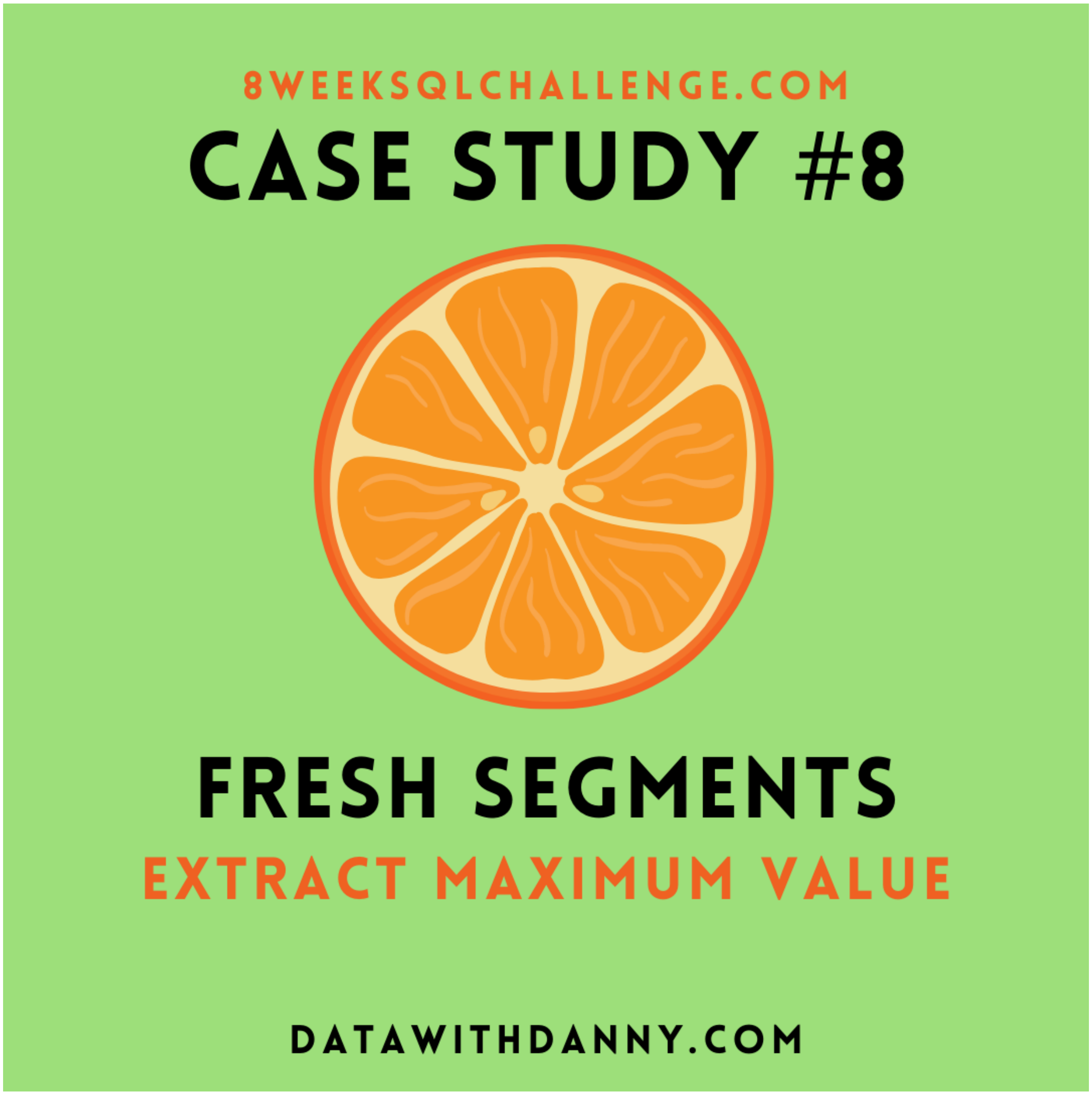
8WeekSQLCHALLENGE

DIGITAL FOOTPRINTS: UNVEILING CONSUMER ENGAGEMENT THROUGH INTEREST METRICS

Fatih Sahin



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

NOVEMBER 2023

CONTENTS

Introduction

This report delineates the analytical journey undertaken through the "Fresh Segments" case study, a component of the insightful '8WeekSQLChallenge' curated by Danny Ma. As part of this exploration, I assumed the role of a data analyst to unravel patterns in ad click behavior for a prominent client of Fresh Segments, a digital marketing agency. The challenge presented an opportunity to sift through complex datasets, extracting pivotal insights on customer interests and their digital footprints over various months.

The "Fresh Segments Challenge" steered me through a deep dive into the digital marketing analytics, parsing through extensive data that chronicled customer interactions and preferences. Keen learners looking to delve deeper into this case study can find more details at https://8weeksqlchallenge.com/case-study-8/ .

The analysis required moving beyond the typical confines of SQL queries to address real-world business inquiries, compelling me to draw upon my accumulated SQL expertise. Leveraging MS SQL Server Management Studio, I endeavored to distill vast amounts of data into coherent, actionable insights.

The dual purpose of this report is to highlight key findings from the Fresh Segments data and to showcase the transformative power of SQL in extracting business intelligence. It is my sincere hope that this report not only conveys valuable insights but also showcases the critical role that data analytics plays in shaping the strategies of digital marketing entities.

Problem Statement

Fresh Segments is in the business of making sense of how people react to online ads. They help companies understand which ads catch the eye of their customers by looking at which ones they click on the most. Before we could start finding patterns and insights in the data, we had to make sure the data was clean and organized:

- **Data Cleaning**: We made sure all the customer interest data was consistent and clear, setting the stage for reliable analysis.
- **Time Tracking**: We organized the data by months, so we could see how customer interests changed over time.
- **Sorting Interests**: We grouped the data to show which types of ads were getting attention over several months and which weren't.
- **Watching Trends**: We looked at which ads stayed popular over time and which ones didn't, to get a sense of what keeps customers interested.

With everything set up right, we faced a few key challenges:

- **Tracking Changes**: We needed to find out how customer interests changed from month to month and figure out why.
- **Identifying Favorites**: We wanted to see which types of ads were consistently popular to help companies know where to focus their efforts.
- **Understanding Lifespan**: We had to figure out which customer interests lasted and which ones were just temporary.
- **Getting to Know Customers**: By looking at the data, we aimed to paint a picture of what the company's customers are like and what they're into.
- **Explaining Fluctuations**: We explored why some ads got more or less popular over time, which could tell us if there's something bigger going on with the business or the market.

Fresh Segments wanted to give their clients a clear view of their customers' habits to help them make smarter decisions about their online ads.

Data Structure

The data architecture for the Fresh Segments case study consists of two main tables: **Interest Metrics** and **Interest Map**. These tables are designed to provide comprehensive insights into customer behavior and interest interactions over time.

Interest Metrics

- This table holds aggregated data about customer interactions with online advertisements.
- Primary Key: Composite of (**month_year**, **interest_id**)
- Columns:
  - **month**: Numeric month of the data record.
  - **year**: Numeric year of the data record.
  - **month_year**: Concatenation of _month and _year, representing the date of the record.
  - **interest_id**: A unique identifier for the type of customer interest.
  - **composition**: Represents the level of interaction with the ads related to the interest.
  - **index_value**: A reference value to calculate the average composition.
  - **ranking**: The rank of interest based on interaction levels.
  - **percentile_ranking**: Percentile rank comparing different interests.

Interest Map

- This table maps interest_id to a readable interest_name and provides additional summary information.
- Primary Key: id
- Columns:
  - **id**: A unique identifier for each interest, which correlates with interest_id from the Interest Metrics table.
  - **interest_name**: The descriptive name of the interest.
  - **interest_summary**: A brief description of the interest.

- **created_at**: The creation date of the interest record.
- **last_modified**: The last modified date of the interest record.

Data Flow:

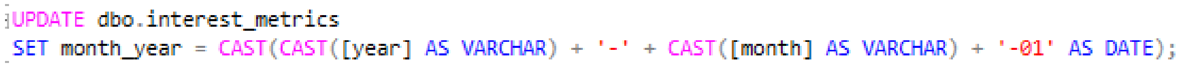
- Customer interaction data is collected and aggregated in the **Interest Metrics** table.
- The **Interest Map** table is used to provide human-readable names and descriptions for each unique **interest_id**.
- Analysts can join these two tables on **interest_id** to **id** to analyze trends, identify the popularity of interests over time, and derive insights for targeted marketing strategies.

Using this data structure, Fresh Segments can analyze the client's customer base and their engagement with various interests over time, helping to refine digital marketing campaigns for better reach and effectiveness.

Case Study Questions

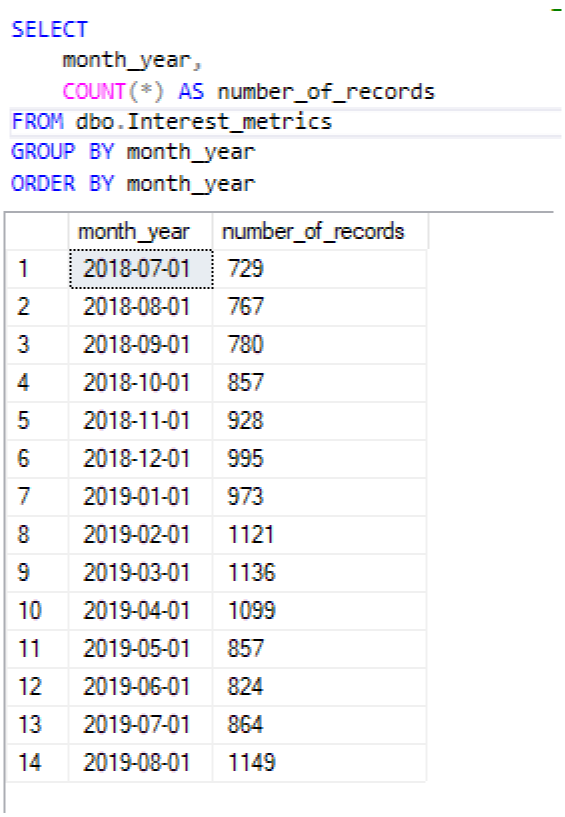
A.Data Exploration and Cleansing

1. **Update the fresh_segments.interest_metrics table by modifying the month_year column to be a date data type with the start of the month**



```
UPDATE dbo.interest_metrics
SET month_year = CAST(CAST([year] AS VARCHAR) + '-' + CAST([month] AS VARCHAR) + '-01' AS DATE);
```

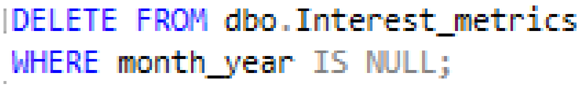
2. **What is count of records in the fresh_segments.interest_metrics for each month_year value sorted in chronological order (earliest to latest) with the null values appearing first?**



```
SELECT
    month_year,
    COUNT(*) AS number_of_records
FROM dbo.Interest_metrics
GROUP BY month_year
ORDER BY month_year
```

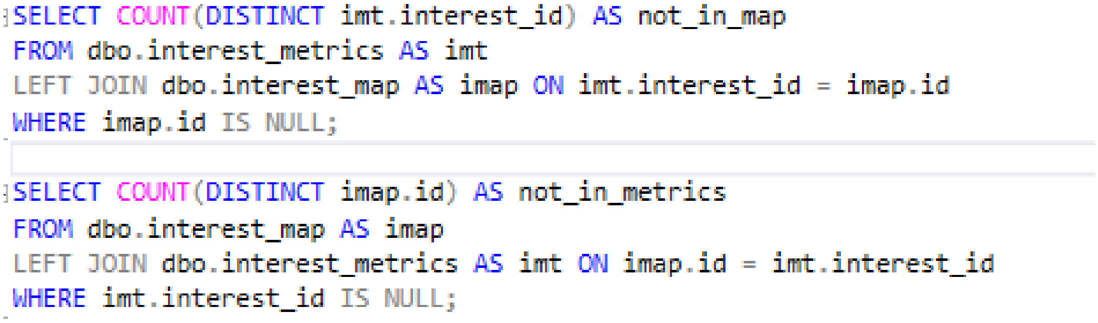
	month_year	number_of_records
1	2018-07-01	729
2	2018-08-01	767
3	2018-09-01	780
4	2018-10-01	857
5	2018-11-01	928
6	2018-12-01	995
7	2019-01-01	973
8	2019-02-01	1121
9	2019-03-01	1136
10	2019-04-01	1099
11	2019-05-01	857
12	2019-06-01	824
13	2019-07-01	864
14	2019-08-01	1149

3. **What do you think we should do with these null values in the fresh_segments.interest_metrics**



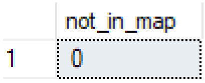
```
DELETE FROM dbo.Interest_metrics
WHERE month_year IS NULL;
```

4. **How many interest_id values exist in the fresh_segments.interest_metrics table but not in the fresh_segments.interest_map table? What about the other way around?**

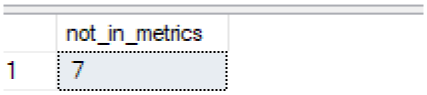


```
SELECT COUNT(DISTINCT imt.interest_id) AS not_in_map
FROM dbo.interest_metrics AS imt
LEFT JOIN dbo.interest_map AS imap ON imt.interest_id = imap.id
WHERE imap.id IS NULL;

SELECT COUNT(DISTINCT imap.id) AS not_in_metrics
FROM dbo.interest_map AS imap
LEFT JOIN dbo.interest_metrics AS imt ON imap.id = imt.interest_id
WHERE imt.interest_id IS NULL;
```

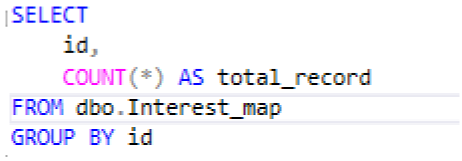


	not_in_map
1	0

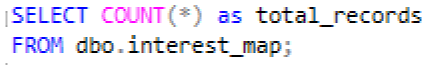


	not_in_metrics
1	7

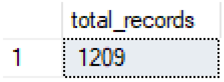
5. **Summarise the id values in the fresh_segments.interest_map by its total record count in this table.**



```
SELECT
    id,
    COUNT(*) AS total_record
FROM dbo.Interest_map
GROUP BY id
```



```
SELECT COUNT(*) as total_records
FROM dbo.interest_map;
```

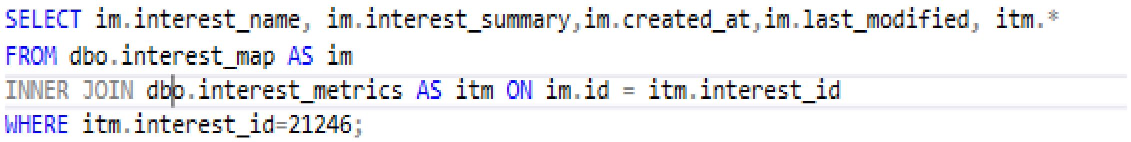


	total_records
1	1209

6. **What sort of table join should we perform for our analysis and why? Check your logic by checking the rows where interest_id = 21246 in your joined output and include all columns from fresh_segments.interest_metrics and all columns from fresh_segments.interest_map except from the id column.**

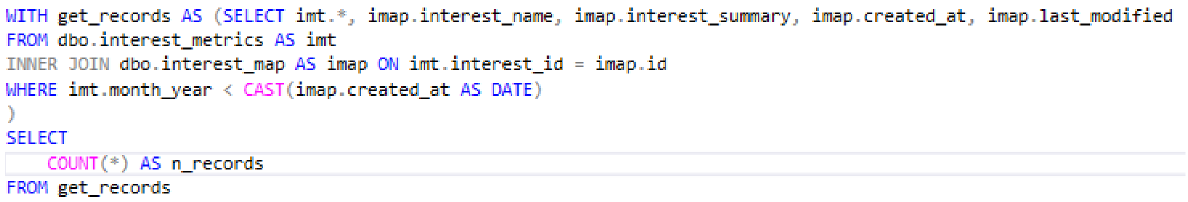
All values of interest_id from interest_metrics are also in interest_map.

- id's in interest_map are unique.
- An inner join would work in this scenario.

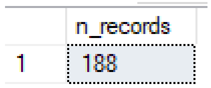


```
SELECT im.interest_name, im.interest_summary,im.created_at,im.last_modified, itm.*
FROM dbo.interest_map AS im
INNER JOIN dbo.interest_metrics AS itm ON im.id = itm.interest_id
WHERE itm.interest_id=21246;
```

7. **Are there any records in your joined table where the month_year value is before the created_at value from the fresh_segments.interest_map table?**



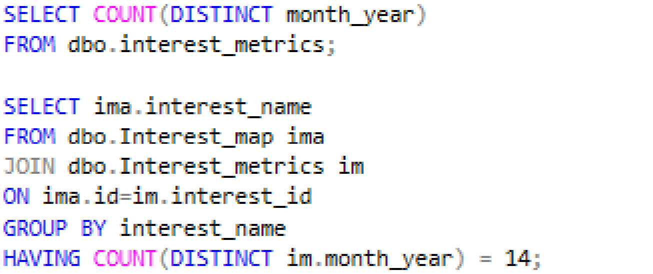
```
WITH get_records AS (SELECT imt.*, imap.interest_name, imap.interest_summary, imap.created_at, imap.last_modified
FROM dbo.interest_metrics AS imt
INNER JOIN dbo.interest_map AS imap ON imt.interest_id = imap.id
WHERE imt.month_year < CAST(imap.created_at AS DATE)
)
SELECT
    COUNT(*) AS n_records
FROM get_records
```



	n_records
1	188

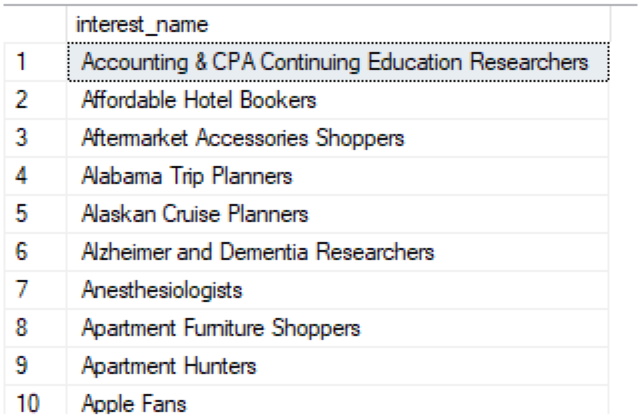
B. INTEREST ANALYSIS

1. Which interests have been present in all month_year dates in our dataset?



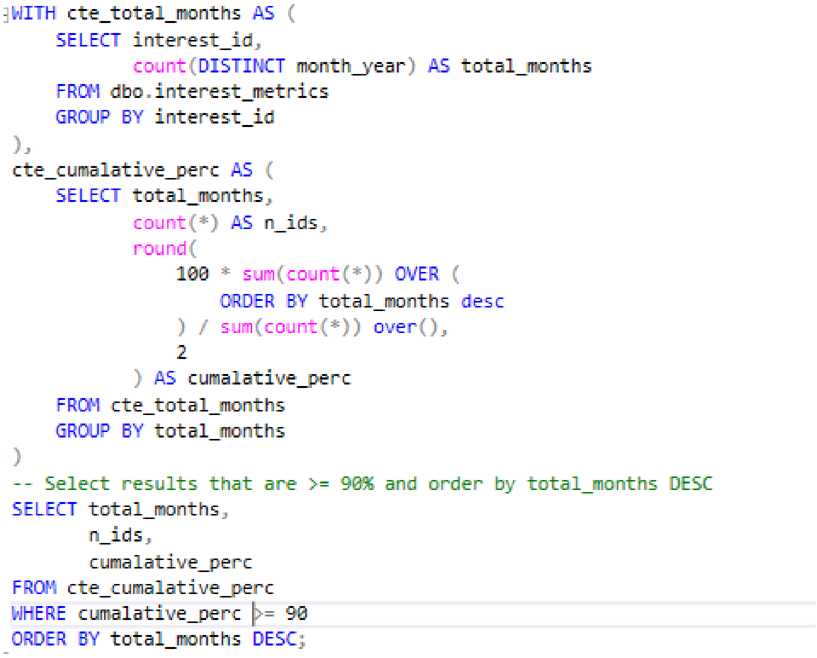
```
SELECT COUNT(DISTINCT month_year)
FROM dbo.interest_metrics;

SELECT ima.interest_name
FROM dbo.Interest_map ima
JOIN dbo.Interest_metrics im
ON ima.id=im.interest_id
GROUP BY interest_name
HAVING COUNT(DISTINCT im.month_year) = 14;
```

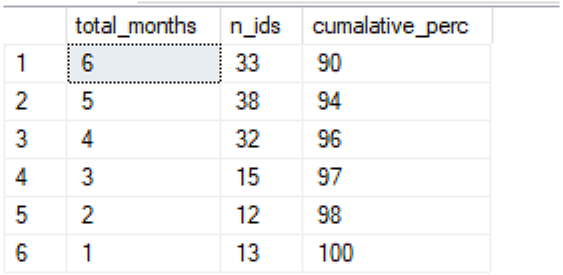


	interest_name
1	Accounting & CPA Continuing Education Researchers
2	Affordable Hotel Bookers
3	Aftermarket Accessories Shoppers
4	Alabama Trip Planners
5	Alaskan Cruise Planners
6	Alzheimer and Dementia Researchers
7	Anesthesiologists
8	Apartment Furniture Shoppers
9	Apartment Hunters
10	Apple Fans

2. Using this same total_months measure - calculate the cumulative percentage of all records starting at 14 months - which total_months value passes the 90% cumulative percentage value?

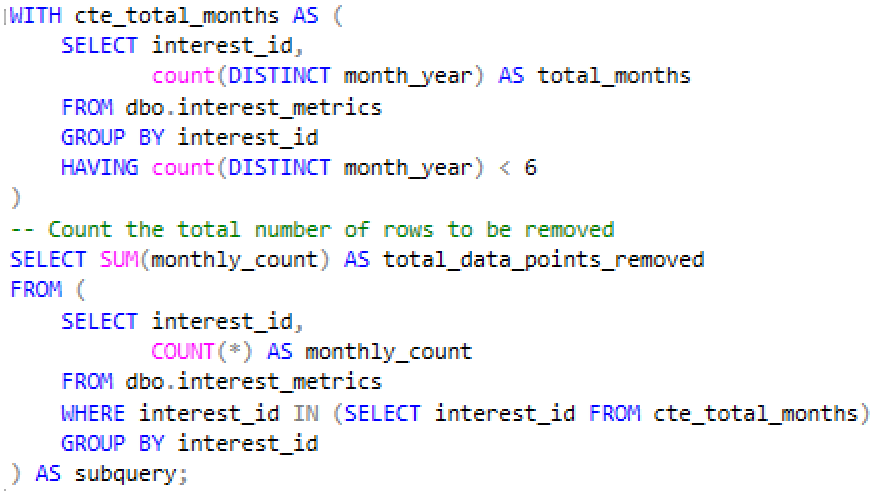


```
WITH cte_total_months AS (
     SELECT interest_id,
            count(DISTINCT month_year) AS total_months
     FROM dbo.interest_metrics
     GROUP BY interest_id
),
cte_cumalative_perc AS (
     SELECT total_months,
            count(*) AS n_ids,
            round(
                100 * sum(count(*)) OVER (
                    ORDER BY total_months desc
                ) / sum(count(*)) over(),
                2
            ) AS cumalative_perc
     FROM cte_total_months
     GROUP BY total_months
)
-- Select results that are >= 90% and order by total_months DESC
SELECT total_months,
       n_ids,
       cumalative_perc
FROM cte_cumalative_perc
WHERE cumalative_perc >= 90
ORDER BY total_months DESC;
```

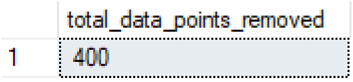


	total_months	n_ids	cumalative_perc
1	6	33	90
2	5	38	94
3	4	32	96
4	3	15	97
5	2	12	98
6	1	13	100

3. If we were to remove all interest_id values which are lower than the total_months value we found in the previous question - how many total data points would we be removing?



```
WITH cte_total_months AS (
    SELECT interest_id,
           count(DISTINCT month_year) AS total_months
    FROM dbo.interest_metrics
    GROUP BY interest_id
    HAVING count(DISTINCT month_year) < 6
)
-- Count the total number of rows to be removed
SELECT SUM(monthly_count) AS total_data_points_removed
FROM (
    SELECT interest_id,
           COUNT(*) AS monthly_count
    FROM dbo.interest_metrics
    WHERE interest_id IN (SELECT interest_id FROM cte_total_months)
    GROUP BY interest_id
) AS subquery;
```

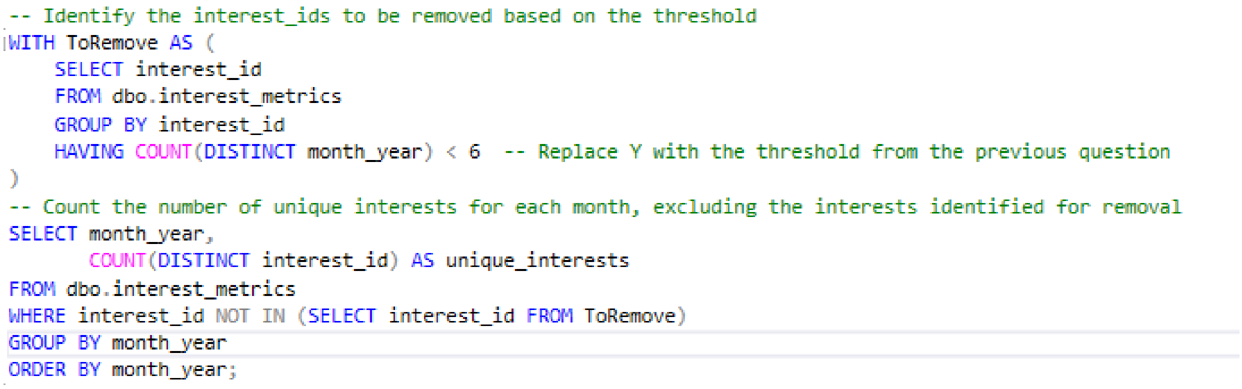


	total_data_points_removed
1	400

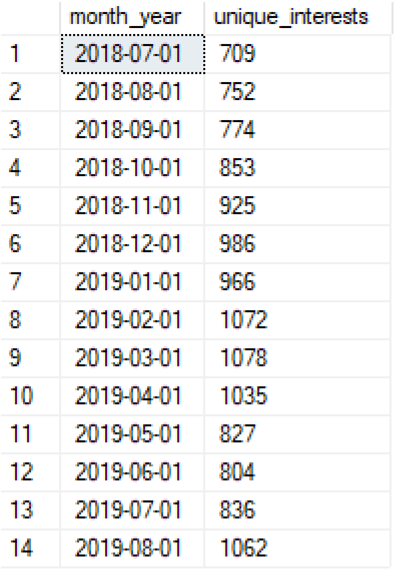
4. Does this decision make sense to remove these data points from a business perspective? Use an example where there are all 14 months present to a removed interest example for your arguments - think about what it means to have less months present from a segment perspective.

While removing infrequent interests can streamline the dataset and make analyses more manageable, it's essential to understand the business context. If the goal is to focus on the most significant segments, trimming might make sense. However, if the aim is to capture a comprehensive view of the market, including niche or emerging segments, it might be best to retain these data points.

5. After removing these interests - how many unique interests are there for each month?



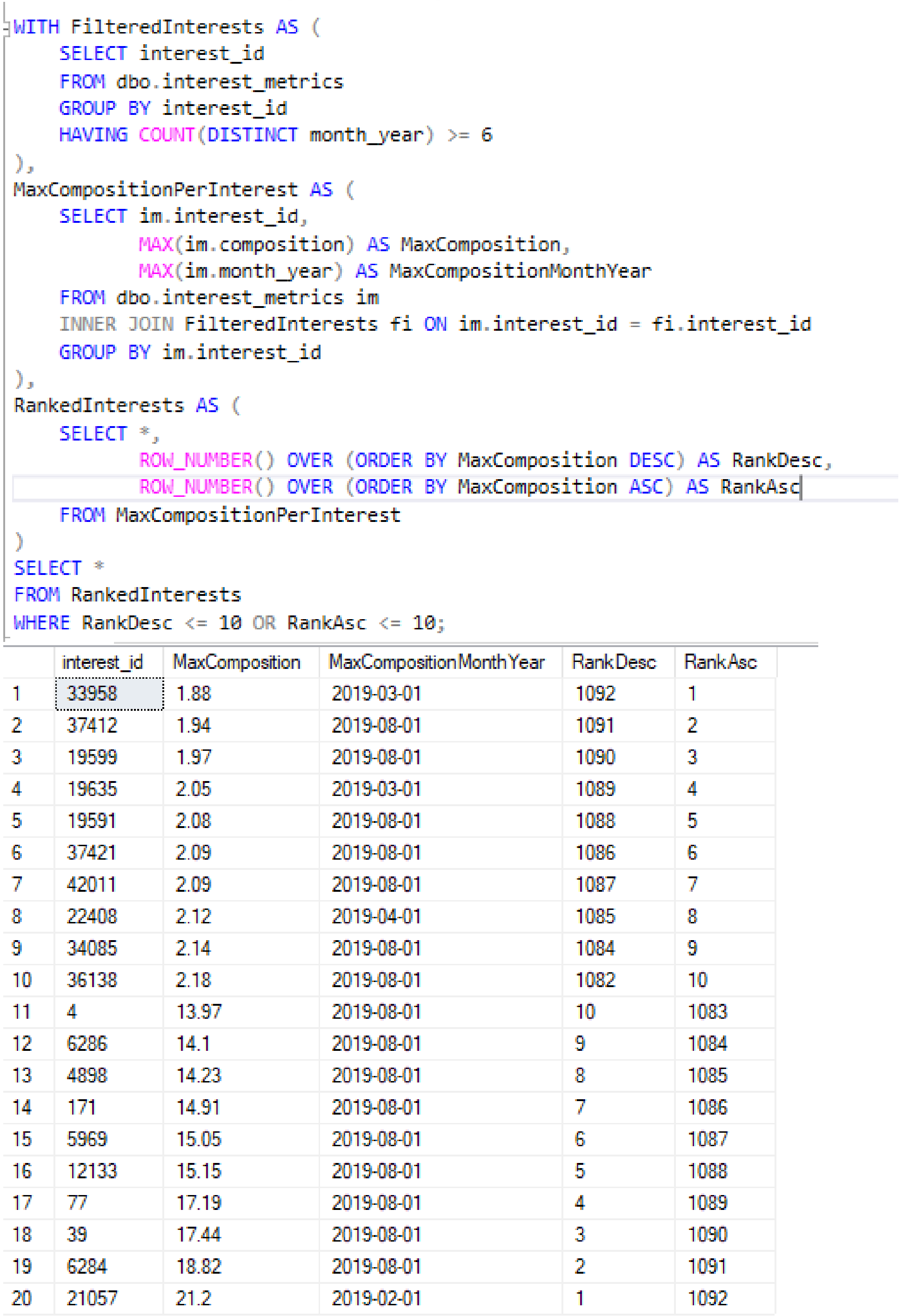
```
-- Identify the interest_ids to be removed based on the threshold
WITH ToRemove AS (
    SELECT interest_id
    FROM dbo.interest_metrics
    GROUP BY interest_id
    HAVING COUNT(DISTINCT month_year) < 6  -- Replace Y with the threshold from the previous question
)
-- Count the number of unique interests for each month, excluding the interests identified for removal
SELECT month_year,
       COUNT(DISTINCT interest_id) AS unique_interests
FROM dbo.interest_metrics
WHERE interest_id NOT IN (SELECT interest_id FROM ToRemove)
GROUP BY month_year
ORDER BY month_year;
```



	month_year	unique_interests
1	2018-07-01	709
2	2018-08-01	752
3	2018-09-01	774
4	2018-10-01	853
5	2018-11-01	925
6	2018-12-01	986
7	2019-01-01	966
8	2019-02-01	1072
9	2019-03-01	1078
10	2019-04-01	1035
11	2019-05-01	827
12	2019-06-01	804
13	2019-07-01	836
14	2019-08-01	1062

C.SEGMENT ANALYSIS

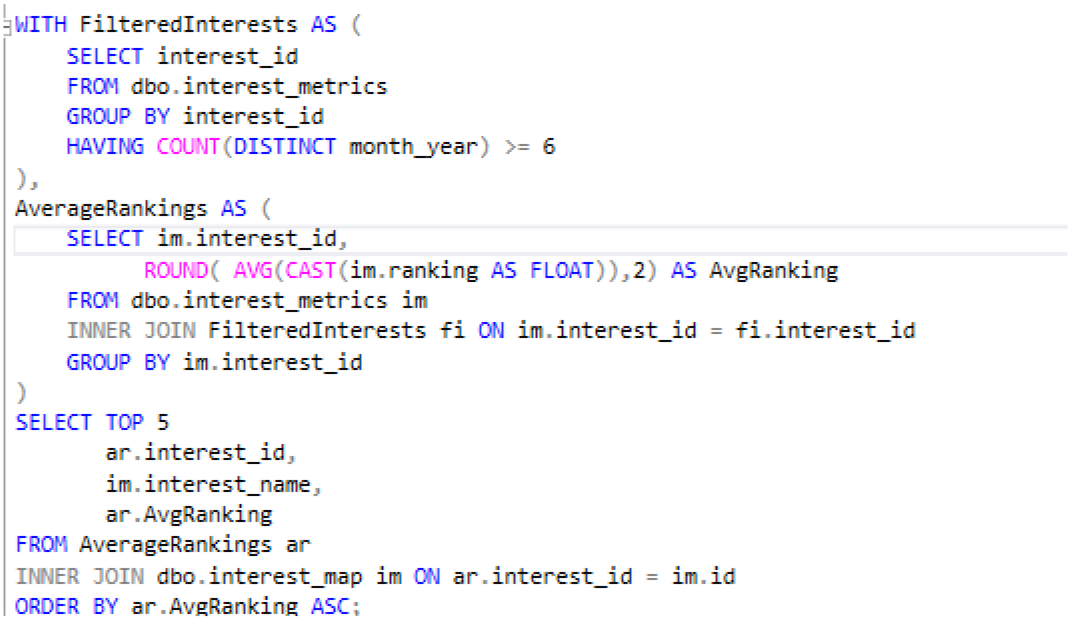
1. **Using our filtered dataset by removing the interests with less than 6 months worth of data,which are the top 10 and bottom 10 interests which have the largest composition values in any month_year? Only use the maximum composition value for each interest but you must keep the corresponding month_year**



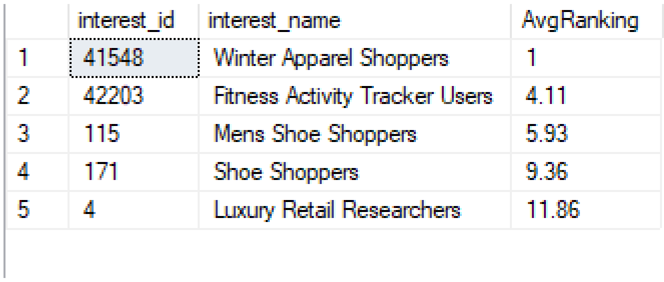
```
WITH FilteredInterests AS (
    SELECT interest_id
    FROM dbo.interest_metrics
    GROUP BY interest_id
    HAVING COUNT(DISTINCT month_year) >= 6
),
MaxCompositionPerInterest AS (
    SELECT im.interest_id,
           MAX(im.composition) AS MaxComposition,
           MAX(im.month_year) AS MaxCompositionMonthYear
    FROM dbo.interest_metrics im
    INNER JOIN FilteredInterests fi ON im.interest_id = fi.interest_id
    GROUP BY im.interest_id
),
RankedInterests AS (
    SELECT *,
           ROW_NUMBER() OVER (ORDER BY MaxComposition DESC) AS RankDesc,
           ROW_NUMBER() OVER (ORDER BY MaxComposition ASC) AS RankAsc
    FROM MaxCompositionPerInterest
)
SELECT *
FROM RankedInterests
WHERE RankDesc <= 10 OR RankAsc <= 10;
```

	interest_id	MaxComposition	MaxCompositionMonthYear	RankDesc	RankAsc
1	33958	1.88	2019-03-01	1092	1
2	37412	1.94	2019-08-01	1091	2
3	19599	1.97	2019-08-01	1090	3
4	19635	2.05	2019-03-01	1089	4
5	19591	2.08	2019-08-01	1088	5
6	37421	2.09	2019-08-01	1086	6
7	42011	2.09	2019-08-01	1087	7
8	22408	2.12	2019-04-01	1085	8
9	34085	2.14	2019-08-01	1084	9
10	36138	2.18	2019-08-01	1082	10
11	4	13.97	2019-08-01	10	1083
12	6286	14.1	2019-08-01	9	1084
13	4898	14.23	2019-08-01	8	1085
14	171	14.91	2019-08-01	7	1086
15	5969	15.05	2019-08-01	6	1087
16	12133	15.15	2019-08-01	5	1088
17	77	17.19	2019-08-01	4	1089
18	39	17.44	2019-08-01	3	1090
19	6284	18.82	2019-08-01	2	1091
20	21057	21.2	2019-02-01	1	1092

2. Which 5 interests had the lowest average ranking value?

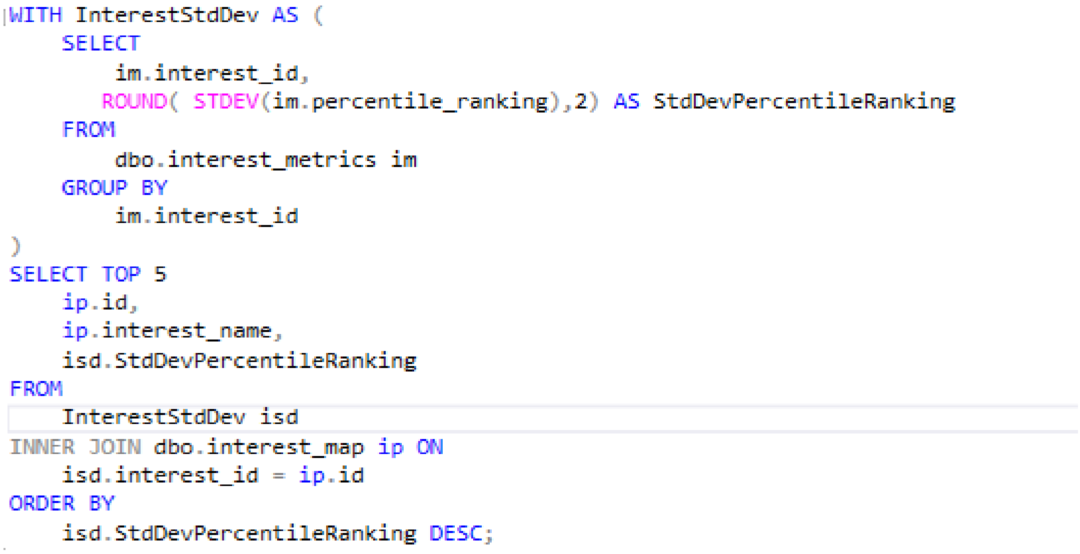


```
WITH FilteredInterests AS (
    SELECT interest_id
    FROM dbo.interest_metrics
    GROUP BY interest_id
    HAVING COUNT(DISTINCT month_year) >= 6
),
AverageRankings AS (
    SELECT im.interest_id,
           ROUND( AVG(CAST(im.ranking AS FLOAT)),2) AS AvgRanking
    FROM dbo.interest_metrics im
    INNER JOIN FilteredInterests fi ON im.interest_id = fi.interest_id
    GROUP BY im.interest_id
)
SELECT TOP 5
       ar.interest_id,
       im.interest_name,
       ar.AvgRanking
FROM AverageRankings ar
INNER JOIN dbo.interest_map im ON ar.interest_id = im.id
ORDER BY ar.AvgRanking ASC;
```

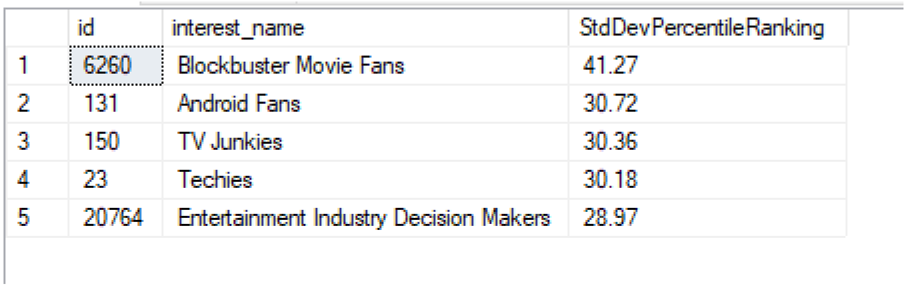


	interest_id	interest_name	AvgRanking
1	41548	Winter Apparel Shoppers	1
2	42203	Fitness Activity Tracker Users	4.11
3	115	Mens Shoe Shoppers	5.93
4	171	Shoe Shoppers	9.36
5	4	Luxury Retail Researchers	11.86

3. Which 5 interests had the largest standard deviation in their percentile_ranking value?

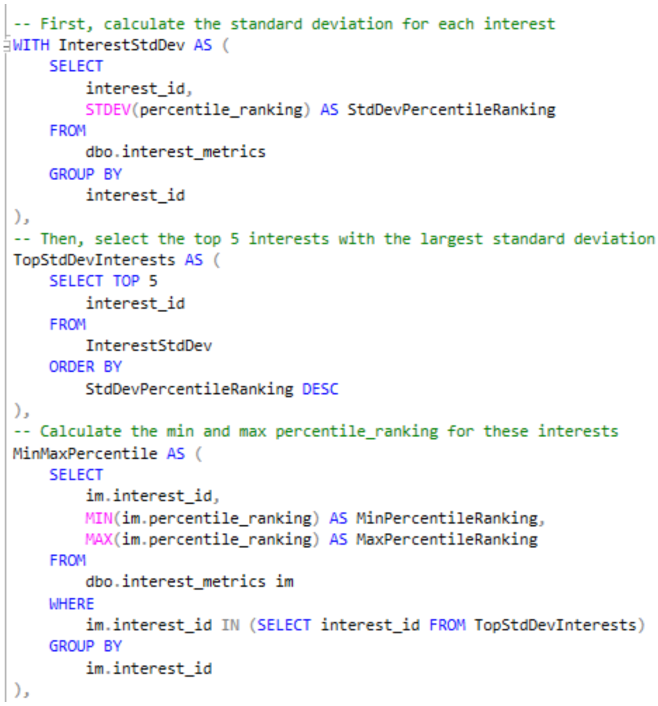


```
WITH InterestStdDev AS (
    SELECT
        im.interest_id,
       ROUND( STDEV(im.percentile_ranking),2) AS StdDevPercentileRanking
    FROM
        dbo.interest_metrics im
    GROUP BY
        im.interest_id
)
SELECT TOP 5
    ip.id,
    ip.interest_name,
    isd.StdDevPercentileRanking
FROM
    InterestStdDev isd
INNER JOIN dbo.interest_map ip ON
    isd.interest_id = ip.id
ORDER BY
    isd.StdDevPercentileRanking DESC;
```

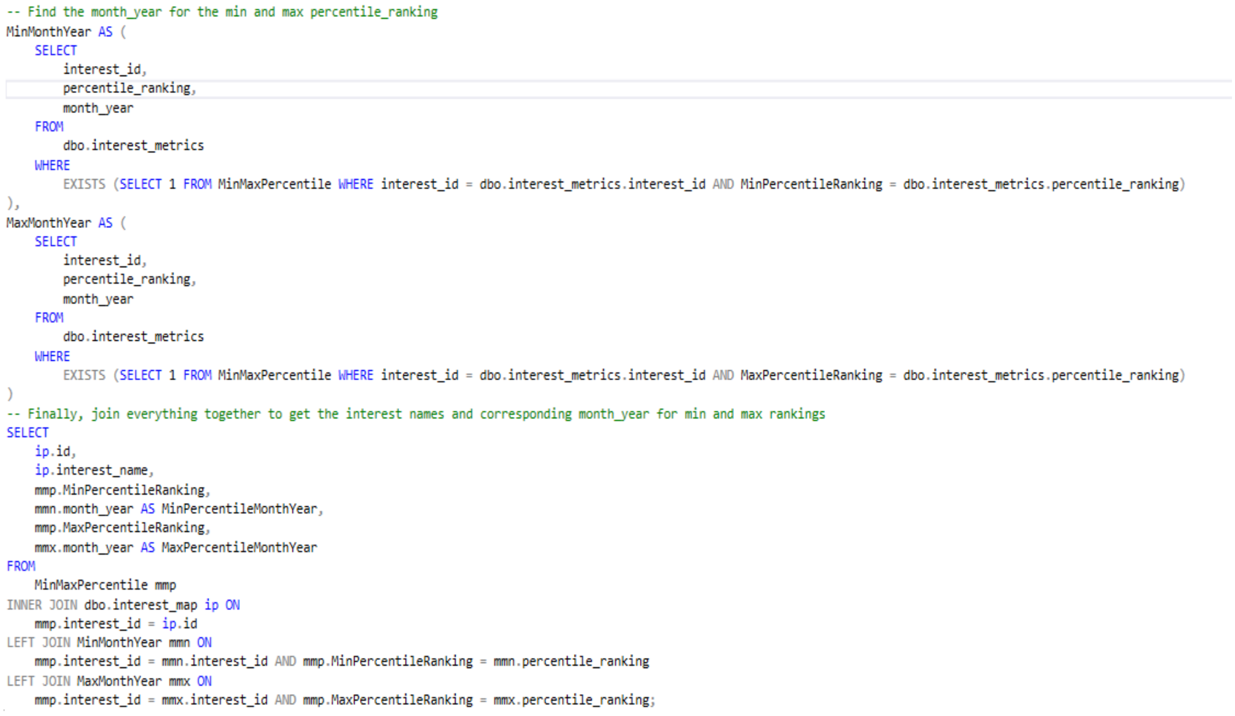


	id	interest_name	StdDevPercentileRanking
1	6260	Blockbuster Movie Fans	41.27
2	131	Android Fans	30.72
3	150	TV Junkies	30.36
4	23	Techies	30.18
5	20764	Entertainment Industry Decision Makers	28.97

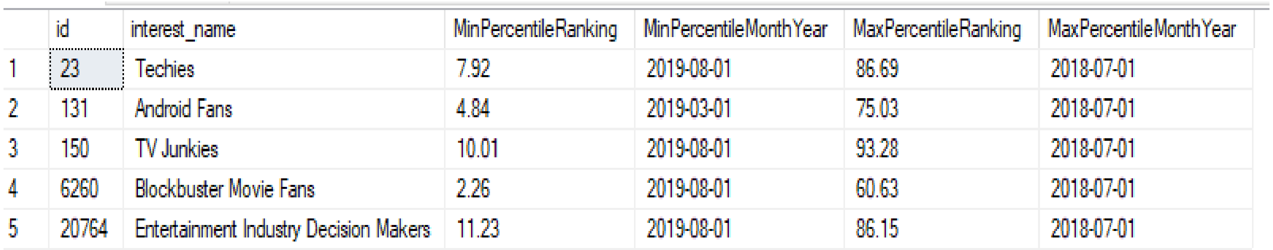
4. For the 5 interests found in the previous question, what was minimum and maximum percentile_ranking values for each interest and its corresponding year_month value? Can you describe what is happening for these 5 interests?



```
-- First, calculate the standard deviation for each interest
WITH InterestStdDev AS (
    SELECT
        interest_id,
        STDEV(percentile_ranking) AS StdDevPercentileRanking
    FROM
        dbo.interest_metrics
    GROUP BY
        interest_id
),
-- Then, select the top 5 interests with the largest standard deviation
TopStdDevInterests AS (
    SELECT TOP 5
        interest_id
    FROM
        InterestStdDev
    ORDER BY
        StdDevPercentileRanking DESC
),
-- Calculate the min and max percentile_ranking for these interests
MinMaxPercentile AS (
    SELECT
        im.interest_id,
        MIN(im.percentile_ranking) AS MinPercentileRanking,
        MAX(im.percentile_ranking) AS MaxPercentileRanking
    FROM
        dbo.interest_metrics im
    WHERE
        im.interest_id IN (SELECT interest_id FROM TopStdDevInterests)
    GROUP BY
        im.interest_id
),
```



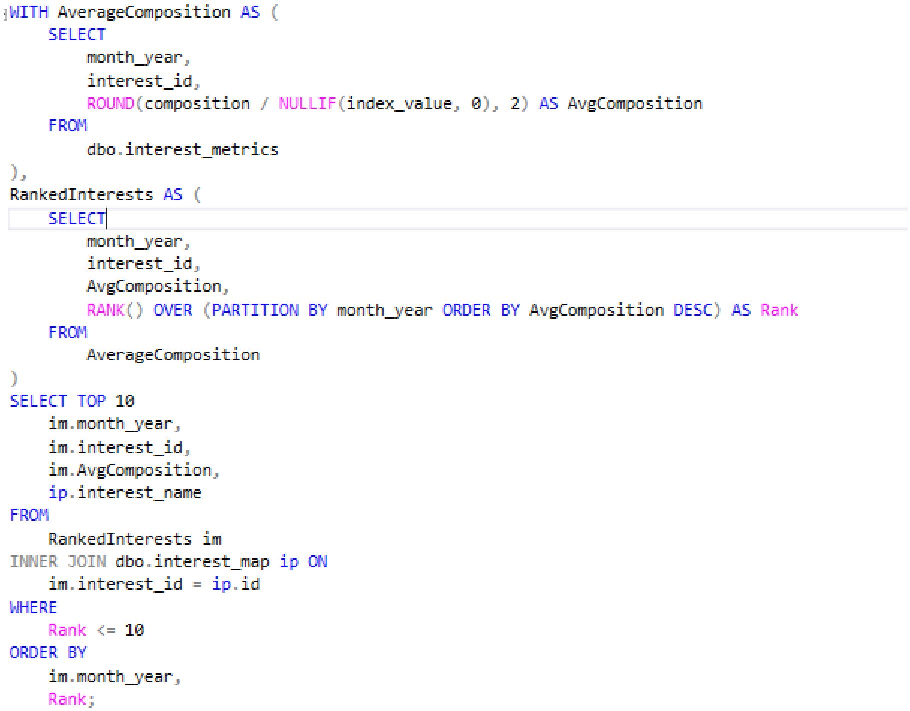
```
-- Find the month_year for the min and max percentile_ranking
MinMonthYear AS (
    SELECT
        interest_id,
        percentile_ranking,
        month_year
    FROM
        dbo.interest_metrics
    WHERE
        EXISTS (SELECT 1 FROM MinMaxPercentile WHERE interest_id = dbo.interest_metrics.interest_id AND MinPercentileRanking = dbo.interest_metrics.percentile_ranking)
),
MaxMonthYear AS (
    SELECT
        interest_id,
        percentile_ranking,
        month_year
    FROM
        dbo.interest_metrics
    WHERE
        EXISTS (SELECT 1 FROM MinMaxPercentile WHERE interest_id = dbo.interest_metrics.interest_id AND MaxPercentileRanking = dbo.interest_metrics.percentile_ranking)
)
-- Finally, join everything together to get the interest names and corresponding month_year for min and max rankings
SELECT
    ip.id,
    ip.interest_name,
    mmp.MinPercentileRanking,
    mmn.month_year AS MinPercentileMonthYear,
    mmp.MaxPercentileRanking,
    mmx.month_year AS MaxPercentileMonthYear
FROM
    MinMaxPercentile mmp
INNER JOIN dbo.interest_map ip ON
    mmp.interest_id = ip.id
LEFT JOIN MinMonthYear mmn ON
    mmp.interest_id = mmn.interest_id AND mmp.MinPercentileRanking = mmn.percentile_ranking
LEFT JOIN MaxMonthYear mmx ON
    mmp.interest_id = mmx.interest_id AND mmp.MaxPercentileRanking = mmx.percentile_ranking;
```



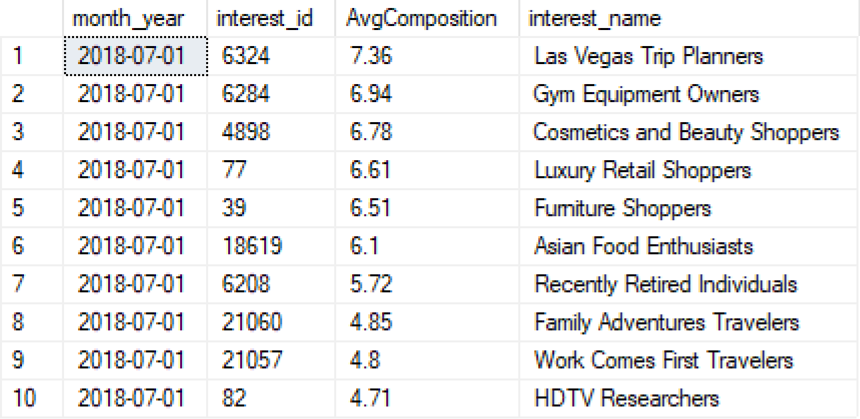
	id	interest_name	MinPercentileRanking	MinPercentileMonthYear	MaxPercentileRanking	MaxPercentileMonthYear
1	23	Techies	7.92	2019-08-01	86.69	2018-07-01
2	131	Android Fans	4.84	2019-03-01	75.03	2018-07-01
3	150	TV Junkies	10.01	2019-08-01	93.28	2018-07-01
4	6260	Blockbuster Movie Fans	2.26	2019-08-01	60.63	2018-07-01
5	20764	Entertainment Industry Decision Makers	11.23	2019-08-01	86.15	2018-07-01

D.INDEX ANALYSIS

1. What is the top 10 interests by the average composition for each month?

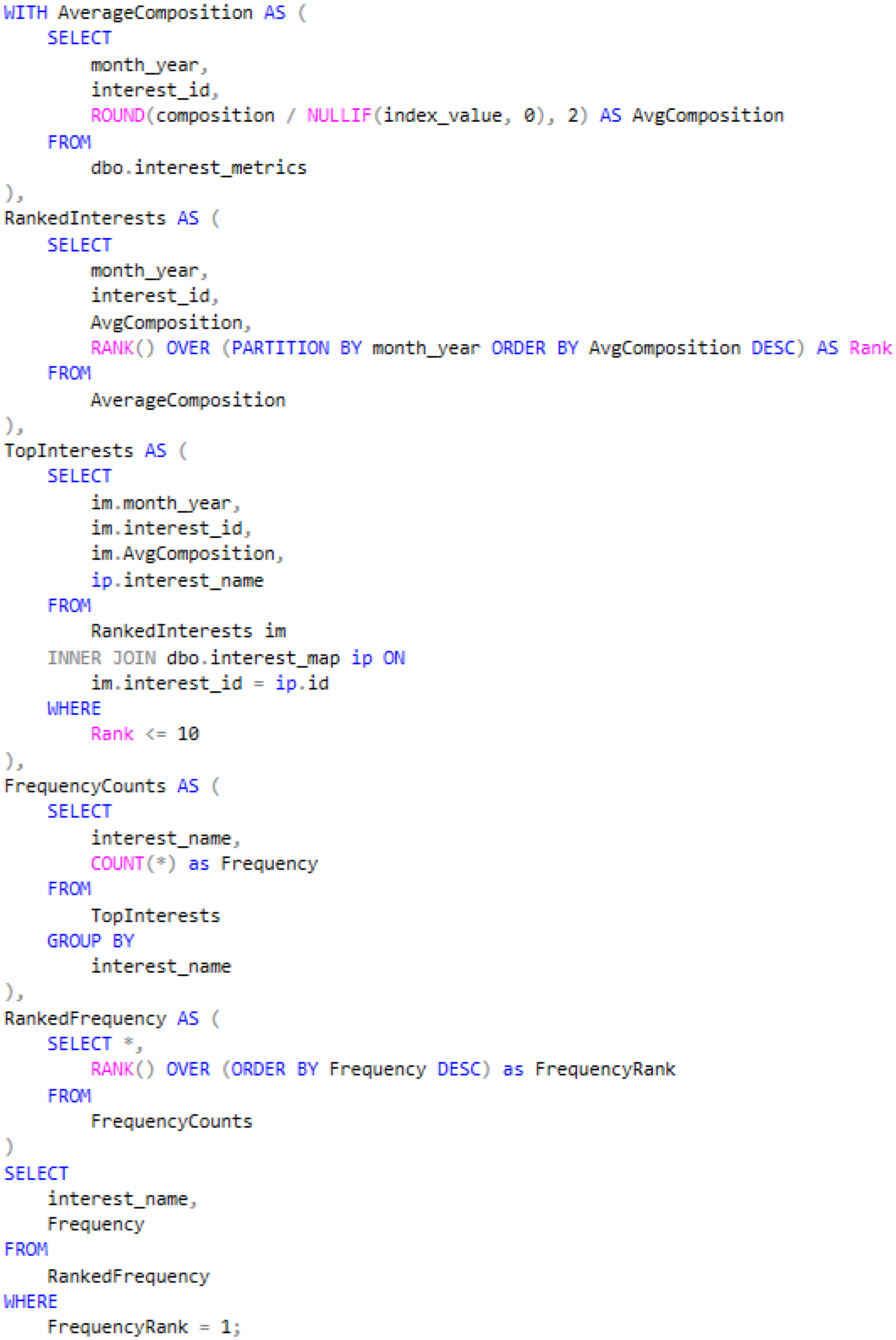


```
WITH AverageComposition AS (
    SELECT
        month_year,
        interest_id,
        ROUND(composition / NULLIF(index_value, 0), 2) AS AvgComposition
    FROM
        dbo.interest_metrics
),
RankedInterests AS (
    SELECT
        month_year,
        interest_id,
        AvgComposition,
        RANK() OVER (PARTITION BY month_year ORDER BY AvgComposition DESC) AS Rank
    FROM
        AverageComposition
)
SELECT TOP 10
    im.month_year,
    im.interest_id,
    im.AvgComposition,
    ip.interest_name
FROM
    RankedInterests im
INNER JOIN dbo.interest_map ip ON
    im.interest_id = ip.id
WHERE
    Rank <= 10
ORDER BY
    im.month_year,
    Rank;
```

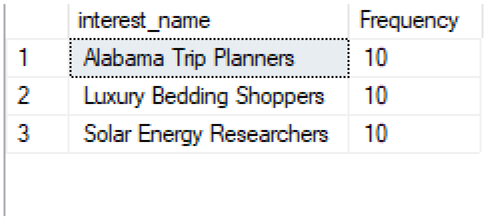


	month_year	interest_id	AvgComposition	interest_name
1	2018-07-01	6324	7.36	Las Vegas Trip Planners
2	2018-07-01	6284	6.94	Gym Equipment Owners
3	2018-07-01	4898	6.78	Cosmetics and Beauty Shoppers
4	2018-07-01	77	6.61	Luxury Retail Shoppers
5	2018-07-01	39	6.51	Furniture Shoppers
6	2018-07-01	18619	6.1	Asian Food Enthusiasts
7	2018-07-01	6208	5.72	Recently Retired Individuals
8	2018-07-01	21060	4.85	Family Adventures Travelers
9	2018-07-01	21057	4.8	Work Comes First Travelers
10	2018-07-01	82	4.71	HDTV Researchers

2. For all of these top 10 interests - which interest appears the most often?

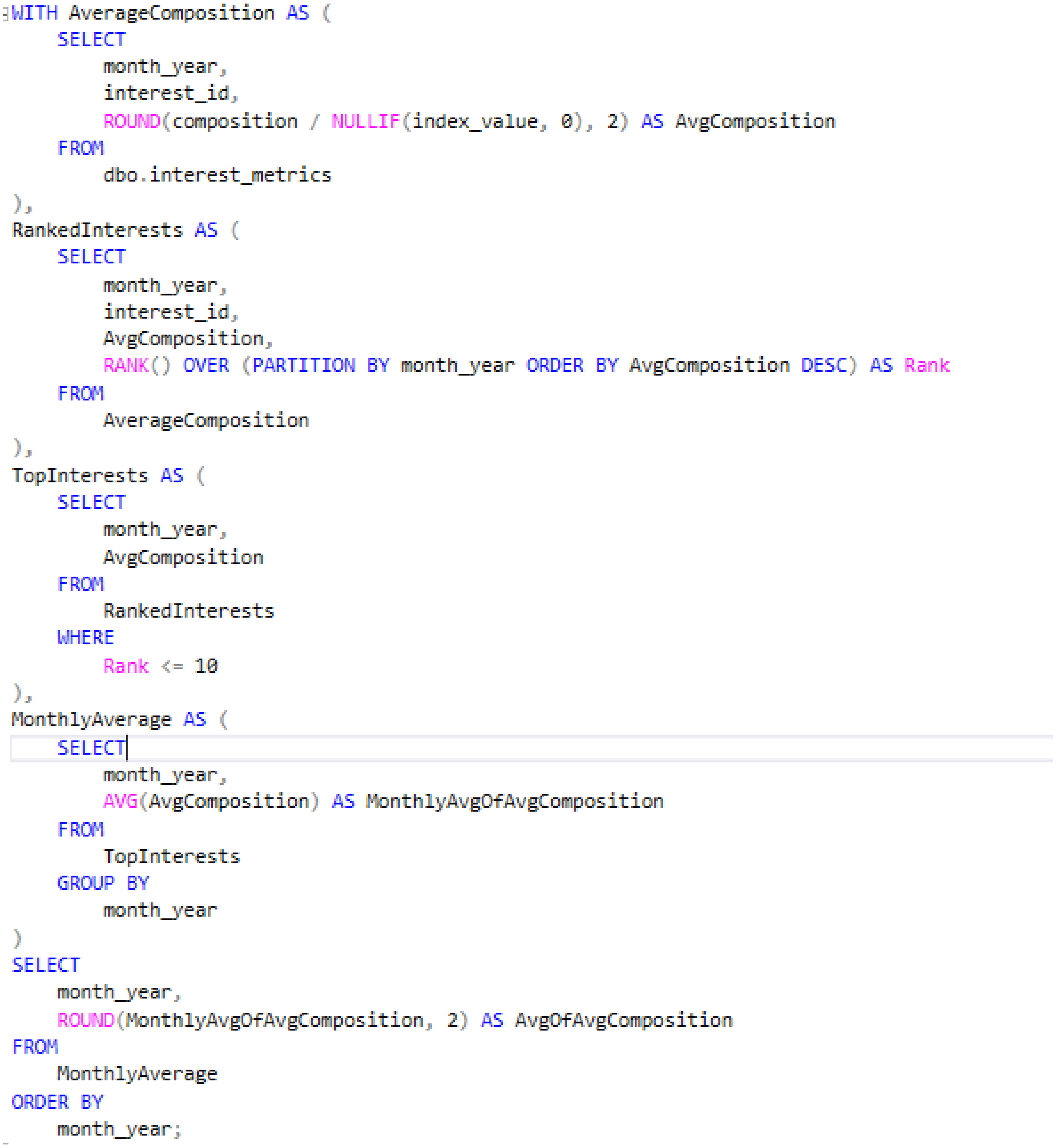


```sql
WITH AverageComposition AS (
    SELECT
        month_year,
        interest_id,
        ROUND(composition / NULLIF(index_value, 0), 2) AS AvgComposition
    FROM
        dbo.interest_metrics
),
RankedInterests AS (
    SELECT
        month_year,
        interest_id,
        AvgComposition,
        RANK() OVER (PARTITION BY month_year ORDER BY AvgComposition DESC) AS Rank
    FROM
        AverageComposition
),
TopInterests AS (
    SELECT
        im.month_year,
        im.interest_id,
        im.AvgComposition,
        ip.interest_name
    FROM
        RankedInterests im
    INNER JOIN dbo.interest_map ip ON
        im.interest_id = ip.id
    WHERE
        Rank <= 10
),
FrequencyCounts AS (
    SELECT
        interest_name,
        COUNT(*) as Frequency
    FROM
        TopInterests
    GROUP BY
        interest_name
),
RankedFrequency AS (
    SELECT *,
        RANK() OVER (ORDER BY Frequency DESC) as FrequencyRank
    FROM
        FrequencyCounts
)
SELECT
    interest_name,
    Frequency
FROM
    RankedFrequency
WHERE
    FrequencyRank = 1;
```

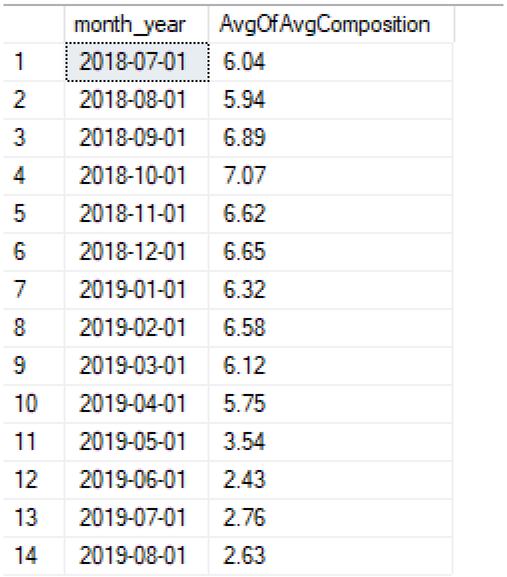


	interest_name	Frequency
1	Alabama Trip Planners	10
2	Luxury Bedding Shoppers	10
3	Solar Energy Researchers	10

3. What is the average of the average composition for the top 10 interests for each month?

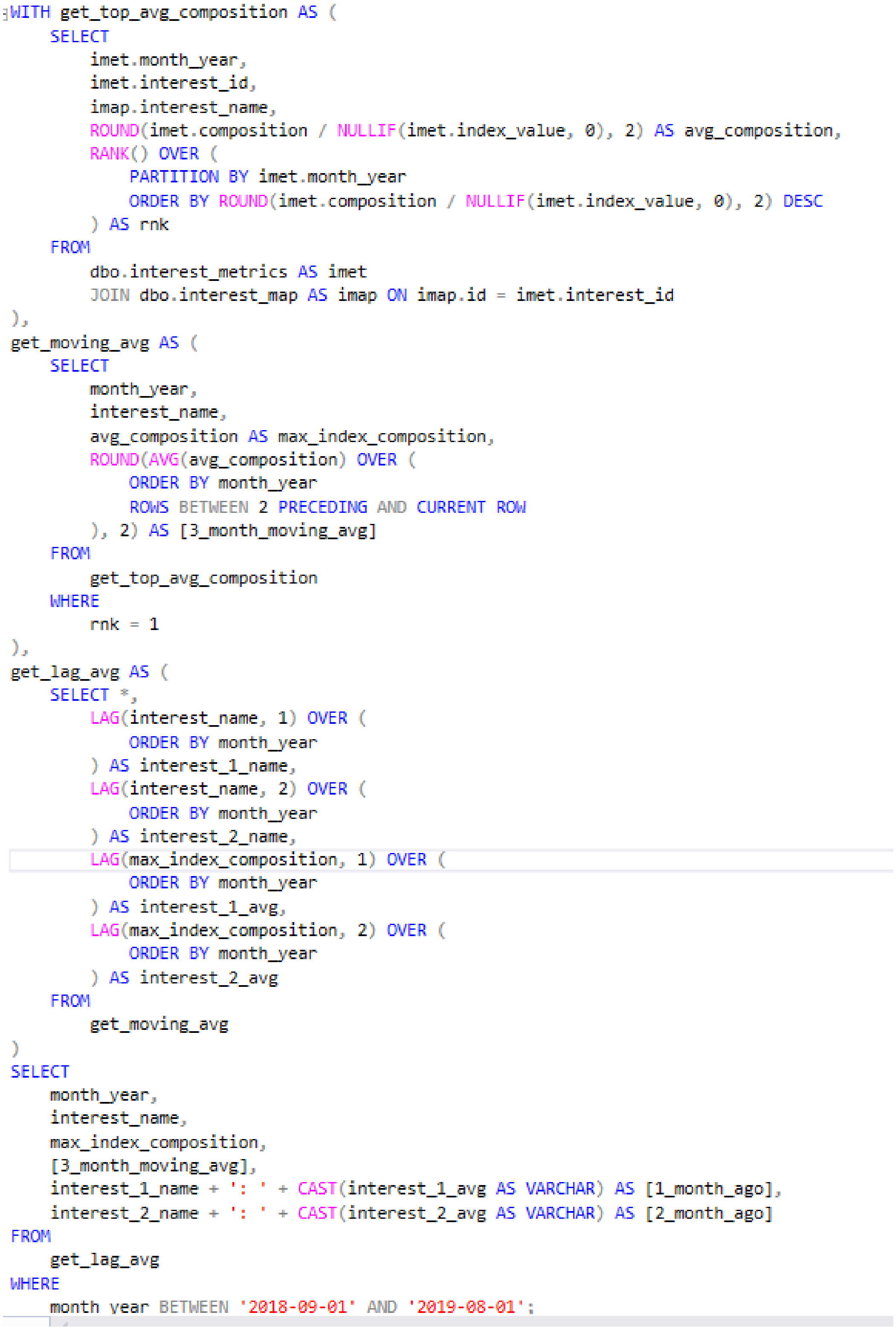


```sql
WITH AverageComposition AS (
    SELECT
        month_year,
        interest_id,
        ROUND(composition / NULLIF(index_value, 0), 2) AS AvgComposition
    FROM
        dbo.interest_metrics
),
RankedInterests AS (
    SELECT
        month_year,
        interest_id,
        AvgComposition,
        RANK() OVER (PARTITION BY month_year ORDER BY AvgComposition DESC) AS Rank
    FROM
        AverageComposition
),
TopInterests AS (
    SELECT
        month_year,
        AvgComposition
    FROM
        RankedInterests
    WHERE
        Rank <= 10
),
MonthlyAverage AS (
    SELECT
        month_year,
        AVG(AvgComposition) AS MonthlyAvgOfAvgComposition
    FROM
        TopInterests
    GROUP BY
        month_year
)
SELECT
    month_year,
    ROUND(MonthlyAvgOfAvgComposition, 2) AS AvgOfAvgComposition
FROM
    MonthlyAverage
ORDER BY
    month_year;
```

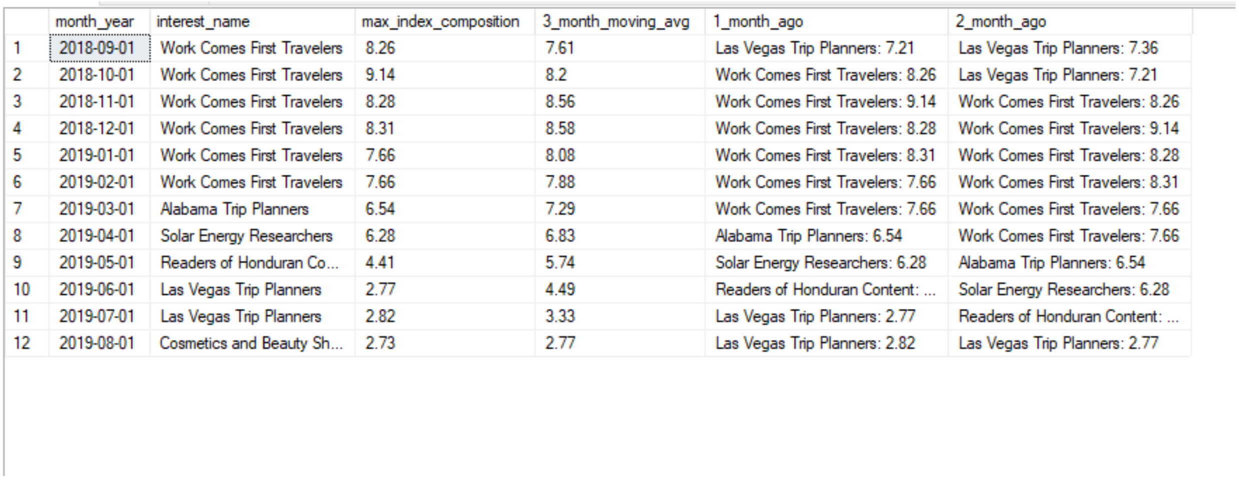


	month_year	AvgOfAvgComposition
1	2018-07-01	6.04
2	2018-08-01	5.94
3	2018-09-01	6.89
4	2018-10-01	7.07
5	2018-11-01	6.62
6	2018-12-01	6.65
7	2019-01-01	6.32
8	2019-02-01	6.58
9	2019-03-01	6.12
10	2019-04-01	5.75
11	2019-05-01	3.54
12	2019-06-01	2.43
13	2019-07-01	2.76
14	2019-08-01	2.63

4. What is the 3 month rolling average of the max average composition value from September 2018 to August 2019 and include the previous top ranking interests in the same output shown below.



```
WITH get_top_avg_composition AS (
    SELECT
        imet.month_year,
        imet.interest_id,
        imap.interest_name,
        ROUND(imet.composition / NULLIF(imet.index_value, 0), 2) AS avg_composition,
        RANK() OVER (
            PARTITION BY imet.month_year
            ORDER BY ROUND(imet.composition / NULLIF(imet.index_value, 0), 2) DESC
        ) AS rnk
    FROM
        dbo.interest_metrics AS imet
        JOIN dbo.interest_map AS imap ON imap.id = imet.interest_id
),
get_moving_avg AS (
    SELECT
        month_year,
        interest_name,
        avg_composition AS max_index_composition,
        ROUND(AVG(avg_composition) OVER (
            ORDER BY month_year
            ROWS BETWEEN 2 PRECEDING AND CURRENT ROW
        ), 2) AS [3_month_moving_avg]
    FROM
        get_top_avg_composition
    WHERE
        rnk = 1
),
get_lag_avg AS (
    SELECT *,
        LAG(interest_name, 1) OVER (
            ORDER BY month_year
        ) AS interest_1_name,
        LAG(interest_name, 2) OVER (
            ORDER BY month_year
        ) AS interest_2_name,
        LAG(max_index_composition, 1) OVER (
            ORDER BY month_year
        ) AS interest_1_avg,
        LAG(max_index_composition, 2) OVER (
            ORDER BY month_year
        ) AS interest_2_avg
    FROM
        get_moving_avg
)
SELECT
    month_year,
    interest_name,
    max_index_composition,
    [3_month_moving_avg],
    interest_1_name + ': ' + CAST(interest_1_avg AS VARCHAR) AS [1_month_ago],
    interest_2_name + ': ' + CAST(interest_2_avg AS VARCHAR) AS [2_month_ago]
FROM
    get_lag_avg
WHERE
    month_year BETWEEN '2018-09-01' AND '2019-08-01';
```



	month_year	interest_name	max_index_composition	3_month_moving_avg	1_month_ago	2_month_ago
1	2018-09-01	Work Comes First Travelers	8.26	7.61	Las Vegas Trip Planners: 7.21	Las Vegas Trip Planners: 7.36
2	2018-10-01	Work Comes First Travelers	9.14	8.2	Work Comes First Travelers: 8.26	Las Vegas Trip Planners: 7.21
3	2018-11-01	Work Comes First Travelers	8.28	8.56	Work Comes First Travelers: 9.14	Work Comes First Travelers: 8.26
4	2018-12-01	Work Comes First Travelers	8.31	8.58	Work Comes First Travelers: 8.28	Work Comes First Travelers: 9.14
5	2019-01-01	Work Comes First Travelers	7.66	8.08	Work Comes First Travelers: 8.31	Work Comes First Travelers: 8.28
6	2019-02-01	Work Comes First Travelers	7.66	7.88	Work Comes First Travelers: 7.66	Work Comes First Travelers: 8.31
7	2019-03-01	Alabama Trip Planners	6.54	7.29	Work Comes First Travelers: 7.66	Work Comes First Travelers: 7.66
8	2019-04-01	Solar Energy Researchers	6.28	6.83	Alabama Trip Planners: 6.54	Work Comes First Travelers: 7.66
9	2019-05-01	Readers of Honduran Co...	4.41	5.74	Solar Energy Researchers: 6.28	Alabama Trip Planners: 6.54
10	2019-06-01	Las Vegas Trip Planners	2.77	4.49	Readers of Honduran Content: ...	Solar Energy Researchers: 6.28
11	2019-07-01	Las Vegas Trip Planners	2.82	3.33	Las Vegas Trip Planners: 2.77	Readers of Honduran Content: ...
12	2019-08-01	Cosmetics and Beauty Sh...	2.73	2.77	Las Vegas Trip Planners: 2.82	Las Vegas Trip Planners: 2.77

5. Provide a possible reason why the max average composition might change from month to month? Could it signal something is not quite right with the overall business model for Fresh Segments?

Seasonal Trends: Certain interests may naturally fluctuate due to seasonal trends. For example, travel-related interests might peak during holiday seasons, while retail might see a rise around big sale periods like Black Friday.

Key Findings and Observations

Interest Metrics and Data Integrity:

- A **peak in records** early 2019 suggests **increased engagement** or data collection.

Consistency and Fluctuations in Interests:

- **480 interests** were consistently present, indicating **core segments**.
- "**Work Comes First Travelers**" consistently showed the highest **composition values**.
- Interests like "**Mowing Equipment Shoppers**" had the lowest, showing **niche appeal**.

Volatility in Engagement:

- Significant **standard deviation** in percentile rankings for interests like "**Techies**" indicates **fluctuating consumer engagement**.

Segment and Index Analysis:

- Removing interests with **less than 6 months of data** removed many data points, suggesting **transient interests**.
- "**Luxury Bedding Shoppers**" frequently appeared in top compositions, showing sustained interest.

Trends and Seasonality:

- A **decline in average composition** mid-2019 could signal **changing preferences** or market conditions.

Strategic Implications:

- Recognizing **trends and seasonality** can inform **optimal campaign timing** and content relevance.

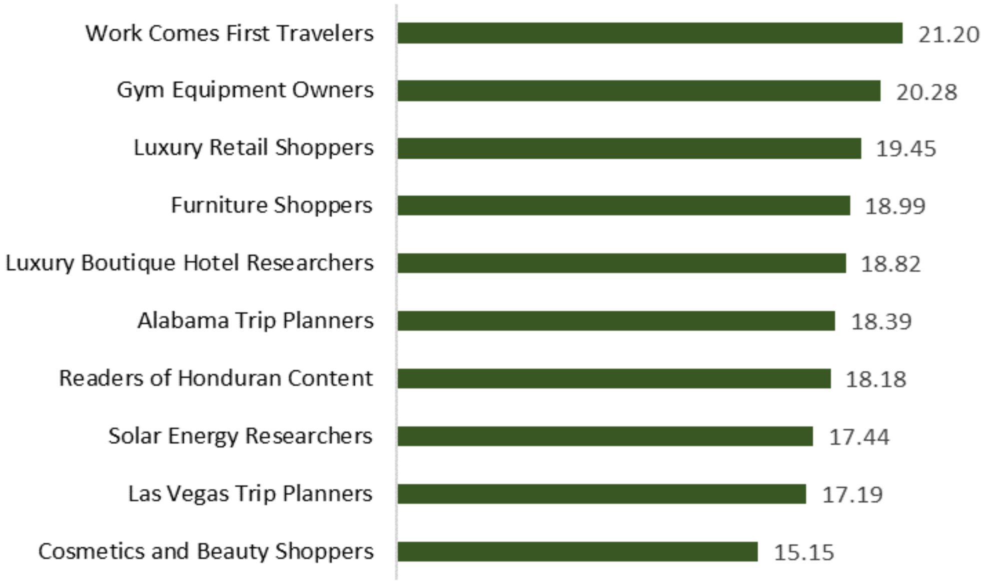
These findings underscore the importance of **regular data review** for timely adjustments to marketing strategies.

Conclusions from Analysis:

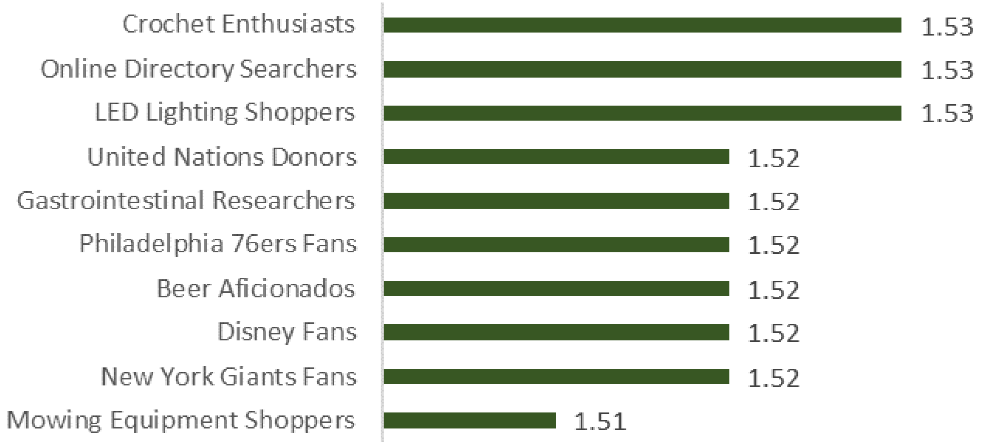
- These findings offer a compelling glimpse into the dynamic nature of customer interests and their impact on digital marketing strategies.
- Understanding these patterns can enable the client to optimize their marketing efforts, tailor content more effectively, and anticipate shifts in consumer interest with greater agility.

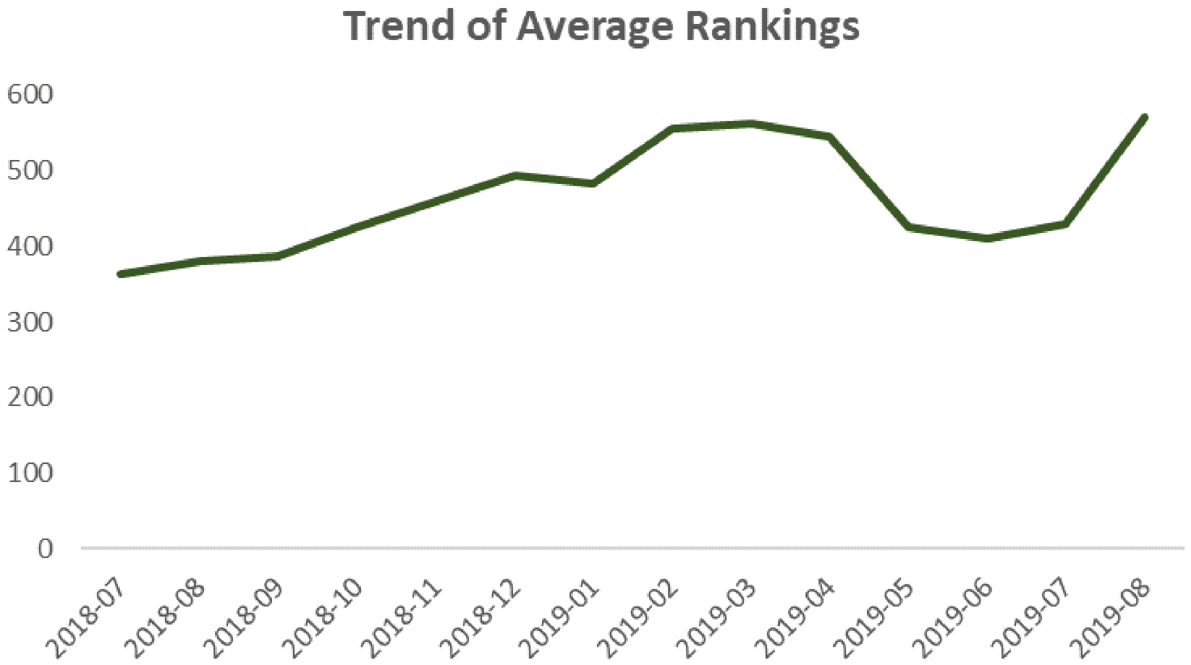
By examining these patterns and trends, businesses can better align their marketing strategies with consumer interests, ensuring that they capture attention effectively and efficiently.

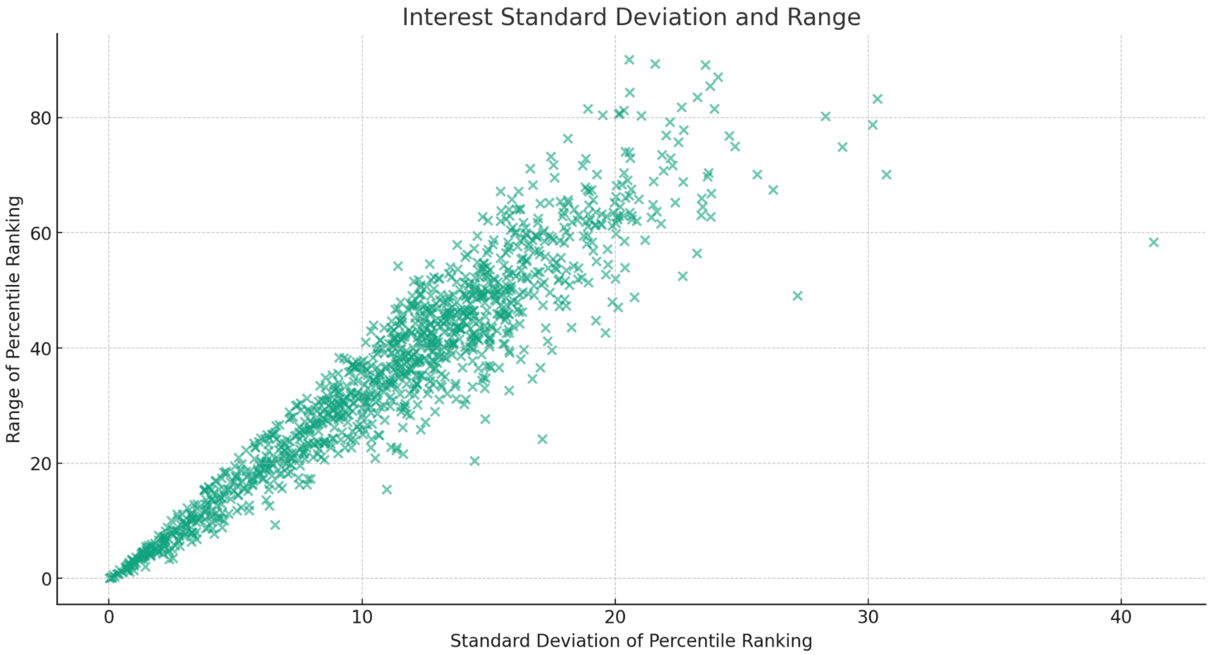
Top 10 Interests by Maximum Composition Value



Bottom 10 Interests by Minimum Composition Value




Trend of Average Rankings
600
500
400
300
200
100
0
2018-07
2018-08
2018-09
2018-10
2018-11
2018-12
2019-01
2019-02
2019-03
2019-04
2019-05
2019-06
2019-07
2019-08


Interest Standard Deviation and Range
Range of Percentile Ranking
80
60
40
20
0
0
10
20
30
40
Standard Deviation of Percentile Ranking

CONCLUSION

This case study takes us behind the scenes of how companies figure out what their customers like online, especially when it comes to ads. If you've ever wondered why certain ads pop up when you're online, it's because of customer segments – basically groups of people with similar interests.

Big online companies and social media sites use these segments to show ads that are more likely to interest you. They look at the things you like or click on and use smart computer programs to find other people who like the same things.

Our look at Fresh Segments showed us just how they use numbers and data to make sense of all this. It's like putting together a big puzzle to see what picture it makes about what customers are into. For companies, this is super useful because it helps them to be smarter about the ads they put out there.

So next time you see an ad online that seems to know exactly what you like, remember it's all about those customer segments working behind the scenes. This case study gives us a sneak peek at how it all works and why it's important for the ads we see every day.