LIST OF SQL Queries
used in Power BI to transform our database tables

TABLE: dim_customer

```
-- SQL statement to join dim_customers with dim_geography to enrich customer data with geographic information

SELECT
    c.CustomerID,  -- Selects the unique identifier for each customer
    c.CustomerName,  -- Selects the name of each customer
    c.Email,  -- Selects the email of each customer
    c.Gender,  -- Selects the gender of each customer
    c.Age,  -- Selects the age of each customer
    g.Country,  -- Selects the country from the geography table to enrich customer data
    g.City  -- Selects the city from the geography table to enrich customer data
FROM
    dbo.customers as c  -- Specifies the alias 'c' for the dim_customers table
LEFT JOIN
-- RIGHT JOIN
-- INNER JOIN
-- FULL OUTER JOIN
    dbo.geography g  -- Specifies the alias 'g' for the dim_geography table
ON
    c.GeographyID = g.GeographyID;  -- Joins the two tables on the GeographyID field to match customers with their geographic information
```

TABLE: products

```
-- SQL Query to categorize products based on their price

SELECT
    ProductID,  -- Selects the unique identifier for each product
    ProductName,  -- Selects the name of each product
    Price,  -- Selects the price of each product
          -- Category, -- Selects the product category for each product

    CASE -- Categorizes the products into price categories: Low, Medium, or High
        WHEN Price < 50 THEN 'Low'  -- If the price is less than 50, categorize as 'Low'
        WHEN Price BETWEEN 50 AND 200 THEN 'Medium'  -- If the price is between 50 and 200 (inclusive), categorize as 'Medium'
        ELSE 'High'  -- If the price is greater than 200, categorize as 'High'
    END AS PriceCategory  -- Names the new column as PriceCategory

FROM
```

```
    dbo.products;  -- Specifies the source table from which to select the data
```

TABLE: fact_customer_reviews

```
-- Query to clean whitespace issues in the ReviewText column

SELECT
    ReviewID,  -- Selects the unique identifier for each review
    CustomerID,  -- Selects the unique identifier for each customer
    ProductID,  -- Selects the unique identifier for each product
    ReviewDate,  -- Selects the date when the review was written
    Rating,  -- Selects the numerical rating given by the customer (e.g., 1 to 5 stars)
    -- Cleans up the ReviewText by replacing double spaces with single spaces to ensure the text is more readable and standardized
    REPLACE(ReviewText, '  ', ' ') AS ReviewText
FROM
    dbo.customer_reviews;  -- Specifies the source table from which to select the data
```

TABLE: fact_engagement_data

```
-- Query to clean and normalize the engagement_data table

SELECT
    EngagementID,  -- Selects the unique identifier for each engagement record
    ContentID,  -- Selects the unique identifier for each piece of content
        CampaignID,  -- Selects the unique identifier for each marketing campaign
    ProductID,  -- Selects the unique identifier for each product
    UPPER(REPLACE(ContentType, 'Socialmedia', 'Social Media')) AS ContentType,  -- Replaces "Socialmedia" with "Social Media" and then converts all ContentType values to uppercase
    LEFT(ViewsClicksCombined, CHARINDEX('-', ViewsClicksCombined) - 1) AS Views,  -- Extracts the Views part from the ViewsClicksCombined column by taking the substring before the '-' character
    RIGHT(ViewsClicksCombined, LEN(ViewsClicksCombined) - CHARINDEX('-', ViewsClicksCombined)) AS Clicks,  -- Extracts the Clicks part from the ViewsClicksCombined column by taking the substring after the '-' character
    Likes,  -- Selects the number of likes the content received
    -- Converts the EngagementDate to the dd.mm.yyyy format
    FORMAT(CONVERT(DATE, EngagementDate), 'dd.MM.yyyy') AS EngagementDate  -- Converts and formats the date as dd.mm.yyyy
FROM
```

```
    dbo.engagement_data  -- Specifies the source table from which to select the data
WHERE
    ContentType != 'Newsletter';  -- Filters out rows where ContentType is 'Newsletter' as these are not relevant for our analysis
```

TABLE: fact_customer_journey

```
-- Outer query selects the final cleaned and standardized data

SELECT
    JourneyID,  -- Selects the unique identifier for each journey to ensure data traceability
    CustomerID,  -- Selects the unique identifier for each customer to link journeys to specific customers
    ProductID,  -- Selects the unique identifier for each product to analyze customer interactions with different products
    VisitDate,  -- Selects the date of the visit to understand the timeline of customer interactions
    Stage,  -- Uses the uppercased stage value from the subquery for consistency in analysis
    Action,  -- Selects the action taken by the customer (e.g., View, Click, Purchase)
    COALESCE(Duration, avg_duration) AS Duration  -- Replaces missing durations with the average duration for the corresponding date
FROM
    (
        -- Subquery to process and clean the data
        SELECT
            JourneyID,  -- Selects the unique identifier for each journey to ensure data traceability
            CustomerID,  -- Selects the unique identifier for each customer to link journeys to specific customers
            ProductID,  -- Selects the unique identifier for each product to analyze customer interactions with different products
            VisitDate,  -- Selects the date of the visit to understand the timeline of customer interactions
            UPPER(Stage) AS Stage,  -- Converts Stage values to uppercase for consistency in data analysis
            Action,  -- Selects the action taken by the customer (e.g., View, Click, Purchase)
            Duration,  -- Uses Duration directly, assuming it's already a numeric type
            AVG(Duration) OVER (PARTITION BY VisitDate) AS avg_duration,  -- Calculates the average duration for each date, using only numeric values
```

```
        ROW_NUMBER() OVER (
            PARTITION BY CustomerID, ProductID, VisitDate, UPPER(Stage), Action
-- Groups by these columns to identify duplicate records
            ORDER BY JourneyID  -- Orders by JourneyID to keep the first occurrence of
each duplicate
        ) AS row_num  -- Assigns a row number to each row within the partition to
identify duplicates
    FROM
        dbo.customer_journey  -- Specifies the source table from which to select the data
) AS subquery  -- Names the subquery for reference in the outer query
WHERE
  row_num = 1;  -- Keeps only the first occurrence of each duplicate group identified in
the subquery
```