ANALYTICAL CASE STUDY

Name: Ahmed Sami Abdeflattah Sliem

Track: Data Management

Contents

Q1: Exploratory Data Analysis

1.Correlation between Total Sales and Total Quantity Sold

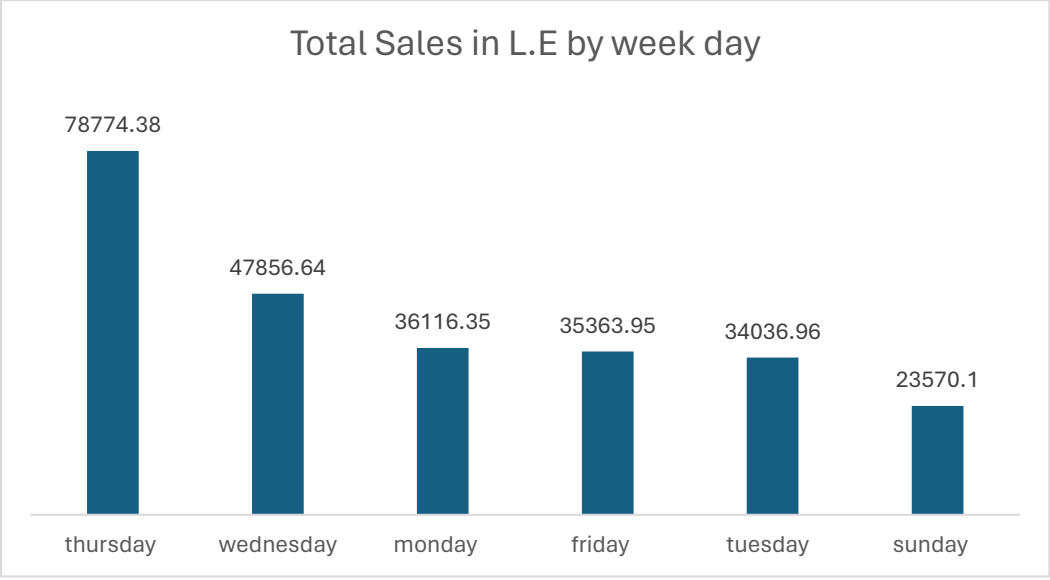
```
select corr(total_sales,total_quantity) as correlation
from (
select STOCKCODE, sum(price*quantity) as total_sales , sum(quantity) as total_quantity
from TABLERETAIL
group by STOCKCODE) s1;
```

- **Description:** This query calculates the correlation between total sales revenue and total quantity sold for each product.
- **Business Benefit:** Helps in understanding the relationship between sales revenue and product quantity sold. The value was **0.75** and this indicates a strong positive relationship and also provides an indication that almost my products majority is not expensive so to achieve a good sales revenue I need to sell high quantity of moderate to small, priced product, This can provide guiding pricing strategies and product positioning.

2. Daily Sales Analysis:

```
SELECT day, totalsales
FROM (
   SELECT DISTINCT
      TO_CHAR(invoicedate, 'day') AS day,
      SUM(QUANTITY * PRICE) OVER (PARTITION BY
TO_CHAR(invoicedate, 'day')) AS totalsales
   FROM
      TABLERETAIL
   ORDER BY
      totalsales DESC
);
```

- **Description:** This query calculates total sales for each day of the week, allowing businesses to analyse daily sales trends.
- **Business Benefit:** Facilitates the identification of weekdays with higher sales volumes, enabling businesses to schedule promotions, offers, and targeted marketing campaigns on optimal days.



3.Customer Segmentation based on Total Purchase Amount:

```
WITH CustomerSegmentation AS (
   SELECT
      CUSTOMER_ID,
      SUM(QUANTITY * PRICE) AS TOTAL_PURCHASE_AMOUNT
   FROM
      TABLERETAIL
   GROUP BY
      CUSTOMER_ID
)
SELECT
   CUSTOMER_ID,
   CASE
      WHEN TOTAL_PURCHASE_AMOUNT >= 1000 THEN 'High Value'
      WHEN TOTAL_PURCHASE_AMOUNT >= 500 AND
TOTAL_PURCHASE_AMOUNT < 1000 THEN 'Mid Value'
      ELSE 'Low Value'
   END AS CUSTOMER_SEGMENT
FROM
   CustomerSegmentation;
```

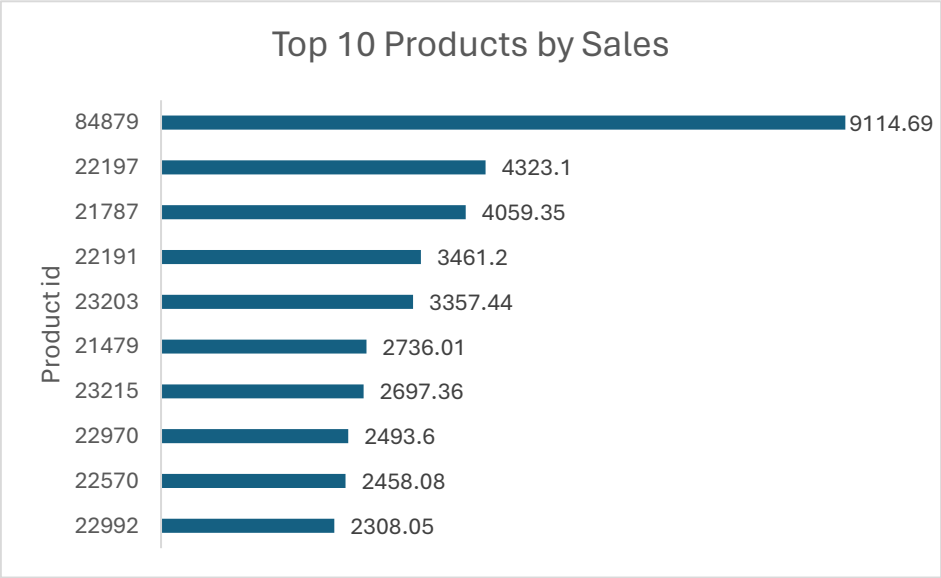
- **Description:** This SQL query segments customers into categories based on their total purchase amount. Customers are classified as High Value if their total purchase amount is 1000 units or more, Mid Value if it falls between 500 and 999 units, and Low Value otherwise.
- **Business Benefit:** Customer segmentation based on total purchase amount provides valuable insights into the value each customer brings to the business. By categorizing customers into segments, businesses can tailor marketing strategies, promotions, and customer service initiatives to meet the specific needs and preferences of each segment. High-value customers may receive exclusive offers or VIP treatment to foster loyalty, while low-value customers may be targeted with incentives to increase their spending or frequency of purchases. This segmentation approach enables businesses to allocate resources effectively, optimize customer engagement, and enhance overall profitability.

CUSTOMER_ID	CUSTOMER_SEGMENT
12828	High Value
12829	Low Value
12833	Low Value
12841	High Value
12844	Low Value
12856	High Value
12883	Mid Value
12884	Low Value
12916	High Value
12921	High Value
12933	Mid Value

4. Product Ranking by Total Sales (Top 10 Products)

```
select STOCKCODE,product_total_sales,Total_Sales_rank_per_stock from (
SELECT
   STOCKCODE,SUM(QUANTITY * PRICE) as product_total_sales,
   DENSE_RANK() OVER(ORDER BY SUM(QUANTITY * PRICE) DESC) AS
Total_Sales_rank_per_stock
FROM
   TABLERETAIL
   group by STOCKCODE
)
where  Total_Sales_rank_per_stock <=10;
```

- **Description:** This query ranks products based on their total sales volume. It complements the previous query by highlighting products that contribute the most to revenue.
- Provides recommendations for best seller products to the customers, guiding marketing strategies, product promotions, and inventory decisions. It ensures that marketing efforts are focused on products with high sales potential.



5. Annual Sales Analysis with Percentage Increase

```
select year,total_sales ,round( ((total_sales - lag(total_sales, 1) OVER
(ORDER BY Year)) / lag(total_sales, 1) OVER (ORDER BY Year)) * 100,2) AS
Sales_annual_increase from (

select distinct to_char(invoicedate,'yyyy') Year, SUM(QUANTITY * PRICE)
over(partition by to_char(invoicedate,'yyyy') order by
to_char(invoicedate,'yyyy') )  as total_sales

from TABLERETAIL

);
```

- **Description:** This query calculates total sales for each year and computes the percentage increase from the previous year. It helps in understanding the growth trajectory of the business.
- **Business Benefit:** Enables the identification of growth trends and patterns over time. It assists in strategic planning, budget allocation, and forecasting by highlighting periods of significant growth or decline.

YEAR	TOTAL_SALES	SALES_ANNUAL_INCREASE
2010	13422.96	
2011	242295.42	1705.08

Q2: Customer Segmentation

A lookup table with the permutations of the scores is created to avoid the headache of case when conditions to specify the category of a customer

```
CREATE TABLE Customer_Category (
    Category_name VARCHAR2(100),
    R_factor NUMBER,
    fm_factor NUMBER
);
```

Here's the Query itself

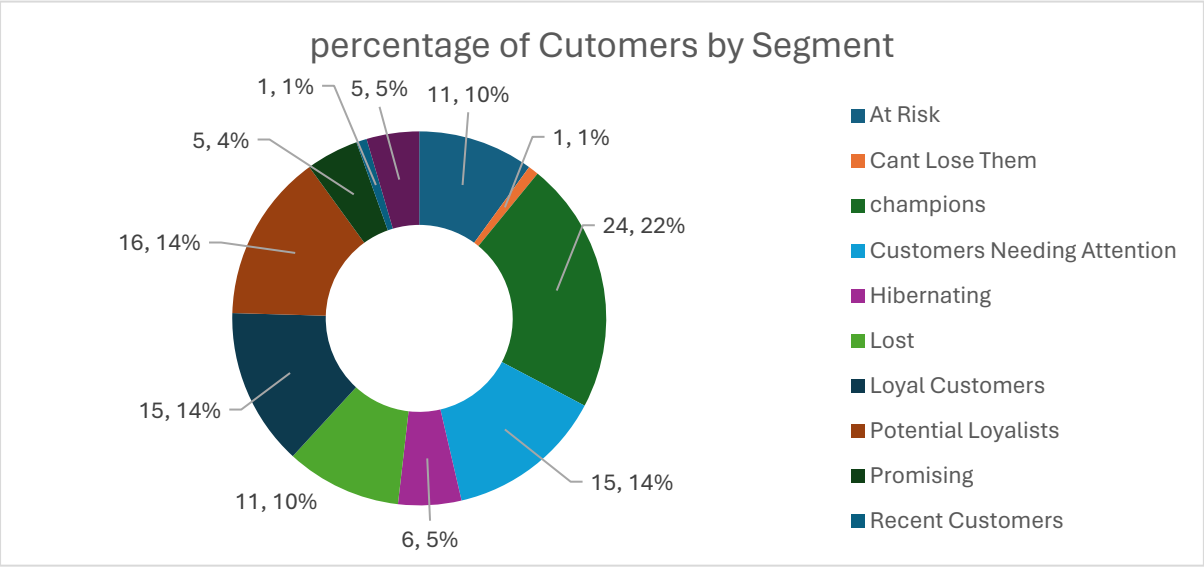
```
with s1 as (
select customer_id,
  count(distinct invoice) Frequency ,
 min(round((select max(invoicedate) from TABLERETAIL )-invoicedate)) as
recency ,
 SUM(QUANTITY * PRICE) AS monetary ,
 ntile(5) over(order by count(distinct invoice) ) as  F_factor,
 ntile(5) over(order by min(trunc((select max(invoicedate) from TABLERETAIL
)-invoicedate))desc) as  R_factor,
 ntile(5) over(order by   SUM(QUANTITY * PRICE)) as  m_factor
from TABLERETAIL
group by customer_id
order by recency desc
), s2 as
(
select s1.*,  ntile(5) over(order by ((m_factor+F_factor)/2)) as fm_factor
from s1
)
select s2.*, COALESCE(cc.category_name, 'Uncategorized') AS category_name
from s2 left join customer_category cc on s2.R_factor=cc.R_factor and
s2.fm_factor =cc.fm_factor
 /*and cc.category_name ='champions' */
order by customer_id desc;
```

Description: This SQL query performs customer segmentation based on three key metrics: frequency, recency, and monetary value (often referred to as RFM analysis). It categorizes customers into segments using quintiles (ntile(5)) for each metric and calculates a combined factor (fm_factor) derived from the average of F_factor and M_factor. The resulting segments are then joined with predefined customer category names based on their recency and combined factor.

Business Benefit:

Customer Understanding: By segmenting customers based on their transaction behavior, businesses gain insights into different customer segments' purchasing patterns and preferences.

Targeted Marketing: Segmentation allows for personalized and targeted marketing campaigns tailored to each customer segment's specific needs, increasing the effectiveness of marketing efforts and customer engagement.



CUSTOMER_ID	FREQUENCY	RECENCY	MONETARY	F_FACTOR	R_FACTOR	M_FACTOR	FM_FACTOR	CATEGORY_NAME
12971	45	168	5190.74	5	2	5	5	At Risk
12970	4	7	452.24	3	5	2	3	Loyal Customers
12968	1	112	135.95	1	2	1	1	Uncategorized
12967	2	358	1660.9	2	1	4	3	At Risk
12966	1	9	160.18	1	4	1	1	Promising
12965	1	89	771.91	2	2	3	2	Customers Needing Attention
12963	8	8	1856.63	5	5	4	5	champions
12962	2	7	266.39	2	5	1	1	Recent Customers
12957	8	9	4017.54	5	4	5	5	champions
12956	1	306	108.07	1	1	1	1	Lost
12955	11	1	4757.16	5	5	5	5	champions
12953	1	9	329.85	1	4	2	2	Potential Loyalists
12952	4	5	1387.79	4	5	4	4	champions
12951	6	8	1064.07	4	4	3	3	Potential Loyalists
12950	3	2	1843	3	5	4	4	champions

Q3: Customer Behavior Analysis

Query 1: Maximum Consecutive Purchase Days

```
with customerconsecutivedays as (
select cust_id, calendar_dt,
       case
           when calendar_dt - lag(calendar_dt) over (partition by cust_id order by
calendar_dt) = 1
           then 0
           else 1
       end as consecutive_flag
   from customers
),
Groupss as (
select cust_id,calendar_dt,sum(consecutive_flag) over (partition by cust_id order by
calendar_dt) as group_id
from customerconsecutivedays),

Consecutive_occurence as
 (
 select cust_id, COUNT(cust_id) Consecutive_day
from Groupss /*where cust_id=26592*/
group by cust_id, group_id )

select cust_id, max(Consecutive_day) as max_consecutive_days
from Consecutive_occurence
group by cust_id ;
```

CUST_ID	MAX_CONSECUTIVE_DAYS
150488	9
259866	8
480780	11
839622	61
1311280	2
1327831	8
1331618	9
1807965	6
1822477	8
2211812	7
2341278	5
2877195	18
3342799	9
3788435	3

Query 2: Average Days to Reach a Spending Threshold

```
-- Calculate total spending for each customer over time

with total_spending as (
select cust_id,sum(amt_le)over(partition by cust_id order by CUSTOMERS.CALENDAR_DT) as
total_spend ,CALENDAR_DT
from customers
),
 -- Determine the first order date for each customer

First_order_date as (
 select distinct cust_id, first_value(CALENDAR_DT) over(partition by cust_id order by CALENDAR_DT)
firts_order_date
 from customers ),

-- Calculate the number of days it takes for each customer to reach $250 spending milestone from
their first order date

 Final_result as (
 select distinct  t.cust_id,(first_value(t.CALENDAR_DT) over(partition by t.cust_id order by
t.CALENDAR_DT)  - f.firts_order_date) as Way_to_250, -- left part of this equation is used to get the
first date that customer exceeded 250 ,
  min(t.total_spend) over (partition by t.cust_id order by t.CALENDAR_DT) as First_250_Hit -- this
column only to show the value of sales that cutomers first exceeded 250
 from total_spending t , First_order_date f
 where t.cust_id=f.cust_id and
  t.total_spend>=250
   /*and t.cust_id=26592*/
 )
 -- Calculate the average number of days across all customers to reach the $250 spending milestone

 select round(avg(Way_to_250),2) as avg_no_days
 from Final_result;
```

AVG_NO_DAYS
11.35