE plpgsql
        AS
        $$
        BEGIN
        TRUNCATE product_summary;
        INSERT INTO product_summary(
                product_id,
                product_name,
                quantity,
                total
        )
        SELECT
                Product_id,
                product_name,
                SUM(quantity),
                SUM(item_total) AS Total
        FROM detailed_table
        GROUP BY product_id, product_name
        ORDER BY Total DESC;
        EXCEPTION
        WHEN others THEN
        RAISE NOTICE 'Error in populate_product_summary(): %', SQLERRM;
        END;
        $$;
```

	product_id [PK] smallint	product_name text	quantity integer	total numeric
1	310	Relaxed Ribbed Trousers	36	2379.30
2	60	Modern Cotton Tee	25	1919.95
3	116	Modern High-Waist Trousers	26	1907.66
4	469	Dresses Drop 1	29	1858.32
5	459	Bold High-Waist Dress	24	1804.08
6	224	Relaxed Linen Dress	22	1658.8
7	297	Essential High-Waist Shoes	25	1603.18
8	307	Essential Crew Tee	24	1598.19
9	239	Polished Wrap Trousers	25	1594.37
10	10	Soft Ribbed Trousers	20	1594.0
11	464	Bold Ribbed Shoes	26	1549.08
12	19	Soft Silk Dress	24	1527.08
13	445	Bold Cotton Set	30	1489.98
14	2	Soft Wrap Tee	20	1474.86
15	428	Elegant High-Waist Shoes	25	1438.67

Campaign Summary Table

The campaign_summary table answers the third business question: “Which sales channels performed the best?” To answer this, we summarize key fields from the detailed_table into the campaign_summary table –

- Channel – The platform or method used to make the sale (e.g., E-commerce, Retail, Social Media)
- Campaign – Specific marketing or sales campaigns under each channel
- Quantity – The total number of products sold per campaign
- Total – The total revenue generated through each campaign

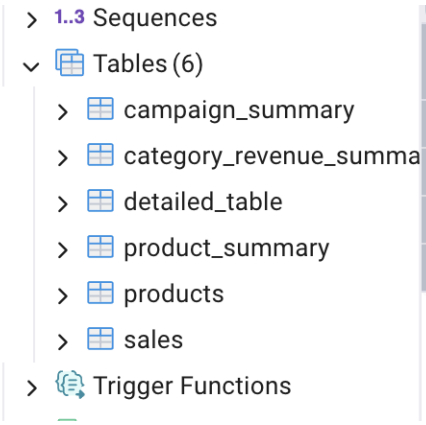
This table helps evaluate the effectiveness of different sales channels and campaigns, enabling the business to allocate resources to the most profitable marketing efforts

Creating Campaign Summary Table

```
 CREATE TABLE campaign_summary(
        channel TEXT,
        campaign TEXT,
        quantity INTEGER,
        total NUMERIC,
        PRIMARY KEY (channel, campaign)
);
```

Populate Campaign Summary Table

```
CREATE OR REPLACE FUNCTION populate_campaign_summary()
RETURNS VOID
LANGUAGE plpgsql
AS
$$
BEGIN
TRUNCATE campaign_summary;
INSERT INTO campaign_summary(
        channel,
        campaign,
        quantity,
        total
)
SELECT
        channel,
        channel_campaigns,
        SUM(quantity),
        SUM(item_total) AS Total
FROM detailed_table
GROUP BY channel, channel_campaigns
ORDER BY Total DESC;
EXCEPTION
WHEN others THEN
RAISE NOTICE 'Error in populate_campaign_summary(): %', SQLERRM;
END;
$$;
```



	channel [PK] text	campaign [PK] text	quantity integer	total numeric
1	E-commerce	Website Banner	3480	170513.31
2	App Mobile	App Mobile	2940	142892.00
3	App Mobile	Social Media	268	9668.94
4	E-commerce	Email	27	1162.41

Update Summary Function

The update_summaries_function is a trigger function that refreshes all three summary tables—product_summary, category_revenue_summary, and campaign_summary. This is done by using the populate function for each table whenever changes are made to the detailed_table. This ensures all summaries stay up to date automatically with every data change.

```
CREATE OR REPLACE FUNCTION update_summaries_function()
RETURNS TRIGGER
LANGUAGE plpgsql
AS
$$
BEGIN
PERFORM populate_product_summary();
PERFORM populate_category_summary();
PERFORM populate_campaign_summary();
RETURN NULL;
END;
$$;
```

Update Summary Trigger

The update_summaries trigger activates the update_summaries_function after any insert, update, or delete operation on the detailed_table. It runs once per statement to automatically refresh all summary tables, keeping the data consistent with any changes made to the detailed data.

```
CREATE TRIGGER update_summaries
AFTER INSERT OR UPDATE OR DELETE
ON detailed_table
FOR EACH STATEMENT
EXECUTE PROCEDURE update_summaries_function();
```

Refresh Tables Procedure

The refresh_tables procedure ensures all necessary tables exist by creating them if they don't. It then clears existing data from these tables and repopulates them by calling their respective populate functions. This procedure helps reset and refresh the entire dataset and summary tables to maintain up-to-date and accurate analytics.

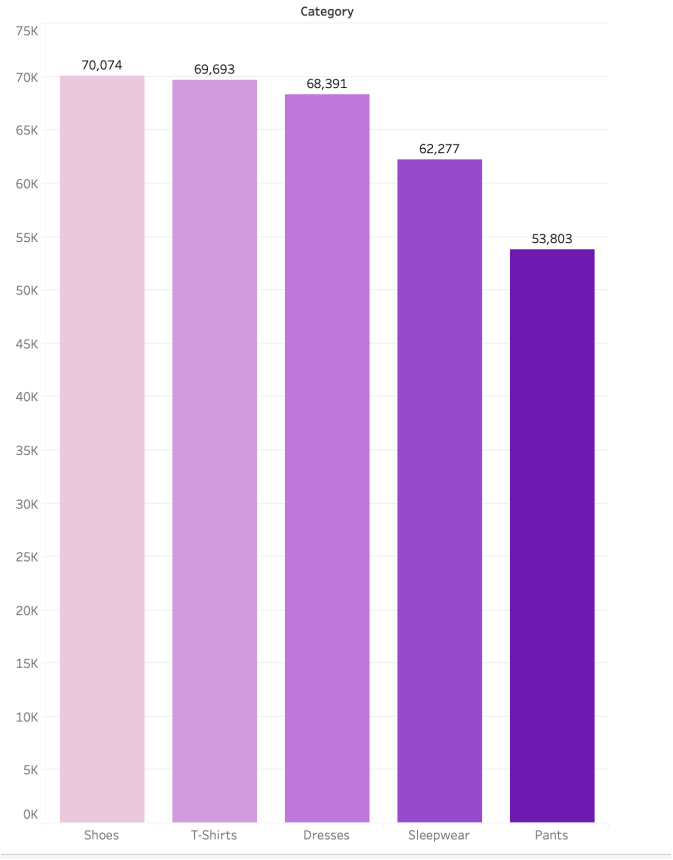
```
CREATE OR REPLACE PROCEDURE refresh_tables()
LANGUAGE plpgsql
AS
$$
BEGIN
  EXECUTE '
    CREATE TABLE IF NOT EXISTS detailed_table(
      product_id SMALLINT,
      product_name TEXT,
      product_category TEXT,
      quantity SMALLINT,
      item_total NUMERIC,
      sales_date DATE,
      channel TEXT,
      channel_campaigns TEXT,
      PRIMARY KEY (product_id, sales_date, channel, channel_campaigns)
    )
  ';

  EXECUTE '
    CREATE TABLE IF NOT EXISTS category_revenue_summary(
      category TEXT PRIMARY KEY,
      quantity SMALLINT,
      total NUMERIC
    )
  ';

  EXECUTE '
```

```
   CREATE TABLE IF NOT EXISTS product_summary(
     product_id SMALLINT PRIMARY KEY,
     product_name TEXT,
     quantity INTEGER,
     total NUMERIC
   )
 ';

 EXECUTE '
   CREATE TABLE IF NOT EXISTS campaign_summary(
     channel TEXT,
     campaign TEXT,
     quantity INTEGER,
     total NUMERIC,
     PRIMARY KEY (channel, campaign)
   )
 ';
 -- Now delete existing data
 TRUNCATE detailed_table;
 TRUNCATE category_revenue_summary;
 TRUNCATE product_summary;
 TRUNCATE campaign_summary;
 -- Populate tables
 PERFORM populate_detailed();
 PERFORM populate_product_summary();
 PERFORM populate_category_summary();
 PERFORM populate_campaign_summary();
END;
$$;


CALL refresh_tables ()
```

Conclusions –

Using the data from each summary table, we were able to draw clear conclusions addressing each business question. By leveraging Tableau for visualization, we translated raw numbers into intuitive and digestible insights. These visualizations enhanced our understanding of the data and provided actionable information that can support strategic business decisions.

Which product categories generate the highest revenue?

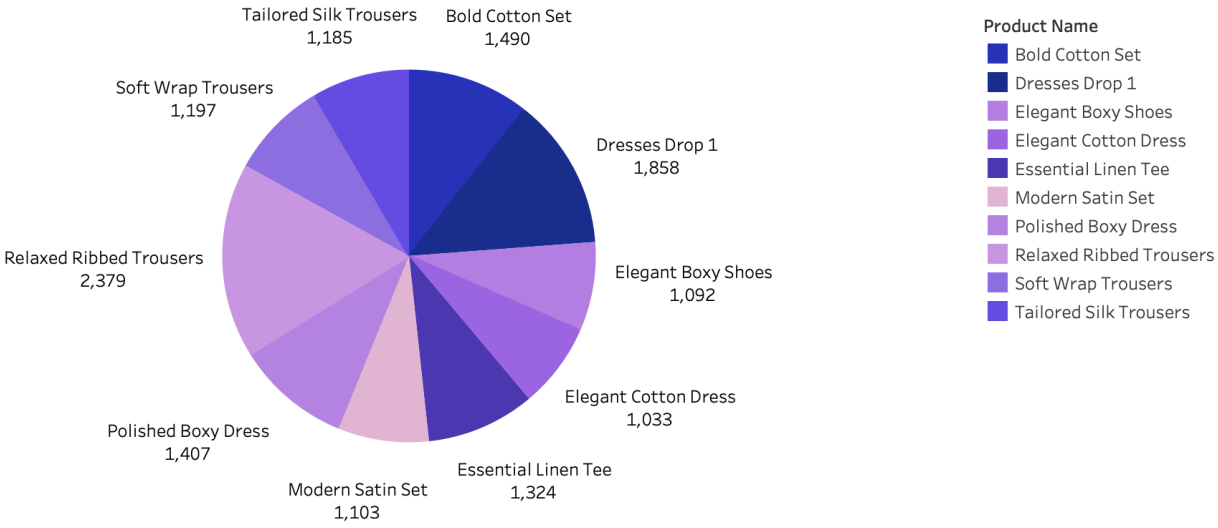


Among the product categories analyzed—Shoes, T-Shirts, Dresses, Sleepwear, and Pants—**Shoes** emerged as the top-performing category. Over the two-month period, the **Shoes** category generated a total revenue of **$70,074**, making it the most profitable product category during this time frame.

This insight helps prioritize inventory planning, marketing focus, and promotional efforts toward high-performing categories like footwear.

What are the top-selling products?

Top 10 Products Revenue



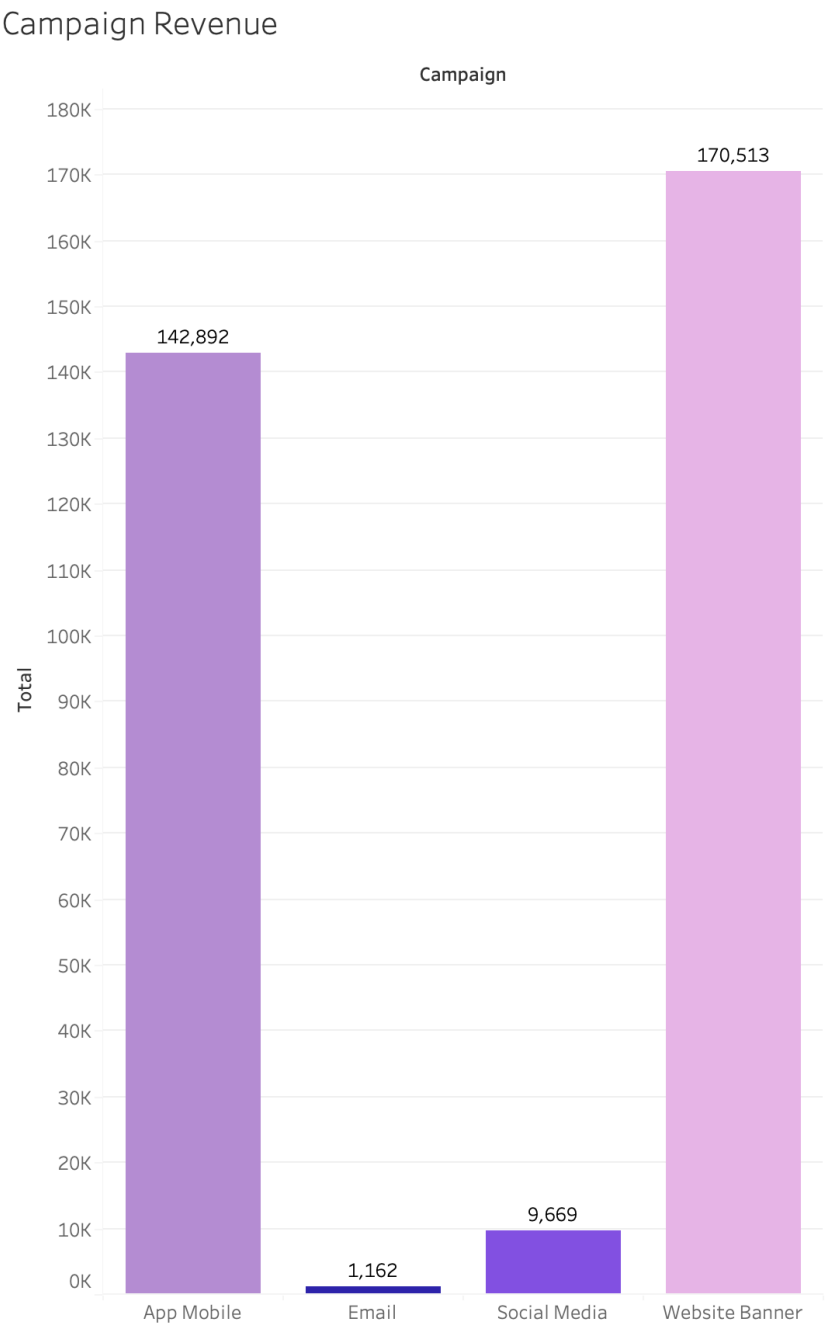
The charts above highlight the top 10 best-selling products for a comprehensive view. For the purpose of this analysis, we focus on the top 5 products based on two key performance metrics: quantity sold and total revenue.

- **Top 5 Products by Quantity Sold –**
  Modern Satin Set, Relaxed Ripped Trousers, Polished Boxy Dress, Elegant Boxy Shorts, and Elegant Cotton Dress.

- **Top 5 Products by Revenue Generated –**
  Relaxed Ripped Trousers, Dress Drop 1, Bold Cotton Set, Polished Boxy Dress, and Essential Linen Tee.

This dual perspective provides both volume-based and revenue-based insights, helping the business understand not only which products are popular but also which are the most profitable.

Which sales channel Campaigns Performed Best?



Based on the campaign summary data, the Website Banner campaign under the **E-commerce channel was the most successful, generating approximately $170,513 in revenue.** This was followed by the Mobile App, which brought in around $142,892.

In comparison, Social Media campaigns generated $9,669, while Email campaigns contributed the least with $1,162 in revenue.

These results suggest that digital storefront visibility—particularly through website banners and mobile apps—has the highest impact on sales performance, whereas social and email campaigns may need further optimization or a reallocation of marketing efforts.