FROM PLASTIC TO SUSTAINABLE: EVALUATING DATA MART'S SALES DYNAMICS IN A CHANGING LANDSCAPE

Fatih Sahin

https://8weeksqlchallenge.com/case-study-5/

CONTENTS

Introduction

This report presents the findings from the fifth case study of the '8WeekSQLChallenge' designed by Danny Ma.This case study dives deep into the dynamic world of Data Mart, a leading retail chain that has been at the forefront of adopting sustainable business practices. In a bold move, Data Mart introduced sustainable packaging across its product range, aiming to reduce its carbon footprint and cater to an increasingly eco-conscious consumer base

My analytical journey began when I took on the "Data Mart Business Analysis Challenge", a unique opportunity to explore the impact of sustainable initiatives on business metrics. This challenge provides a simulated data environment, reflecting real-world sales, customer demographics, and other essential business parameters. For those interested in a deeper dive into the challenge's specifics, you can explore more at https://8weeksqlchallenge.com/case-study-5/

Such challenges are not just exercises in SQL proficiency; they mirror real business scenarios, pushing analysts to derive actionable insights from raw data.

For the analysis presented in this report, I utilized the MS SQL Server Management Studio, a robust tool that offers unparalleled capabilities for data querying and manipulation. All insights and findings shared in this document result from careful analysis, and the expertise I've developed through extensive SQL practice.

The goal of this report is twofold: to present key findings from the Data Mart dataset and to showcase the unparalleled power of SQL in deriving business insights. I hope that this case study serves as both an informative read and an inspiration for budding data analysts and professionals, emphasizing the importance of data-driven decision-making in today's business landscape.

Problem Statement

Data Mart, a trailblazer in the retail domain, is at a pivotal juncture. Spearheaded by Danny, a visionary who believes in sustainable business practices, the company is looking to evaluate the impact of its recent sustainable packaging initiative on its sales and business metrics. However, before delving into the deeper analysis, a vital foundational step was the meticulous data cleansing process:

- **Data Format Standardization**: The week date values were converted to a consistent DATE format, ensuring uniformity in date-related operations.
- **Enhancement of Date Metrics**: The data was enriched with week numbers, month numbers, and calendar years to aid in time-series analysis.
- **Segmentation Enhancement**: New columns were introduced, such as age_band and demographic, based on the segment values for a more granular customer analysis.
- **Null Handling**: Null values in the segment column, as well as in the newly introduced age_band and demographic columns, were replaced with the string "unknown" for clarity.
- **Transaction Metrics**: A new avg_transaction column was generated by dividing sales by transactions, providing a clearer per-transaction view.

With the data cleaned and enhanced, the challenges posed to the business became multifaceted:

- **Sales Dynamics Analysis**: Explore the shifts in sales figures before and after the sustainable packaging introduction, providing insights into the overall business impact of the initiative.
- **Demographic & Age Analysis**: Dive deep into customer segments to identify which groups have been most responsive to the sustainable packaging.

- **Regional Analysis**: Evaluate regional responses to the packaging changes, helping to pinpoint areas of significant impact or potential areas of concern.
- **Platform Performance**: Understand the sales differences between the Retail and Shopify platforms, offering insights into where the sustainable initiatives are most resonating.
- **Customer Type Insights**: Break down the data by new, existing, and guest customers to ascertain how each segment reacts to the sustainable packaging and what it might mean for future marketing and sales strategies.

Through this multifaceted approach, and with the foundation of a clean and enhanced dataset, Data Mart aims to gather comprehensive insights, ensuring the sustainable packaging initiative not only aligns with its eco-friendly vision but also drives business growth.

Creating Schema and Tables

All datasets exist within the `data_mart` database schema.

Schema SQL

```sql
CREATE SCHEMA data_mart;
SET search_path = data_mart;


DROP TABLE IF EXISTS data_mart.weekly_sales;
CREATE TABLE data_mart.weekly_sales (
  "week_date" VARCHAR(7),
  "region" VARCHAR(13),
  "platform" VARCHAR(7),
  "segment" VARCHAR(4),
  "customer_type" VARCHAR(8),
  "transactions" INTEGER,
  "sales" INTEGER
);

INSERT INTO data_mart.weekly_sales
  ("week_date", "region", "platform", "segment", "customer_type", "transactions", "sales")
VALUES
  ('31/8/20', 'ASIA', 'Retail', 'C3', 'New', '120631', '3656163'),
  ('31/8/20', 'ASIA', 'Retail', 'F1', 'New', '31574', '996575'),
  ('31/8/20', 'USA', 'Retail', 'null', 'Guest', '529151', '16509610'),
  ('31/8/20', 'EUROPE', 'Retail', 'C1', 'New', '4517', '141942'),
  ('31/8/20', 'AFRICA', 'Retail', 'C2', 'New', '58046', '1758388'),
  ('31/8/20', 'CANADA', 'Shopify', 'F2', 'Existing', '1336', '243878'),
  ('31/8/20', 'AFRICA', 'Shopify', 'F3', 'Existing', '2514', '519502'),
  ('31/8/20', 'ASIA', 'Shopify', 'F1', 'Existing', '2158', '371417'),
  ('31/8/20', 'AFRICA', 'Shopify', 'F2', 'New', '318', '49557'),
  ('31/8/20', 'AFRICA', 'Retail', 'C3', 'New', '111032', '3888162'),
  ('31/8/20', 'USA', 'Shopify', 'F1', 'Existing', '1398', '260773'),
  ('31/8/20', 'OCEANIA', 'Shopify', 'C2', 'Existing', '4661', '882690'),
  ('31/8/20', 'SOUTH AMERICA', 'Retail', 'C2', 'Existing', '1029', '38762'),
  ('31/8/20', 'SOUTH AMERICA', 'Shopify', 'C4', 'New', '6', '917'),
  ('31/8/20', 'EUROPE', 'Shopify', 'F3', 'Existing', '115', '35215'),
  ('31/8/20', 'OCEANIA', 'Retail', 'F3', 'Existing', '551905', '30371770'),
  ('31/8/20', 'ASIA', 'Shopify', 'C3', 'Existing', '1969', '374327'),
  ('31/8/20', 'AFRICA', 'Retail', 'F1', 'Existing', '97604', '5185233'),
  ('31/8/20', 'OCEANIA', 'Retail', 'C2', 'New', '111219', '2980673'),
  ('31/8/20', 'USA', 'Retail', 'F1', 'New', '11820', '463738'),
  ('31/8/20', 'SOUTH AMERICA', 'Retail', 'F3', 'Existing', '1363', '65730'),
  ('31/8/20', 'AFRICA', 'Retail', 'C3', 'Existing', '284971', '14430196'),
  ('31/8/20', 'ASIA', 'Retail', 'F2', 'New', '70496', '2176980'),
  ('31/8/20', 'AFRICA', 'Shopify', 'F1', 'Existing', '2678', '478756'),
  ('31/8/20', 'USA', 'Shopify', 'C4', 'New', '22', '3319'),
  ('31/8/20', 'CANADA', 'Retail', 'F3', 'Existing', '94274', '5306746'),
  ('31/8/20', 'ASIA', 'Retail', 'F1', 'Existing', '94287', '4511841'),
  ('31/8/20', 'EUROPE', 'Retail', 'null', 'New', '3064', '134249'),
  ('31/8/20', 'EUROPE', 'Shopify', 'F1', 'New', '7', '1579'),
  ('31/8/20', 'SOUTH AMERICA', 'Retail', 'C4', 'New', '329', '11451'),
  ('31/8/20', 'SOUTH AMERICA', 'Retail', 'F1', 'Existing', '854', '31589'),
  ('31/8/20', 'EUROPE', 'Shopify', 'C2', 'Existing', '180', '53567'),
  ('31/8/20', 'EUROPE', 'Shopify', 'F2', 'New', '15', '4023'),
  ('31/8/20', 'AFRICA', 'Retail', 'C2', 'Existing', '112361', '4768214'),
  ('31/8/20', 'ASIA', 'Shopify', 'C2', 'Existing', '2269', '396909'),
  ('31/8/20', 'AFRICA', 'Shopify', 'C4', 'New', '58', '8562'),
  ('31/8/20', 'USA', 'Retail', 'F3', 'Existing', '142898', '8723663'),
  ('31/8/20', 'OCEANIA', 'Shopify', 'C3', 'Existing', '4703', '957939'),
  ('31/8/20', 'SOUTH AMERICA', 'Shopify', 'null', 'New', '40', '7625'),
  ('31/8/20', 'USA', 'Shopify', 'C1', 'New', '164', '20635'),
  ('31/8/20', 'SOUTH AMERICA', 'Shopify', 'C1', 'New', '20', '2265'),
  ('31/8/20', 'EUROPE', 'Retail', 'C1', 'Existing', '14583', '633917'),
  ('31/8/20', 'OCEANIA', 'Shopify', 'F2', 'New', '452', '72293'),
  ('31/8/20', 'SOUTH AMERICA', 'Retail', 'F2', 'New', '382', '13839'),
  ('31/8/20', 'SOUTH AMERICA', 'Retail', 'C3', 'Existing', '1503', '68009'),
  ('31/8/20', 'CANADA', 'Retail', 'F1', 'New', '7033', '269176'),
  ('31/8/20', 'SOUTH AMERICA', 'Retail', 'null', 'Existing', '329', '10874'),
  ('31/8/20', 'CANADA', 'Retail', 'F1', 'Existing', '24153', '1306932'),
  ('31/8/20', 'EUROPE', 'Shopify', 'null', 'Existing', '15', '3507'),
  ('31/8/20', 'EUROPE', 'Retail', 'F3', 'Existing', '18676', '1144376'),
  ('31/8/20', 'SOUTH AMERICA', 'Shopify', 'C3', 'New', '19', '3325'),
  ('31/8/20', 'CANADA', 'Shopify', 'null', 'Existing', '87', '15074'),
  ('31/8/20', 'CANADA', 'Retail', 'C4', 'Existing', '29577', '1473178'),
  ('31/8/20', 'OCEANIA', 'Retail', 'F1', 'New', '41631', '1369115'),
  ('31/8/20', 'ASIA', 'Retail', 'F3', 'Existing', '402652', '20255591'),
  ('31/8/20', 'ASIA', 'Retail', 'null', 'Guest', '1936351', '48773122'),
  ('31/8/20', 'ASIA', 'Shopify', 'C2', 'New', '413', '55780'),
  ('31/8/20', 'EUROPE', 'Shopify', 'F3', 'New', '4', '554'),
```

Entity Relationship Diagram

Data Structure

Data Mart utilizes a streamlined approach in managing its vast array of sales data, primarily focusing on a single table that captures weekly sales metrics. However, in the spirit of data refinement, the initial table underwent crucial cleansing steps, resulting in the creation of an enhanced table. Here's a breakdown of the tables:

1. **data_mart.weekly_sales**: This table is the primary source of all sales data for Data Mart. The columns encapsulate crucial information such as:
    - **week_date**: The specific week of the sales data.
    - **region**: Geographical regions like ASIA, EUROPE, etc.
    - **platform**: The platform through which sales were made, like Retail or Shopify.
    - **segment**: A unique identifier used for customer segmentation.
    - **customer_type**: Classification of the customers, such as New, Existing, or Guest.
    - **transactions**: The number of transactions made during the specified week.
    - **sales**: Total sales for the week.
2. **data_mart.clean_weekly_sales**: This is an enhanced version of the weekly_sales table, born out of the data cleansing process. Apart from the columns present in the original table, it boasts of additional columns that provide more granular insights. Some of the new columns include:
    - **week_number**: The specific number of the week in the year.
    - **month_number**: The calendar month of the sales data.
    - **calendar_year**: The specific year of the sales data, such as 2018, 2019, or 2020.
    - **age_band**: Categories like Young Adults, Middle Aged, or Retirees, derived from the segment column.
    - **demographic**: Categories based on customer types, like Couples or Families.
    - **avg_transaction**: Average transaction size, calculated as sales divided by transactions.

These tables, particularly data_mart.clean_weekly_sales, serve as the backbone of Data Mart's analytics endeavors. Their structure and relationships not only facilitate

comprehensive insights into sales dynamics but also provide a flexible framework for future data enrichment and analysis.

data_mart.weekly_sales	
week_date	VARCHAR(7)
region	VARCHAR(13)
platform	VARCHAR(7)
segment	VARCHAR(4)
customer_type	VARCHAR(8)
transactions	INTEGER
sales	INTEGER

data_mart.clean_weekly_sales	
week_date	date
week_number	integer
month_number	integer
calendar_year	integer
region	varchar
platform	varchar
segment	varchar
age_band	varchar
demographic	varchar
customer_type	varchar
transactions	integer
sales	integer
avg_transaction	float

Case Study Questions

A. 1. Data Cleansing Steps

1. Convert week_date to DATE format
2. Add week_number column
3. Add month_number column
4. Add calendar_year column
5. Add age_band column
6. Add demographic column
7. Handle NULL values
8. Generate avg_transaction column

```sql
--1. Data Cleansing Steps

-- If the table exists, drop it first
IF OBJECT_ID('data_mart.clean_weekly_sales', 'U') IS NOT NULL
  DROP TABLE data_mart.clean_weekly_sales;
GO

-- Create the new cleaned table
SELECT
    -- Convert week_date to DATE format
    CONVERT(DATE, week_date, 3) AS week_date,

    -- Add week_number column
    DATEPART(WEEK, CONVERT(DATE, week_date, 3)) AS week_number,

    -- Add month_number column
    DATEPART(MONTH, CONVERT(DATE, week_date, 3)) AS month_number,

    -- Add calendar_year column
    DATEPART(YEAR, CONVERT(DATE, week_date, 3)) AS calendar_year,

    -- Existing columns
    region, platform,

    -- Handle NULL values in segment
    CASE
        WHEN segment='null' THEN 'unknown'
        ELSE segment
    END AS segment,

    -- Add age_band column
    CASE
        WHEN segment LIKE '%1' THEN 'Young Adults'
        WHEN segment LIKE '%2' THEN 'Middle Aged'
        WHEN segment LIKE '%3' OR segment LIKE '%4' THEN 'Retirees'
        ELSE 'unknown'
    END AS age_band,

    -- Add demographic column
    CASE
        WHEN segment LIKE 'C%' THEN 'Couples'
        WHEN segment LIKE 'F%' THEN 'Families'
        ELSE 'unknown'
    END AS demographic,

    -- Existing columns
    customer_type, transactions, sales,

    -- Generate avg_transaction column
    ROUND(CAST(sales AS FLOAT) / transactions, 2) AS avg_transaction

INTO data_mart.clean_weekly_sales

FROM data_mart.weekly_sales;
GO
```

1. What day of the week is used for each week_date value?

```
SELECT week_date,
       DATEPART(WEEKDAY, CONVERT(DATE, week_date, 3)) AS DayOfWeekNumber,
       DATENAME(WEEKDAY, CONVERT(DATE, week_date, 3)) AS DayOfWeekName
FROM data_mart.clean_weekly_sales
GROUP BY week_date
ORDER BY week_date;
```

	week_date	DayOfWeekNumber	DayOfWeekName
1	2018-03-26	2	Monday
2	2018-04-02	2	Monday
3	2018-04-09	2	Monday
4	2018-04-16	2	Monday
5	2018-04-23	2	Monday
6	2018-04-30	2	Monday
7	2018-05-07	2	Monday
8	2018-05-14	2	Monday
9	2018-05-21	2	Monday
10	2018-05-28	2	Monday
11	2018-06-04	2	Monday
12	2018-06-11	2	Monday
13	2018-06-18	2	Monday
14	2018-06-25	2	Monday
15	2018-07-02	2	Monday
16	2018-07-09	2	Monday
17	2018-07-16	2	Monday
18	2018-07-23	2	Monday
19	2018-07-30	2	Monday
20	2018-08-06	2	Monday
21	2018-08-13	2	Monday
22	2018-08-20	2	Monday

2. What range of week numbers are missing from the dataset?

```
WITH Weeks AS (
    SELECT DISTINCT DATEPART(WEEK, CONVERT(DATE, week_date, 3)) AS week_num,
                    DATEPART(YEAR, CONVERT(DATE, week_date, 3)) AS year_num
    FROM data_mart.clean_weekly_sales
)

, AllWeeks AS (
    SELECT DISTINCT a.week_num, w.year_num
    FROM (VALUES (1), (2), (3), (4), (5), (6), (7), (8), (9), (10), (11), (12),
                 (13), (14), (15), (16), (17), (18), (19), (20), (21), (22), (23), (24),
                 (25), (26), (27), (28), (29), (30), (31), (32), (33), (34), (35), (36),
                 (37), (38), (39), (40), (41), (42), (43), (44), (45), (46), (47), (48),
                 (49), (50), (51), (52)) AS a(week_num),
                 Weeks w
)

SELECT a.week_num as missing_weeks, a.year_num
FROM AllWeeks a
LEFT JOIN Weeks w ON a.week_num = w.week_num AND a.year_num = w.year_num
WHERE w.week_num IS NULL
ORDER BY a.year_num, a.week_num;
```

	missing_weeks	year_num
1	1	2018
2	2	2018
3	3	2018
4	4	2018
5	5	2018
6	6	2018
7	7	2018
8	8	2018
9	9	2018
10	10	2018
11	11	2018
12	12	2018
13	37	2018
14	38	2018
15	39	2018
16	40	2018
17	41	2018
18	42	2018
19	43	2018
20	44	2018
21	45	2018
22	46	2018
23	47	2018
24	48	2018
25	49	2018
26	50	2018
27	51	2018
28	52	2018
29	1	2019
30	2	2019
31	3	2019
32	4	2019
33	5	2019
34	6	2019
35	7	2019
36	8	2019
37	9	2019
38	10	2019
39	11	2019
40	12	2019
41	37	2019
42	38	2019
43	39	2019
44	40	2019
45	41	2019
46	42	2019
47	43	2019
48	44	2019
49	45	2019
50	46	2019
51	47	2019
52	48	2019
53	49	2019
54	50	2019
55	51	2019
56	52	2019
57	1	2020
58	2	2020
59	3	2020
60	4	2020
61	5	2020
62	6	2020
63	7	2020
64	8	2020
65	9	2020
66	10	2020
67	11	2020
68	12	2020
69	37	2020
70	38	2020
71	39	2020
72	40	2020
73	41	2020
74	42	2020
75	43	2020
76	44	2020
77	45	2020
78	46	2020
79	47	2020
80	48	2020
81	49	2020
82	50	2020
83	51	2020
84	52	2020

3. How many total transactions were there for each year in the dataset?

```
SELECT calendar_year,
       SUM(transactions) AS total_transactions
FROM data_mart.clean_weekly_sales
GROUP BY calendar_year
ORDER BY calendar_year;
```

	calendar_year	total_transactions
1	2018	346406460
2	2019	365639285
3	2020	375813651

4. What is the total sales for each region for each month?

```
SELECT
     calendar_year,
     month_number,
     region,
     SUM (CAST(sales as BIGINT)) as total_sales
FROM data_mart.clean_weekly_sales
GROUP BY month_number,region,calendar_year
ORDER BY calendar_year,month_number
```

	calendar_year	month_number	region	total_sales
1	2018	3	EUROPE	8402183
2	2018	3	CANADA	33815571
3	2018	3	AFRICA	130542213
4	2018	3	SOUTH AMERICA	16302144
5	2018	3	OCEANIA	175777460
6	2018	3	USA	52734998
7	2018	3	ASIA	119180883
8	2018	4	USA	260725717
9	2018	4	SOUTH AMERICA	80814046
10	2018	4	CANADA	163479820
11	2018	4	EUROPE	44549418
12	2018	4	ASIA	603716301
13	2018	4	OCEANIA	869324594
14	2018	4	AFRICA	650194751
15	2018	5	USA	210050720
16	2018	5	ASIA	472634283
17	2018	5	SOUTH AMERICA	63685837
18	2018	5	OCEANIA	692610094
19	2018	5	EUROPE	36492553
20	2018	5	CANADA	130367940
21	2018	5	AFRICA	522814997
22	2018	6	USA	206372070
23	2018	6	ASIA	462233474
24	2018	6	OCEANIA	687546255
25	2018	6	AFRICA	519127094
26	2018	6	EUROPE	38998277
27	2018	6	CANADA	130410790
28	2018	6	SOUTH AMERICA	63764243
29	2018	7	EUROPE	50535910
30	2018	7	CANADA	164198426
31	2018	7	OCEANIA	871333919
32	2018	7	SOUTH AMERICA	81690746
33	2018	7	AFRICA	674135866

5. What is the total count of transactions for each platform?

```
SELECT
    platform,
    SUM(transactions) as total_count_transactions
FROM data_mart.clean_weekly_sales
GROUP BY platform
```

	platform	total_count_transactions
1	Retail	1081934227
2	Shopify	5925169

6. What is the percentage of sales for Retail vs Shopify for each month?

```
WITH MonthlySales AS (
    -- Aggregating total sales by month, year, and platform
    SELECT
        calendar_year,
        month_number,
        platform,
        SUM(CAST(sales AS BIGINT)) AS platform_sales
    FROM data_mart.clean_weekly_sales
    GROUP BY calendar_year, month_number, platform
),
TotalMonthlySales AS (
    -- Aggregating total sales by month and year (regardless of platform)
    SELECT
        calendar_year,
        month_number,
        SUM(platform_sales) AS total_month_sales
    FROM MonthlySales
    GROUP BY calendar_year, month_number
)
SELECT
    ms.calendar_year,
    ms.month_number,
    ms.platform,
    ms.platform_sales,
    ROUND((CAST(ms.platform_sales AS FLOAT) / tms.total_month_sales) * 100,2) AS percentage_of_total
FROM MonthlySales ms
JOIN TotalMonthlySales tms ON ms.calendar_year = tms.calendar_year AND ms.month_number = tms.month_number
ORDER BY ms.calendar_year, ms.month_number, ms.platform;
```

	calendar_year	month_number	platform	platform_sales	percentage_of_total
1	2018	3	Retail	525583061	97.92
2	2018	3	Shopify	11172391	2.08
3	2018	4	Retail	2617369077	97.93
4	2018	4	Shopify	55435570	2.07
5	2018	5	Retail	2080290488	97.73
6	2018	5	Shopify	48365936	2.27
7	2018	6	Retail	2061128568	97.76
8	2018	6	Shopify	47323635	2.24
9	2018	7	Retail	2646368290	97.75
10	2018	7	Shopify	60830182	2.25
11	2018	8	Retail	2140297292	97.71
12	2018	8	Shopify	50244975	2.29
13	2018	9	Retail	540134542	97.68
14	2018	9	Shopify	12836820	2.32
15	2019	3	Retail	567984858	97.71
16	2019	3	Shopify	13332196	2.29
17	2019	4	Retail	2836349313	97.8
18	2019	4	Shopify	63798008	2.2
19	2019	5	Retail	2221160706	97.52
20	2019	5	Shopify	56371106	2.48
21	2019	6	Retail	2181126868	97.42
22	2019	6	Shopify	57727053	2.58
23	2019	7	Retail	2785870177	97.35
24	2019	7	Shopify	75766614	2.65
25	2019	8	Retail	2240942490	97.21
26	2019	8	Shopify	64297818	2.79
27	2019	9	Retail	564372315	97.09
28	2019	9	Shopify	16932978	2.91
29	2020	3	Retail	1205620498	97.3
30	2020	3	Shopify	33475731	2.7
31	2020	4	Retail	2281873844	96.96
32	2020	4	Shopify	71478722	3.04
33	2020	5	Retail	2284387029	96.71
34	2020	5	Shopify	77687860	3.29
35	2020	6	Retail	2807693824	96.8
36	2020	6	Shopify	92714414	3.2
37	2020	7	Retail	2255852981	96.67
38	2020	7	Shopify	77642565	3.33
39	2020	8	Retail	2810210216	96.51
40	2020	8	Shopify	101583216	3.49

7. What is the percentage of sales by demographic for each year in the dataset?

```
WITH yearly_sales AS(
SELECT
    calendar_year,
    demographic,
    SUM(CAST(sales AS BIGINT)) AS demographic_sales
FROM data_mart.clean_weekly_sales
GROUP BY calendar_year, demographic
)
,
total_yearly_sales AS (
     SELECT
        calendar_year,
        SUM(demographic_sales) AS total_yearly_sales
    FROM yearly_sales
    GROUP BY calendar_year)
SELECT
    ys.calendar_year,
    ys.demographic,
    ys.demographic_sales,
    ROUND((CAST(ys.demographic_sales AS FLOAT) / tys.total_yearly_sales) * 100,2) AS percentage_of_total
FROM yearly_sales ys
JOIN total_yearly_sales tys ON ys.calendar_year = tys.calendar_year
ORDER BY ys.calendar_year, ys.demographic;
```

	calendar_year	demographic	demographic_sales	percentage_of_total
1	2018	Couples	3402388688	26.38
2	2018	Families	4125558033	31.99
3	2018	unknown	5369434106	41.63
4	2019	Couples	3749251935	27.28
5	2019	Families	4463918344	32.47
6	2019	unknown	5532862221	40.25
7	2020	Couples	4049566928	28.72
8	2020	Families	4614338065	32.73
9	2020	unknown	5436315907	38.55

8. Which age_band and demographic values contribute the most to Retail sales?

```
WITH all_sales AS (
SELECT
    age_band,
    demographic,
    SUM(CAST(sales AS BIGINT)) as total_sales,
    rank() OVER(ORDER BY SUM(CAST(sales AS BIGINT)) desc) AS rank
FROM data_mart.clean_weekly_sales
WHERE platform='Retail' AND age_band!='unknown'
GROUP BY age_band,demographic
)

SELECT
    age_band,
    demographic,
    total_sales
FROM all_sales
WHERE rank=1
```

	age_band	demographic	total_sales
1	Retirees	Families	6634686916

9. Can we use the avg_transaction column to find the average transaction size for each year for Retail vs Shopify? If not - how would you calculate it instead?

If we want to find the average transaction size for each year for Retail vs. Shopify, we'd need to do a weighted average based on the number of transactions, rather than simply averaging the avg_transaction values. This is because each avg_transaction value pertains to a different number of transactions

To illustrate why, consider two weeks as an example:

Week 1: 100 transactions, $1000 total, so average transaction is $10.

Week 2: 10 transactions, $500 total, so average transaction is $50.

To compute the average transaction size for each year for Retail vs Shopify, we would sum up all the sales for each platform for each year, and then divide by the sum of transactions for each platform for each year.

```sql
SELECT
    calendar_year,
    platform,
    SUM(CAST(sales AS BIGINT)) / SUM(transactions) AS avg_transaction_size
FROM data_mart.clean_weekly_sales
WHERE platform IN ('Retail', 'Shopify')
GROUP BY calendar_year, platform
ORDER BY calendar_year, platform;
```

	calendar_year	platform	avg_transaction_size
1	2018	Retail	36
2	2018	Shopify	192
3	2019	Retail	36
4	2019	Shopify	183
5	2020	Retail	36
6	2020	Shopify	179

B. BEFORE & AFTER ANALYSIS

1. What is the total sales for the 4 weeks before and after 2020-06-15?
2. What is the growth or reduction rate in actual values and percentage of sales?

```sql
WITH BeforeAfterSales AS (
    SELECT
    CASE
        WHEN week_date < '2020-06-15' THEN 'Before'
        ELSE 'After'
    END AS period,
    SUM(CAST(sales AS BIGINT)) AS total_sales
FROM data_mart.clean_weekly_sales
WHERE
    -- 4 weeks before 2020-06-15 up to the day before 2020-06-15
    (week_date BETWEEN DATEADD(WEEK, -4, '2020-06-15') AND DATEADD(DAY, -1, '2020-06-15'))
    -- Or, the week of 2020-06-15 and the 3 weeks after
    OR (week_date BETWEEN '2020-06-15' AND DATEADD(WEEK, 3, '2020-06-15'))
GROUP BY
    CASE
        WHEN week_date < '2020-06-15' THEN 'Before'
        ELSE 'After'
    END
)

--Calculate the growth or reduction in actual values and percentage
SELECT
    b.total_sales as BeforeSales,
    a.total_sales as AfterSales,
    (a.total_sales - b.total_sales) as Difference,
    ROUND(CAST((a.total_sales - b.total_sales) * 100.0 / b.total_sales AS DECIMAL(10,2)), 2) as GrowthReductionRate
FROM BeforeAfterSales a, BeforeAfterSales b
WHERE a.Period = 'After' AND b.Period = 'Before';
```

	BeforeSales	AfterSales	Difference	GrowthReductionRate
1	2345878357	2318994169	-26884188	-1.15

3. What about the entire 12 weeks before and after?

```sql
WITH BeforeAfterSales AS (
    SELECT
    CASE
        WHEN week_date < '2020-06-15' THEN 'Before'
        ELSE 'After'
    END AS period,
    SUM(CAST(sales AS BIGINT)) AS total_sales
FROM data_mart.clean_weekly_sales
WHERE
    -- 12 weeks before 2020-06-15 up to the day before 2020-06-15
    (week_date BETWEEN DATEADD(WEEK, -12, '2020-06-15') AND DATEADD(DAY, -1, '2020-06-15'))
    -- Or, the week of 2020-06-15 and the 11 weeks after
    OR (week_date BETWEEN '2020-06-15' AND DATEADD(WEEK, 11, '2020-06-15'))
GROUP BY
    CASE
        WHEN week_date < '2020-06-15' THEN 'Before'
        ELSE 'After'
    END
)

-- Calculate the growth or reduction in actual values and percentage
SELECT
    b.total_sales as BeforeSales,
    a.total_sales as AfterSales,
    (a.total_sales - b.total_sales) as Difference,
    ROUND(CAST((a.total_sales - b.total_sales) * 100.0 / b.total_sales AS DECIMAL(10,2)), 2) as GrowthReductionRate
FROM BeforeAfterSales a, BeforeAfterSales b
WHERE a.Period = 'After' AND b.Period = 'Before';
```

BeforeSales	AfterSales	Difference	GrowthReductionRate
7126273147	6973947753	-152325394	-2.14

4. **How do the sale metrics for these 2 periods before and after compare with the previous years in 2018 and 2019?**

```
SELECT
    calendar_year,
    SUM(CAST(sales AS BIGINT)) AS total_sales
FROM data_mart.clean_weekly_sales
WHERE calendar_year in ('2018','2019')
GROUP BY calendar_year
ORDER BY calendar_year
```

	calendar_year	total_sales
1	2018	12897380827
2	2019	13746032500

C.BONUS QUESTION

1. **Which areas of the business have the highest negative impact in sales metrics performance in 2020 for the 12 weeks before and after period?**

-region

-platform

-age_band

-demographic

-customer_type

For Region:

```
WITH BeforeSales AS (
    SELECT
        region,
        SUM(CAST(sales AS BIGINT)) as total_sales_before
    FROM data_mart.clean_weekly_sales
    WHERE week_date BETWEEN DATEADD(WEEK, -12, '2020-06-15') AND '2020-06-14'
    GROUP BY region
),
AfterSales AS (
    SELECT
        region,
        SUM(CAST(sales AS BIGINT)) as total_sales_after
    FROM data_mart.clean_weekly_sales
    WHERE week_date BETWEEN '2020-06-15' AND DATEADD(WEEK, 12, '2020-06-15')
    GROUP BY region
)
SELECT
    b.region,
    b.total_sales_before,
    a.total_sales_after,
    (a.total_sales_after - b.total_sales_before) as Difference
FROM AfterSales a
JOIN BeforeSales b ON a.region = b.region
ORDER BY Difference;
```

	region	total_sales_before	total_sales_after	Difference
1	OCEANIA	2354116790	2282795690	-71321100
2	ASIA	1637244466	1583807621	-53436845
3	USA	677013558	666198715	-10814843
4	AFRICA	1709537105	1700390294	-9146811
5	CANADA	426438454	418264441	-8174013
6	SOUTH AMERICA	213036207	208452033	-4584174
7	EUROPE	108886567	114038959	5152392

For Platform:

```
WITH BeforeSales AS (
    SELECT
        platform,
        SUM(CAST(sales AS BIGINT)) as total_sales_before
    FROM data_mart.clean_weekly_sales
    WHERE week_date BETWEEN DATEADD(WEEK, -12, '2020-06-15') AND '2020-06-14'
    GROUP BY platform
),
AfterSales AS (
    SELECT
        platform,
        SUM(CAST(sales AS BIGINT)) as total_sales_after
    FROM data_mart.clean_weekly_sales
    WHERE week_date BETWEEN '2020-06-15' AND DATEADD(WEEK, 12, '2020-06-15')
    GROUP BY platform
)
SELECT
    b.platform,
    b.total_sales_before,
    a.total_sales_after,
    (a.total_sales_after - b.total_sales_before) as Difference
FROM AfterSales a
JOIN BeforeSales b ON a.platform = b.platform
ORDER BY Difference;
```

	platform	total_sales_before	total_sales_after	Difference
1	Retail	6906861113	6738777279	-168083834
2	Shopify	219412034	235170474	15758440

For Age band:

```
WITH BeforeSales AS (
    SELECT
        age_band,
        SUM(CAST(sales AS BIGINT)) as total_sales_before
    FROM data_mart.clean_weekly_sales
    WHERE week_date BETWEEN DATEADD(WEEK, -12, '2020-06-15') AND '2020-06-14'
    GROUP BY age_band
),
AfterSales AS (
    SELECT
        age_band,
        SUM(CAST(sales AS BIGINT)) as total_sales_after
    FROM data_mart.clean_weekly_sales
    WHERE week_date BETWEEN '2020-06-15' AND DATEADD(WEEK, 12, '2020-06-15')
    GROUP BY age_band
)
SELECT
    b.age_band,
    b.total_sales_before,
    a.total_sales_after,
    (a.total_sales_after - b.total_sales_before) as Difference
FROM AfterSales a
JOIN BeforeSales b ON a.age_band = b.age_band
ORDER BY Difference;
```

	age_band	total_sales_before	total_sales_after	Difference
1	unknown	2764354464	2671961443	-92393021
2	Retirees	2395264515	2365714994	-29549521
3	Middle Aged	1164847640	1141853348	-22994292
4	Young Adults	801806528	794417968	-7388560

For Demographic:

```
WITH BeforeSales AS (
    SELECT
        demographic,
        SUM(CAST(sales AS BIGINT)) as total_sales_before
    FROM data_mart.clean_weekly_sales
    WHERE week_date BETWEEN DATEADD(WEEK, -12, '2020-06-15') AND '2020-06-14'
    GROUP BY demographic
),
AfterSales AS (
    SELECT
        demographic,
        SUM(CAST(sales AS BIGINT)) as total_sales_after
    FROM data_mart.clean_weekly_sales
    WHERE week_date BETWEEN '2020-06-15' AND DATEADD(WEEK, 12, '2020-06-15')
    GROUP BY demographic
)
SELECT
    b.demographic,
    b.total_sales_before,
    a.total_sales_after,
    (a.total_sales_after - b.total_sales_before) as Difference
FROM AfterSales a
JOIN BeforeSales b ON a.demographic = b.demographic
ORDER BY Difference;
```

	age_band	total_sales_before	total_sales_after	Difference
1	unknown	2764354464	2671961443	-92393021
2	Retirees	2395264515	2365714994	-29549521
3	Middle Aged	1164847640	1141853348	-22994292
4	Young Adults	801806528	794417968	-7388560

For customer type:

```
WITH BeforeSales AS (
    SELECT
        customer_type,
        SUM(CAST(sales AS BIGINT)) as total_sales_before
    FROM data_mart.clean_weekly_sales
    WHERE week_date BETWEEN DATEADD(WEEK, -12, '2020-06-15') AND '2020-06-14'
    GROUP BY customer_type
),
AfterSales AS (
    SELECT
        customer_type,
        SUM(CAST(sales AS BIGINT)) as total_sales_after
    FROM data_mart.clean_weekly_sales
    WHERE week_date BETWEEN '2020-06-15' AND DATEADD(WEEK, 12, '2020-06-15')
    GROUP BY customer_type
)
SELECT
    b.customer_type,
    b.total_sales_before,
    a.total_sales_after,
    (a.total_sales_after - b.total_sales_before) as Difference
FROM AfterSales a
JOIN BeforeSales b ON a.customer_type = b.customer_type
ORDER BY Difference;
```

	customer_type	total_sales_before	total_sales_after	Difference
1	Existing	3690116427	3606243454	-83872973
2	Guest	2573436301	2496233635	-77202666
3	New	862720419	871470664	8750245

Key Findings and Observations

1. **Data Consistency:**
    - The dataset consistently uses **Monday** as the **week's starting day across all data points**. This consistency aids in streamlining analyses based on weekly metrics.
2. **Data Gaps:**
    - The dataset has **significant gaps**, with weeks missing across the years 2018, 2019, and 2020. Specifically, the first 12 weeks and the last 16 weeks of each year are absent. This could lead to **potential inaccuracies** if yearly aggregates are used without accounting for these missing weeks.
3. **Transactional Volume:**
    - The total transactions have shown a **consistent annual increase** from 2018 to 2020. This indicates a growing customer base or increased activity over the years.
4. **Regional Sales Analysis:**
    - The sales distribution by region for each month indicates a significant volume in regions like **Oceania**, **Asia**, and **Africa**. These regions could be pivotal markets for the business.
5. **Platform Analysis:**
    - **Retail dominates** the platform usage, with over a billion transactions, while Shopify only accounts for around six million. This indicates that the core of the business lies in the retail platform, and Shopify might be a newer or less emphasized channel.
    - In terms of sales value, Retail's average transaction size remains consistent at $36 across the years. Meanwhile, Shopify's average transaction size decreased from $192 in 2018 to $179 in 2020, suggesting a slight reduction in average purchase value on this platform.
6. **Demographics:**
    - **Retiree** families **contribute the most to retail sales**, with a significant total of 6.63 billion. This demographic could be a key segment to target for marketing or product offerings.
    - The sales distribution by demographic over the years shows a fairly consistent pattern, with "unknown" demographics accounting for the largest

portion. This indicates a potential opportunity for better customer data collection and segmentation.

7. **Impact of Sustainable Packaging:**

    - A before-and-after analysis of the introduction of sustainable packaging reveals a slight **decrease in sales**. For the 4 weeks around the change, sales decreased by 1.15%, and considering 12 weeks, the drop was 2.14%. This might indicate a short-term impact, possibly due to customer adaptation to the new packaging or other related factors.

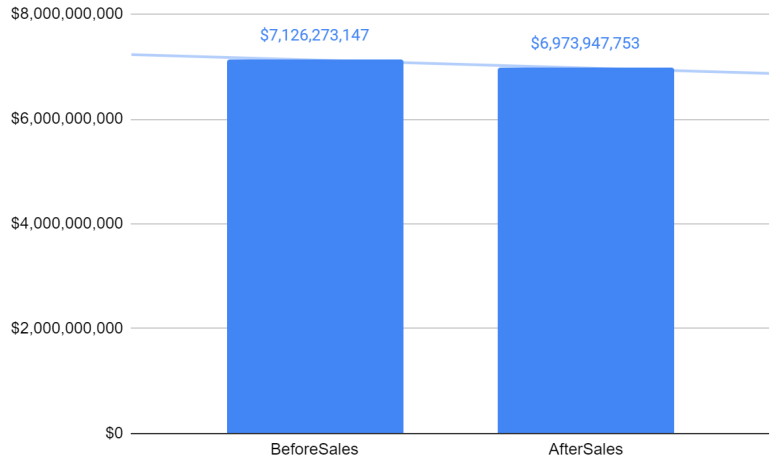
    - Comparatively, the total sales for the years 2018 and 2019 were significantly higher than the 12-week periods analyzed in 2020. This context is essential as it shows that the periods taken for the before-and-after analysis in 2020 are not necessarily representative of a typical sales trend for the business.

8. **Growth Perspective:**

    - Comparing the yearly totals, there was a **growth** of approximately **6.6% in sales** from 2018 to 2019. This positive growth trend emphasizes the importance of understanding the slight decrease post the sustainable packaging initiative in 2020.

In conclusion, while the business demonstrates robust growth and a **strong retail platform presence**, there are areas of **opportunity**, particularly in better understanding the impact of significant changes (like **sustainable packaging**) and improving **data collection for customer** demographics.

1. Sales Before vs. After Sustainable Packaging Change:

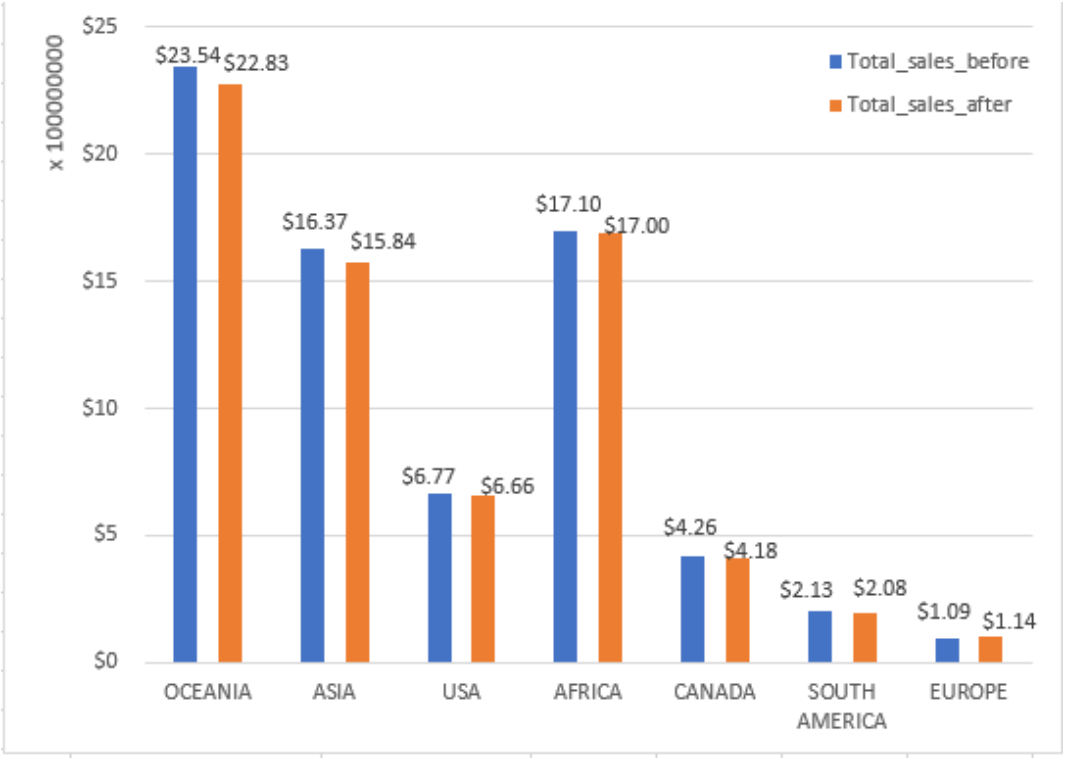


This chart has two main bars, one representing sales for the 12 weeks before June 15, 2020, and the other for the 12 weeks after. This visual representation shows the impact of the packaging change on sales.

2. Sales Distribution by Platform (2018-2020):

While **Retail** remains the **dominant platform** in terms of total sales, **Shopify** is showing a **strong growth trajectory** that, if sustained, might make it a significant competitor or alternative in the future.

3. Regional Sales Differences Before and After the Change:



The combined **negative difference across all the regions (excluding Europe)** is substantial. This suggests that the **change had a significant impact on sales** in most regions.

Further Recommendations:

- **Feedback Collection**: Run surveys or feedback campaigns targeting customers who made purchases during the periods before and after the introduction of sustainable packaging. Understand their perception and any hesitations.
- **Educational Marketing**: Launch educational campaigns emphasizing the benefits of sustainable packaging. Highlight the company's commitment to the environment and how customers play a role in that by choosing Data Mart.
- **Incentives**: Offer incentives, such as discounts or loyalty points, to encourage customers to embrace the new packaging.
- **A/B Testing**: Consider testing different marketing messages or packaging designs in smaller markets or segments to determine what resonates most with customers.
- **Review Pricing**: Ensure that the cost passed onto consumers due to sustainable packaging is not significantly high. If it is, consider ways to optimize the supply chain or offer value propositions in other areas.

CONCLUSION

This case study, inspired by real changes in Australian retail where free plastic bags were phased out, shows how small decisions can make big waves in business. The change in bag policy wasn't just about the environment; it changed how people shopped and how businesses operated.

Using Microsoft SQL Server Management Studio, I dove into the data. With SQL's help, patterns started to emerge, and the story became clearer. After crunching the numbers, I turned to Excel to visualize the findings. Simple charts made complex data easy to understand, showing the ripple effect of the no-free-bags decision.

To wrap up, this exploration reaffirms the indispensable role of SQL in data analytics and the power of clear visualization. The revelations from this study accentuate the transformative potential of informed, data-backed decisions in sculpting the retail sector's future..