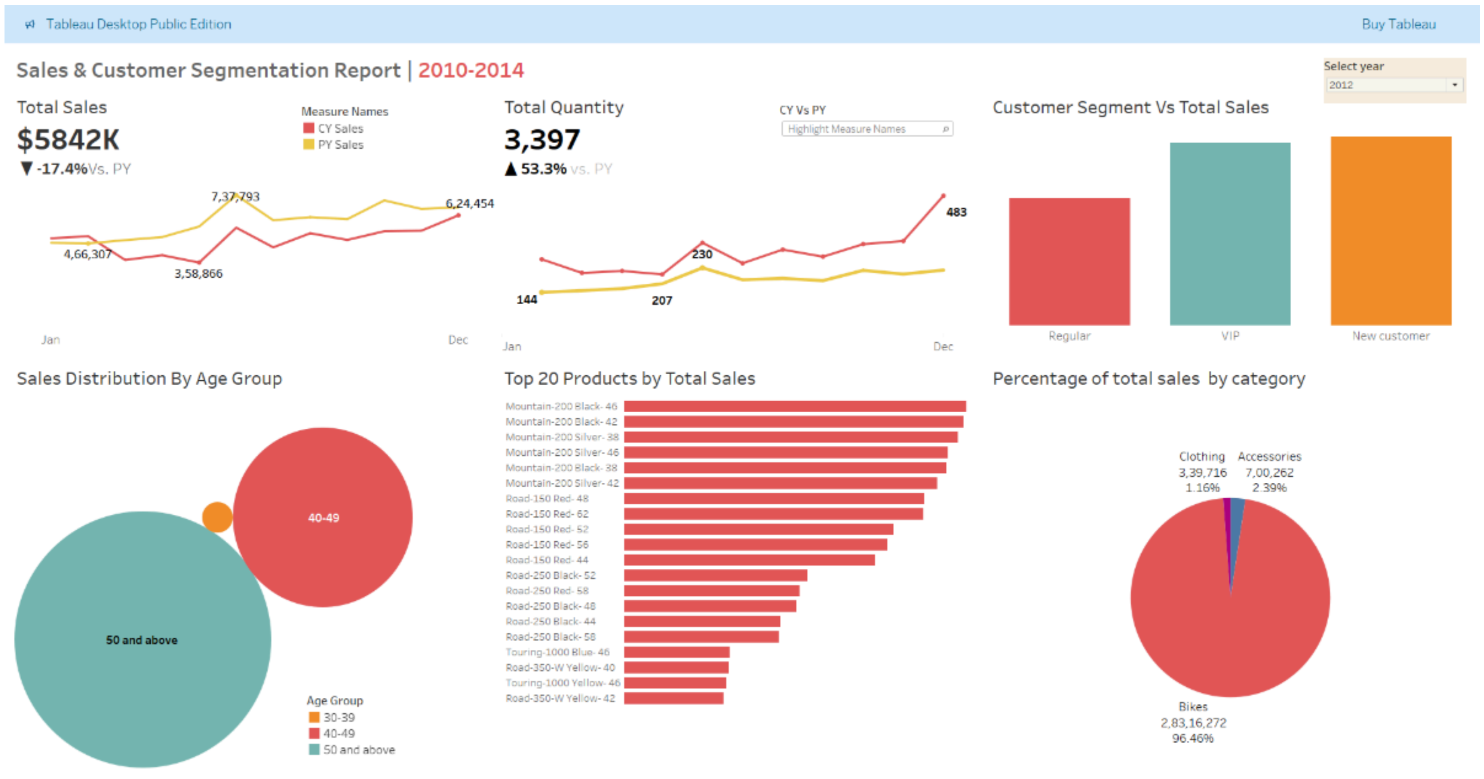
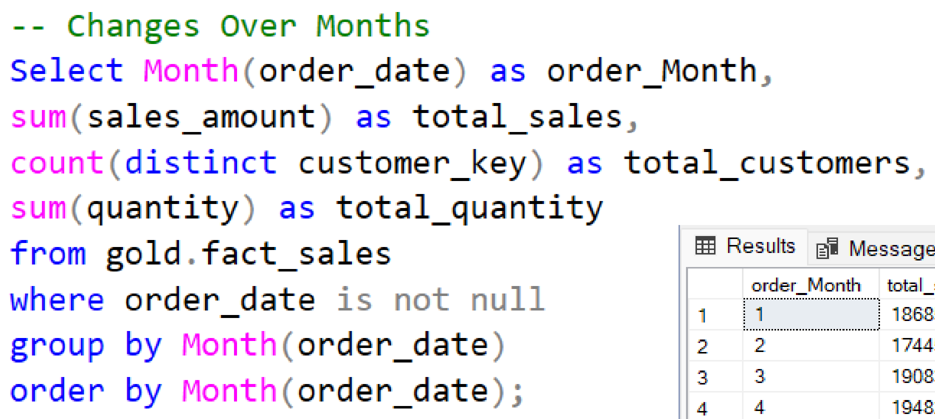
Customer Segmentation & Advanced Sales Data Analysis

Project-By Siddhartha Srivastava

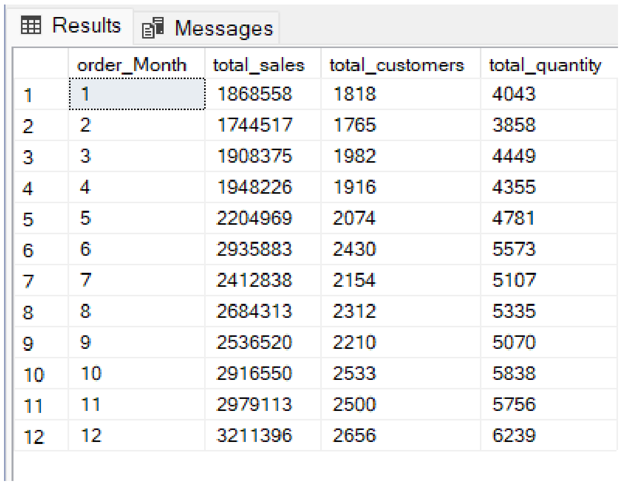


-- Changes Over Months

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);

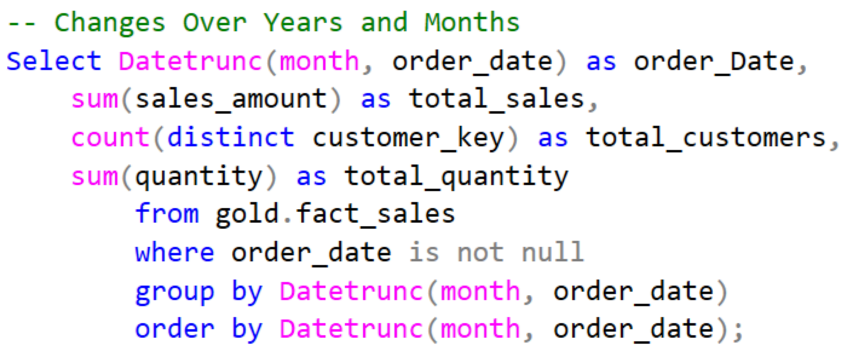


```sql
-- Changes Over Months
Select Month(order_date) as order_Month,
sum(sales_amount) as total_sales,
count(distinct customer_key) as total_customers,
sum(quantity) as total_quantity
from gold.fact_sales
where order_date is not null
group by Month(order_date)
order by Month(order_date);
```

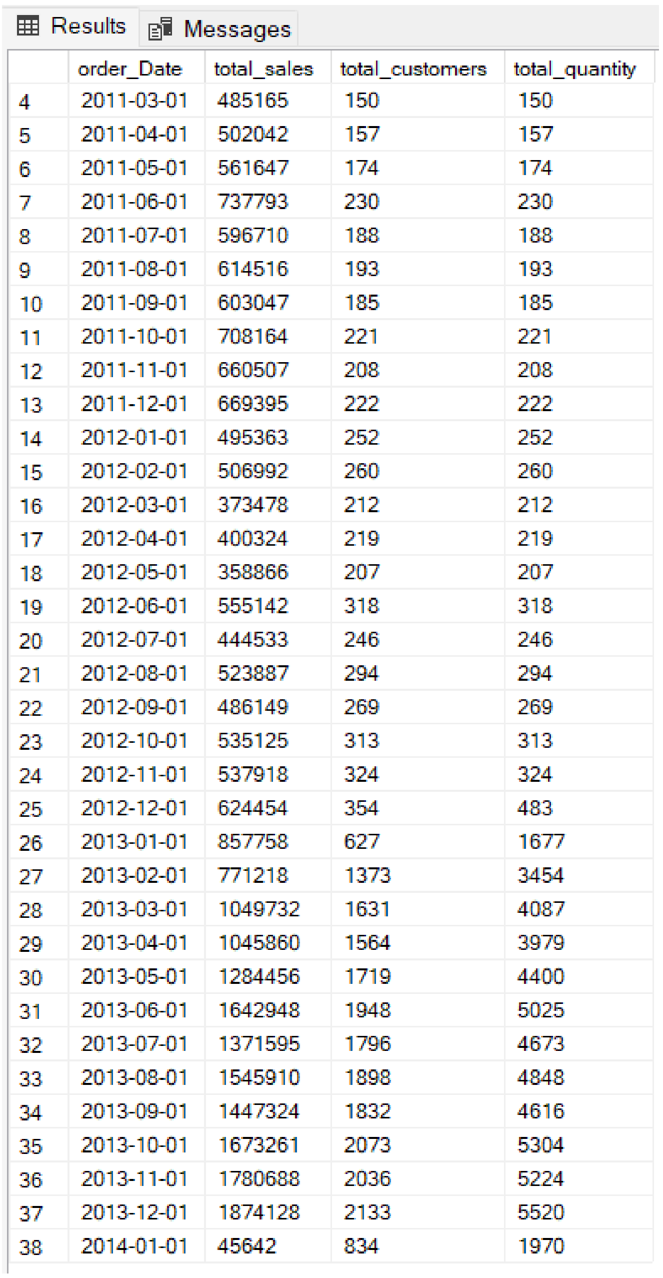

Results
Messages

	order_Month	total_sales	total_customers	total_quantity
1	1	1868558	1818	4043
2	2	1744517	1765	3858
3	3	1908375	1982	4449
4	4	1948226	1916	4355
5	5	2204969	2074	4781
6	6	2935883	2430	5573
7	7	2412838	2154	5107
8	8	2684313	2312	5335
9	9	2536520	2210	5070
10	10	2916550	2533	5838
11	11	2979113	2500	5756
12	12	3211396	2656	6239

```
-- 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);
```

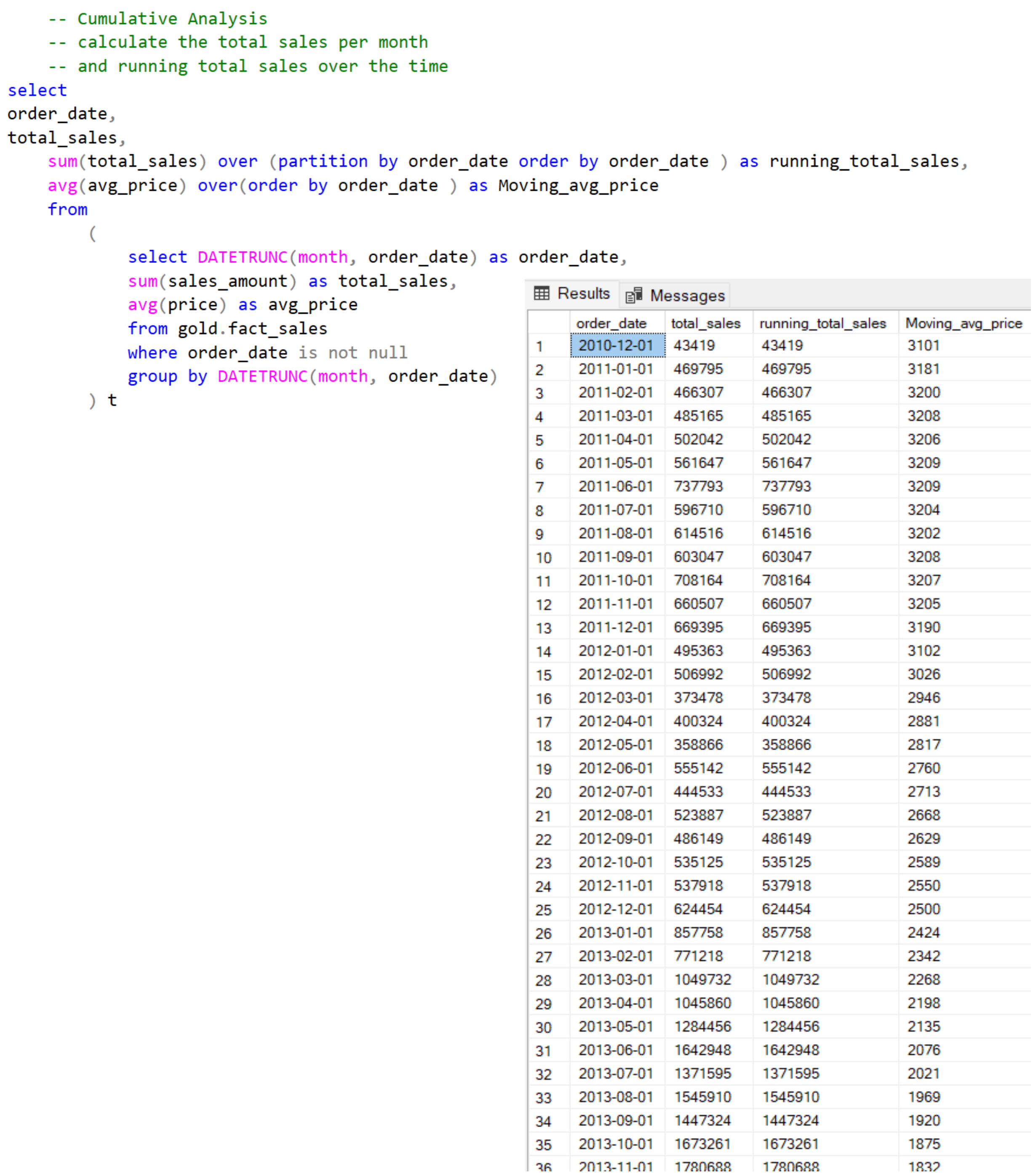


```
-- Changes Over Years and Months
Select Datetrunc(month, order_date) as order_Date,
    sum(sales_amount) as total_sales,
    count(distinct customer_key) as total_customers,
    sum(quantity) as total_quantity
        from gold.fact_sales
        where order_date is not null
        group by Datetrunc(month, order_date)
        order by Datetrunc(month, order_date);
```



	order_Date	total_sales	total_customers	total_quantity
4	2011-03-01	485165	150	150
5	2011-04-01	502042	157	157
6	2011-05-01	561647	174	174
7	2011-06-01	737793	230	230
8	2011-07-01	596710	188	188
9	2011-08-01	614516	193	193
10	2011-09-01	603047	185	185
11	2011-10-01	708164	221	221
12	2011-11-01	660507	208	208
13	2011-12-01	669395	222	222
14	2012-01-01	495363	252	252
15	2012-02-01	506992	260	260
16	2012-03-01	373478	212	212
17	2012-04-01	400324	219	219
18	2012-05-01	358866	207	207
19	2012-06-01	555142	318	318
20	2012-07-01	444533	246	246
21	2012-08-01	523887	294	294
22	2012-09-01	486149	269	269
23	2012-10-01	535125	313	313
24	2012-11-01	537918	324	324
25	2012-12-01	624454	354	483
26	2013-01-01	857758	627	1677
27	2013-02-01	771218	1373	3454
28	2013-03-01	1049732	1631	4087
29	2013-04-01	1045860	1564	3979
30	2013-05-01	1284456	1719	4400
31	2013-06-01	1642948	1948	5025
32	2013-07-01	1371595	1796	4673
33	2013-08-01	1545910	1898	4848
34	2013-09-01	1447324	1832	4616
35	2013-10-01	1673261	2073	5304
36	2013-11-01	1780688	2036	5224
37	2013-12-01	1874128	2133	5520
38	2014-01-01	45642	834	1970

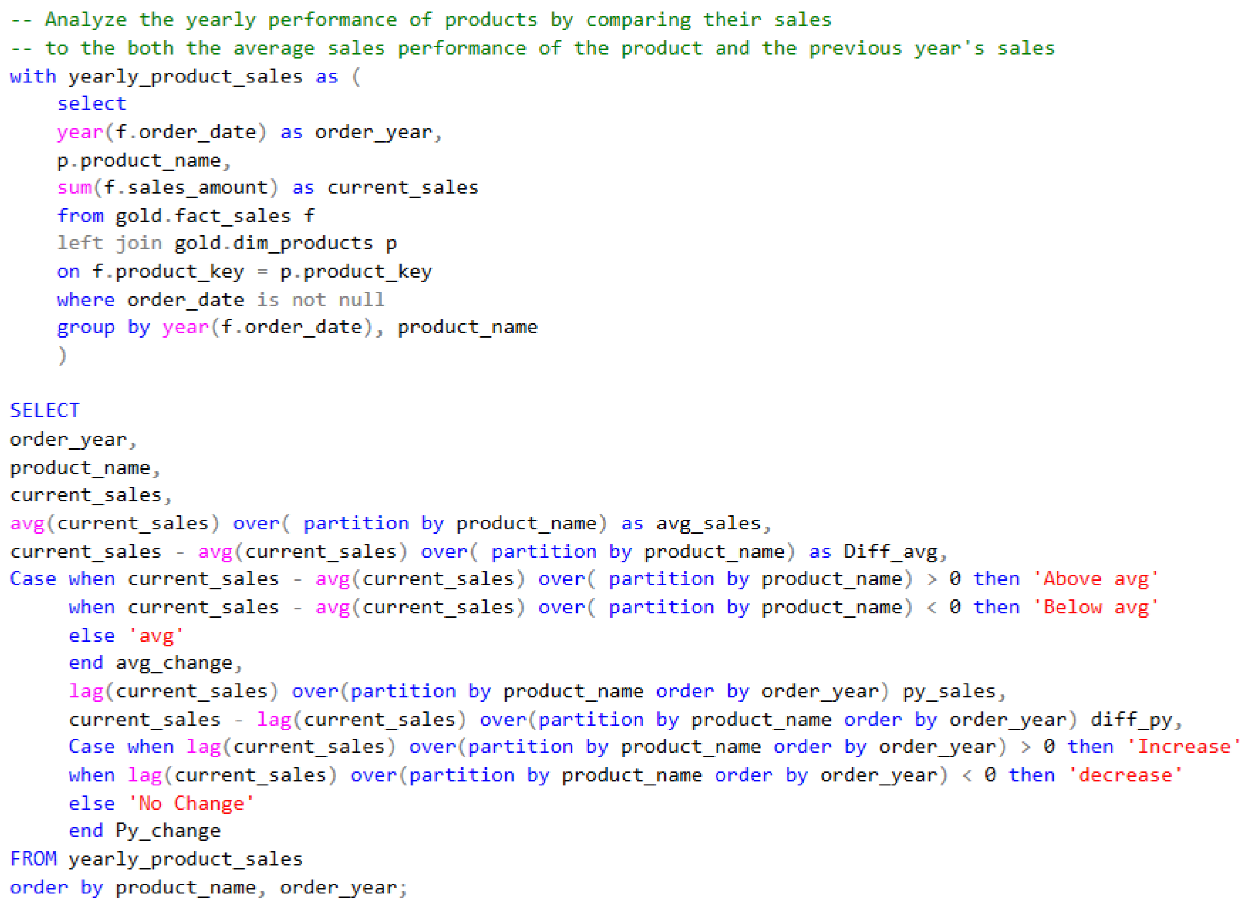
```
-- Cumulative Analysis

  -- 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
```



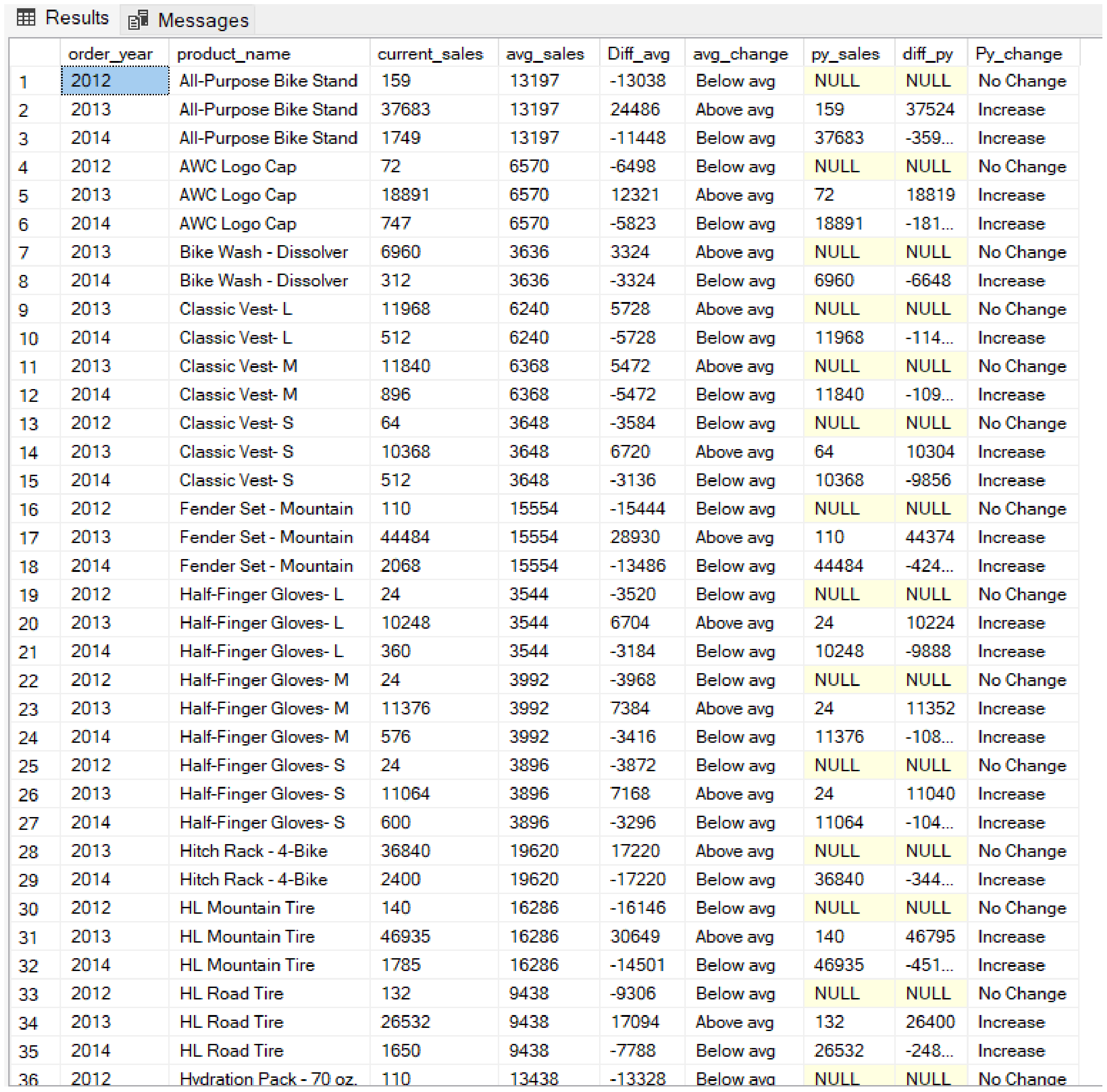
```
    -- Cumulative Analysis
    -- calculate the total sales per month
    -- and running total sales over the time
select
order_date,
total_sales,
    sum(total_sales) over (partition by order_date order by order_date ) as running_total_sales,
    avg(avg_price) over(order by order_date ) as Moving_avg_price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

	order_date	total_sales	running_total_sales	Moving_avg_price
1	2010-12-01	43419	43419	3101
2	2011-01-01	469795	469795	3181
3	2011-02-01	466307	466307	3200
4	2011-03-01	485165	485165	3208
5	2011-04-01	502042	502042	3206
6	2011-05-01	561647	561647	3209
7	2011-06-01	737793	737793	3209
8	2011-07-01	596710	596710	3204
9	2011-08-01	614516	614516	3202
10	2011-09-01	603047	603047	3208
11	2011-10-01	708164	708164	3207
12	2011-11-01	660507	660507	3205
13	2011-12-01	669395	669395	3190
14	2012-01-01	495363	495363	3102
15	2012-02-01	506992	506992	3026
16	2012-03-01	373478	373478	2946
17	2012-04-01	400324	400324	2881
18	2012-05-01	358866	358866	2817
19	2012-06-01	555142	555142	2760
20	2012-07-01	444533	444533	2713
21	2012-08-01	523887	523887	2668
22	2012-09-01	486149	486149	2629
23	2012-10-01	535125	535125	2589
24	2012-11-01	537918	537918	2550
25	2012-12-01	624454	624454	2500
26	2013-01-01	857758	857758	2424
27	2013-02-01	771218	771218	2342
28	2013-03-01	1049732	1049732	2268
29	2013-04-01	1045860	1045860	2198
30	2013-05-01	1284456	1284456	2135
31	2013-06-01	1642948	1642948	2076
32	2013-07-01	1371595	1371595	2021
33	2013-08-01	1545910	1545910	1969
34	2013-09-01	1447324	1447324	1920
35	2013-10-01	1673261	1673261	1875
36	2013-11-01	1780688	1780688	1832

```
-- 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;
```

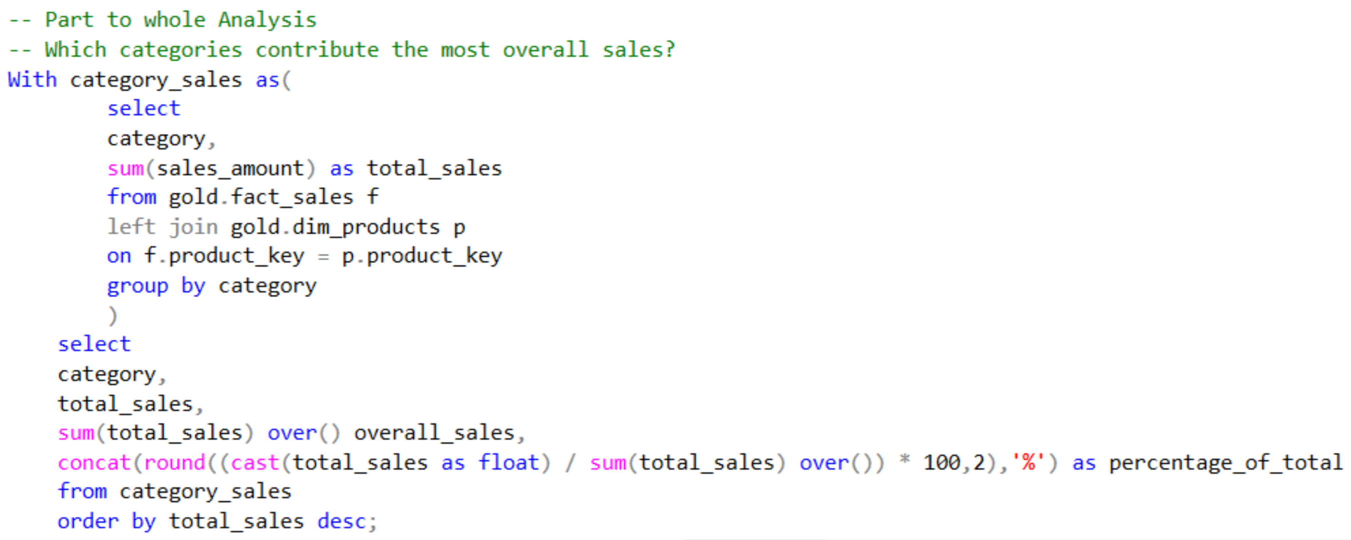


```
-- Analyze the yearly performance of products by comparing their sales
-- to the both the average sales performance of the product and the previous year's sales
with yearly_product_sales as (
    select
    year(f.order_date) as order_year,
    p.product_name,
    sum(f.sales_amount) as current_sales
    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;
```

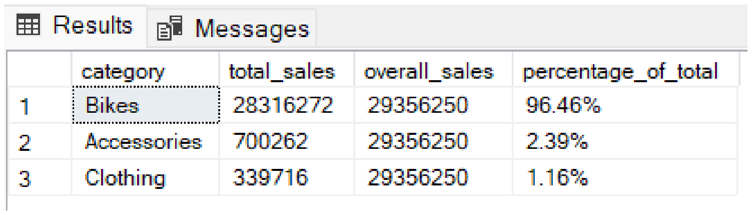


	order_year	product_name	current_sales	avg_sales	Diff_avg	avg_change	py_sales	diff_py	Py_change
1	2012	All-Purpose Bike Stand	159	13197	-13038	Below avg	NULL	NULL	No Change
2	2013	All-Purpose Bike Stand	37683	13197	24486	Above avg	159	37524	Increase
3	2014	All-Purpose Bike Stand	1749	13197	-11448	Below avg	37683	-359...	Increase
4	2012	AWC Logo Cap	72	6570	-6498	Below avg	NULL	NULL	No Change
5	2013	AWC Logo Cap	18891	6570	12321	Above avg	72	18819	Increase
6	2014	AWC Logo Cap	747	6570	-5823	Below avg	18891	-181...	Increase
7	2013	Bike Wash - Dissolver	6960	3636	3324	Above avg	NULL	NULL	No Change
8	2014	Bike Wash - Dissolver	312	3636	-3324	Below avg	6960	-6648	Increase
9	2013	Classic Vest- L	11968	6240	5728	Above avg	NULL	NULL	No Change
10	2014	Classic Vest- L	512	6240	-5728	Below avg	11968	-114...	Increase
11	2013	Classic Vest- M	11840	6368	5472	Above avg	NULL	NULL	No Change
12	2014	Classic Vest- M	896	6368	-5472	Below avg	11840	-109...	Increase
13	2012	Classic Vest- S	64	3648	-3584	Below avg	NULL	NULL	No Change
14	2013	Classic Vest- S	10368	3648	6720	Above avg	64	10304	Increase
15	2014	Classic Vest- S	512	3648	-3136	Below avg	10368	-9856	Increase
16	2012	Fender Set - Mountain	110	15554	-15444	Below avg	NULL	NULL	No Change
17	2013	Fender Set - Mountain	44484	15554	28930	Above avg	110	44374	Increase
18	2014	Fender Set - Mountain	2068	15554	-13486	Below avg	44484	-424...	Increase
19	2012	Half-Finger Gloves- L	24	3544	-3520	Below avg	NULL	NULL	No Change
20	2013	Half-Finger Gloves- L	10248	3544	6704	Above avg	24	10224	Increase
21	2014	Half-Finger Gloves- L	360	3544	-3184	Below avg	10248	-9888	Increase
22	2012	Half-Finger Gloves- M	24	3992	-3968	Below avg	NULL	NULL	No Change
23	2013	Half-Finger Gloves- M	11376	3992	7384	Above avg	24	11352	Increase
24	2014	Half-Finger Gloves- M	576	3992	-3416	Below avg	11376	-108...	Increase
25	2012	Half-Finger Gloves- S	24	3896	-3872	Below avg	NULL	NULL	No Change
26	2013	Half-Finger Gloves- S	11064	3896	7168	Above avg	24	11040	Increase
27	2014	Half-Finger Gloves- S	600	3896	-3296	Below avg	11064	-104...	Increase
28	2013	Hitch Rack - 4-Bike	36840	19620	17220	Above avg	NULL	NULL	No Change
29	2014	Hitch Rack - 4-Bike	2400	19620	-17220	Below avg	36840	-344...	Increase
30	2012	HL Mountain Tire	140	16286	-16146	Below avg	NULL	NULL	No Change
31	2013	HL Mountain Tire	46935	16286	30649	Above avg	140	46795	Increase
32	2014	HL Mountain Tire	1785	16286	-14501	Below avg	46935	-451...	Increase
33	2012	HL Road Tire	132	9438	-9306	Below avg	NULL	NULL	No Change
34	2013	HL Road Tire	26532	9438	17094	Above avg	132	26400	Increase
35	2014	HL Road Tire	1650	9438	-7788	Below avg	26532	-248...	Increase
36	2012	Hydration Pack - 70 oz.	110	13438	-13328	Below avg	NULL	NULL	No Change

```
-- 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;
```



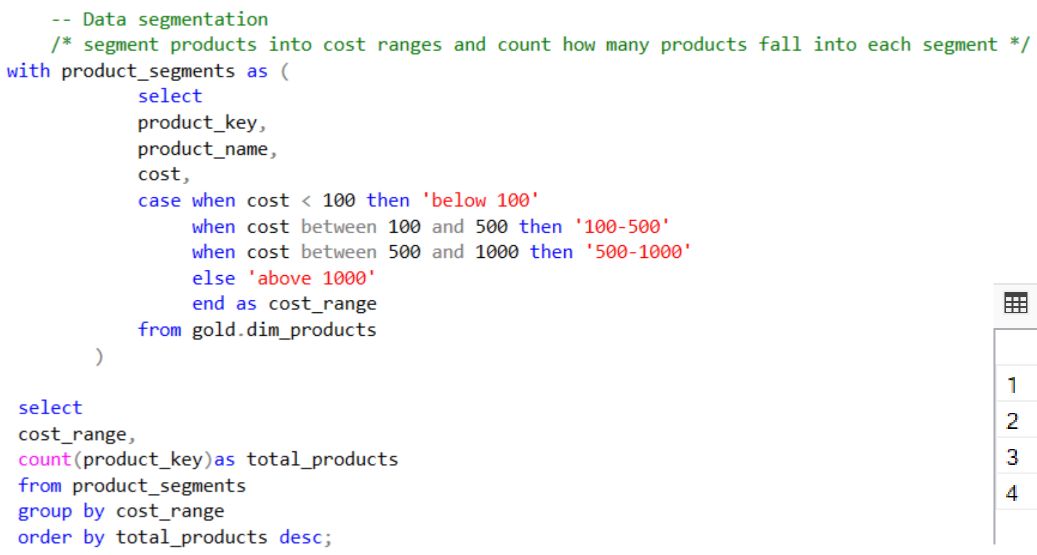
```sql
-- Part to whole Analysis
-- Which categories contribute the most overall sales?
With category_sales as(
        select
        category,
        sum(sales_amount) as total_sales
        from gold.fact_sales f
        left join gold.dim_products p
        on f.product_key = p.product_key
    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



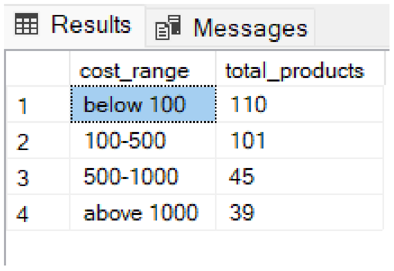
Results | Messages

	category	total_sales	overall_sales	percentage_of_total
1	Bikes	28316272	29356250	96.46%
2	Accessories	700262	29356250	2.39%
3	Clothing	339716	29356250	1.16%

```sql
-- 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;
```



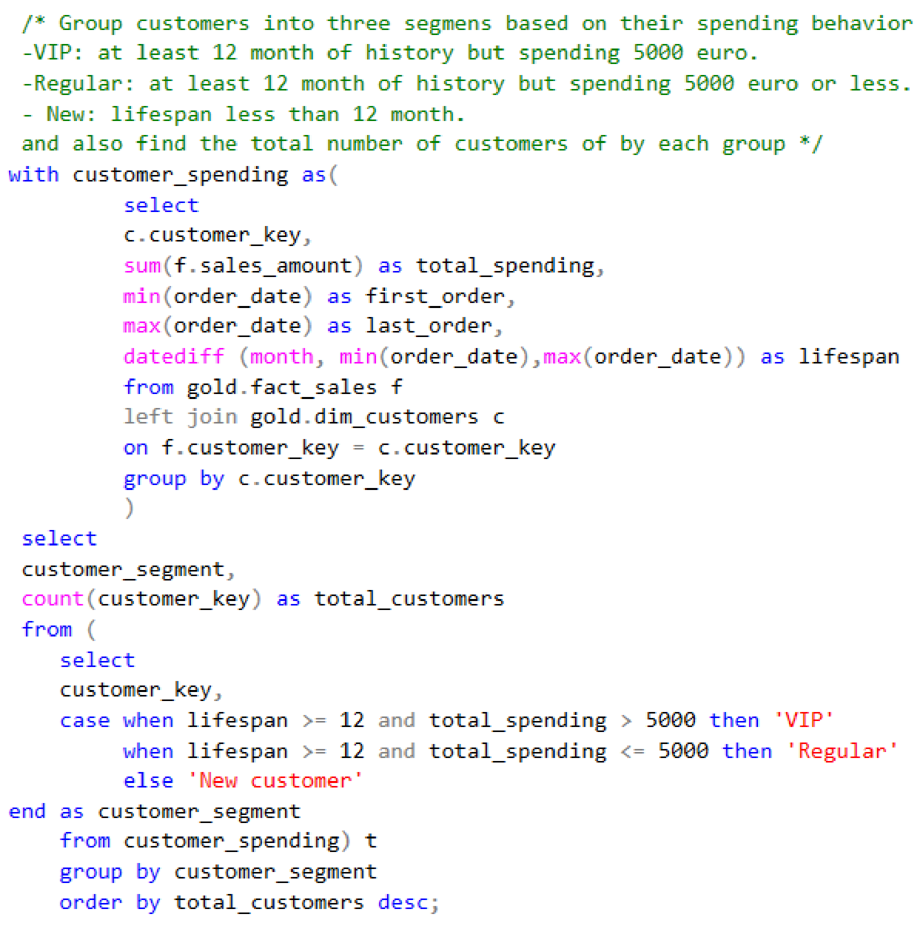
```sql
    -- Data segmentation
    /* segment products into cost ranges and count how many products fall into each segment */
with product_segments as (
            select
            product_key,
            product_name,
            cost,
            case when cost < 100 then 'below 100'
                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;
```



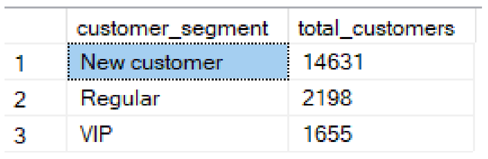
Results | Messages

	cost_range	total_products
1	below 100	110
2	100-500	101
3	500-1000	45
4	above 1000	39

```
/* 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;
```

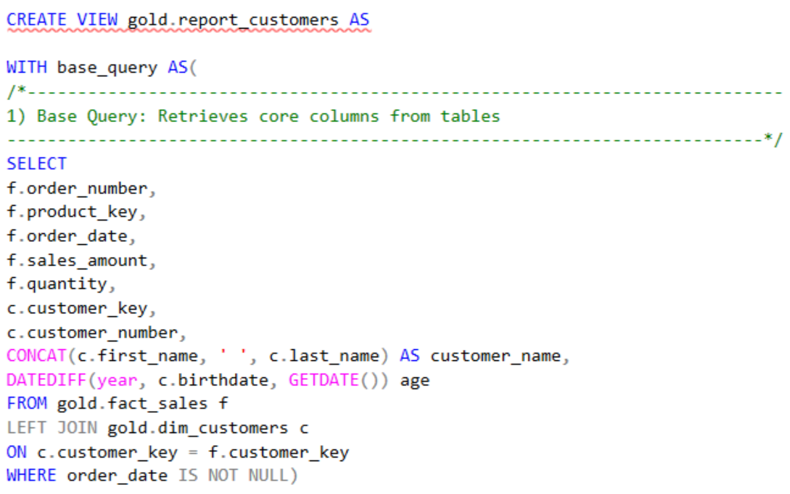


```
 /* Group customers into three segmens based on their spending behavior
 -VIP: at least 12 month of history but spending 5000 euro.
 -Regular: at least 12 month of history but spending 5000 euro or less.
 - New: lifespan less than 12 month.
 and also find the total number of customers of by each 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;
```

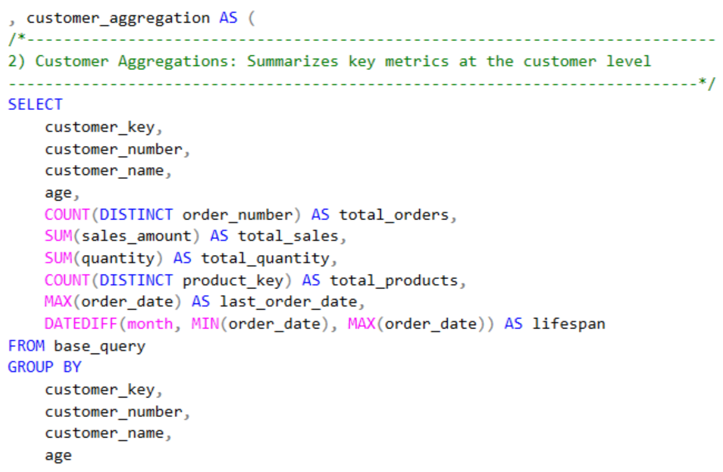


	customer_segment	total_customers
1	New customer	14631
2	Regular	2198
3	VIP	1655

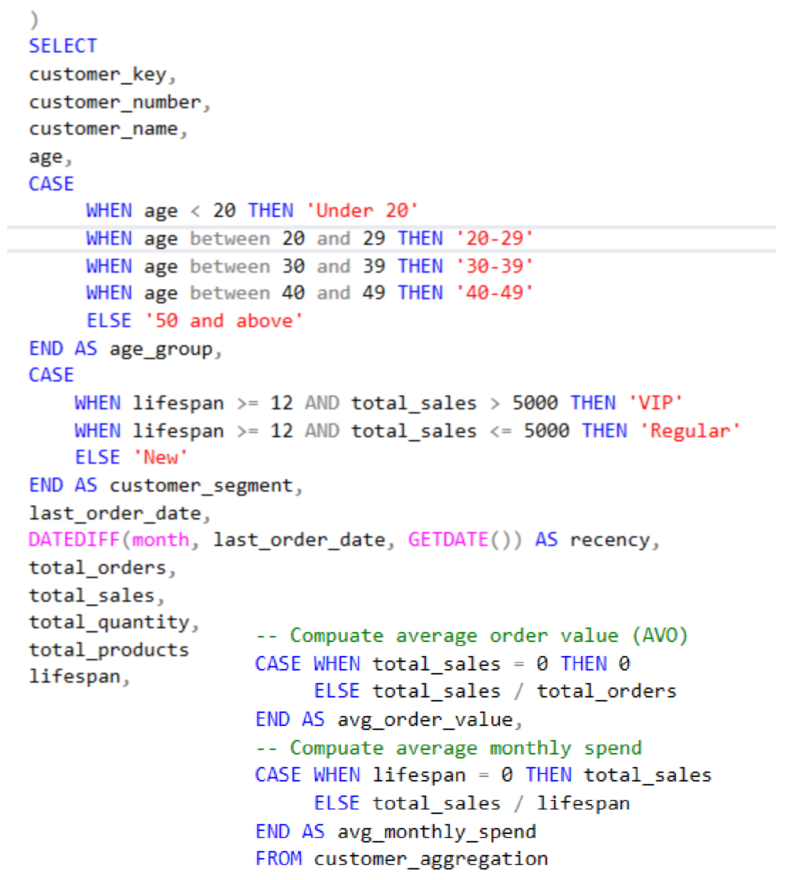
```
/*
===============================================================================
Customer Report
===============================================================================
Purpose:
    - This report consolidates key customer metrics and behaviors


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_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_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
```



```
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)
```

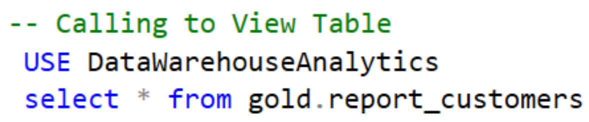


```
, customer_aggregation AS (
/*---------------------------------------------------------------------------
2) Customer Aggregations: Summarizes key metrics at the customer level
---------------------------------------------------------------------------*/
SELECT
    customer_key,
    customer_number,
    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
```

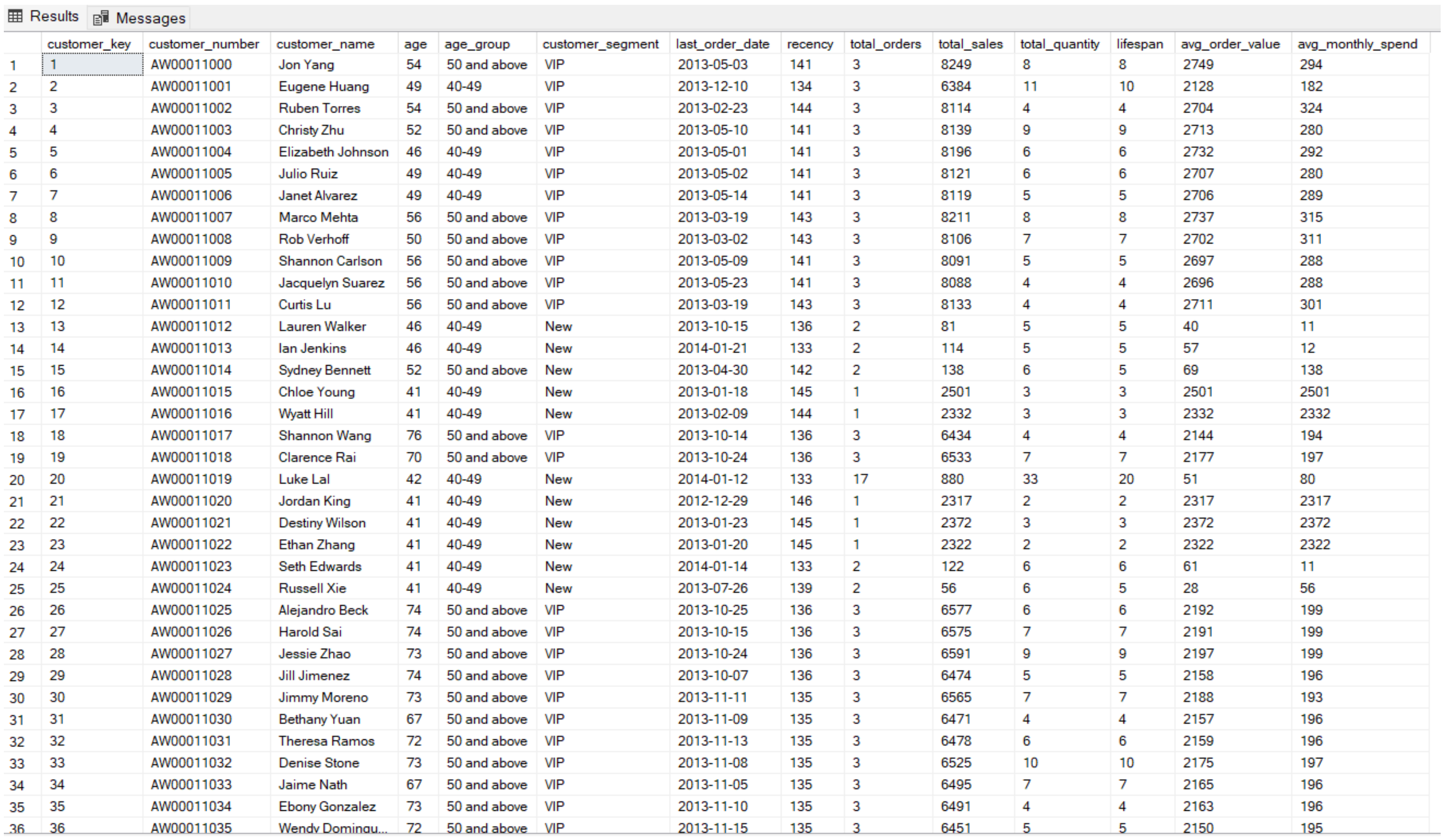


```
)
SELECT
customer_key,
customer_number,
customer_name,
age,
CASE
     WHEN age < 20 THEN 'Under 20'
     WHEN age between 20 and 29 THEN '20-29'
     WHEN age between 30 and 39 THEN '30-39'
     WHEN age between 40 and 49 THEN '40-49'
     ELSE '50 and above'
END AS age_group,
CASE
    WHEN lifespan >= 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_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
```

```
-- Calling to View Table
   use DataWarehouseAnalytics
   select * from gold.report_customers
```



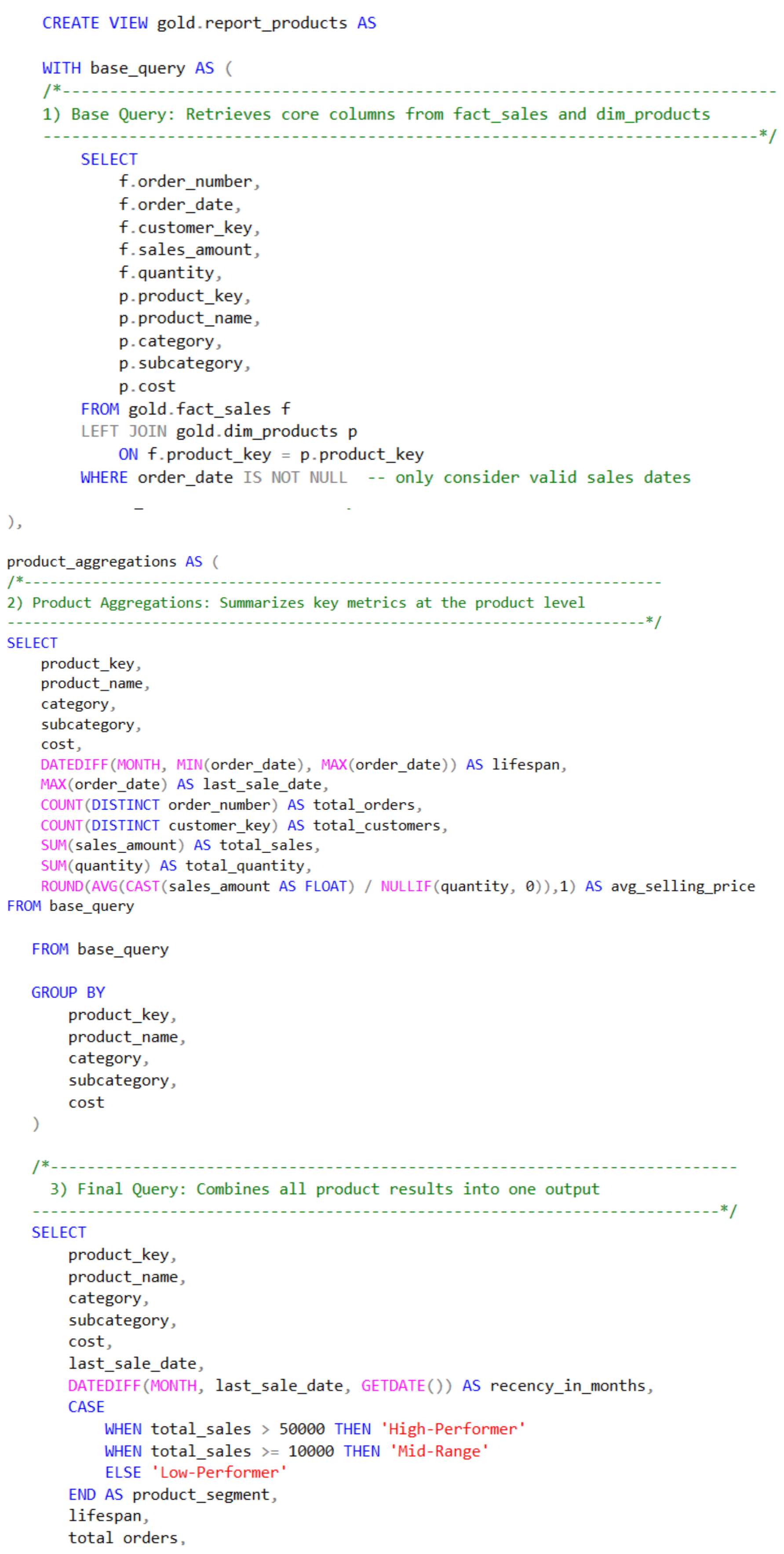
```
-- Calling to View Table
 USE DataWarehouseAnalytics
 select * from gold.report_customers
```



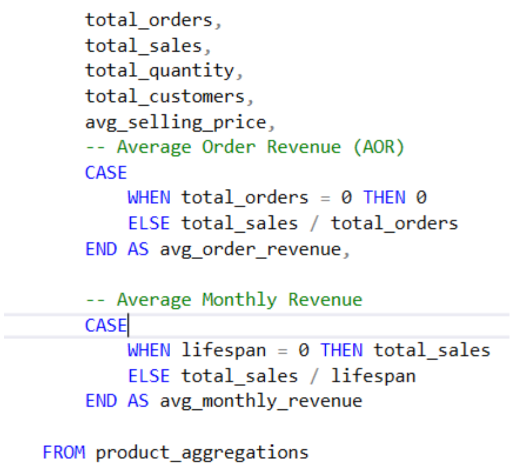
Results | Messages

	customer_key	customer_number	customer_name	age	age_group	customer_segment	last_order_date	recency	total_orders	total_sales	total_quantity	lifespan	avg_order_value	avg_monthly_spend
1	1	AW00011000	Jon Yang	54	50 and above	VIP	2013-05-03	141	3	8249	8	8	2749	294
2	2	AW00011001	Eugene Huang	49	40-49	VIP	2013-12-10	134	3	6384	11	10	2128	182
3	3	AW00011002	Ruben Torres	54	50 and above	VIP	2013-02-23	144	3	8114	4	4	2704	324
4	4	AW00011003	Christy Zhu	52	50 and above	VIP	2013-05-10	141	3	8139	9	9	2713	280
5	5	AW00011004	Elizabeth Johnson	46	40-49	VIP	2013-05-01	141	3	8196	6	6	2732	292
6	6	AW00011005	Julio Ruiz	49	40-49	VIP	2013-05-02	141	3	8121	6	6	2707	280
7	7	AW00011006	Janet Alvarez	49	40-49	VIP	2013-05-14	141	3	8119	5	5	2706	289
8	8	AW00011007	Marco Mehta	56	50 and above	VIP	2013-03-19	143	3	8211	8	8	2737	315
9	9	AW00011008	Rob Verhoff	50	50 and above	VIP	2013-03-02	143	3	8106	7	7	2702	311
10	10	AW00011009	Shannon Carlson	56	50 and above	VIP	2013-05-09	141	3	8091	5	5	2697	288
11	11	AW00011010	Jacquelyn Suarez	56	50 and above	VIP	2013-05-23	141	3	8088	4	4	2696	288
12	12	AW00011011	Curtis Lu	56	50 and above	VIP	2013-03-19	143	3	8133	4	4	2711	301
13	13	AW00011012	Lauren Walker	46	40-49	New	2013-10-15	136	2	81	5	5	40	11
14	14	AW00011013	Ian Jenkins	46	40-49	New	2014-01-21	133	2	114	5	5	57	12
15	15	AW00011014	Sydney Bennett	52	50 and above	New	2013-04-30	142	2	138	6	5	69	138
16	16	AW00011015	Chloe Young	41	40-49	New	2013-01-18	145	1	2501	3	3	2501	2501
17	17	AW00011016	Wyatt Hill	41	40-49	New	2013-02-09	144	1	2332	3	3	2332	2332
18	18	AW00011017	Shannon Wang	76	50 and above	VIP	2013-10-14	136	3	6434	4	4	2144	194
19	19	AW00011018	Clarence Rai	70	50 and above	VIP	2013-10-24	136	3	6533	7	7	2177	197
20	20	AW00011019	Luke Lal	42	40-49	New	2014-01-12	133	17	880	33	20	51	80
21	21	AW00011020	Jordan King	41	40-49	New	2012-12-29	146	1	2317	2	2	2317	2317
22	22	AW00011021	Destiny Wilson	41	40-49	New	2013-01-23	145	1	2372	3	3	2372	2372
23	23	AW00011022	Ethan Zhang	41	40-49	New	2013-01-20	145	1	2322	2	2	2322	2322
24	24	AW00011023	Seth Edwards	41	40-49	New	2014-01-14	133	2	122	6	6	61	11
25	25	AW00011024	Russell Xie	41	40-49	New	2013-07-26	139	2	56	6	5	28	56
26	26	AW00011025	Alejandro Beck	74	50 and above	VIP	2013-10-25	136	3	6577	6	6	2192	199
27	27	AW00011026	Harold Sai	74	50 and above	VIP	2013-10-15	136	3	6575	7	7	2191	199
28	28	AW00011027	Jessie Zhao	73	50 and above	VIP	2013-10-24	136	3	6591	9	9	2197	199
29	29	AW00011028	Jill Jimenez	74	50 and above	VIP	2013-10-07	136	3	6474	5	5	2158	196
30	30	AW00011029	Jimmy Moreno	73	50 and above	VIP	2013-11-11	135	3	6565	7	7	2188	193
31	31	AW00011030	Bethany Yuan	67	50 and above	VIP	2013-11-09	135	3	6471	4	4	2157	196
32	32	AW00011031	Theresa Ramos	72	50 and above	VIP	2013-11-13	135	3	6478	6	6	2159	196
33	33	AW00011032	Denise Stone	73	50 and above	VIP	2013-11-08	135	3	6525	10	10	2175	197
34	34	AW00011033	Jaime Nath	67	50 and above	VIP	2013-11-05	135	3	6495	7	7	2165	196
35	35	AW00011034	Ebony Gonzalez	73	50 and above	VIP	2013-11-10	135	3	6491	4	4	2163	196
36	36	AW00011035	Wendy Domingu...	72	50 and above	VIP	2013-11-15	135	3	6451	5	5	2150	195

```
/*
===============================================================================
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_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  -- 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
```

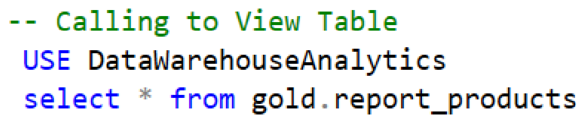


```
    CREATE VIEW gold.report_products AS

    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_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  -- 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

  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,
```

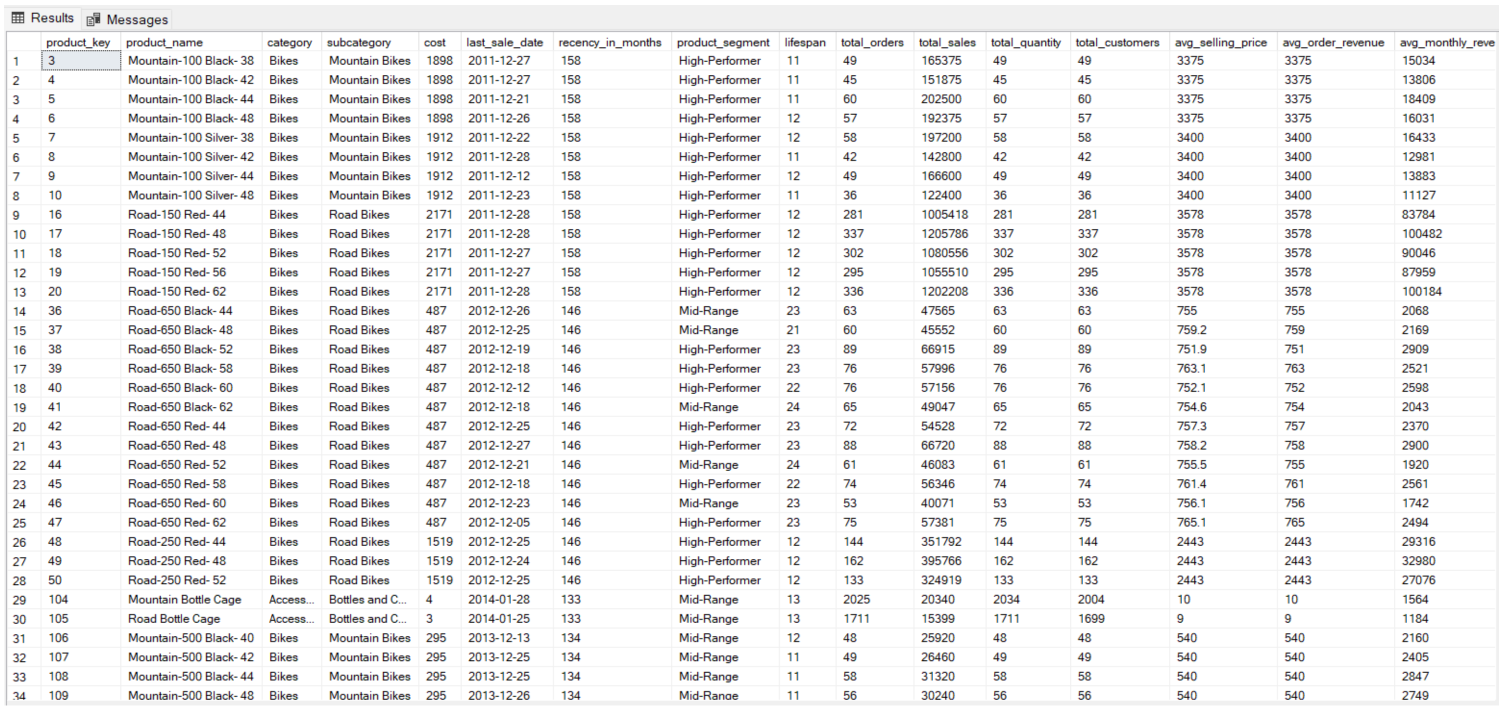


```sql
            total_orders,
            total_sales,
            total_quantity,
            total_customers,
            avg_selling_price,
            -- Average Order Revenue (AOR)
            CASE
                WHEN total_orders = 0 THEN 0
                ELSE total_sales / total_orders
            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
```

-- Calling to View Table
USE DataWarehouseAnalytics
select * from gold.report_products



```sql
-- Calling to View Table
 USE DataWarehouseAnalytics
 select * from gold.report_products
```



Results | Messages

	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	avg_order_revenue	avg_monthly_reve
1	3	Mountain-100 Black- 38	Bikes	Mountain Bikes	1898	2011-12-27	158	High-Performer	11	49	165375	49	49	3375	3375	15034
2	4	Mountain-100 Black- 42	Bikes	Mountain Bikes	1898	2011-12-27	158	High-Performer	11	45	151875	45	45	3375	3375	13806
3	5	Mountain-100 Black- 44	Bikes	Mountain Bikes	1898	2011-12-21	158	High-Performer	11	60	202500	60	60	3375	3375	18409
4	6	Mountain-100 Black- 48	Bikes	Mountain Bikes	1898	2011-12-26	158	High-Performer	12	57	192375	57	57	3375	3375	16031
5	7	Mountain-100 Silver- 38	Bikes	Mountain Bikes	1912	2011-12-22	158	High-Performer	12	58	197200	58	58	3400	3400	16433
6	8	Mountain-100 Silver- 42	Bikes	Mountain Bikes	1912	2011-12-28	158	High-Performer	11	42	142800	42	42	3400	3400	12981
7	9	Mountain-100 Silver- 44	Bikes	Mountain Bikes	1912	2011-12-12	158	High-Performer	12	49	166600	49	49	3400	3400	13883
8	10	Mountain-100 Silver- 48	Bikes	Mountain Bikes	1912	2011-12-23	158	High-Performer	11	36	122400	36	36	3400	3400	11127
9	16	Road-150 Red- 44	Bikes	Road Bikes	2171	2011-12-28	158	High-Performer	12	281	1005418	281	281	3578	3578	83784
10	17	Road-150 Red- 48	Bikes	Road Bikes	2171	2011-12-28	158	High-Performer	12	337	1205786	337	337	3578	3578	100482
11	18	Road-150 Red- 52	Bikes	Road Bikes	2171	2011-12-27	158	High-Performer	12	302	1080556	302	302	3578	3578	90046
12	19	Road-150 Red- 56	Bikes	Road Bikes	2171	2011-12-27	158	High-Performer	12	295	1055510	295	295	3578	3578	87959
13	20	Road-150 Red- 62	Bikes	Road Bikes	2171	2011-12-28	158	High-Performer	12	336	1202208	336	336	3578	3578	100184
14	36	Road-650 Black- 44	Bikes	Road Bikes	487	2012-12-26	146	Mid-Range	23	63	47565	63	63	755	755	2068
15	37	Road-650 Black- 48	Bikes	Road Bikes	487	2012-12-25	146	Mid-Range	21	60	45552	60	60	759.2	759	2169
16	38	Road-650 Black- 52	Bikes	Road Bikes	487	2012-12-19	146	High-Performer	23	89	66915	89	89	751.9	751	2909
17	39	Road-650 Black- 58	Bikes	Road Bikes	487	2012-12-18	146	High-Performer	23	76	57996	76	76	763.1	763	2521
18	40	Road-650 Black- 60	Bikes	Road Bikes	487	2012-12-12	146	High-Performer	22	76	57156	76	76	752.1	752	2598
19	41	Road-650 Black- 62	Bikes	Road Bikes	487	2012-12-18	146	Mid-Range	24	65	49047	65	65	754.6	754	2043
20	42	Road-650 Red- 44	Bikes	Road Bikes	487	2012-12-25	146	High-Performer	23	72	54528	72	72	757.3	757	2370
21	43	Road-650 Red- 48	Bikes	Road Bikes	487	2012-12-27	146	High-Performer	23	88	66720	88	88	758.2	758	2900
22	44	Road-650 Red- 52	Bikes	Road Bikes	487	2012-12-21	146	Mid-Range	24	61	46083	61	61	755.5	755	1920
23	45	Road-650 Red- 58	Bikes	Road Bikes	487	2012-12-18	146	High-Performer	22	74	56346	74	74	761.4	761	2561
24	46	Road-650 Red- 60	Bikes	Road Bikes	487	2012-12-23	146	Mid-Range	23	53	40071	53	53	756.1	756	1742
25	47	Road-650 Red- 62	Bikes	Road Bikes	487	2012-12-05	146	High-Performer	23	75	57381	75	75	765.1	765	2494
26	48	Road-250 Red- 44	Bikes	Road Bikes	1519	2012-12-25	146	High-Performer	12	144	351792	144	144	2443	2443	29316
27	49	Road-250 Red- 48	Bikes	Road Bikes	1519	2012-12-24	146	High-Performer	12	162	395766	162	162	2443	2443	32980
28	50	Road-250 Red- 52	Bikes	Road Bikes	1519	2012-12-25	146	High-Performer	12	133	324919	133	133	2443	2443	27076
29	104	Mountain Bottle Cage	Access...	Bottles and C...	4	2014-01-28	133	Mid-Range	13	2025	20340	2034	2004	10	10	1564
30	105	Road Bottle Cage	Access...	Bottles and C...	3	2014-01-25	133	Mid-Range	13	1711	15399	1711	1699	9	9	1184
31	106	Mountain-500 Black- 40	Bikes	Mountain Bikes	295	2013-12-13	134	Mid-Range	12	48	25920	48	48	540	540	2160
32	107	Mountain-500 Black- 42	Bikes	Mountain Bikes	295	2013-12-25	134	Mid-Range	11	49	26460	49	49	540	540	2405
33	108	Mountain-500 Black- 44	Bikes	Mountain Bikes	295	2013-12-25	134	Mid-Range	11	58	31320	58	58	540	540	2847
34	109	Mountain-500 Black- 48	Bikes	Mountain Bikes	295	2013-12-26	134	Mid-Range	11	56	30240	56	56	540	540	2749

Visual Reporting

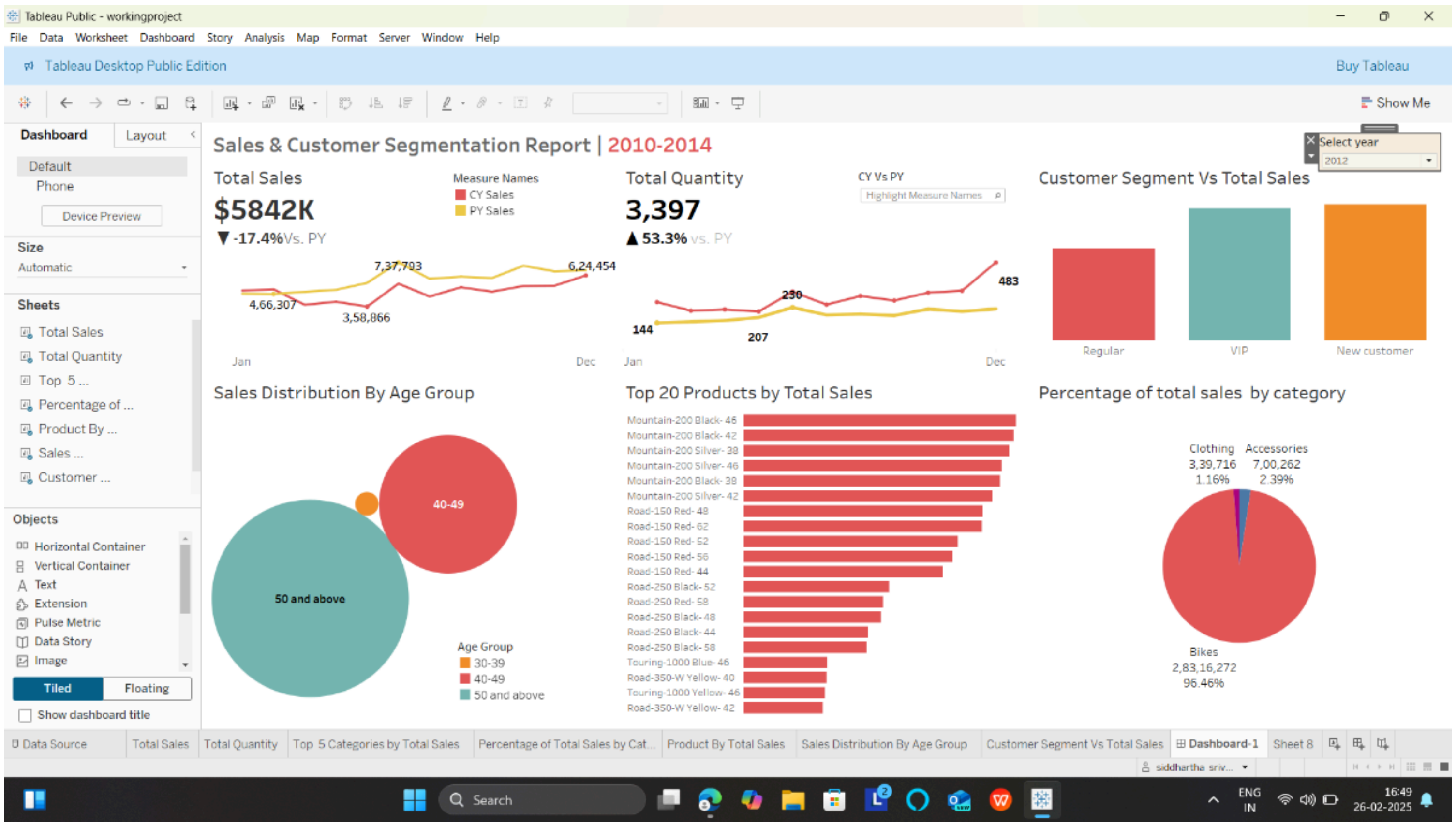


Tableau Public - workingproject
File Data Worksheet Dashboard Story Analysis Map Format Server Window Help
Tableau Desktop Public Edition
Buy Tableau
Show Me
Dashboard
Layout
Default
Phone
Device Preview
Size
Automatic
Sheets
Total Sales
Total Quantity
Top 5 ...
Percentage of ...
Product By ...
Sales ...
Customer ...
Objects
Horizontal Container
Vertical Container
Text
Extension
Pulse Metric
Data Story
Image
Tiled
Floating
Show dashboard title
Sales & Customer Segmentation Report | 2010-2014
Select year
2012
Total Sales
$5842K
▼ -17.4%Vs. PY
Measure Names
CY Sales
PY Sales
7,37,793
6,24,454
4,66,307
3,58,866
Jan
Dec
Total Quantity
3,397
▲ 53.3% vs. PY
CY Vs PY
Highlight Measure Names
230
483
144
207
Jan
Dec
Customer Segment Vs Total Sales
Regular
VIP
New customer
Sales Distribution By Age Group
40-49
50 and above
Age Group
30-39
40-49
50 and above
Top 20 Products by Total Sales
Mountain-200 Black- 46
Mountain-200 Black- 42
Mountain-200 Silver- 38
Mountain-200 Silver- 46
Mountain-200 Black- 38
Mountain-200 Silver- 42
Road-150 Red- 48
Road-150 Red- 62
Road-150 Red- 52
Road-150 Red- 56
Road-150 Red- 44
Road-250 Black- 52
Road-250 Red- 58
Road-250 Black- 48
Road-250 Black- 44
Road-250 Black- 58
Touring-1000 Blue- 46
Road-350-W Yellow- 40
Touring-1000 Yellow- 46
Road-350-W Yellow- 42
Percentage of total sales by category
Clothing
3,39,716
1.16%
Accessories
7,00,262
2.39%
Bikes
2,83,16,272
96.46%
Data Source
Total Sales
Total Quantity
Top 5 Categories by Total Sales
Percentage of Total Sales by Cat...
Product By Total Sales
Sales Distribution By Age Group
Customer Segment Vs Total Sales
Dashboard-1
Sheet 8
siddhartha sriv...
Search
ENG
IN
16:49
26-02-2025

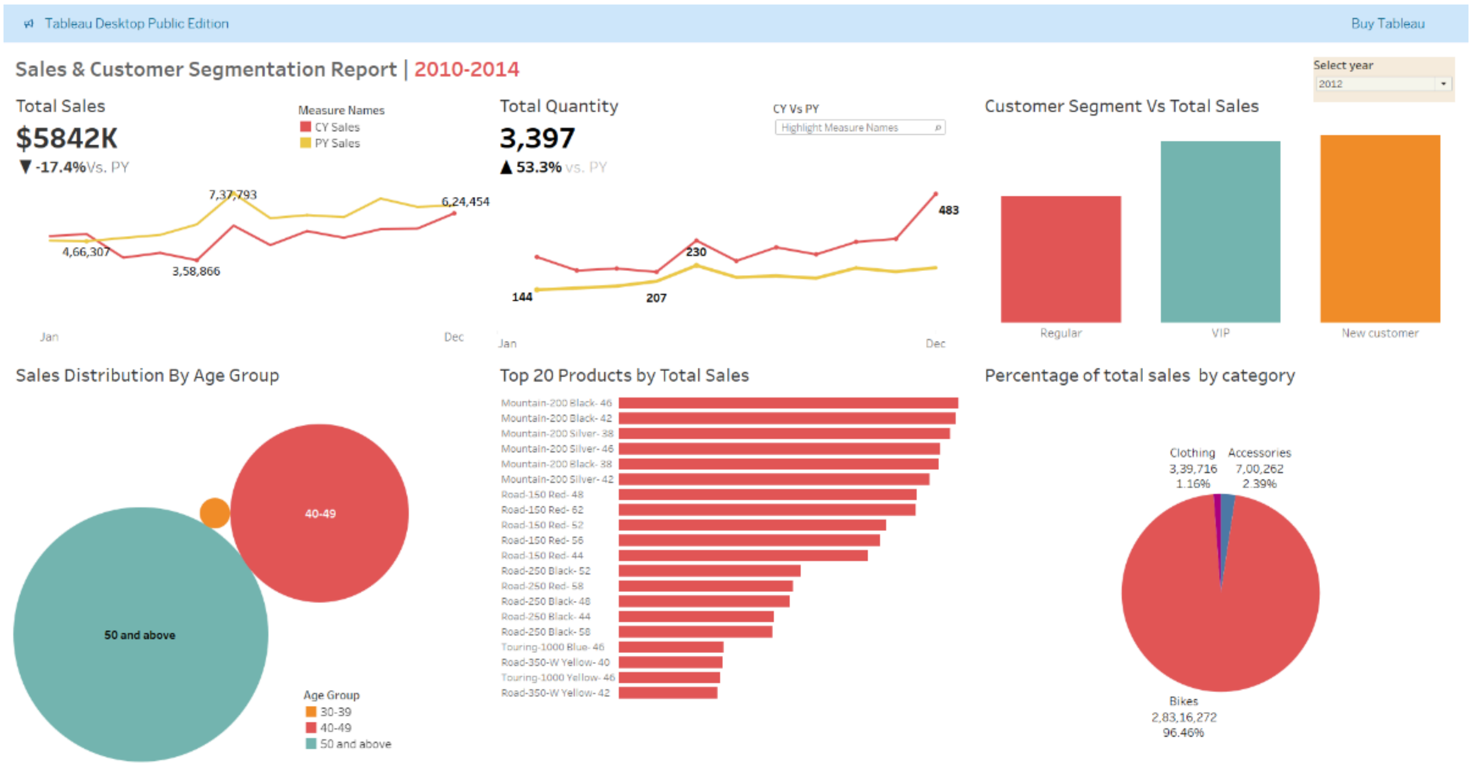


Tableau Desktop Public Edition
Buy Tableau
Sales & Customer Segmentation Report | 2010-2014
Select year
2012
Total Sales
$5842K
▼ -17.4%Vs. PY
Measure Names
CY Sales
PY Sales
7,37,793
6,24,454
4,66,307
3,58,866
Jan
Dec
Total Quantity
3,397
▲ 53.3% vs. PY
CY Vs PY
Highlight Measure Names
230
483
144
207
Jan
Dec
Customer Segment Vs Total Sales
Regular
VIP
New customer
Sales Distribution By Age Group
40-49
50 and above
Age Group
30-39
40-49
50 and above
Top 20 Products by Total Sales
Mountain-200 Black- 46
Mountain-200 Black- 42
Mountain-200 Silver- 38
Mountain-200 Silver- 46
Mountain-200 Black- 38
Mountain-200 Silver- 42
Road-150 Red- 48
Road-150 Red- 62
Road-150 Red- 52
Road-150 Red- 56
Road-150 Red- 44
Road-250 Black- 52
Road-250 Red- 58
Road-250 Black- 48
Road-250 Black- 44
Road-250 Black- 58
Touring-1000 Blue- 46
Road-350-W Yellow- 40
Touring-1000 Yellow- 46
Road-350-W Yellow- 42
Percentage of total sales by category
Clothing
3,39,716
1.16%
Accessories
7,00,262
2.39%
Bikes
2,83,16,272
96.46%