PRODUCT SALES ANALYSIS - SQL PROJECT

Project Overview :

I've completed a comprehensive SQL project analyzing product sales data with a focus on identifying and addressing underperforming products. The project implements various analytical techniques to gain actionable insights from sales data.

Project Outcomes :

The analysis revealed several underperforming products that could benefit from strategic interventions. Based on the data insights, I've developed actionable recommendations to boost sales, including pricing strategies, visibility improvements, and targeted marketing approaches.

Dataset : fact_sales, dim_customers and dim_products

KEY ANALYTICAL APPROACHES

Changes Over Time Analysis

- I tracked sales trends across different time periods to identify seasonal patterns and long-term trends in product performance.

Cumulative Analysis

- By aggregating data progressively over time, I was able to visualize growth curves and determine whether specific product categories are growing or declining.

Performance Analysis

- I compared actual sales metrics against target values to measure success and evaluate performance across different product lines.

Part-To-Whole Analysis

- This analysis helped determine which product categories have the greatest impact on overall business performance by comparing individual product contributions to total sales.

Data Segmentation

- I grouped data based on specific ranges to uncover correlations between different sales measures and identify potential improvement opportunities.

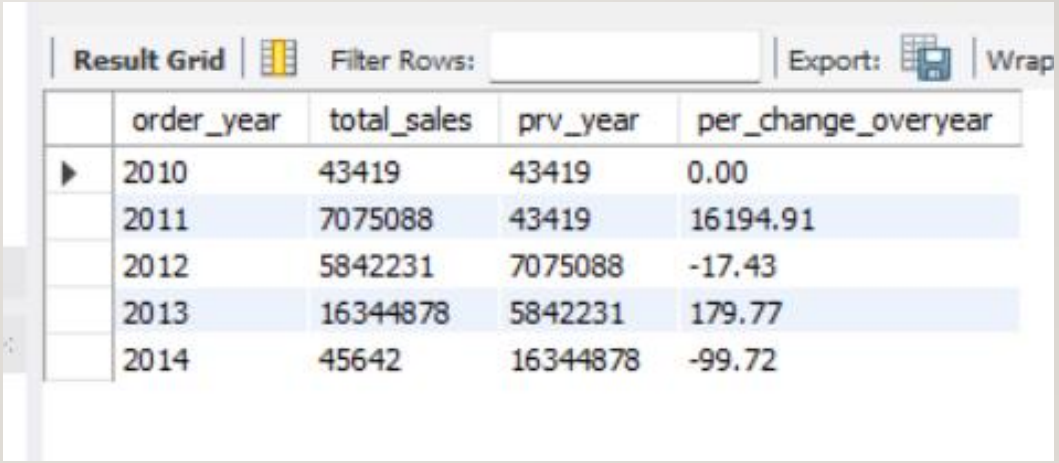
Top-N Analysis

- By identifying our top performing products, I was able to recommend targeted strategies for optimizing our product portfolio.

Changes Over Time Analysis : Helps track trends and identify seasonality in your data

```
with cte as (select year(order_date) as order_year , sum(sales_amount) as total_sales,count(distinct customer_key)
as total_customer,sum(quantity) as total_quantity from fact_sales where year(order_date) is not null group by
year(order_date) order by order_year)

select order_year,total_sales,lag(total_sales,1,total_sales) over(order by order_year) as prv_year,round((total_sales -
lag(total_sales,1,total_sales) over(order by order_year))*100/lag(total_sales,1,total_sales)       over(order by
order_year),2) as per_change_overyear from cte;
```



order_year	total_sales	prv_year	per_change_overyear
2010	43419	43419	0.00
2011	7075088	43419	16194.91
2012	5842231	7075088	-17.43
2013	16344878	5842231	179.77
2014	45642	16344878	-99.72

Cumulative Analysis : Calculate the total sales per month and the running total of sales over time?

```
with cte as(select date_format(order_date,"%Y-%m-01") as order_date, sum(sales_amount) as total_sales,avg(price) as avg_price from fact_sales where date_format(order_date,"%Y-%m-01") is not null group by date_format(order_date,"%Y-%m-01") order by order_date)

select order_date,total_sales,sum(total_sales) over(partition by year(order_date) order by order_date asc) as running_total_sales,round(avg(avg_price) over(partition by year(order_date) order by order_date asc),0) as moving_avg_price from cte;
```

order_date	total_sales	running_total_sales	moving_avg_price
2010-12-01	43419	43419	3101
2011-01-01	469795	469795	3262
2011-02-01	466307	936102	3250
2011-03-01	485165	1421267	3245
2011-04-01	502042	1923309	3233
2011-05-01	561647	2484956	3232
2011-06-01	737793	3222749	3228
2011-07-01	596710	3819459	3220
2011-08-01	614516	4433975	3216
2011-09-01	603047	5037022	3221
2011-10-01	708164	5745186	3219
2011-11-01	660507	6405693	3215
2011-12-01	669395	7075088	3198
2012-01-01	495363	495363	1966
2012-02-01	506992	1002355	1958
2012-03-01	373478	1375833	1892
2012-04-01	400324	1776157	1876

Result 2

Helps to understand whether our business is growing or declining

Performance Analysis : Analyze the yearly performance of products by comparing each product's sales to both its average sales performance and the previous year's sales?

```
with cte as(select year(f.order_date) as order_year,p.product_name,sum(f.sales_amount) as current_sales from fact_sales as f left join dim_products as p on f.product_key=p.product_key where year(f.order_date) is not null group by year(f.order_date),p.product_name order by order_year)

select order_year,product_name,current_sales,ceiling(avg(current_sales) over(partition by product_name)) as avg_sales,current_sales - ceiling(avg(current_sales) over(partition by product_name)) as diff_avg,case when current_sales - ceiling(avg(current_sales) over(partition by product_name))>0 then "Above Avg" when current_sales - ceiling(avg(current_sales) over(partition by product_name)) <0 then "Below Avg"else "Avg" end as avg_change,lag(current_sales,1,current_sales) over(partition by product_name order by order_year) as py_sales,current_sales - lag(current_sales,1,current_sales) over(partition by product_name order by order_year) as diff_py,case when current_sales - lag(current_sales,1,current_sales) over(partition by product_name order by order_year) >0 then "Increase" when current_sales - lag(current_sales,1,current_sales) over(partition by product_name order by order_year) <0 then "Decrease" else "No Change" end as py_change from cte order by product_name,order_year;
```

order_year	product_name	current_sales	avg_sales	diff_avg	avg_change	py_sales	diff_py	py_change
2012	All-Purpose Bike Stand	159	13197	-13038	Below Avg	159	0	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	72	0	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	6960	0	No Change
2014	Bike Wash - Dissolver	312	3636	-3324	Below Avg	6960	-6648	Decrease
2013	Classic Vest- L	11968	6240	5728	Above Avg	11968	0	No Change
2014	Classic Vest- L	512	6240	-5728	Below Avg	11968	-11456	Decrease

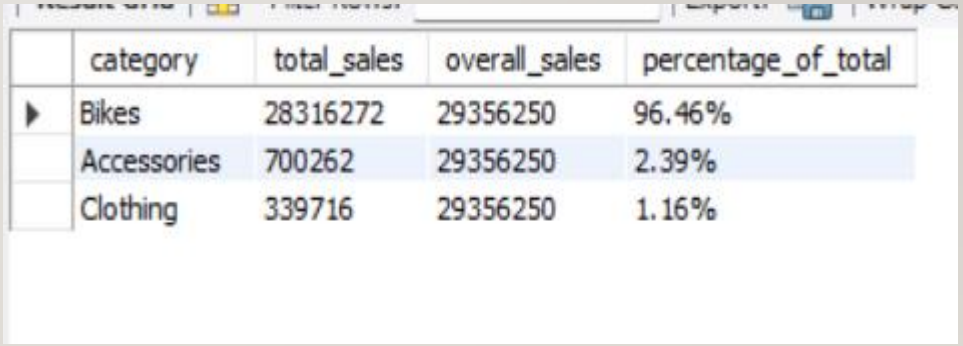
Comparing the current value to a target value.
Helps measure success and compare performance

Part-To-Whole Analysis :Which categories contribute the most to overall sales?

```
with cte as(select p.category,sum(f.sales_amount) as total_sales from fact_sales as f join dim_products as p on
f.product_key = p.product_key group by p.category)

select category,total_sales,sum(total_sales) over() as overall_sales,concat(round((total_sales*100/sum(total_sales)
over()),2),"%") as percentage_of_total from cte order by total_sales desc;
```

Analyze how an individual part is performing compared to the overall, allowing us to understand which category has the greatest impact on the business

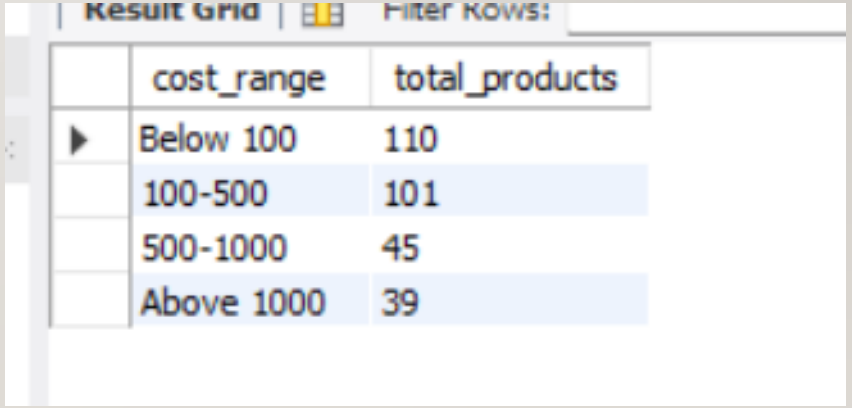


category	total_sales	overall_sales	percentage_of_total
Bikes	28316272	29356250	96.46%
Accessories	700262	29356250	2.39%
Clothing	339716	29356250	1.16%

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 dim_products)

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


Result Grid | Filter Rows:

cost_range	total_products
Below 100	110
100-500	101
500-1000	45
Above 1000	39

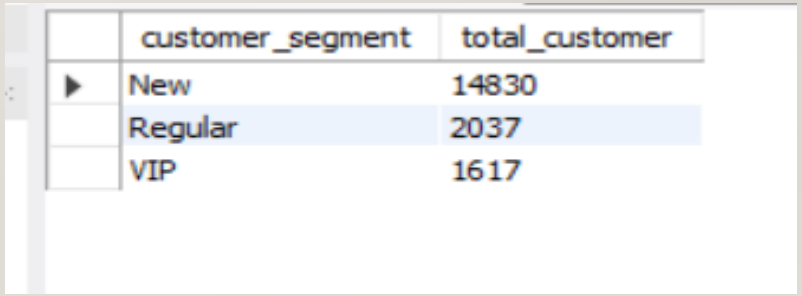
Group the data based on a specific range.
Helps understand the correlation between two measures

Group customers into three segments based on their spending behavior and find the total number of customers by each group?

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.

```
with cte as (select c.customer_key,sum(f.sales_amount) as total_spending,min(f.order_date) as first_order,max(f.order_date) as last_order,timestampdiff(month,min(f.order_date),max(f.order_date)) as life_span from fact_sales as f left join dim_customers as c on f.customer_key=c.customer_key group by c.customer_key), cte2 as (select customer_key,total_spending,life_span, case when life_span >=12 and total_spending >5000 then "VIP"when life_span >=12 and total_spending <=5000 then "Regular" else "New" end as customer_segment  from cte)

select customer_segment,count(customer_key) as total_customer from cte2 group by customer_segment order by total_customer desc;
```



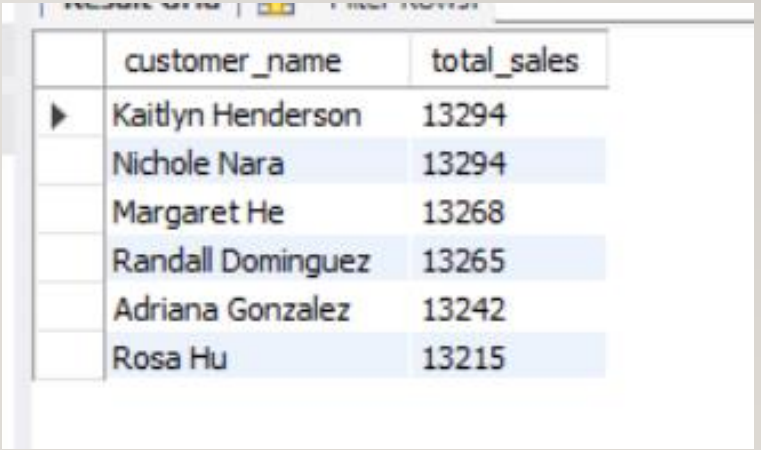
customer_segment	total_customer
New	14830
Regular	2037
VIP	1617

Top-N Analysis : Write an SQL query to find the top 5 customers who have contributed the highest total sales?

```
with cte as (select customer_key,total_sales from (select customer_key,sum(sales_amount) as total_sales ,dense_rank() over(order by sum(sales_amount) desc) as rnk from fact_sales group by customer_key) sal where rnk<=5)

select concat(c.first_name," ",c.last_name) as customer_name,ct.total_sales from cte as ct join dim_customers as c on ct.customer_key=c.customer_key order by ct.total_sales desc;
```

Identifying top performers allows companies to allocate resources more effectively, focusing on areas with the highest potential for growth and profit.



customer_name	total_sales
Kaitlyn Henderson	13294
Nichole Nara	13294
Margaret He	13268
Randall Dominguez	13265
Adriana Gonzalez	13242
Rosa Hu	13215

Key Learnings:

- Gained hands-on experience in data transformation and SQL query optimization.
- Implemented window functions, aggregate functions, and CTEs for advanced data analysis.
- Developed a structured approach to solving real-world business problems using SQL.
- Applied advanced SQL querying and data manipulation for insightful data retrieval.
- Performed data aggregation and statistical analysis to derive meaningful interpretations.
- Designed performance metrics to measure and compare business success.