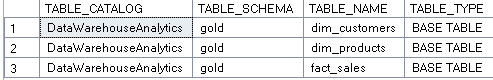
Document on the Queries

This Document contains queries and their results related to EDA and Advance Data Analytics

Step 1. DATABASE EXPLORATION

1. Explore All Objects in the Database

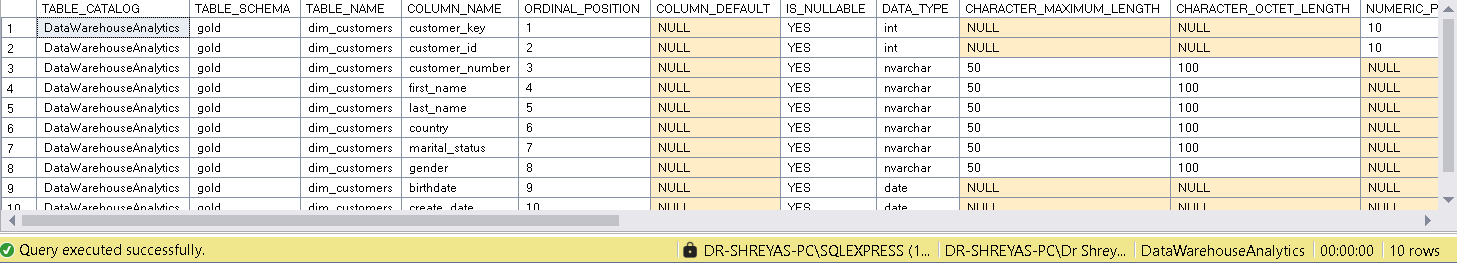
SELECT \* FROM INFORMATION\_SCHEMA.TABLES



1. Explore All Columns in Database

SELECT \* FROM INFORMATION\_SCHEMA.COLUMNS

WHERE TABLE\_NAME = 'dim\_customers'



Step 2. DIMENSIONS EXPLORATIO

Identifying the unique values (or categories) in each dimensions &

Recognizing how data might be grouped or segmented. Which is useful for later analysis

1. Explore All Countries our Customers come from

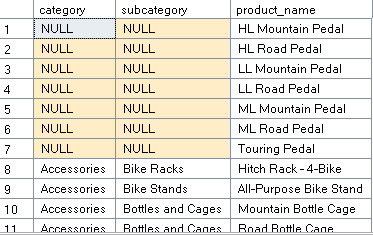
SELECT DISTINCT country from gold.dim\_customers;



1. Explore All Categories "The major Divisions", Subcategories & Products

SELECT DISTINCT category, subcategory, product\_name FROM gold.dim\_products

ORDER BY 1,2,3



In Total we have 295 different Products

Step 3. DATE EXPLORATION

Identify the earliest and latest dates (boundaries). Understand the scope of data and the timespan

1. Find the date of the first and last order

SELECT

MIN(order\_date) first\_order\_date,

MAX(order\_date) last\_order\_date

FROM gold.fact\_sales;



1. How many years of sales are available

SELECT

MIN(order\_date) first\_order\_date,

MAX(order\_date) last\_order\_date,

DATEDIFF(year, MIN(order\_date), MAX(order\_date)) AS order\_range\_years

FROM gold.fact\_sales;



1. --Find the youngest and oldest customer

SELECT

MIN(birthdate) oldest\_birthdate,

DATEDIFF(year, MIN(birthdate), GETDATE()) AS youngest\_age,

MAX(birthdate) youngest\_birthdate,

DATEDIFF(year, MAX(birthdate), GETDATE()) AS youngest\_age

FROM gold.dim\_customers;



Step 4. MEASURES EXPLORATION

Calculate the key metric of the business (Big Numbers)

-Highest Level of Aggregation

1. --Find the Total Sales

SELECT SUM(sales\_amount) AS total\_sales FROM gold.fact\_sales



1. --Find how many items are sold

SELECT SUM(quantity) FROM gold.fact\_sales



1. --Find the average selling price

SELECT AVG(price) AS avg\_selling\_price FROM gold.fact\_sales



1. --Find the total number of orders

SELECT COUNT(DISTINCT order\_number) as total\_orders from gold.fact\_sales



1. --Find the total number of products

SELECT COUNT(product\_name) AS total\_products FROM gold.dim\_products



1. --Find the total number of customers

SELECT COUNT(customer\_key) AS total\_customers FROM gold.dim\_customers



1. --Find the total number of customers that has placed an order

SELECT COUNT(DISTINCT customer\_key) AS total\_customers FROM gold.fact\_sales



1. --Generate a Report that shows all key metrics of the business

SELECT 'Total Sales' AS meaure\_name, SUM(sales\_amount) AS measure\_value FROM gold.fact\_sales

UNION ALL

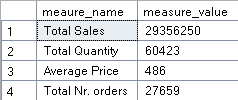
SELECT 'Total Quantity', SUM(quantity) FROM gold.fact\_sales

UNION ALL

SELECT 'Average Price', AVG(price) AS avg\_selling\_price FROM gold.fact\_sales

UNION ALL

SELECT 'Total Nr. orders', COUNT(DISTINCT order\_number) as total\_orders from gold.fact\_sales



Step 5. Magnitude

Compare the measure values by categories

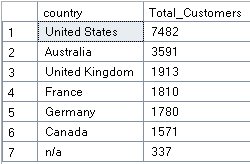
It helps us understand the importance of different categories

1. --Find total customers by countries

SELECT country, COUNT(customer\_id) AS Total\_Customers FROM gold.dim\_customers

GROUP BY country

ORDER BY Total\_Customers DESC

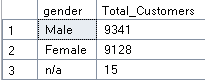


1. --Find total customers by gender

SELECT gender, COUNT(customer\_id) AS Total\_Customers FROM gold.dim\_customers

GROUP BY gender

ORDER BY Total\_Customers DESC

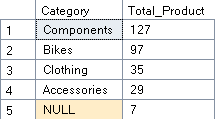


1. --Find total products by category

SELECT Category, COUNT(DISTINCT product\_key) AS Total\_Product FROM gold.dim\_products

GROUP BY category

ORDER BY Total\_Product DESC

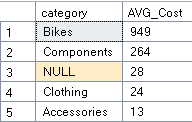


1. --What is the average cost in each category

SELECT category, AVG(cost) AS AVG\_Cost From gold.dim\_products

GROUP BY category

ORDER BY AVG\_Cost DESC



1. --What is the total revenue generated for each category

SELECT

p.category,

SUM(f.sales\_amount) total\_revenue

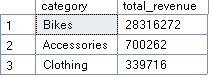
FROM gold.fact\_sales f

LEFT JOIN gold.dim\_products p

ON p.product\_key = f.product\_key

GROUP BY p.category

ORDER BY total\_revenue DESC



1. --What is the total revenue generated by each customer

SELECT DISTINCT

c.customer\_key,

c.first\_name,

c.last\_name,

SUM(s.sales\_amount) AS Total\_Revenue

FROM gold.dim\_customers c

LEFT JOIN gold.fact\_sales s

ON c.customer\_key = s.customer\_key

GROUP BY

c.customer\_key,

c.first\_name,

c.last\_name

ORDER BY Total\_Revenue DESC



1. --What is the distribution of items sold across countries

SELECT DISTINCT

c.country,

SUM(s.quantity) AS Total\_sold\_items

FROM gold.dim\_customers c

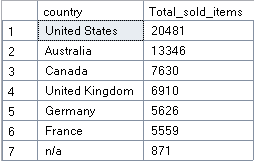
LEFT JOIN gold.fact\_sales s

ON c.customer\_key = s.customer\_key

GROUP BY

c.country

ORDER BY Total\_sold\_items DESC



Step 6. Ranking (Top N – Bottom N)

Order the values of dimensions by measure

1. --Which 5 Products generate the highest revenue

SELECT Top 5

p.product\_name,

SUM(f.sales\_amount) total\_revenue,

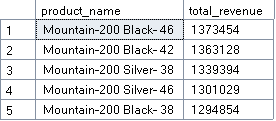
ROW\_NUMBER() OVER (ORDER BY SUM(f.sales\_amount) DESC) AS rank\_products

FROM gold.fact\_sales f

LEFT JOIN gold.dim\_products p

ON p.product\_key = f.product\_key

GROUP BY p.product\_name



1. --What are the 5 worst-performing products in terms of sales

SELECT Top 5

p.product\_name,

SUM(f.sales\_amount) total\_revenue

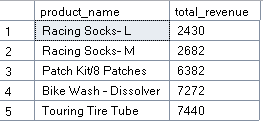
FROM gold.fact\_sales f

LEFT JOIN gold.dim\_products p

ON p.product\_key = f.product\_key

GROUP BY p.product\_name

ORDER BY total\_revenue DESC



1. --Find the top 10 customers who have generated the highest revenue

SELECT TOP 10

c.customer\_key,

c.first\_name,

c.last\_name,

SUM(f.sales\_amount) AS total\_revenue

FROM gold.fact\_sales f

LEFT JOIN gold.dim\_customers c

ON c.customer\_key = f.customer\_key

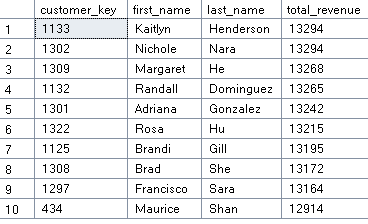
GROUP BY

c.customer\_key,

c.first\_name,

c.last\_name

ORDER BY total\_revenue DESC



1. --The 3 customers with the fewest orders placed

SELECT TOP 3

c.customer\_key,

c.first\_name,

c.last\_name,

COUNT(DISTINCT order\_number) AS total\_orders

FROM gold.fact\_sales f

LEFT JOIN gold.dim\_customers c

ON c.customer\_key = f.customer\_key

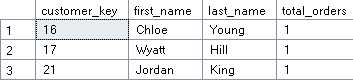
GROUP BY

c.customer\_key,

c.first\_name,

c.last\_name

ORDER BY total\_orders



Step 7 – Change Over Time

Analyze how a measure evolves over time.

Helps track trends and identify seasonality in your data.

1. --Analyze Sales Performance Change Over Time

SELECT

DATETRUNC(month,order\_date) AS order\_year,

SUM(sales\_amount) AS total\_sales,

COUNT(DISTINCT customer\_key) AS total\_customers,

SUM(quantity) as total\_quantity

From gold.fact\_sales

WHERE order\_date IS NOT NULL

GROUP BY DATETRUNC(month,order\_date)

ORDER BY DATETRUNC(month,order\_date)



Step 8 – Cumulative Analysis

Aggregate the data progressively over time

Helps to understand whether our business is growing or declining.

1. --Calculate the total sales per year

SELECT

order\_date,

total\_sales,

SUM(total\_sales) OVER (ORDER BY order\_date) AS running\_total\_sales,

SUM(avg\_price) OVER (ORDER BY order\_date) AS moving\_average\_price

FROM

(

SELECT

DATETRUNC(year, order\_date) AS order\_date,

SUM(sales\_amount) AS total\_sales,

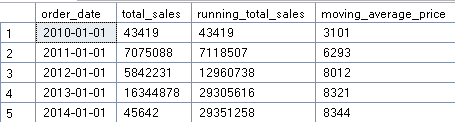
AVG(price) AS avg\_price

FROM gold.fact\_sales

WHERE order\_date IS NOT NULL

Group BY DATETRUNC(year,order\_date)

) t



Step 9. Performance Analysis

Comparing the current value to the target value.

Help measure success and compare performance.

1. --Analyze the yearly performance of products by comparing each product's sale to both its average sales performance and the previous year's sales.

WITH yearly\_product\_sales AS (

SELECT

YEAR(f.order\_date) AS order\_year,

p.product\_name,

SUM(f.sales\_amount) AS current\_sales

FROM gold.fact\_sales f

LEFT JOIN gold.dim\_products p

ON f.product\_key = p.product\_key

WHERE f.order\_date IS NOT NULL

GROUP BY

YEAR(f.order\_date),

p.product\_name

)

SELECT

order\_year,

product\_name,

current\_sales,

AVG(current\_sales) OVER (PARTITION BY product\_name) avg\_sales,

current\_sales - AVG(current\_sales) OVER (PARTITION BY product\_name) AS diff\_avg,

CASE WHEN current\_sales - AVG(current\_sales) OVER (PARTITION BY product\_name) > 0 THEN 'Above Avg'

WHEN current\_sales - AVG(current\_sales) OVER (PARTITION BY product\_name) < 0 THEN 'Below Avg'

ELSE 'Avg'

END avg\_change,

LAG(current\_sales) OVER (PARTITION BY product\_name ORDER BY order\_year) py\_sales,

current\_sales - LAG(current\_sales) OVER (PARTITION BY product\_name ORDER BY order\_year) AS diff\_py,

CASE WHEN current\_sales - LAG(current\_sales) OVER (PARTITION BY product\_name ORDER BY order\_year) > 0 THEN 'Increase'

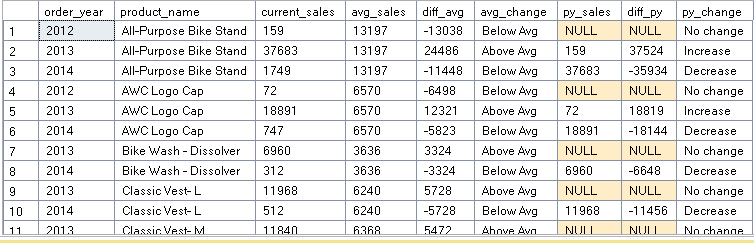
WHEN current\_sales - LAG(current\_sales) OVER (PARTITION BY product\_name ORDER BY order\_year) < 0 THEN 'Decrease'

ELSE 'No change'

END py\_change

FROM yearly\_product\_sales

ORDER BY product\_name, order\_year



Step 10 – Part To Whole Analysis

Anlayze how an individual part is performing compared to the overall.

Allowing us to understand which category has the greatest impact on the business

1. Which categories contribute the most to overall sales

WITH category\_sales AS (

SELECT

category,

SUM(sales\_amount) total\_sales

FROM gold.fact\_sales f

LEFT JOIN gold.dim\_products p

ON

p.product\_key = f.product\_key

GROUP BY category)

SELECT

category,

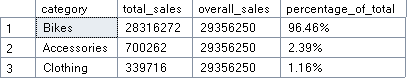
total\_sales,

SUM(total\_sales) OVER() overall\_sales,

CONCAT(ROUND((CAST (total\_sales AS float)/ SUM(total\_sales) OVER())\*100,2), '%') AS percentage\_of\_total

FROM category\_sales

ORDER BY total\_sales DESC



Step 11. Data Segmentation

Group the data base don a specific range.

Helps understand the correlation between two measures.

1. /\*Segment products into cost ranges and count how many products fall into each segment\*/

WITH product\_segments AS(

SELECT

product\_key,

product\_name,

cost,

CASE WHEN cost < 100 THEN 'Below 100'

WHEN cost BETWEEN 100 AND 500 THEN '100-500'

WHEN cost BETWEEN 500 AND 1000 THEN '500-1000'

ELSE 'Above 1000'

END cost\_range

FROM gold.dim\_products)

SELECT

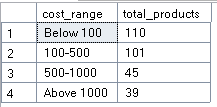
cost\_range,

COUNT(product\_key) AS total\_products

FROM product\_segments

GROUP BY cost\_range

ORDER BY total\_products DESC



Step 5: Reporting

CREATE VIEW gold.report\_customers AS

WITH base\_query AS (

/\*-------------------------------------------------

1) Base Query: Retrieves core columns from tables

---------------------------------------------------\*/

SELECT

f.order\_number,

f.product\_key,

f.order\_date,

f.sales\_amount,

f.quantity,

c.customer\_key,

c.customer\_number,

CONCAT(c.first\_name, ' ', c.last\_name) AS customer\_name,

DATEDIFF(year, c.birthdate, GETDATE()) age

FROM gold.fact\_sales f

LEFT JOIN gold.dim\_customers c

ON c.customer\_key = f.customer\_key

WHERE order\_date IS NOT NULL

)

, customer\_aggregation AS (

/\*-------------------------------------------------------------

2) Customer Aggregations: Summarizes key metrics at the customer level

\*/

SELECT

customer\_key,

customer\_number,

customer\_name,

age,

COUNT(DISTINCT order\_number) AS total\_orders,

SUM(sales\_amount) AS total\_sales,

SUM(quantity) AS total\_quantity,

COUNT(DISTINCT product\_key) AS total\_products,

MAX(order\_date) AS last\_order\_date,

DATEDIFF(month, MIN(order\_date), MAX(order\_date)) AS lifespan

FROM base\_query

GROUP BY

customer\_key,

customer\_number,

customer\_name,

age

)

SELECT

customer\_key,

customer\_number,

customer\_name,

age,

CASE

WHEN age < 20 THEN 'Under 20'

WHEN age BETWEEN 20 AND 29 THEN '20-29'

WHEN age BETWEEN 30 AND 39 THEN '30-39'

WHEN age BETWEEN 40 AND 49 THEN '40-49'

ELSE '50 and above'

END AS age\_group,

CASE

WHEN lifespan >= 12 AND total\_sales > 5000 THEN 'VIP'

WHEN lifespan >= 12 AND total\_sales <= 5000 THEN 'Regular'

ELSE 'New'

END AS customer\_segment,

last\_order\_date,

DATEDIFF(month, last\_order\_date, GETDATE()) AS recency,

total\_orders,

total\_sales,

total\_quantity,

total\_products,

lifespan,

--Compuate average order value (AVO)

CASE WHEN total\_orders = 0 THEN 0

ELSE total\_sales / total\_orders

END AS avg\_order\_value,

--Compuate average monthly spend

CASE WHEN lifespan = 0 THEN total\_sales

ELSE total\_sales / lifespan

END AS avg\_monthly\_spend

FROM customer\_aggregation

/\* Product Report \*/

CREATE VIEW gold.report\_products AS

WITH base\_query AS (

SELECT

f.order\_number,

f.order\_date,

f.customer\_key,

f.sales\_amount,

f.quantity,

p.product\_key,

p.product\_name,

p.category,

p.subcategory,

p.cost

FROM gold.fact\_sales f

LEFT JOIN gold.dim\_products p

ON f.product\_key = p.product\_key

WHERE order\_date IS NOT NULL

),

product\_aggregations AS (

SELECT

product\_key,

product\_name,

category,

subcategory,

cost,

DATEDIFF(MONTH, MIN(order\_date), MAX(order\_date)) AS lifespan,

MAX(order\_date) AS last\_sale\_date,

COUNT(DISTINCT order\_number) AS total\_orders,

COUNT(DISTINCT customer\_key) AS total\_customers,

SUM(sales\_amount) AS total\_sales,

SUM(quantity) AS total\_quantity,

ROUND(AVG(CAST(Sales\_amount AS FLOAT) / NULLIF(quantity, 0)), 1) AS avg\_selling\_price

FROM base\_query

GROUP BY

product\_key,

product\_name,

category,

subcategory,

cost

)

SELECT

product\_key,

product\_name,

category,

subcategory,

cost,

last\_sale\_date,

DATEDIFF(MONTH, last\_sale\_date, GETDATE()) AS recency\_in\_months,

CASE

WHEN total\_sales > 50000 THEN 'High-Performance'

WHEN total\_sales >= 10000 THEN 'Mid-Range'

ELSE 'Low-Performance'

END AS product\_segment,

lifespan,

total\_orders,

total\_sales,

total\_quantity,

total\_customers,

avg\_selling\_price,

CASE

WHEN total\_orders = 0 THEN 0

ELSE total\_sales / lifespan

END AS avg\_monthly\_revenue

FROM product\_aggregations