Query 1: Sales by Customer Age Group

In this query, I examined sales data based on consumer age groupings for each country. By calculating customer age using birth dates and order dates, I categorized the results by country and age group. This analysis provides valuable insights into purchasing patterns across different age segments and regions, enabling me to tailor marketing strategies to meet specific customer requirements and preferences.

```
WITH CTE_MAIN AS
(
SELECT T3.EnglishCountryRegionName,
DATEDIFF(YEAR, BirthDate, ORDERDATE) AS AGE, ---Shows customer's ages when they ordered
SalesOrderNumber
FROM [AdventureWorksDW2019].[dbo].[FactInternetSales] T1
JOIN AdventureWorksDW2019.DBO.DimCustomer T2
ON T1.CustomerKey = T2.CustomerKey
JOIN AdventureWorksDW2019.DBO.DimGeography T3
ON T2.GeographyKey = T3.GeographyKey
)
---Creating the age groups using CASE WHEN Statements to show the sales by age groups---
SELECT EnglishCountryRegionName,
   CASE WHEN AGE < 30 THEN 'a: Under 30'
   WHEN AGE BETWEEN 30 AND 40 THEN 'b: 30 - 40'
   WHEN AGE BETWEEN 40 AND 50 THEN 'c: 40 - 50'
   WHEN AGE BETWEEN 50 AND 60 THEN 'd: 50 - 60'
   WHEN AGE > 60 THEN 'e: over 60'
   ELSE 'Other'
   END AS Age_group,
   COUNT(SalesOrderNumber) AS Sales
FROM CTE_MAIN
GROUP BY EnglishCountryRegionName,
   CASE WHEN AGE < 30 THEN 'a: Under 30'
   WHEN AGE BETWEEN 30 AND 40 THEN 'b: 30 - 40'
   WHEN AGE BETWEEN 40 AND 50 THEN 'c: 40 - 50'
   WHEN AGE BETWEEN 50 AND 60 THEN 'd: 50 - 60'
   WHEN AGE > 60 THEN 'e: over 60'
   ELSE 'Other'
   END
   ORDER BY EnglishCountryRegionName, Age_group
```

Query 2: Product Sales by Age Group

This search aims to study product sales based on client age groups. To do this, I calculated customer age using birthdates and order dates, categorizing the results by product kind and age group. The analysis can uncover patterns and age-based preferences, providing insights into how different age groups shop. Understanding these interactions can help marketers and product developers better cater to their diverse client base.

```
WITH CTE_MAIN AS
```

```
(
SELECT T4.EnglishProductName,
   DATEDIFF(YEAR, BirthDate, ORDERDATE) AS AGE,
---Shows customer's when they ordered---
   SalesOrderNumber,
   T5.EnglishProductSubcategoryName
FROM [AdventureWorksDW2019].[dbo].[FactInternetSales] T1
JOIN AdventureWorksDW2019.DBO.DimCustomer T2
ON T1.CustomerKey = T2.CustomerKey
JOIN AdventureWorksDW2019.DBO.DimGeography T3
ON T2.GeographyKey = T3.GeographyKey
JOIN AdventureWorksDW2019.DBO.DimProduct T4
ON T1.ProductKey = T4.ProductKey
JOIN [AdventureWorksDW2019].[DBO].[DimProductSubcategory] T5
ON T4.ProductSubcategoryKey = T5.ProductSubcategoryKey
)
---Creating the age groups using CASE WHEN Statements to show the product sales by age groups---
SELECT
      EnglishProductSubcategoryName AS Product_Type,
   CASE WHEN AGE < 30 THEN 'a: Under 30'
   WHEN AGE BETWEEN 30 AND 40 THEN 'b: 30 - 40'
   WHEN AGE BETWEEN 40 AND 50 THEN 'c: 40 - 50'
   WHEN AGE BETWEEN 50 AND 60 THEN 'd: 50 - 60'
   WHEN AGE > 60 THEN 'e: over 60'
   ELSE 'Other'
   END AS Age_Group,
   COUNT(SalesOrderNumber) AS Sales
FROM CTE_MAIN
GROUP BY
   EnglishProductSubcategoryName,
   CASE WHEN AGE < 30 THEN 'a: Under 30'
   WHEN AGE BETWEEN 30 AND 40 THEN 'b: 30 - 40'
   WHEN AGE BETWEEN 40 AND 50 THEN 'c: 40 - 50'
   WHEN AGE BETWEEN 50 AND 60 THEN 'd: 50 - 60'
   WHEN AGE > 60 THEN 'e: over 60'
   ELSE 'Other'
   END
   ORDER BY Product_Type, Age_Group
```

Query 3: Monthly sales for Australia and USA compared for year 2012

In this query, I compared the monthly sales between Australia and the USA for the year 2012. My objective was to understand their sales trends and performance during that period. By filtering and including only the sales data from Australia and the USA for 2012, the analysis provided valuable insights, enabling informed decisions and strategies to optimize sales and market performance in both countries.

```
SELECT SUBSTRING(CAST ([OrderDateKey] as char),1,6) AS Month_Key
---The month and year of the sales (formatted as YYYYMM)
    ,[SalesOrderNumber]---The unique identifier of the sales order
    ,[OrderDate]---The date of the sales order
    ,T2.SalesTerritoryCountry---The country of the sales territory
 FROM [AdventureWorksDW2019].[dbo].[FactInternetSales] T1
 JOIN [AdventureWorksDW2019].[dbo].DimSalesTerritory T2
 ON T1.SalesTerritoryKey = T2.SalesTerritoryKey
```

```
 WHERE SalesTerritoryCountry IN ('Australia', 'United States')
 AND SUBSTRING(CAST ([OrderDateKey] as char),1,4) = '2012'
---Limits to year 2012
```

Query 4: First Reorder Date for Each Product

The purpose of this query is to find the first reorder date for each product based on sales data. By executing this query, I gained valuable insights into the best timing for reordering, ensuring sufficient stock levels to meet customer demand without excess inventory. This analysis improved inventory management, optimized supply chain operations, and enhanced customer satisfaction through timely product availability.

```
--- Query Utilizes two Common Table Expressions (CTEs) - 'MAIN_CTE' and 'REORDER_DATE '
WITH MAIN_CTE AS
--- Calculates the total sales quantity for each product by aggregating the data from the `DimProduct` and `FactInternetSales` tables
(
SELECT
     [EnglishProductName] --- Product name in english
     ,OrderDateKey
     ,[SafetyStockLevel]
     ,[ReorderPoint]
     ,SUM(T2.OrderQuantity) AS SALES
  FROM [AdventureWorksDW2019].[dbo].[DimProduct] T1
  JOIN DBO.FactInternetSales T2 ON T1.ProductKey = T2.ProductKey
  GROUP BY
    [EnglishProductName]
    ,OrderDateKey
     ,[SafetyStockLevel]
     ,[ReorderPoint]
),
REORDER_DATE
---Determines the first reorder date for each product based on the safety stock level and running total sales quantity
AS
(
       SELECT *, CASE WHEN (SAFETYSTOCKLEVEL - RUNNING_TOTAL_SALES) <= REORDERPOINT THEN 1 ELSE 0 END AS
REORDER_FLAG
       FROM
(SELECT *, SUM(SALES) OVER (PARTITION BY ENGLISHPRODUCTNAME ORDER BY ORDERDATEKEY) AS RUNNING_TOTAL_SALES
FROM MAIN_CTE
GROUP BY
    [EnglishProductName]
     ,OrderDateKey
     ,[SafetyStockLevel]
     ,[ReorderPoint]
     ,SALES) MAIN_SQ
     )
SELECT EnglishProductName, MIN(ORDERDATEKEY) AS First_Reorder_Date---The first date when the product needs to be reordered
FROM REORDER_DATE
WHERE REORDER_FLAG = 1---identifies if the safety stock level is lower than or equal to the running total sales quantity. If the flag is 1, it indicates
that a reorder is required.
GROUP BY EnglishProductName
```

Query 5: Days Between First Order and First Reorder Date

I aimed to calculate the days between a product's first order and its first reorder in this query. I identify products taking over a year to reorder, suggesting possible excess stock. By adding a new column to the results from Query 4, this analysis offers valuable insights to optimize inventory levels and enhance supply chain efficiency.

```
---Query Utilizes two Common Table Expressions (CTEs) - 'MAIN_CTE' and 'REORDER_DATE '
WITH MAIN_CTE AS
---retrieves sales and relevant details for each product by joining the DimProduct and FactInternetSales tables and grouping the data
(
SELECT
     [EnglishProductName]
     ,OrderDateKey
     ,[SafetyStockLevel]
     ,[ReorderPoint]
     ,SUM(T2.OrderQuantity) AS SALES
  FROM [AdventureWorksDW2019].[dbo].[DimProduct] T1
  JOIN DBO.FactInternetSales T2 ON T1.ProductKey = T2.ProductKey
  GROUP BY
    [EnglishProductName]
    ,OrderDateKey
     ,[SafetyStockLevel]
     ,[ReorderPoint]
),
REORDER_DATE
---determines the first reorder date for each product based on the safety stock level and running total sales quantity. It includes a REORDER_FLAG column to identify if a reorder is needed.
AS
(
SELECT *,
CASE WHEN (SAFETY STOCK LEVEL - RUNNING_TOTAL_SALES) <= REORDERPOINT         THEN 1 ELSE 0 END AS REORDER_FLAG
FROM
(SELECT *, SUM(SALES) OVER (PARTITION BY ENGLISHPRODUCTNAME ORDER BY ORDERDATEKEY) AS RUNNING_TOTAL_SALES
FROM MAIN_CTE
GROUP BY
    [EnglishProductName]
     ,OrderDateKey
     ,[SafetyStockLevel]
     ,[ReorderPoint]
     ,SALES) MAIN_SQ
     )
SELECT EnglishProductName, MAX(products_first_orderdate) AS products_first_orderdate, MAX(first_reorder_date) AS first_reorder_date,
datediff(day, MAX(CAST(CAST(products_first_orderdate AS char) AS date)), MAX(CAST(CAST(first_reorder_date AS char) AS date))) AS
days_to_reorder
---Retrieves the number of days between the first order and first reorder date
FROM
(
SELECT EnglishProductName, MIN(ORDERDATEKEY) AS products_first_orderdate,NULL AS first_reorder_date
FROM MAIN_CTE
GROUP BY EnglishProductName
UNION ALL---Vertically Combines results from the two subqueries into a single result
SELECT EnglishProductName, NULL AS products_first_orderdate, MIN(ORDERDATEKEY) AS first_reorder_date
FROM REORDER_DATE
WHERE REORDER_FLAG = 1---identifies if the safety stock level is lower than or equal to the running total sales quantity. If the flag is 1, it indicates that a reorder is required.
GROUP BY EnglishProductName
```

```
) main_subquery
GROUP BY EnglishProductName
HAVING DATEDIFF(day, MAX(CAST(CAST(products_first_orderdate AS char) AS date)), MAX(CAST(CAST(first_reorder_date AS char) AS date))) > 365
--DATEDIFF operator used to calculate the difference in days between first order date and first reorder date
```

Query 6: Sales Promotion Discount Analysis

In this SQL query, my purpose was to retrieve all sales transactions linked to a promotion and calculate the new sales value by applying a 25% discount. By running this query, I will obtain the updated sales figures after the discount has been applied to each transaction, which will provide valuable insights into the impact of promotions on our sales revenue. This analysis will help assess the effectiveness of promotions and their contribution to overall business performance.

```
-- Select the OrderDate, SalesReasonName, SalesOrderNumber, and calculate the sales_amount_after_discount
SELECT OrderDate, SalesReasonName, t1.SalesOrderNumber, (SalesAmount*0.75) AS sales_amount_after_discount
FROM [AdventureWorksDW2019].[dbo].[FactInternetSales] t1
-- Join with the FactInternetSalesReason table to link sales transactions with reasons
JOIN [AdventureWorksDW2019].[dbo].[FactInternetSalesReason] t2
  ON t1.SalesOrderNumber = t2.SalesOrderNumber
-- Join with the DimSalesReason table to retrieve the sales reason name
JOIN [AdventureWorksDW2019].[dbo].[DimSalesReason] t3
  ON t2.SalesReasonKey = t3.SalesReasonKey
-- Filter the results to include only sales transactions associated with the promotion
WHERE SalesReasonName = 'On Promotion'
```

Query 7: Customer First and Last Sale Analysis

This SQL query's objective is to examine each customer's sales data, with an emphasis on the earliest and most recent sales. The customer key, the sales value of their initial purchase, the sales value of their most recent purchase, and the computed difference between the two were all included in the information that I created the query to get. This query offers insightful information about clients' purchasing patterns by contrasting their initial and most recent purchases. A more thorough understanding of consumer behavior and preferences can be achieved by using the differences to identify clients who have seen substantial changes in their purchasing habits or overall sales trends.

```
WITH first_purchase AS
(
---Retrieves the customer key, sales amount, and order date for the first purchase of each customer
SELECT
  [CustomerKey],
  [SalesAmount],
  [OrderDate],
  ROW_NUMBER() OVER (PARTITION BY [CustomerKey] ORDER BY [OrderDate] ASC) AS purchase_num
```

```
FROM [AdventureWorksDW2019].[dbo].[FactInternetSales]
),
last_purchase AS
(
---Retrieve the customer key, sales amount, and order date for the last purchase of each customer
SELECT
    [CustomerKey],
    [SalesAmount],
    [OrderDate],
    ROW_NUMBER() OVER (PARTITION BY [CustomerKey] ORDER BY [OrderDate] ASC) AS purchase_num
FROM [AdventureWorksDW2019].[dbo].[FactInternetSales]
)

SELECT
    customerkey,
    SUM(first_purchase_value) AS first_purchase_value,
    SUM(last_purchase_value) AS last_purchase_value,
    (SUM(first_purchase_value) - SUM(last_purchase_value)) AS difference
FROM
(
--- Combine the results of the first and last purchases using union all join
SELECT
    CustomerKey,
    SalesAmount AS first_purchase_value, --- Sales value of the first purchase
    NULL AS last_purchase_value --- Set as NULL since this represents the first purchase
 FROM first_purchase
 WHERE purchase_num = 1 --- Filter to select only the first purchase records

UNION ALL

SELECT
    CustomerKey,
    NULL AS first_purchase_value, --- Set as NULL since this represents the last purchase
    SalesAmount AS last_purchase_value --- Sales value of the last purchase
FROM last_purchase
WHERE purchase_num = 1 --- Filter to select only the last purchase records
) main_subquery
GROUP BY customerkey
HAVING (SUM(first_purchase_value) - SUM(last_purchase_value)) <> 0 --- Exclude customers with equal first and last purchase values
ORDER BY customerkey
```

Query 8: Filtered Bike Sales Analysis by Commute Distance and Country

I built upon the previous Query to create this one. I concentrated on using our clients' income ranges to filter bike purchases in this query. To do this, I divided consumer revenue into different categories and determined the number of purchases made in each category. The analysis produced by this query will give us important information about the spending habits of our clients in relation to their income, enabling us to spot trends and patterns that can help us better understand our target market.

```
WITH bike_sales AS
(
---Retrieves relevant columns for bike sales, joining necessary tables and applying filters
SELECT
    orderdatekey,
```

```
    OrderDate,
    CustomerKey,
    BirthDate,
    YearlyIncome,
    TotalChildren,
    CommuteDistance,
    englishcountryregionname AS Country,
    SalesAmount,
    SalesOrderNumber
FROM [AdventureWorksDW2019].[dbo].[FactInternetSales] T1
 JOIN [AdventureWorksDW2019].[dbo].[DimCustomer] T2 ON T1.CustomerKey = T2.CustomerKey
 JOIN [AdventureWorksDW2019].[dbo].[DimGeography] T3 ON T2.Geographykey = T3.Geographykey
 JOIN [AdventureWorksDW2019].[dbo].[DimProduct] T4 ON T1.ProductKey = T4.ProductKey
 JOIN [AdventureWorksDW2019].[dbo].[DimProductSubcategory] T5 ON T4.ProductSubcategoryKey = T5.ProductSubcategoryKey
 WHERE EnglishProductSubcategoryName IN ('Mountain Bikes', 'Touring Bikes', 'Road Bikes')
)

SELECT
    Country,
    CommuteDistance,
    COUNT(DISTINCT SalesOrderNumber) AS sales
FROM bike_sales
GROUP BY Country, CommuteDistance
ORDER BY Country, CommuteDistance
```

Query 9: Bike Sales Analysis by Customer and Income Brackets.

In this query, I build upon the previous one. Here, I use the customer's salary ranges to filter bike sales. To do this, I divide income into brackets and count how many things each bracket has bought. This study provides insightful information on consumer spending habits tied to customer income, allowing me to spot trends and patterns. In order to effectively cater marketing strategies and product offerings to customers with different income levels, it is important to understand how income affects bike sales.

```
SELECT CustomerKey,
---Income brackets created using the CASE Statement based on their yearly income
CASE WHEN YearlyIncome < 50000 THEN 'Less than 50K'
WHEN YearlyIncome BETWEEN 50000 AND 75000 THEN '50K - 75K'
WHEN YearlyIncome BETWEEN 75000 AND 100000 THEN '75K - 100K'
WHEN YearlyIncome > 100000 THEN 'Over 100K'
ELSE 'Other'
END AS Income,
COUNT(salesordernumber) AS purchases
FROM bike_sales
GROUP BY CustomerKey,
                                        YearlyIncome
```

Query 10: Analysis of Customers with and without Children

I created this query with the purpose of identifying customers who have children and those who do not. I wanted to compare their buying patterns by collecting their sales figures and dates. In order to better understand our client base, this analysis will help identify any potential trends in sales among customer segments for the year 2012.

```sql
--- Selects the substring of the Orderdatekey column, representing the month, and aliases it as "month_key"
--- Determines whether the customer has children or not based on the TotalChildren column and aliases it as "Has Children"
--- Calculates the count of sales orders and aliases it as "sales"
SELECT
 SUBSTRING(CAST(Orderdatekey AS char), 1, 6) AS month_key,
 CASE
   WHEN TotalChildren = 0 THEN 'No Children'
   ELSE 'Has Children'
 END AS 'Has Children',
 COUNT(salesordernumber) AS sales
--- Retrieves data from the "bike_sales" CTE
--- Filters the data to include only sales orders from the year 2012 based on the substring of the Orderdatekey column
WHERE SUBSTRING(CAST(Orderdatekey AS char), 1, 4) = '2012'
--- Groups the results by the month_key and TotalChildren
GROUP BY
 SUBSTRING(CAST(Orderdatekey AS char), 1, 6),
 TotalChildren
```

Last Steps:

I inputted the data results into Google Sheets to create visualizations that represent the analysis carried out using the SQL queries. Additionally, I prepared the data for Tableau by creating a comprehensive table with all the required data using JOINS. This data was then used to build an interactive dashboard on Tableau, leveraging the same dataset for further insights and data exploration.

```sql
SELECT
OrderDateKey,
OrderDate,
SalesOrderNumber,
SalesAmount,
BirthDate,
YearlyIncome,
TotalChildren,
TotalProductCost,
EnglishCountryRegionName,
EnglishProductName,
EnglishProductSubcategoryName
FROM AdventureWorksDW2019.DBO.FactInternetSales T1
JOIN DBO.DimCustomer T2
ON T1.CustomerKey = T2.CustomerKey
JOIN DBO.DimGeography T3
ON T2.GEOGRAPHYKEY = T3.GeographyKey
JOIN DBO.DimProduct T4
ON T1.ProductKey = T4.ProductKey
JOIN DBO.DimProductSubcategory T5
ON T4.ProductSubcategoryKey = T5.ProductSubcategoryKey
WHERE OrderDate BETWEEN '20130101' AND '20131231'
```