```
-- Advanced Data Analytics Project
-- Changes Over Months
Select Month(order_date) as order_Month,
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 Month(order_date)
order by Month(order_date);


-- Changes Over Years and Months
Select Datetrunc(month, order_date) as order_Date,
    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);

    -- Cumulative Analysis
    -- calculate the total sales per month
    -- and running total sales over the time
select
order_date,
total_sales,
    sum(total_sales) over (partition by order_date order by order_date ) as
      running_total_sales,
    avg(avg_price)  over(order  by  order_date  )  as  Moving_avg_price
    from (

            select DATETRUNC(month, 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(month, order_date)
        ) t

-- Performance Analysis

-- Analyze the yearly performance of products by comparing their sales
-- to the both the average sales performance of the product 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 order_date is not null
    group by year(f.order_date), product_name
    )

SELECT order_year, product_name, current_sales, avg(current_sales) over( partition by
product_name) as 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) diff_py,
     Case when lag(current_sales) over(partition by product_name order by
       order_year) > 0 then 'Increase'
     when 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;


-- Part to whole Analysis
-- Which categories contribute the most overall sales?
With category_sales as(
        select
        category,
        sum(sales_amount) as total_sales
        from gold.fact_sales f
        left join gold.dim_products p
        on f.product_key = p.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;



        -- Data segmentation
        /* 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 as 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;


 /* Group customers into three segmens based on their spending behavior
  -VIP: at least 12 month of history but spending 5000 euro.
 -Regular: at least 12 month of history but spending 5000 euro or less.
- New: lifespan less than 12 month.
and also find the total number of customers of by each group */
with customer_spending as(
                select
                c.customer_key,
                sum(f.sales_amount) as total_spending,
                min(order_date) as first_order,
                max(order_date) as last_order,
                datediff (month, min(order_date),max(order_date)) as lifespan
                from gold.fact_sales f
                left join gold.dim_customers c
                on f.customer_key = c.customer_key
                group by c.customer_key
           )
     select
customer_segment,
count(customer_key) as total_customers
  from (
        select
        customer_key,
        case when lifespan >= 12 and total_spending > 5000 then 'VIP'
             when lifespan >= 12 and total_spending <= 5000 then 'Regular'
             else 'New customer'
end as customer_segment
        from customer_spending) t
        group by customer_segment
        order by total_customers desc;

/*
===============================================================================
Customer Report
===============================================================================
Purpose:
        - This report consolidates key customer metrics and behaviors

Highlights:
        1. Gathers essential fields such as names, ages, and transaction details. 2.
        Segments customers into categories (VIP, Regular, New) and age groups. 3.
        Aggregates customer-level metrics:
        - total orders
        - total sales
        - total quantity purchased
        - total products
        - lifespan (in months)
        4. Calculates valuable KPIs:
        - recency (months since last order)
        - average order value
        - average monthly spend
===============================================================================
*/

-- =============================================================================
-- Create Report: gold.report_customers
-- =============================================================================
IF OBJECT_ID('gold.report_customers', 'V') IS NOT NULL
        DROP VIEW gold.report_customers;
GO

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_numb
er,
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_sales = 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
===============================================================================
Purpose:
      - This report consolidates key product metrics and behaviors.

Highlights:
      1. Gathers essential fields such as product name, category, subcategory, and
         cost.
      2. Segments products by revenue to identify High-Performers, Mid-Range, or Low-
         Performers.
      3. Aggregates product-level metrics:
      - total orders
      - total sales
      - total quantity sold
      - total customers (unique)
      - lifespan (in months)
      4. Calculates valuable KPIs:
          - recency (months since last sale)
          - average order revenue (AOR)
          - average monthly revenue
===============================================================================
*/
```

```
-- =============================================================================
-- Create Report: gold.report_products
-- =============================================================================
IF OBJECT_ID('gold.report_products', 'V') IS NOT NULL
    DROP VIEW gold.report_products;
GO

CREATE VIEW gold.report_products AS

WITH base_query AS (
/*---------------------------------------------------------------------------
1) Base Query: Retrieves core columns from fact_sales and dim_products
---------------------------------------------------------------------------*/
    SELECT
        f.order_number
        ,   f.order_date,
        f.customer_key,
        f.sales_amount,
        f.quantity,
        p.product_key,
        p.product_nam
        e,     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-- only consider valid sales dates
),

product_aggregations AS (
/*---------------------------------------------------------------------------
2) Product Aggregations: Summarizes key metrics at the product level
---------------------------------------------------------------------------*/
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
)

/*---------------------------------------------------------------------------
   3) Final Query: Combines all product results into one output
---------------------------------------------------------------------------*/
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-Performer'
		WHEN total_sales >= 10000 THEN 'Mid-Range'
		ELSE 'Low-Performer'
	END AS product_segment,
	lifespan,
	total_orders,
	total_sales,
	total_quantity,
	total_customers,
	avg_selling_price,
	-- Average Order Revenue (AOR)
	CASE
		WHEN total_orders = 0 THEN 0
		ELSE total_sales / total_orders
	END AS avg_order_revenue,

	-- Average Monthly Revenue
	CASE
		WHEN lifespan = 0 THEN total_sales
		ELSE total_sales / lifespan
	END AS avg_monthly_revenue

FROM product_aggregations
```