```sql
-- Querying original table
SELECT *
FROM weekly_sales
LIMIT 10;
```

week_date	region	platform	segment	customer_type	transactions	sales
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

```sql
-- A. DATA CLEANING STEPS

-- In a single query, perform the following operations and generate a new table in the
data_mart schema named clean_weekly_sales:

-- 1. Convert the week_date to a DATE format
-- 2. Add a week_number as the second column for each week_date value, for example any
value from the 1st of January to 7th of January will be 1, 8th to 14th will be 2 etc
-- 3. Add a month_number with the calendar month for each week_date value as the 3rd column
-- 4. Add a calendar_year column as the 4th column containing either 2018, 2019 or 2020
values
-- 5. Add a new column called age_band after the original segment column using the
following mapping on the number inside the segment value
            --  segment age_band
            -- 1        Young Adults
            -- 2        Middle Aged
            -- 3 or 4   Retirees
-- 6. Add a new demographic column using the following mapping for the first letter in the
segment values:
            --  segment demographic
            -- C         Couples
            -- F         Families
-- 7. Ensure all null string values with an "unknown" string value in the original segment
column as well as the new age_band and demographic columns
-- 8. Generate a new avg_transaction column as the sales value divided by transactions
rounded to 2 decimal places for each record

DROP TABLE IF EXISTS clean_weekly_sales;
CREATE TABLE clean_weekly_sales AS
SELECT TO_DATE(week_date, 'DD/MM/Y') AS week_date,
        DATE_PART('week', week_date :: DATE) AS week_number,
        EXTRACT(MONTH FROM week_date :: DATE) AS month_number,
        EXTRACT(YEAR FROM week_date :: DATE) AS calendar_year,
        region,
        platform,
        segment,
        CASE
            WHEN RIGHT(segment, 1) = '1' THEN 'Young Adults'
            WHEN RIGHT(segment, 1) = '2' THEN 'Middle Aged'
            WHEN RIGHT(segment, 1) = '3' OR RIGHT(segment, 1) = '4' THEN 'Retirees'
            ELSE 'unknown'
        END AS age_band,
```

```
        CASE
            WHEN LEFT(segment, 1) = 'F' THEN 'Families'
            WHEN LEFT(segment, 1) = 'C' THEN 'Couples'
            ELSE 'unknown'
        END AS demographic,
        customer_type,
        transactions,
        sales,
        ROUND(sales / transactions, 2) AS avg_transaction
FROM weekly_sales;

-- Querying newly created table
SELECT *
FROM clean_weekly_sales
LIMIT 10;
```

week_date	week_number	month_number	calendar_year	region	platform	segment	age_band	demographic	customer_type	transactions	sales	avg_transaction
2020-08-31	36.0	8	2020	ASIA	Retail	C3	Retirees	Couples	New	120631	3656163	30.00
2020-08-31	36.0	8	2020	ASIA	Retail	F1	Young Adults	Families	New	31574	996575	31.00
2020-08-31	36.0	8	2020	USA	Retail	null	unknown	unknown	Guest	529151	16509610	31.00
2020-08-31	36.0	8	2020	EUROPE	Retail	C1	Young Adults	Couples	New	4517	141942	31.00
2020-08-31	36.0	8	2020	AFRICA	Retail	C2	Middle Aged	Couples	New	58046	1758388	30.00
2020-08-31	36.0	8	2020	CANADA	Shopify	F2	Middle Aged	Families	Existing	1336	243878	182.00
2020-08-31	36.0	8	2020	AFRICA	Shopify	F3	Retirees	Families	Existing	2514	519502	206.00
2020-08-31	36.0	8	2020	ASIA	Shopify	F1	Young Adults	Families	Existing	2158	371417	172.00
2020-08-31	36.0	8	2020	AFRICA	Shopify	F2	Middle Aged	Families	New	318	49557	155.00
2020-08-31	36.0	8	2020	AFRICA	Retail	C3	Retirees	Couples	New	111032	3888162	35.00

```
-- 2. DATA EXPLORATION

-- 1. What day of the week is used for each week_date value?
SELECT DISTINCT week_date,
    CASE
        WHEN EXTRACT(DAY FROM week_date) % 7 = 1 THEN 'Monday'
        WHEN EXTRACT(DAY FROM week_date) % 7 = 2 THEN 'Tuesday'
        WHEN EXTRACT(DAY FROM week_date) % 7 = 3 THEN 'Wednesday'
        WHEN EXTRACT(DAY FROM week_date) % 7 = 4 THEN 'Thursday'
        WHEN EXTRACT(DAY FROM week_date) % 7 = 5 THEN 'Friday'
        WHEN EXTRACT(DAY FROM week_date) % 7 = 6 THEN 'Saturday'
        ELSE 'Sunday'
    END AS day_of_week
FROM clean_weekly_sales;
```

- Only part of the output is displayed.

week_date	day_of_week
2019-05-27	Saturday
2020-04-20	Saturday
2018-06-04	Thursday
2019-04-08	Monday
2019-06-10	Wednesday
2020-05-18	Thursday
2020-04-06	Saturday
2018-08-13	Saturday
2019-07-15	Monday
2018-05-21	Sunday
2018-06-25	Thursday
2019-07-01	Monday
2018-05-14	Sunday
2019-05-06	Saturday
2020-06-01	Monday
2020-08-10	Wednesday

```sql
-- 2. What range of week numbers are missing from the dataset?
WITH all_weeks AS (
        SELECT GENERATE_SERIES(1, 52, 1) AS week
    ),

    weeks_in_table AS (
        SELECT DISTINCT(week_number) AS week
        FROM clean_weekly_sales
    )

SELECT AW.week
FROM all_weeks AS AW
LEFT JOIN weeks_in_table AS WT
ON AW.week = WT.week
WHERE WT.week IS NULL;
```

3
4
5
6
7
8
9
10
11
12
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52

```sql
/* We can see that weeks ranging from 1-12 and 37 - 52 are not present in our
clean_weekly_sales table.*/

-- 3. How many total transactions were there for each year in the dataset?
SELECT calendar_year,
    SUM(transactions) AS total_transactions
FROM clean_weekly_sales
GROUP BY calendar_year
ORDER BY calendar_year ASC;
```

calendar_year	total_transactions
2018	346406460
2019	365639285
2020	375813651

```sql
-- 4. What is the total sales for each region for each month?
SELECT region,
    month_number,
    SUM(sales) AS total_sales
FROM clean_weekly_sales
GROUP BY region, month_number
ORDER BY region ASC, month_number ASC;
```

region	month_number	total_sales
AFRICA	3	567767480
AFRICA	4	1911783504
AFRICA	5	1647244738
AFRICA	6	1767559760
AFRICA	7	1960219710
AFRICA	8	1809596890
AFRICA	9	276320987
ASIA	3	529770793
ASIA	4	1804628707
ASIA	5	1526285399
ASIA	6	1619482889
ASIA	7	1768844756
ASIA	8	1663320609
ASIA	9	252836807
CANADA	3	144634329
CANADA	4	484552594
CANADA	5	412378365
CANADA	6	443846698
CANADA	7	477134947
CANADA	8	447073019
CANADA	9	69067959
EUROPE	3	35337093
EUROPE	4	127334255
EUROPE	5	109338389
EUROPE	6	122813826
EUROPE	7	136757466
EUROPE	8	122102995
EUROPE	9	18877433
OCEANIA	3	783282888
OCEANIA	4	2599767620
OCEANIA	5	2215657304
OCEANIA	6	2371884744
OCEANIA	7	2563459400
OCEANIA	8	2432313652
OCEANIA	9	372465518
SOUTH AMERICA	3	71023109
SOUTH AMERICA	4	238451531
SOUTH AMERICA	5	201391809
SOUTH AMERICA	6	218247455
SOUTH AMERICA	7	235582776
SOUTH AMERICA	8	221166052
SOUTH AMERICA	9	34175583
USA	3	225353043
USA	4	759786323
USA	5	655967121
USA	6	703878990
USA	7	760331754
USA	8	712002790
USA	9	110532368

```sql
-- 5. What is the total count of transactions for each platform?
SELECT platform,
    SUM(transactions) AS total_count_of_transactions
FROM clean_weekly_sales
GROUP BY platform;
```

platform	total_count_of_transactions
Shopify	5925169
Retail	1081934227

```sql
-- 6. What is the percentage of sales for Retail vs Shopify for each month?
WITH platform_monthly_sales AS (
        SELECT TO_CHAR(week_date, 'Month') AS month_name,
            month_number,
            platform,
            SUM(sales) AS total_sales,
            SUM(SUM(sales)) OVER(PARTITION BY month_number ORDER BY month_number) AS
monthly_sales
        FROM clean_weekly_sales
        GROUP BY month_name, month_number, platform
    )

SELECT month_name,
    platform,
    ROUND(CAST(total_sales AS NUMERIC) / monthly_sales * 100, 2) AS percent_sales
FROM platform_monthly_sales;
```

month_name	platform	percent_sales
March	Retail	97.54
March	Shopify	2.46
April	Retail	97.59
April	Shopify	2.41
May	Shopify	2.70
May	Retail	97.30
June	Shopify	2.73
June	Retail	97.27
July	Retail	97.29
July	Shopify	2.71
August	Retail	97.08
August	Shopify	2.92
September	Retail	97.38
September	Shopify	2.62

```sql
-- 7. What is the percentage of sales by demographic for each year in the dataset?
WITH yearly_demographic_sales AS (
        SELECT calendar_year,
            demographic,
            SUM(sales) AS total_sales,
            SUM(SUM(sales)) OVER(PARTITION BY calendar_year ORDER BY calendar_year) AS
yearly_sales
        FROM clean_weekly_sales
        GROUP BY calendar_year, demographic
    )

SELECT calendar_year,
    demographic,
    ROUND(CAST(total_sales AS NUMERIC) / yearly_sales * 100, 2) AS percentage_of_sales
FROM yearly_demographic_sales
ORDER BY calendar_year ASC;
```

calendar_year	demographic	percentage_of_sales
2018	Couples	26.38
2018	Families	31.99
2018	unknown	41.63
2019	Couples	27.28
2019	unknown	40.25
2019	Families	32.47
2020	Couples	28.72
2020	Families	32.73
2020	unknown	38.55

```sql
-- 8. Which age_band and demographic values contribute the most to Retail sales?
SELECT age_band,
    demographic,
    SUM(sales) AS total_sales
FROM clean_weekly_sales
WHERE age_band != 'unknown'
    AND demographic != 'unknown'
GROUP BY age_band, demographic
ORDER BY total_sales DESC
LIMIT 1;
```

age_band	demographic	total_sales
Retirees	Families	6750457132

```sql
-- 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?

/* No, we cannot use the avg_transaction column because it is already averaged.
If we use it, we will get a wrong answer. Instead we will use the sum of sales and divide
it by
sum of transactions for a grouped category.*/

WITH platform_transactions AS (
        SELECT calendar_year,
            platform,
            SUM(sales) AS total_sales,
            SUM(transactions) AS total_transactions
        FROM clean_weekly_sales
        GROUP BY calendar_year, platform
    )

SELECT calendar_year,
    platform,
    total_sales / total_transactions AS avg_transaction_size
FROM platform_transactions
ORDER BY calendar_year ASC, avg_transaction_size DESC;
```

calendar_year	platform	avg_transaction_size
2018	Shopify	192
2018	Retail	36
2019	Shopify	183
2019	Retail	36
2020	Shopify	179
2020	Retail	36

```sql
-- BEFORE & AFTER ANALYSIS

-- This technique is usually used when we inspect an important event and want to inspect the impact before and after a certain point in time.
-- Taking the week_date value of 2020-06-15 as the baseline week where the Data Mart sustainable packaging changes came into effect.
-- We would include all week_date values for 2020-06-15 as the start of the period after the change and the previous week_date values would be before

-- Setting '2020-05-16' as the baseline week
SELECT DISTINCT(week_number) AS baseline_week
FROM clean_weekly_sales
WHERE calendar_year = 2020
    AND week_date = '2020-06-15';
```

baseline_week
25.0

```sql
/* We can see that baseline week number comes out to be 25.
We are going to use this to find answers to questions.*/

-- Using this analysis approach - answer the following questions:

-- 1. What is the total sales for the 4 weeks before and after 2020-06-15? What is the growth or reduction rate in actual values and percentage of sales?
WITH before_sales AS (
        SELECT SUM(sales) AS sales
        FROM clean_weekly_sales
        WHERE calendar_year = 2020
            AND week_number BETWEEN (25 - 4) AND (25 - 1)
    ),

    after_sales AS (
        SELECT SUM(sales) AS sales
        FROM clean_weekly_sales
        WHERE calendar_year = 2020
            AND week_number BETWEEN (25) AND (25 + 3)
    )

SELECT before_sales.sales - after_sales.sales AS absolute_change,
```

```
    ROUND(CAST((after_sales.sales - before_sales.sales) AS NUMERIC) / before_sales.sales *
100, 2) AS percent_change
FROM before_sales, after_sales;
```

absolute_change	percent_change
26884188	-1.15

```
-- 2. What about the entire 12 weeks before and after?
WITH before_sales AS (
        SELECT SUM(sales) AS sales
        FROM clean_weekly_sales
        WHERE calendar_year = 2020
            AND week_number BETWEEN (25 - 12) AND (25 - 1)
    ),

    after_sales AS (
        SELECT SUM(sales) AS sales
        FROM clean_weekly_sales
        WHERE calendar_year = 2020
            AND week_number BETWEEN (25) AND (25 + 11)
    )

SELECT before_sales.sales - after_sales.sales AS absolute_change,
    ROUND(CAST((after_sales.sales - before_sales.sales) AS NUMERIC) / before_sales.sales *
100, 2) AS percent_change
FROM before_sales, after_sales;
```

absolute_change	percent_change
152325394	-2.14

```
-- 3. How does the sale metrics for these 2 periods before and after compare with the
previous years in 2018 and 2019?
DROP VIEW IF EXISTS all_years_change_sales;
CREATE VIEW all_years_change_sales AS
WITH change_all_years AS (

    -- 2020
    (WITH change_2020 AS (

        (WITH before_sales AS (
            SELECT SUM(sales) AS sales
            FROM clean_weekly_sales
            WHERE calendar_year = 2020
                AND week_number BETWEEN (25 - 4) AND (25 - 1)
        ),
```

```
        after_sales AS (
            SELECT SUM(sales) AS sales
            FROM clean_weekly_sales
            WHERE calendar_year = 2020
                AND week_number BETWEEN (25) AND (25 + 3)
        )

        SELECT 2020 AS year,
            4 AS time_period_weeks,
            after_sales.sales - before_sales.sales AS absolute_change,
            ROUND(CAST((after_sales.sales - before_sales.sales) AS NUMERIC) /
before_sales.sales * 100, 2) AS percent_change
        FROM before_sales, after_sales)

        UNION

        (WITH before_sales AS (
            SELECT SUM(sales) AS sales
            FROM clean_weekly_sales
            WHERE calendar_year = 2020
                AND week_number BETWEEN (25 - 12) AND (25 - 1)
        ),

        after_sales AS (
            SELECT SUM(sales) AS sales
            FROM clean_weekly_sales
            WHERE calendar_year = 2020
                AND week_number BETWEEN (25) AND (25 + 11)
        )

        SELECT 2020 AS year,
            12 AS time_period_weeks,
            after_sales.sales - before_sales.sales AS absolute_change,
            ROUND(CAST((after_sales.sales - before_sales.sales) AS NUMERIC) /
before_sales.sales * 100, 2) AS percent_change
        FROM before_sales, after_sales)
        )

    SELECT *
    FROM change_2020)

    UNION

    --2018
    (WITH change_2018 AS (

        (WITH before_sales AS (
            SELECT SUM(sales) AS sales
            FROM clean_weekly_sales
            WHERE calendar_year = 2018
                AND week_number BETWEEN (25 - 4) AND (25 - 1)
        ),

        after_sales AS (
            SELECT SUM(sales) AS sales
            FROM clean_weekly_sales
            WHERE calendar_year = 2018
```

```
                AND week_number BETWEEN (25) AND (25 + 3)
        )

        SELECT 2018 AS year,
            4 AS time_period_weeks,
            after_sales.sales - before_sales.sales AS absolute_change,
            ROUND(CAST((after_sales.sales - before_sales.sales) AS NUMERIC) /
before_sales.sales * 100, 2) AS percent_change
        FROM before_sales, after_sales)

        UNION

        (WITH before_sales AS (
            SELECT SUM(sales) AS sales
            FROM clean_weekly_sales
            WHERE calendar_year = 2018
                AND week_number BETWEEN (25 - 12) AND (25 - 1)
        ),

        after_sales AS (
            SELECT SUM(sales) AS sales
            FROM clean_weekly_sales
            WHERE calendar_year = 2018
                AND week_number BETWEEN (25) AND (25 + 11)
        )

        SELECT 2018 AS year,
            12 AS time_period_weeks,
            after_sales.sales - before_sales.sales AS absolute_change,
            ROUND(CAST((after_sales.sales - before_sales.sales) AS NUMERIC) /
before_sales.sales * 100, 2) AS percent_change
        FROM before_sales, after_sales)
        )

    SELECT *
    FROM change_2018)

    UNION

    --2019
    (WITH change_2019 AS (

        (WITH before_sales AS (
            SELECT SUM(sales) AS sales
            FROM clean_weekly_sales
            WHERE calendar_year = 2019
                AND week_number BETWEEN (25 - 4) AND (25 - 1)
        ),

        after_sales AS (
            SELECT SUM(sales) AS sales
            FROM clean_weekly_sales
            WHERE calendar_year = 2019
                AND week_number BETWEEN (25) AND (25 + 3)
        )

        SELECT 2019 AS year,
            4 AS time_period_weeks,
```

```
            after_sales.sales - before_sales.sales AS absolute_change,
            ROUND(CAST((after_sales.sales - before_sales.sales) AS NUMERIC) /
before_sales.sales * 100, 2) AS percent_change
        FROM before_sales, after_sales)

        UNION

        (WITH before_sales AS (
            SELECT SUM(sales) AS sales
            FROM clean_weekly_sales
            WHERE calendar_year = 2019
                AND week_number BETWEEN (25 - 12) AND (25 - 1)
        ),

        after_sales AS (
            SELECT SUM(sales) AS sales
            FROM clean_weekly_sales
            WHERE calendar_year = 2019
                AND week_number BETWEEN (25) AND (25 + 11)
        )

        SELECT 2019 AS year,
            12 AS time_period_weeks,
            after_sales.sales - before_sales.sales AS absolute_change,
            ROUND(CAST((after_sales.sales - before_sales.sales) AS NUMERIC) /
before_sales.sales * 100, 2) AS percent_change
        FROM before_sales, after_sales)
        )

    SELECT *
    FROM change_2019))

SELECT *
FROM change_all_years;

-- 4 weeks changes in sales in all years
SELECT *
FROM all_years_change_sales
WHERE time_period_weeks = 4
ORDER BY year ASC;
```

year	time_period_weeks	absolute_change	percent_change
2018	4	4102105	0.19
2019	4	2336594	0.10
2020	4	-26884188	-1.15

```
-- 12 weeks changes in sales in all years
SELECT *
FROM all_years_change_sales
WHERE time_period_weeks = 12
ORDER BY year ASC;
```

year	time_period_weeks	absolute_change	percent_change
2018	12	104256193	1.63
2019	12	-20740294	-0.30
2020	12	-152325394	-2.14

/* There's a definite reduction in sales in both the 4 weeks and 12 weeks time period.
Comparing to 2018 and 2019 also, we see a significant change.
This shows the change we see is not random. Packaging change did affect sales.*/