
SQL
DATA
ANALYTICS
PROJECT
setting
config
connecting
connecting

Advanced Data Analytics

Answer Business Problems:

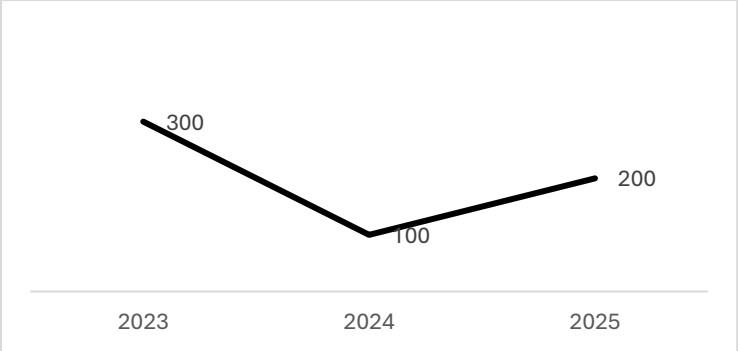
- **Complex Queries**
- **Window Functions**
- **Subqueries**
- **CTE**
- **Reports**

Change Over Time, Trends

$\sum[measure]\ by\ [Date\ Dimension]$

- Total sales by year
- Average cost by year

Year	Sales
2023	300
2024	100
2025	200



Problem: Analyze sales performance over time.

```
create database data_warehouse;          -- create a database
use data_warehouse;                      -- use database

select*from customers;
select*from products;
select*from sales;
select*from report_customers;
select*from report_products;

SELECT
year(order_date) as order_year,
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 sales
WHERE order_date IS NOT NULL
group by year(order_date), month(order_date)
ORDER BY year(order_date),month(order_date)
LIMIT 184467440 OFFSET 1;        -- here we have to give maximum limits
```

order_year	order_month	total_sales	total_customers	total_quantity
2010	12	43419	14	14
2011	1	469795	144	144
2011	2	466307	144	144
2011	3	485165	150	150
2011	4	502042	157	157
2011	5	561647	174	174
2011	6	737793	230	230
2011	7	596710	188	188
2011	8	614516	193	193
2011	9	603047	185	185
2011	10	708164	221	221
2011	11	660507	208	208
2011	12	669395	222	222
2012	1	495363	252	252
2012	2	506992	260	260
2012	3	373478	212	212
2012	4	400324	219	219

order_year	order_month	total_sales	total_customers	total_quantity
2012	5	358866	207	207
2012	6	555142	318	318
2012	7	444533	246	246
2012	8	523887	294	294
2012	9	486149	269	269
2012	10	535125	313	313
2012	11	537918	324	324
2012	12	624454	354	483
2013	1	857758	627	1677
2013	2	771218	1373	3454
2013	3	1049732	1631	4087
2013	4	1045860	1564	3979
2013	5	1284456	1719	4400
2013	6	1642948	1948	5025
2013	7	1371595	1796	4673
2013	8	1545910	1898	4848
2013	9	1447324	1832	4616
2013	10	1673261	2073	5304

```
SELECT
  DATE_FORMAT(order_date, '%Y-%m-01') AS order_date,     -- date formatting
  SUM(sales_amount) AS total_sales,
  COUNT(DISTINCT customer_key) AS total_customers,
  SUM(quantity) AS total_quantity
FROM sales
WHERE order_date IS NOT NULL
GROUP BY DATE_FORMAT(order_date, '%Y-%m-01')
ORDER BY DATE_FORMAT(order_date, '%Y-%m-01')
LIMIT 18446744073 OFFSET 1;
```

order_date	total_sales	total_customers	total_quantity
2010-12-01	43419	14	14
2011-01-01	469795	144	144
2011-02-01	466307	144	144
2011-03-01	485165	150	150
2011-04-01	502042	157	157
2011-05-01	561647	174	174
2011-06-01	737793	230	230
2011-07-01	596710	188	188
2011-08-01	614516	193	193
2011-09-01	603047	185	185
2011-10-01	708164	221	221
2011-11-01	660507	208	208
2011-12-01	669395	222	222
2012-01-01	495363	252	252
2012-02-01	506992	260	260
2012-03-01	373478	212	212
2012-04-01	400324	219	219

order_date	total_sales	total_customers	total_quantity
2012-05-01	358866	207	207
2012-06-01	555142	318	318
2012-07-01	444533	246	246
2012-08-01	523887	294	294
2012-09-01	486149	269	269
2012-10-01	535125	313	313
2012-11-01	537918	324	324
2012-12-01	624454	354	483
2013-01-01	857758	627	1677
2013-02-01	771218	1373	3454
2013-03-01	1049732	1631	4087
2013-04-01	1045860	1564	3979
2013-05-01	1284456	1719	4400
2013-06-01	1642948	1948	5025
2013-07-01	1371595	1796	4673
2013-08-01	1545910	1898	4848
2013-09-01	1447324	1832	4616
2013-10-01	1673261	2073	5304

Cumulative Analysis

$$\sum [\,Cumulative\ Measure] \ by\ [Date\ Dimension]$$

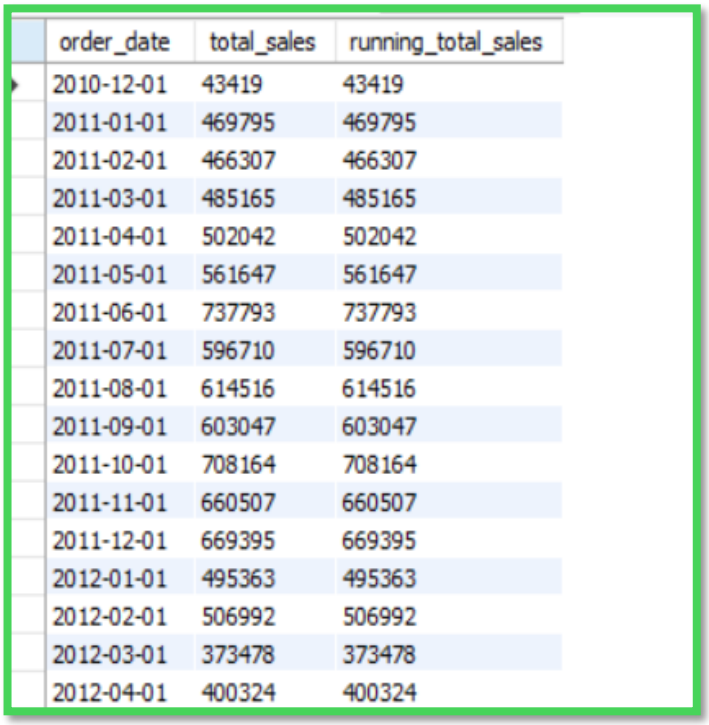
- Running total sales by year
- Moving the average sales by month

Cumulative value

2024	300	300
2025	100	400
2026	200	600

Problem: Calculate the total sales per month and the running total of overtime sales.

```
SELECT
  order_date,
  total_sales,
  SUM(total_sales) OVER (partition by order_date order by order_date) AS
running_total_sales        --window fcn
FROM
(
  SELECT
    DATE_FORMAT(order_date, '%Y-%m-01') AS order_date,
    SUM(sales_amount) AS total_sales
  FROM sales
  WHERE order_date IS NOT NULL
  GROUP BY DATE_FORMAT(order_date, '%Y-%m-01')
  order by total_sales
  LIMIT 184467440 OFFSET 1
) AS monthly_sales
ORDER BY order_date;
```



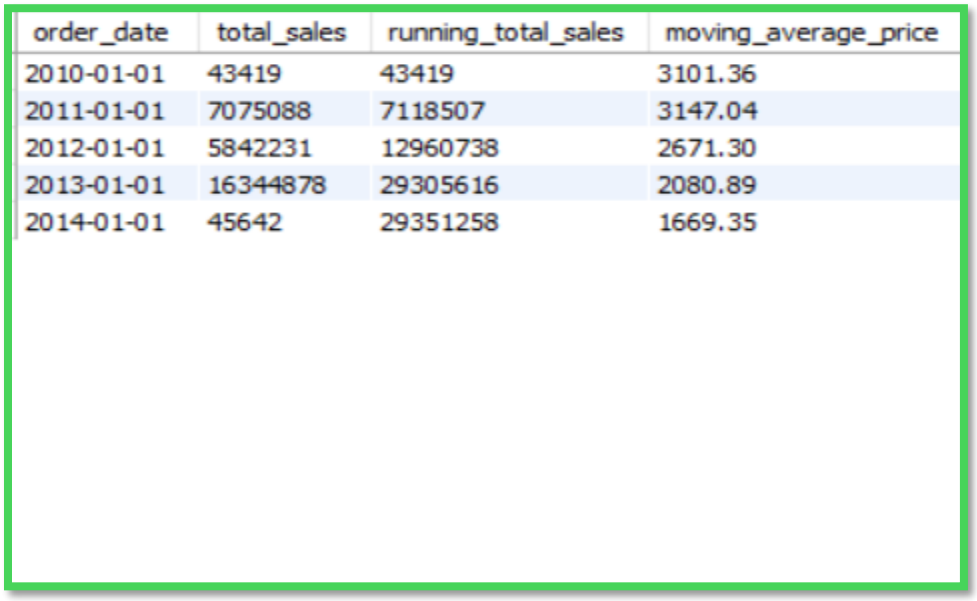
order_date	total_sales	running_total_sales
2010-12-01	43419	43419
2011-01-01	469795	469795
2011-02-01	466307	466307
2011-03-01	485165	485165
2011-04-01	502042	502042
2011-05-01	561647	561647
2011-06-01	737793	737793
2011-07-01	596710	596710
2011-08-01	614516	614516
2011-09-01	603047	603047
2011-10-01	708164	708164
2011-11-01	660507	660507
2011-12-01	669395	669395
2012-01-01	495363	495363
2012-02-01	506992	506992
2012-03-01	373478	373478
2012-04-01	400324	400324

order_date	total_sales	running_total_sales
2012-05-01	358866	358866
2012-06-01	555142	555142
2012-07-01	444533	444533
2012-08-01	523887	523887
2012-09-01	486149	486149
2012-10-01	535125	535125
2012-11-01	537918	537918
2012-12-01	624454	624454
2013-01-01	857758	857758
2013-02-01	771218	771218
2013-03-01	1049732	1049732
2013-04-01	1045860	1045860
2013-05-01	1284456	1284456
2013-06-01	1642948	1642948
2013-07-01	1371595	1371595
2013-08-01	1545910	1545910
2013-09-01	1447324	1447324

```
SELECT
  STR_TO_DATE(CONCAT(order_year, '-01-01'), '%Y-%m-%d') AS order_date,
  total_sales,
  SUM(total_sales) OVER (ORDER BY order_year) AS running_total_sales,
  ROUND(AVG(avg_price) OVER (ORDER BY order_year), 2) AS
moving_average_price
FROM (
 SELECT
   YEAR(order_date) AS order_year,
    SUM(sales_amount) AS total_sales,
    avg(price) avg_price
```

```
  FROM sales
  WHERE order_date IS NOT NULL
  GROUP BY YEAR(order_date)
  ORDER BY YEAR(order_date)
  LIMIT 184467440737 OFFSET 1

) AS yearly_sales
ORDER BY order_year;
```



order_date	total_sales	running_total_sales	moving_average_price
2010-01-01	43419	43419	3101.36
2011-01-01	7075088	7118507	3147.04
2012-01-01	5842231	12960738	2671.30
2013-01-01	16344878	29305616	2080.89
2014-01-01	45642	29351258	1669.35

Performance Analysis

$$Current\ [Measure] - Target\ [measure]$$

- Current sales – Average sales
- Current year sales – Previous year sales
- Current sales – Lowest sales

	Current	Target	Performance
A	**400**	**400**	**0**
B	**600**	**400**	**200**
C	**200**	**400**	**-200**

Problem: Analyze the yearly performance of each product by comparing its sales to both its average annual sales and the previous year's sales.

```
WITH yearly_product_sales AS (                    -- starting with CTE
  SELECT
    YEAR(s.order_date) AS order_year,
    p.product_name,
    SUM(s.sales_amount) AS current_sales
  FROM sales s
  LEFT JOIN products p ON s.product_key = p.product_key
  WHERE s.order_date IS NOT NULL
  GROUP BY YEAR(s.order_date), p.product_name
)
SELECT
  order_year, product_name, current_sales,
  ROUND(AVG(current_sales) OVER (PARTITION BY product_name)) AS
avg_sales,
  current_sales -  ROUND(AVG(current_sales) OVER (PARTITION BY
product_name)) AS diff_avg,
case when  current_sales -  ROUND(AVG(current_sales) OVER (PARTITION BY
product_name)) > 0 then "above Avg"
when  current_sales -  ROUND(AVG(current_sales) OVER (PARTITION BY
product_name)) < 0 then "below Avg"
    else "Avg" end avg_change,
```

Joined sales with products on product_key using LEFT JOIN to retain all sales data.

```
-- year over year analysis

LAG(current_sales) over (partition by product_name order by order_year) prev_sales,

current_sales - LAG(current_sales) over (partition by product_name order by order_year) as diff_prev,

case
when  current_sales -  ROUND( LAG(current_sales) over (partition by product_name order by order_year)) > 0 then "Increase"

when  current_sales -  ROUND( LAG(current_sales) over (partition by product_name order by order_year)) < 0 then "Decrease"

   else "No change"

   end prev_change

FROM yearly_product_sales

WHERE order_year IS NOT NULL;
```

order_year	product_name	current_sales	avg_sales	diff_avg	avg_change	prev_sales	diff_prev	prev_change
2012	All-Purpose Bike Stand	159	13197	-13038	below Avg	NULL	NULL	No change
2013	All-Purpose Bike Stand	37683	13197	24486	above Avg	159	37524	Increase
2014	All-Purpose Bike Stand	1749	13197	-11448	below Avg	37683	-35934	Decrease
2012	AWC Logo Cap	72	6570	-6498	below Avg	NULL	NULL	No change
2013	AWC Logo Cap	18891	6570	12321	above Avg	72	18819	Increase
2014	AWC Logo Cap	747	6570	-5823	below Avg	18891	-18144	Decrease
2013	Bike Wash - Dissolver	6960	3636	3324	above Avg	NULL	NULL	No change
2014	Bike Wash - Dissolver	312	3636	-3324	below Avg	6960	-6648	Decrease
2013	Classic Vest- L	11968	6240	5728	above Avg	NULL	NULL	No change
2014	Classic Vest- L	512	6240	-5728	below Avg	11968	-11456	Decrease
2013	Classic Vest- M	11840	6368	5472	above Avg	NULL	NULL	No change
2014	Classic Vest- M	896	6368	-5472	below Avg	11840	-10944	Decrease
2012	Classic Vest- S	64	3648	-3584	below Avg	NULL	NULL	No change
2013	Classic Vest- S	10368	3648	6720	above Avg	64	10304	Increase
2014	Classic Vest- S	512	3648	-3136	below Avg	10368	-9856	Decrease
2012	Fender Set - Mountain	110	15554	-15444	below Avg	NULL	NULL	No change
2013	Fender Set - Mountain	44484	15554	28930	above Avg	110	44374	Increase

Proportional Analysis or Part to whole

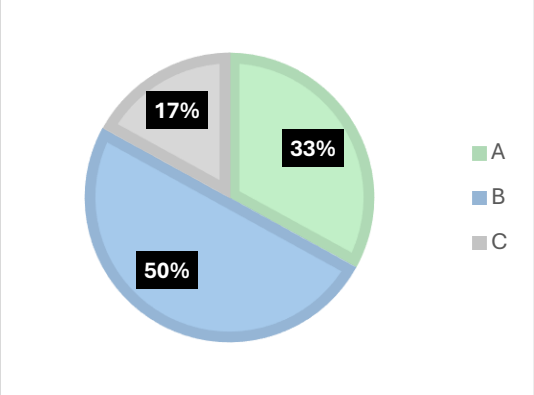
Assess the performance of individual components in relation to the overall business, enabling us to identify which category contributes most significantly to overall impact.

$$([Measure]/Total\ [Measure]) * 100\ \ by\ [Dimension]$$

$$(Sales/Total\ sales) * 100\ \ by\ [Dimension]$$

$$(Quantity/Total\ Quantity) * 100\ \ by\ [Country]$$

A	**200**	**33%**
B	**300**	**50%**
C	**100**	**17%**



Problem: Which categories contribute the most to overall sales?

```
with category_sales as (
select p.category,
sum(sales_amount) as total_sales from sales s
left join products p
on p.product_key= s.product_key group by category order by total_sales desc)

select category, total_sales,
sum(total_sales) over() as overall_sales ,
round((total_sales/ sum(total_sales) over ())*100,2) as  "total(%)"
from category_sales;
```

category	total_sales	overall_sales	total(%)
Bikes	28316272	29356250	96.46
Accessories	700262	29356250	2.39
Clothing	339716	29356250	1.16

Data Segmentation

[*Measure*] *by* [*Measure*]
[*Total products by sales Range*]
[*Total customers by Age*]

Sum	Categorize
3	**50**
4	**100**
5	**150**
1	**200**
10	**250**
5	**300**

Low	7
Medium	6
High	15

Problem: Segment products into cost ranges and count how many products fall into each segment.

```
use data_warehouse;

with product_segment 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 products )
select cost_range,
count(product_key) as total_products from product_segment group by cost_range
order by total_products desc;
```

cost_range	total_products
below 100	110
100-500	101
500-1000	45
above 1000	39

Problem: Group customers into three segments based on their spending behavior:

- VIP: Customers with at least 12 months of history and spending more than €5,000.

- Regular: Customers with at least 12 months of history but spending €5,000 or less.

- New: Customers with a lifespan less than 12 months.

And find the total number of customers by each group

```
with customer_spending as (
SELECT
    c.customer_key,
    SUM(s.sales_amount) AS total_spending,
    MIN(s.order_date) AS first_order,
    MAX(s.order_date) AS last_order,
    TIMESTAMPDIFF(MONTH, MIN(s.order_date), MAX(s.order_date)) AS lifespan
FROM sales s
LEFT JOIN customers c
    ON s.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'
        END AS customer_segment
    FROM customer_spending
) AS t
GROUP BY customer_segment
ORDER BY total_customers DESC;
```

customer_segment	total_customers
New	14830
Regular	2037
VIP	1617

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 expenses

Task:

1. Gathers essential fields such as names, ages, and transaction details (Base Query)

```
WITH base_query as (                    --By CTE
SELECT
   s.order_number, s.product_key, s.order_date, s.sales_amount, s.quantity,
   c.customer_key, c.customer_number,
   CONCAT(c.first_name, ' ', c.last_name) AS customer_name,
   TIMESTAMPDIFF(YEAR, c.birthdate, CURDATE()) AS age
FROM sales s
LEFT JOIN customers c
   ON c.customer_key = s.customer_key
WHERE s.order_date IS NOT NULL)
    select*from base_query;
```

order_number	product_key	order_date	sales_amount	quantity	customer_key	customer_number	customer_name	age
SO54508	244	2013-03-16	35	1	13	AW00011012	Lauren Walker	46
SO54509	284	2013-03-16	35	1	732	AW00011731	Tanya Gill	59
SO54509	289	2013-03-16	5	1	732	AW00011731	Tanya Gill	59
SO54509	174	2013-03-16	22	1	732	AW00011731	Tanya Gill	59
SO54509	234	2013-03-16	24	1	732	AW00011731	Tanya Gill	59
SO54510	284	2013-03-16	35	1	3354	AW00014353	Erin Reed	44
SO54510	289	2013-03-16	5	1	3354	AW00014353	Erin Reed	44
SO54510	246	2013-03-16	35	1	3354	AW00014353	Erin Reed	44
SO54511	105	2013-03-16	9	1	348	AW00011347	Roy Navarro	52
SO54511	295	2013-03-16	5	1	348	AW00011347	Roy Navarro	52
SO54513	136	2013-03-16	1701	1	3084	AW00014083	Nathan Lopez	39
SO54513	251	2013-03-16	50	1	3084	AW00014083	Nathan Lopez	39
SO54514	117	2013-03-16	769	1	3104	AW00014103	Monique Blanco	76
SO54514	104	2013-03-16	10	1	3104	AW00014103	Monique Blanco	76
SO54514	295	2013-03-16	5	1	3104	AW00014103	Monique Blanco	76
SO54519	121	2013-03-16	2295	1	98	AW00011097	Edwin Nara	52
SO54519	104	2013-03-16	10	1	98	AW00011097	Edwin Nara	52
SO54524	165	2013-03-16	2384	1	1631	AW00012630	Kevin Simmons	65

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

```
WITH base_query as (
SELECT
    s.order_number,
    s.product_key,
    s.order_date,
    s.sales_amount,
    s.quantity,
    c.customer_key,
    c.customer_number,
    CONCAT(c.first_name, ' ', c.last_name) AS customer_name,
    TIMESTAMPDIFF(YEAR, c.birthdate, CURDATE()) AS age
FROM sales s
LEFT JOIN customers c
    ON c.customer_key = s.customer_key
WHERE s.order_date IS NOT NULL)

    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,
    TIMESTAMPDIFF(MONTH, MIN(order_date), MAX(order_date)) AS lifespan
FROM base_query
GROUP BY
    customer_key,
    customer_number,
    customer_name,
    age;
```

customer_key	customer_number	customer_name	age	total_orders	total_sales	total_quantity	total_products	last_order_date	lifespan
1	AW00011000	Jon Yang	53	3	8249	8	8	2013-05-03	27
2	AW00011001	Eugene Huang	49	3	6384	11	10	2013-12-10	34
3	AW00011002	Ruben Torres	54	3	8114	4	4	2013-02-23	25
4	AW00011003	Christy Zhu	51	3	8139	9	9	2013-05-10	28
5	AW00011004	Elizabeth Johnson	45	3	8196	6	6	2013-05-01	27
6	AW00011005	Julio Ruiz	48	3	8121	6	6	2013-05-02	28
7	AW00011006	Janet Alvarez	48	3	8119	5	5	2013-05-14	27
8	AW00011007	Marco Mehta	55	3	8211	8	8	2013-03-19	26
9	AW00011008	Rob Verhoff	49	3	8106	7	7	2013-03-02	25
10	AW00011009	Shannon Carlson	55	3	8091	5	5	2013-05-09	27
11	AW00011010	Jacquelyn Suarez	55	3	8088	4	4	2013-05-23	28
12	AW00011011	Curtis Lu	56	3	8133	4	4	2013-03-19	26
13	AW00011012	Lauren Walker	46	2	81	5	5	2013-10-15	6
14	AW00011013	Ian Jenkins	45	2	114	5	5	2014-01-21	9
15	AW00011014	Sydney Bennett	51	2	138	6	5	2013-04-30	1
16	AW00011015	Chloe Young	40	1	2501	3	3	2013-01-18	0
17	AW00011016	Wyatt Hill	40	1	2332	3	3	2013-02-09	0
18	AW00011017	Shannon Wang	75	3	6434	4	4	2013-10-14	33

3. Segments customers into categories (VIP, Regular, New) and age groups.

```
WITH base_query as (
SELECT
   s.order_number,
   s.product_key,
   s.order_date,
   s.sales_amount,
   s.quantity,
   c.customer_key,
   c.customer_number,
   CONCAT(c.first_name, ' ', c.last_name) AS customer_name,
   TIMESTAMPDIFF(YEAR, c.birthdate, CURDATE()) AS age
FROM sales s
LEFT JOIN customers c
   ON c.customer_key = s.customer_key
WHERE s.order_date IS NOT NULL)
,
customer_aggregation AS (
   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,
    TIMESTAMPDIFF(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,

-- Age group classification
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,
```

```
-- Customer segment classification
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,
total_orders,
total_sales,
total_quantity,
total_products,
lifespan
FROM customer_aggregation;
```

customer_key	customer_number	customer_name	age	age_group	customer_segment	last_order_date	total_orders	total_sales	total_quantity	total_products	lifespan
1	AW00011000	Jon Yang	53	50 and above	VIP	2013-05-03	3	8249	8	8	27
2	AW00011001	Eugene Huang	49	40-49	VIP	2013-12-10	3	6384	11	10	34
3	AW00011002	Ruben Torres	54	50 and above	VIP	2013-02-23	3	8114	4	4	25
4	AW00011003	Christy Zhu	51	50 and above	VIP	2013-05-10	3	8139	9	9	28
5	AW00011004	Elizabeth Johnson	45	40-49	VIP	2013-05-01	3	8196	6	6	27
6	AW00011005	Julio Ruiz	48	40-49	VIP	2013-05-02	3	8121	6	6	28
7	AW00011006	Janet Alvarez	48	40-49	VIP	2013-05-14	3	8119	5	5	27
8	AW00011007	Marco Mehta	55	50 and above	VIP	2013-03-19	3	8211	8	8	26
9	AW00011008	Rob Verhoff	49	40-49	VIP	2013-03-02	3	8106	7	7	25
10	AW00011009	Shannon Carlson	55	50 and above	VIP	2013-05-09	3	8091	5	5	27
11	AW00011010	Jacquelyn Suarez	55	50 and above	VIP	2013-05-23	3	8088	4	4	28
12	AW00011011	Curtis Lu	56	50 and above	VIP	2013-03-19	3	8133	4	4	26
13	AW00011012	Lauren Walker	46	40-49	New	2013-10-15	2	81	5	5	6
14	AW00011013	Ian Jenkins	45	40-49	New	2014-01-21	2	114	5	5	9
15	AW00011014	Sydney Bennett	51	50 and above	New	2013-04-30	2	138	6	5	1
16	AW00011015	Chloe Young	40	40-49	New	2013-01-18	1	2501	3	3	0
17	AW00011016	Wyatt Hill	40	40-49	New	2013-02-09	1	2332	3	3	0
18	AW00011017	Shannon Wang	75	50 and above	VIP	2013-10-14	3	6434	4	4	33

4. Calculates valuable KPIs:

```
CREATE view customers_report as                    -- make the View
WITH base_query as (
SELECT
   s.order_number,
   s.product_key,
   s.order_date,
   s.sales_amount,
   s.quantity,
   c.customer_key,
   c.customer_number,
   CONCAT(c.first_name, ' ', c.last_name) AS customer_name,
   TIMESTAMPDIFF(YEAR, c.birthdate, CURDATE()) AS age
FROM sales s
LEFT JOIN customers c
   ON c.customer_key = s.customer_key
WHERE s.order_date IS NOT NULL)
,
customer_aggregation AS (
   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,
    TIMESTAMPDIFF(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,

  -- Age group classification
  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,
```

```
-- Customer segment classification
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,
TIMESTAMPDIFF(MONTH, last_order_date, CURDATE()) AS recency,
total_orders,
total_sales,
total_quantity,
total_products,
lifespan,

-- Compute average order value (AOV).
CASE
    WHEN total_orders = 0 THEN 0
    ELSE round( total_sales / total_orders , 2)
END AS avg_order_value,

-- Compute average monthly spends.
CASE
    WHEN lifespan = 0 THEN round( total_sales , 2)
    ELSE round( total_sales / lifespan , 2)
END AS avg_monthly_spend
FROM customer_aggregation;
```

customer_key	customer_number	customer_name	age	age_group	customer_segment	last_order_date	recency	total_orders	total_sales	total_quantity	total_products	lifespan	avg_order_value	avg_mo
1	AW00011000	Jon Yang	53	50 and above	VIP	2013-05-03	144	3	8249	8	8	27	2749.67	305.52
2	AW00011001	Eugene Huang	49	40-49	VIP	2013-12-10	137	3	6384	11	10	34	2128.00	187.76
3	AW00011002	Ruben Torres	54	50 and above	VIP	2013-02-23	146	3	8114	4	4	25	2704.67	324.56
4	AW00011003	Christy Zhu	51	50 and above	VIP	2013-05-10	144	3	8139	9	9	28	2713.00	290.68
5	AW00011004	Elizabeth Johnson	45	40-49	VIP	2013-05-01	144	3	8196	6	6	27	2732.00	303.56
6	AW00011005	Julio Ruiz	48	40-49	VIP	2013-05-02	144	3	8121	6	6	28	2707.00	290.04
7	AW00011006	Janet Alvarez	48	40-49	VIP	2013-05-14	144	3	8119	5	5	27	2706.33	300.70
8	AW00011007	Marco Mehta	55	50 and above	VIP	2013-03-19	145	3	8211	8	8	26	2737.00	315.81
9	AW00011008	Rob Verhoff	49	40-49	VIP	2013-03-02	146	3	8106	7	7	25	2702.00	324.24
10	AW00011009	Shannon Carlson	55	50 and above	VIP	2013-05-09	144	3	8091	5	5	27	2697.00	299.67
11	AW00011010	Jacquelyn Suarez	55	50 and above	VIP	2013-05-23	143	3	8088	4	4	28	2696.00	288.86
12	AW00011011	Curtis Lu	56	50 and above	VIP	2013-03-19	145	3	8133	4	4	26	2711.00	312.81
13	AW00011012	Lauren Walker	46	40-49	New	2013-10-15	139	2	81	5	5	6	40.50	13.50
14	AW00011013	Ian Jenkins	45	40-49	New	2014-01-21	135	2	114	5	5	9	57.00	12.67
15	AW00011014	Sydney Bennett	51	50 and above	New	2013-04-30	144	2	138	6	5	1	69.00	138.00
16	AW00011015	Chloe Young	40	40-49	New	2013-01-18	148	1	2501	3	3	0	2501.00	2501
17	AW00011016	Wyatt Hill	40	40-49	New	2013-02-09	147	1	2332	3	3	0	2332.00	2332

-- using view

```
SELECT * FROM data_warehouse.customers_report;          -- it showing whole reports


select age_group,
count(customer_number) as total_customers,
sum(total_sales) total_sales from customers_report
group by age_group;
```

age_group	total_customers	total_sales
50 and above	11801	18789571
40-49	6259	9995330
30-39	424	571349

```
select customer_segment,
count(customer_number) as total_customers,
sum(total_sales) total_sales from customers_report
group by customer_segment;
```

customer_segment	total_customers	total_sales
VIP	1617	10549149
New	14830	11807270
Regular	2037	6999831

Products 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)

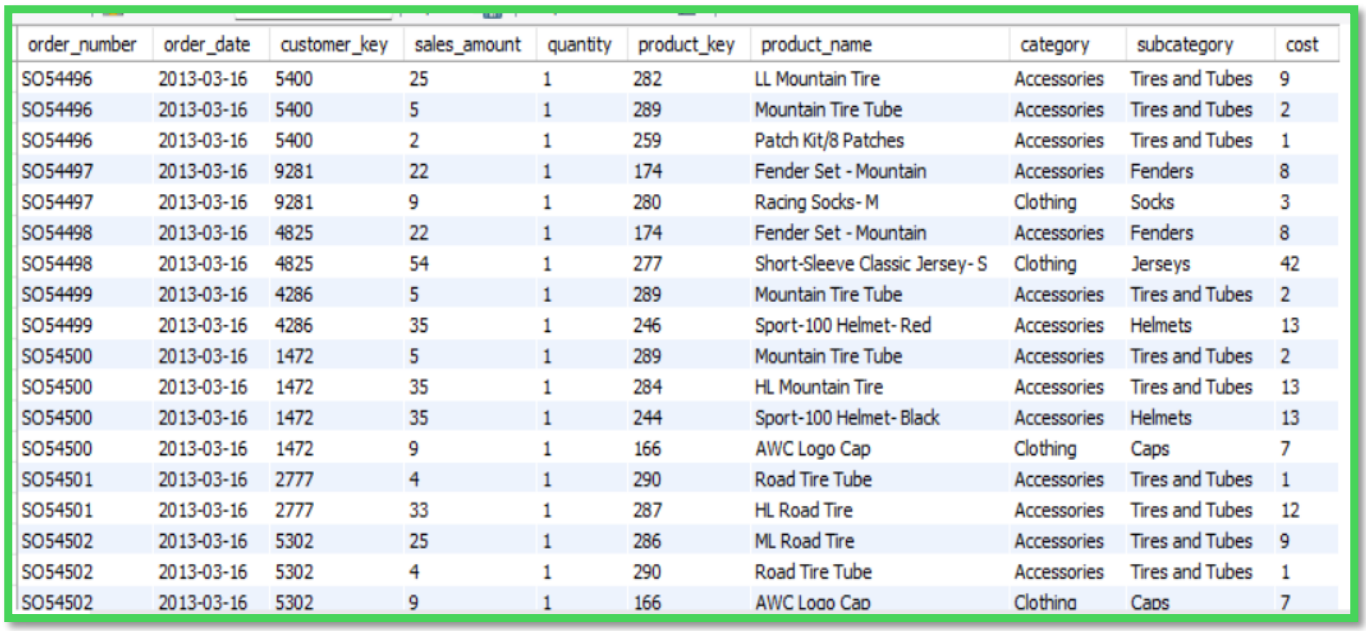
4. Calculates valuable KPIs:

- recency (months since last sale)
- average order revenue (AOR)
- average monthly revenue

Task:

(1) Base Query: Retrieves core columns from sales and products.

```
WITH base_query AS (
  SELECT
      s.order_number,
      s.order_date,
      s.customer_key,
      s.sales_amount,
      s.quantity,
      p.product_key,
      p.product_name,
      p.category,
      p.subcategory,
      p.cost
   FROM sales s
   LEFT JOIN products p
      ON s.product_key = p.product_key
   WHERE s.order_date IS NOT NULL      -- only consider valid sales dates
)
select* from base_query
```



order_number	order_date	customer_key	sales_amount	quantity	product_key	product_name	category	subcategory	cost
SO54496	2013-03-16	5400	25	1	282	LL Mountain Tire	Accessories	Tires and Tubes	9
SO54496	2013-03-16	5400	5	1	289	Mountain Tire Tube	Accessories	Tires and Tubes	2
SO54496	2013-03-16	5400	2	1	259	Patch Kit/8 Patches	Accessories	Tires and Tubes	1
SO54497	2013-03-16	9281	22	1	174	Fender Set - Mountain	Accessories	Fenders	8
SO54497	2013-03-16	9281	9	1	280	Racing Socks- M	Clothing	Socks	3
SO54498	2013-03-16	4825	22	1	174	Fender Set - Mountain	Accessories	Fenders	8
SO54498	2013-03-16	4825	54	1	277	Short-Sleeve Classic Jersey- S	Clothing	Jerseys	42
SO54499	2013-03-16	4286	5	1	289	Mountain Tire Tube	Accessories	Tires and Tubes	2
SO54499	2013-03-16	4286	35	1	246	Sport-100 Helmet- Red	Accessories	Helmets	13
SO54500	2013-03-16	1472	5	1	289	Mountain Tire Tube	Accessories	Tires and Tubes	2
SO54500	2013-03-16	1472	35	1	284	HL Mountain Tire	Accessories	Tires and Tubes	13
SO54500	2013-03-16	1472	35	1	244	Sport-100 Helmet- Black	Accessories	Helmets	13
SO54500	2013-03-16	1472	9	1	166	AWC Logo Cap	Clothing	Caps	7
SO54501	2013-03-16	2777	4	1	290	Road Tire Tube	Accessories	Tires and Tubes	1
SO54501	2013-03-16	2777	33	1	287	HL Road Tire	Accessories	Tires and Tubes	12
SO54502	2013-03-16	5302	25	1	286	ML Road Tire	Accessories	Tires and Tubes	9
SO54502	2013-03-16	5302	4	1	290	Road Tire Tube	Accessories	Tires and Tubes	1
SO54502	2013-03-16	5302	9	1	166	AWC Logo Cap	Clothing	Caps	7

2. Segments Products by revenue to identify High-Performers, Mid-Range, or Low-Performers.

```
WITH base_query AS (
  SELECT
      s.order_number,
      s.order_date,
      s.customer_key,
      s.sales_amount,
      s.quantity,
```

```
        p.product_key,
        p.product_name,
        p.category,
        p.subcategory,
        p.cost
    FROM sales s
    LEFT JOIN products p
        ON s.product_key = p.product_key
    WHERE s.order_date IS NOT NULL        -- only consider valid sales dates
)
,
product_aggregations AS (
```

-- Product Aggregations: Summarizes key metrics at the product level.

```
    SELECT
        product_key,
        product_name,
        category,
        subcategory,
        cost,
        TIMESTAMPDIFF(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(CASE WHEN quantity != 0 THEN sales_amount / quantity
    ELSE  NULL END), 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,
  TIMESTAMPDIFF(MONTH, last_sale_date, CURDATE()) AS recency_in_months,
-- Product Segment based on sales
  CASE
    WHEN total_sales > 50000 THEN 'High-Performer'
    WHEN total_sales >= 10000 THEN 'Mid-Range'
    ELSE 'Low-Performer'
  END AS product_segment
  from product_aggregations;
```

product_key	product_name	category	subcategory	cost	last_sale_date	recency_in_months	product_segment
3	Mountain-100 Black- 38	Bikes	Mountain Bikes	1898	2011-12-27	160	High-Performer
4	Mountain-100 Black- 42	Bikes	Mountain Bikes	1898	2011-12-27	160	High-Performer
5	Mountain-100 Black- 44	Bikes	Mountain Bikes	1898	2011-12-21	161	High-Performer
6	Mountain-100 Black- 48	Bikes	Mountain Bikes	1898	2011-12-26	160	High-Performer
7	Mountain-100 Silver- 38	Bikes	Mountain Bikes	1912	2011-12-22	160	High-Performer
8	Mountain-100 Silver- 42	Bikes	Mountain Bikes	1912	2011-12-28	160	High-Performer
9	Mountain-100 Silver- 44	Bikes	Mountain Bikes	1912	2011-12-12	161	High-Performer
10	Mountain-100 Silver- 48	Bikes	Mountain Bikes	1912	2011-12-23	160	High-Performer
16	Road-150 Red- 44	Bikes	Road Bikes	2171	2011-12-28	160	High-Performer
17	Road-150 Red- 48	Bikes	Road Bikes	2171	2011-12-28	160	High-Performer
18	Road-150 Red- 52	Bikes	Road Bikes	2171	2011-12-27	160	High-Performer
19	Road-150 Red- 56	Bikes	Road Bikes	2171	2011-12-27	160	High-Performer
20	Road-150 Red- 62	Bikes	Road Bikes	2171	2011-12-28	160	High-Performer
36	Road-650 Black- 44	Bikes	Road Bikes	487	2012-12-26	148	Mid-Range
37	Road-650 Black- 48	Bikes	Road Bikes	487	2012-12-25	148	Mid-Range
38	Road-650 Black- 52	Bikes	Road Bikes	487	2012-12-19	149	High-Performer
39	Road-650 Black- 58	Bikes	Road Bikes	487	2012-12-18	149	High-Performer
40	Road-650 Black- 60	Bikes	Road Bikes	487	2012-12-12	149	High-Performer

3. Aggregates product-level metrics: total orders, total sales , total quantity sold, total customers (unique),lifespan (in months).

```
WITH base_query AS (
  SELECT
      s.order_number,
     s.order_date,
      s.customer_key,
      s.sales_amount,
      s.quantity,
      p.product_key,
      p.product_name,
      p.category,
```

```
    p.subcategory,
    p.cost
  FROM sales s
  LEFT JOIN products p
    ON s.product_key = p.product_key
  WHERE s.order_date IS NOT NULL      -- only consider valid sales dates
)
,
product_aggregations AS (
  SELECT
    product_key,
    product_name,
    category,
    subcategory,
    cost,
    TIMESTAMPDIFF(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(CASE WHEN quantity != 0 THEN sales_amount / quantity ELSE
NULL END), 1) AS avg_selling_price
  FROM base_query
  GROUP BY
    product_key,
```

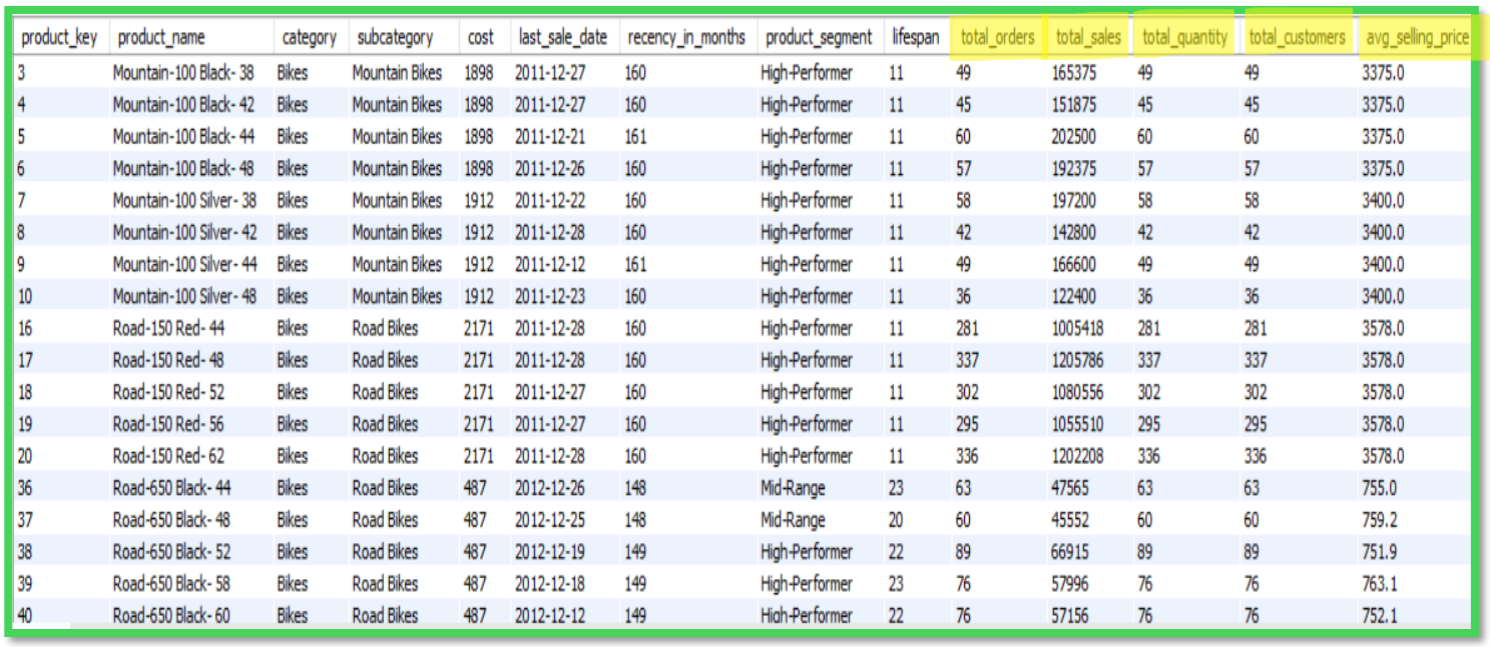
```
    product_name,
    category,
    subcategory,
    cost
)

SELECT
    product_key,
    product_name,
    category,
    subcategory,
    cost,
    last_sale_date,
    TIMESTAMPDIFF(MONTH, last_sale_date, CURDATE()) AS recency_in_months,

    -- Product Segment based on sales.
    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
from product_aggregations;
```



product_key	product_name	category	subcategory	cost	last_sale_date	recency_in_months	product_segment	lifespan	total_orders	total_sales	total_quantity	total_customers	avg_selling_price
3	Mountain-100 Black- 38	Bikes	Mountain Bikes	1898	2011-12-27	160	High-Performer	11	49	165375	49	49	3375.0
4	Mountain-100 Black- 42	Bikes	Mountain Bikes	1898	2011-12-27	160	High-Performer	11	45	151875	45	45	3375.0
5	Mountain-100 Black- 44	Bikes	Mountain Bikes	1898	2011-12-21	161	High-Performer	11	60	202500	60	60	3375.0
6	Mountain-100 Black- 48	Bikes	Mountain Bikes	1898	2011-12-26	160	High-Performer	11	57	192375	57	57	3375.0
7	Mountain-100 Silver- 38	Bikes	Mountain Bikes	1912	2011-12-22	160	High-Performer	11	58	197200	58	58	3400.0
8	Mountain-100 Silver- 42	Bikes	Mountain Bikes	1912	2011-12-28	160	High-Performer	11	42	142800	42	42	3400.0
9	Mountain-100 Silver- 44	Bikes	Mountain Bikes	1912	2011-12-12	161	High-Performer	11	49	166600	49	49	3400.0
10	Mountain-100 Silver- 48	Bikes	Mountain Bikes	1912	2011-12-23	160	High-Performer	11	36	122400	36	36	3400.0
16	Road-150 Red- 44	Bikes	Road Bikes	2171	2011-12-28	160	High-Performer	11	281	1005418	281	281	3578.0
17	Road-150 Red- 48	Bikes	Road Bikes	2171	2011-12-28	160	High-Performer	11	337	1205786	337	337	3578.0
18	Road-150 Red- 52	Bikes	Road Bikes	2171	2011-12-27	160	High-Performer	11	302	1080556	302	302	3578.0
19	Road-150 Red- 56	Bikes	Road Bikes	2171	2011-12-27	160	High-Performer	11	295	1055510	295	295	3578.0
20	Road-150 Red- 62	Bikes	Road Bikes	2171	2011-12-28	160	High-Performer	11	336	1202208	336	336	3578.0
36	Road-650 Black- 44	Bikes	Road Bikes	487	2012-12-26	148	Mid-Range	23	63	47565	63	63	755.0
37	Road-650 Black- 48	Bikes	Road Bikes	487	2012-12-25	148	Mid-Range	20	60	45552	60	60	759.2
38	Road-650 Black- 52	Bikes	Road Bikes	487	2012-12-19	149	High-Performer	22	89	66915	89	89	751.9
39	Road-650 Black- 58	Bikes	Road Bikes	487	2012-12-18	149	High-Performer	23	76	57996	76	76	763.1
40	Road-650 Black- 60	Bikes	Road Bikes	487	2012-12-12	149	High-Performer	22	76	57156	76	76	752.1

4. Calculates valuable KPIs: recency (months since last sale), average order revenue (AOR), average monthly revenue.

```
CREATE VIEW products_report AS          -- create view
WITH base_query AS (
    SELECT
        s.order_number,
        s.order_date,
        s.customer_key,
        s.sales_amount,
```

```
        s.quantity,
        p.product_key,
        p.product_name,
        p.category,
        p.subcategory,
        p.cost
    FROM sales s
    LEFT JOIN products p
        ON s.product_key = p.product_key
    WHERE s.order_date IS NOT NULL
),
product_aggregations AS (
    SELECT
        product_key,
        product_name,
        category,
        subcategory,
        cost,
        TIMESTAMPDIFF(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(CASE WHEN quantity != 0 THEN sales_amount / quantity ELSE
NULL END), 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,
    TIMESTAMPDIFF(MONTH, last_sale_date, CURDATE()) AS recency_in_months,

    -- Product Segment based on sales
    CASE
        WHEN total_sales > 50000 THEN 'High-Performer'
        WHEN total_sales >= 10000 THEN 'Mid-Range'
        ELSE 'Low-Performer'
    END AS product_segment,

    -- Average Order Revenue (AOR)
    CASE
        WHEN total_orders = 0 THEN 0
```

```
    ELSE round(total_sales / total_orders,2)
END AS avg_order_revenue,

-- Average Monthly Revenue
CASE
    WHEN lifespan = 0 THEN round(total_sales,2)
    ELSE round( total_sales / lifespan,2)
    END AS avg_monthly_revenue
FROM product_aggregations;

SELECT * FROM data_warehouse.products_report;          -- view
```

product_key	product_name	category	subcategory	cost	last_sale_date	recency_in_months	product_segment	avg_order_revenue	avg_monthly_revenue
3	Mountain-100 Black- 38	Bikes	Mountain Bikes	1898	2011-12-27	160	High-Performer	3375.00	15034.09
4	Mountain-100 Black- 42	Bikes	Mountain Bikes	1898	2011-12-27	160	High-Performer	3375.00	13806.82
5	Mountain-100 Black- 44	Bikes	Mountain Bikes	1898	2011-12-21	161	High-Performer	3375.00	18409.09
6	Mountain-100 Black- 48	Bikes	Mountain Bikes	1898	2011-12-26	160	High-Performer	3375.00	17488.64
7	Mountain-100 Silver- 38	Bikes	Mountain Bikes	1912	2011-12-22	160	High-Performer	3400.00	17927.27
8	Mountain-100 Silver- 42	Bikes	Mountain Bikes	1912	2011-12-28	160	High-Performer	3400.00	12981.82
9	Mountain-100 Silver- 44	Bikes	Mountain Bikes	1912	2011-12-12	161	High-Performer	3400.00	15145.45
10	Mountain-100 Silver- 48	Bikes	Mountain Bikes	1912	2011-12-23	160	High-Performer	3400.00	11127.27
16	Road-150 Red- 44	Bikes	Road Bikes	2171	2011-12-28	160	High-Performer	3578.00	91401.64
17	Road-150 Red- 48	Bikes	Road Bikes	2171	2011-12-28	160	High-Performer	3578.00	109616.91
18	Road-150 Red- 52	Bikes	Road Bikes	2171	2011-12-27	160	High-Performer	3578.00	98232.36
19	Road-150 Red- 56	Bikes	Road Bikes	2171	2011-12-27	160	High-Performer	3578.00	95955.45
20	Road-150 Red- 62	Bikes	Road Bikes	2171	2011-12-28	160	High-Performer	3578.00	109291.64
36	Road-650 Black- 44	Bikes	Road Bikes	487	2012-12-26	148	Mid-Range	755.00	2068.04
37	Road-650 Black- 48	Bikes	Road Bikes	487	2012-12-25	148	Mid-Range	759.20	2277.60
38	Road-650 Black- 52	Bikes	Road Bikes	487	2012-12-19	149	High-Performer	751.85	3041.59
39	Road-650 Black- 58	Bikes	Road Bikes	487	2012-12-18	149	High-Performer	763.11	2521.57
40	Road-650 Black- 60	Bikes	Road Bikes	487	2012-12-12	149	High-Performer	752.05	2598.00

Thanks !