CSI 4142 - Fundamentals of Data Science

Project Phase 3: OLAP Queries and BI Dashboard

Due Date: April 5th, 2024, 11:59pm

Group 33: Tanner Guiddings, Coralie Ostertag, and Dana Shayakhmetova

I. Introduction

Project Overview:

In Phase 0, we found a dataset containing customer information and their responses to various advertisements. Then, in Phase 1, we did the conceptual design of our dimensional model. Consequently, in Phase 2, we ensured our data's quality by transforming it into a format that allows for effective analysis for Phase 3, where we aim to develop an interactive BI dashboard powered by OLAP (Online Analytical Processing) capabilities.

Objectives:

1. To develop an interactive BI dashboard enabling dynamic data exploration.
2. To implement standard OLAP operations for exploring multidimensional data.
3. To create explorative OLAP operations for analyzing data trends and patterns.

Phase 3 Focus:

During this phase, our primary objective is to develop an interactive BI dashboard that fosters dynamic data exploration. We aim to facilitate the traversal of concept hierarchies, enabling users to perform roll-up and drill-down operations for in-depth analysis. Additionally, we seek to empower users with the capability to slice and dice the data based on various dimensions and attributes. Furthermore, we will implement features to execute Top N or Bottom N queries, facilitating outlier identification and trend analysis. By using diverse visualization techniques, including bar and pie charts, we aim to present this additional information in a visually engaging and easily understandable format.

II. Standard OLAP Operations

Through drill down, roll up, slice, dice, and their combinations, we navigate the multidimensional nature of our data to uncover patterns, trends, and anomalies.

More specifically, these operations allow us to explore the complex relationships between dimensions and measures, offering a comprehensive understanding of the factors influencing customer behaviour, marketing effectiveness, and overall business performance. With these OLAP techniques, we can make informed decisions and strategic planning in our analytical lifecycle more easily.

a) Roll Up and Drill Down by Time Period

```
SELECT EXTRACT(year FROM Dt_Customer) AS Year, COUNT(*) AS Customer_Count
FROM Customer
GROUP BY EXTRACT(year FROM Dt_Customer)
ORDER BY Year;
```

The above rolls up into the number of customers based on the year they joined, providing insight into customer acquisition trends over time.

Output From Running it on Supabase's SQL Editor:

year	customer_count
"2012"	444
"2013"	1082
"2014"	514

Similarly, it would be useful to see if the number of customers joining in each month provides certain trends and insights in our data.

```
SELECT EXTRACT(month FROM Dt_Customer) AS Month,
      COUNT(*) AS Customer_Count
FROM Customer
GROUP BY EXTRACT(month FROM Dt_Customer)
ORDER BY Month;
```

Output from running the above code on Supabase's SQL Editor:

month	customer_count
"1"	176
"2"	166
"3"	189
"4"	167
"5"	176
"6"	153
"7"	130
"8"	192
"9"	153
"10"	190
"11"	166
"12"	182

Conversely, we are able to drill down from year, to month year, to specific date.

To select the general customer information where the Dt_Customer only contains the year or month, please view the standardQueries.sql file in the github repository linked in the appendix. The output of these operations are also in the repository.

Side Note:
Another example of drilling down and rolling up on concept hierarchies includes the marital_status and living_situation concepts. In particular, the column living_situation, where values are "alone" and "partner", can be drilled down to column marital_status, where responses vary from "divorced", "together", "single",and "married". However, this was done in the phase 2 transformation of the data section.

b) Slice by Marital Status

```
SELECT Marital_Status,
      COUNT(*) AS Customer_Count
FROM Customer
WHERE Marital_Status = 'Single'
GROUP BY Marital_Status;
```

This query slices the data based on the marital status attribute, focusing exclusively on customers categorized as "Single" by employing a WHERE clause to filter records accordingly. By isolating this demographic segment, the query aggregates the sliced data by marital status and computes the count of customers within each category. Consequently, the result furnishes insights into the number of single customers, offering a nuanced understanding of this particular demographic segment.

Output from running the above code on Supabase's SQL Editor:

marital_status	customer_count
"Single"	445

To select all customers' general information according to this slice, please view the standardQueries.sql file in the github repository linked in the appendix. The output has been placed on the repository as well.

c) Dice by Elder Age Group and Complaint

```
SELECT
  Age_Group,
  Complaint_Status,
  COUNT(*) AS Customer_Count
FROM (
  SELECT
      CASE
          WHEN age_in_2014 BETWEEN 0 AND 18 THEN 'Young'
          WHEN age_in_2014 BETWEEN 19 AND 50 THEN 'Adult'
          ELSE 'Elder'
      END AS Age_Group,
      CASE
          WHEN s.Complaint_Key = 400001 THEN 'Complained'
          ELSE 'Did Not Complain'
```

```
      END AS Complaint_Status
  FROM
      Customer c
  LEFT JOIN
      SalesAnalysisFact s ON c.Customer_Key = s.Customer_Key
) AS subquery
WHERE Age_Group = 'Elder' AND Complaint_Status = 'Complained'
GROUP BY
  Age_Group,
  Complaint_Status
ORDER BY
  Age_Group;
```

Dicing involves creating a sub-cube by selecting specific values from multiple dimensions simultaneously. In this case, the query creates a subcube of data by only looking at customers considered elderly and that have made a complaint.

Output from running the above code on Supabase's SQL Editor:

age_group	complaint_status	customer_count
"Elder"	"Complained"	8

To select all customers' general information according to this dice, please view the standardQueries.sql file in the github repository linked in the appendix. The output has been placed on the repository as well.

d) Dice by People who are married and spend more than 100 dollars on gold

```
SELECT
   c.*
FROM
   Customer c
JOIN
   SalesAnalysisFact s ON c.Customer_Key = s.Customer_Key
JOIN
   Products p ON s.Products_Key = p.Products_Key
WHERE
   c.Marital_Status = 'Married'
   AND p.MntGoldProds > 100;
```

The query provides diced insights into the spending behavior of married customers specifically on gold products, allowing for a more detailed analysis of their purchasing habits. This is crucial for targeted marketing strategies and product recommendations of perhaps anniversary jewelry, updated engagement rings, etc.

To see the CSV output of this code, please visit the Standard-OLAP directory in the github repository linked in the appendix.

e) Combination

i) Roll Up by Income Group and Slice by Married Marital Status

```
SELECT
   CASE
       WHEN Income < 40000 THEN 'Low Income'
       WHEN Income BETWEEN 40000 AND 80000 THEN 'Middle Income'
       ELSE 'High Income'
   END AS Income_Group,
   COUNT(*) AS Customer_Count
FROM
   Customer
WHERE
   Marital_Status = 'Married'
GROUP BY
   ROLLUP(Income_Group)
ORDER BY
   Income_Group;
```

This query rolls up on income by selecting customers of high income, and then slices by considering only married customers. Thus, this operation provides insights into the income demographics of this specific high-earning marital status group.

Output from running the above code on Supabase's SQL Editor:

income_group	customer_count
"High Income"	72
"Low Income"	259
"Middle Income"	460

To select all customer information according to this combination of operations, please view the combinationQueries.sql file in the github repository linked in the appendix. The folder containing this file will also hold the SQL output in CSV format.

ii) Roll up by Registration Year and Slice by Master Education

```
SELECT EXTRACT(year FROM Dt_Customer) AS Year,
      COUNT(*) AS Customer_Count
FROM Customer
WHERE Education = 'Master'
GROUP BY EXTRACT(year FROM Dt_Customer)
ORDER BY Year;
```

Output from running the above code on Supabase's SQL Editor:

year	customer_count
"2012"	58
"2013"	195
"2014"	84

This query contains a roll up operation as it aggregates customer counts by year, while also implementing a slice operation by including only customers with a Masters degree. This query is helpful as it allows for a focused analysis of customer registration trends among those with a Masters education, providing valuable insights for targeted marketing strategies or tailored product offerings.

To select all customer information according to this combination of operations, please view the combinationQueries.sql file in the github repository linked in the appendix. The folder containing this file will also hold the SQL output in CSV format.

iii) Roll Up by Income Group and Slice with Graduate Education

```
SELECT CASE
        WHEN Income < 40000 THEN 'Low Income'
        WHEN Income BETWEEN 40000 AND 80000 THEN 'Middle Income'
        ELSE 'High Income'
      END AS Income_Group,
```

```
      Education,
      COUNT(*) AS Customer_Count
FROM Customer
WHERE Education IN ('PhD', 'Master')
GROUP BY Income_Group, Education
ORDER BY Income_Group, Education;
```

This query drills down on income, categorizing customers into income groups (low, middle, high) based on their income levels. It specifically focuses on customers with PhD and Master's degrees by adding a WHERE clause to filter the data for these education levels. The analysis provides insights into the income distribution among customers with advanced degrees, which could be valuable for understanding the financial demographics of this specific segment and tailoring marketing strategies accordingly.

income_group	education	customer_count
"High Income"	"Master"	34
"High Income"	"PhD"	46
"Low Income"	"Master"	98
"Low Income"	"PhD"	106
"Middle Income"	"Master"	205
"Middle Income"	"PhD"	291

iv) Roll up by Age decade and Dice with Household Size and 30s

```
WITH Age_Group_CTE AS (
   SELECT
       CASE
           WHEN (AGE_IN_2014 >= 20 AND AGE_IN_2014 < 30) THEN '20s'
           WHEN (AGE_IN_2014 >= 30 AND AGE_IN_2014 < 40) THEN '30s'
           WHEN (AGE_IN_2014 >= 40 AND AGE_IN_2014 < 50) THEN '40s'
           WHEN (AGE_IN_2014 >= 50 AND AGE_IN_2014 < 60) THEN '50s'
           WHEN (AGE_IN_2014 >= 60 AND AGE_IN_2014 < 70) THEN '60s'
```

```
            ELSE '70+'
        END AS Age_Group,
        Household_Size
    FROM
        Customer
)
SELECT
    Age_Group,
    Household_Size,
    COUNT(*) AS Customer_Count
FROM
    Age_Group_CTE
WHERE
    Age_Group = '30s' AND household_size = 3
GROUP BY
    Age_Group, Household_Size
ORDER BY
    Age_Group, Household_Size;
```

This query first performs a roll-up operation on age groups, categorizing customers into broader age groups (20s, 30s, 40s, etc.) based on their age in 2014. Then, it dices on household size and 30s age customers.

Output from running the above code on Supabase's SQL Editor:

age_group	household_size	customer_count
"30s"	3	246

To select the general customer information according to these roll up and dice conditions, please view the ExplorativeQueries.sql file in the github repository linked in the appendix. The folder containing this file will also hold the SQL output in CSV format.

III. Explorative OLAP Operations

a) Iceberg Queries

```
SELECT sf.Customer_Key,
      SUM(p.MntWines + p.MntFruits + p.MntMeatProducts + p.MntFishProducts +
p.MntSweetProducts + p.MntGoldProds) AS Total_Spending
FROM SalesAnalysisFact sf
JOIN Products p ON sf.Products_Key = p.Products_Key
GROUP BY sf.Customer_Key
ORDER BY Total_Spending DESC
LIMIT 10;
```

This query identifies the top 10 customers who spend the most across all product types. Thus, the query provides insights into the most valuable customers in terms of overall spending, allowing businesses to focus their marketing efforts or tailor their services to these high-spending customers.

Output from running the above code on Supabase's SQL Editor:

customer_key	total_spending
101126	2525
101401	2524
100949	2486
101008	2440
101489	2352
101372	2349
101226	2346
100907	2302
101359	2283
101199	2279

b) Windowing Queries

```
WITH CustomerWineSpending AS (
   SELECT c.Generation,
          c.Customer_Key,
          SUM(p.MntWines) AS Total_Wine_Spending,
          ROW_NUMBER() OVER (PARTITION BY c.Generation ORDER BY SUM(p.MntWines)
DESC) AS Wine_Spending_Rank
   FROM SalesAnalysisFact sa
   JOIN Customer c ON sa.Customer_Key = c.Customer_Key
   JOIN Products p ON sa.Products_Key = p.Products_Key
   GROUP BY c.Customer_Key
)
SELECT Generation,
      Customer_Key,
      Total_Wine_Spending,
      Wine_Spending_Rank
FROM CustomerWineSpending;
```

We performed the windowing query to rank customers within each generation based on their total spending on wines, providing insights into the relative wine purchasing behavior across different generations.

The CSV output has been placed in the github repository.

c) Window Clause

```
SELECT
   Customer_Key,
   Income,
   CASE
       WHEN (AGE_IN_2014 >= 20 AND AGE_IN_2014 < 30) THEN '20s'
       WHEN (AGE_IN_2014 >= 30 AND AGE_IN_2014 < 40) THEN '30s'
       WHEN (AGE_IN_2014 >= 40 AND AGE_IN_2014 < 50) THEN '40s'
       WHEN (AGE_IN_2014 >= 50 AND AGE_IN_2014 < 60) THEN '50s'
       WHEN (AGE_IN_2014 >= 60 AND AGE_IN_2014 < 70) THEN '60s'
       ELSE '70+'
   END AS Age_Group,
   Marital_Status,
   ROUND(AVG(Income) OVER W, 2) AS Avg_Income_By_Age_Group_Marital_Status
```

```
FROM
   Customer
WINDOW W AS (PARTITION BY
               CASE
                   WHEN (AGE_IN_2014 >= 20 AND AGE_IN_2014 < 30) THEN '20s'
                   WHEN (AGE_IN_2014 >= 30 AND AGE_IN_2014 < 40) THEN '30s'
                   WHEN (AGE_IN_2014 >= 40 AND AGE_IN_2014 < 50) THEN '40s'
                   WHEN (AGE_IN_2014 >= 50 AND AGE_IN_2014 < 60) THEN '50s'
                   WHEN (AGE_IN_2014 >= 60 AND AGE_IN_2014 < 70) THEN '60s'
                   ELSE '70+'
               END,
               Marital_Status);
```

The above query uses a window clause to calculate the average income within each age group and marital status combination. This is useful for our dataset as it allows us to analyze the distribution of income across different demographic segments while maintaining the granularity of individual records.

The CSV output has been placed in the github repository.

IV. BI Dashboard and Information Visualization

To create our dashboard, we opted for Google Looker Data Studio for several reasons. One crucial factor was compatibility, as most team members use MacBooks, and alternative applications like Power BI are not supported on Apple devices without VMs. Furthermore, Looker Data Studio's robust capabilities and user-friendly interface make it an ideal choice for effectively presenting and analyzing our main data characteristics in a visually appealing and informative manner.

We have created a dashboard that consists of four pages:

<u>First page: General Customer Insights</u>
The first page, titled 'General Customer Insights,' provides an overview of customer details within our dataset. The Customer Acquisition line chart allows users to drill down from a yearly time frame to specific dates using the corner arrows. Similarly, we are able to drill up (i.e. roll up) from the specific dates to years.

Additionally, users can slice the data by double-clicking on specific attributes in the bottom three graphs. For example, to view data sliced by marital status of “Single”, double-click on the corresponding fraction in the pie chart below.
Click on the visualization again to clear the slice filter. Alternatively, users can dice the data by double-clicking on two specific attributes in different visualizations.

Second page: Top-N
This page presents an example of a Top-N query, that displays the top spending customers in our data set. We are able to slice and dice this iceberg data according to the bottom three graphs displayed on that page.

Third page: Bottom-N
Similarly, this page presents an example of a Bottom-N query, that displays the least spending customers in our data set. We are able to slice and dice this iceberg data according to the bottom three graphs displayed on that page.

Fourth page: Other Customer Details
This page visualizes data that was not presented in the previous dashboard pages, but that could be valuable to our data analysis and comprehension.

The link to our BI dashboard is provided in part B of the document appendix.

V. Team Planning

Deliverable Checklist	Responsible Team Members	Completion Date
Standard OLAP Queries	Tanner	27th of March, 2024.
OLAP Combinations	Dana	27th of March, 2024.
Explorative OLAP	Coralie	28th of March, 2024.
Github Documentation of Outputs	Dana	30th of March, 2024.
BI Dashboard	Dana	2nd of April, 2024.
Document creation and organization	Dana	27th of March, 2024.
Document writing	Everyone	4th of April, 2024.
Team planning	Dana	4th of April, 2024.
Note: The distribution of team planning may appear uneven on this tracker because the third phase was led by Dana, while Phase 4 was primarily focused on by Tanner and Coralie. We decided to split the work in this manner because the two phases were due on the same day. We can assure you that this was still a collaborative process and time spent on all tasks was evenly spent. :)		

VI. Conclusion

In conclusion, Phase 3 of our CSI 4142 project has been focused on implementing OLAP queries and developing a BI dashboard to enable dynamic data exploration. By leveraging standard OLAP operations and explorative OLAP techniques, we have gained valuable insights into our dataset, allowing us to uncover trends, patterns, and anomalies. The BI dashboard, built using Google Looker Data Studio, provides an intuitive interface for visualizing key data points and facilitates informed decision-making.

VII. Appendix

Appendix A: Link to Github Repository of Phase 3

You may access the SQL code and CSV output in the following github:
https://github.com/DanaShayakhmetova/CSI4142-Phase3

Appendix B: Link to BI Dashboard

You may access the BI Dashboard created with Google Looker Studio with the following link:
https://lookerstudio.google.com/reporting/12e39931-8d1e-4672-93ce-ccc1ad16f6fc