

1. Let's explore the dataset:

```
1 SELECT
2   'Customers' AS table_name,
3   (SELECT COUNT(*) FROM pragma_table_info('Customers')) AS number_of_attributes,
4   (SELECT COUNT(*) FROM customers) AS number_of_rows
5
6 UNION ALL
7
8 SELECT
9   'Products' AS table_name,
10  (SELECT COUNT(*) FROM pragma_table_info('Products')) AS number_of_attributes,
11  (SELECT COUNT(*) FROM products) AS number_of_rows
12
13 UNION ALL
14
15 SELECT
16   'ProductLines' AS table_name,
17   (SELECT COUNT(*) FROM pragma_table_info('ProductLines')) AS
18   number_of_attributes,
19   (SELECT COUNT(*) FROM productlines) AS number_of_rows
20
21 UNION ALL
22
23 SELECT
24   'Orders' AS table_name,
25   (SELECT COUNT(*) FROM pragma_table_info('Orders')) AS number_of_attributes,
26   (SELECT COUNT(*) FROM orders) AS number_of_rows
27
28 UNION ALL
29
30 SELECT
31   'OrderDetails' AS table_name,
32   (SELECT COUNT(*) FROM pragma_table_info('OrderDetails')) AS
33   number_of_attributes,
34   (SELECT COUNT(*) FROM orderdetails) AS number_of_rows
35
36 UNION ALL
37
38 SELECT
39   'Payments' AS table_name,
40   (SELECT COUNT(*) FROM pragma_table_info('Payments')) AS number_of_attributes,
41   (SELECT COUNT(*) FROM payments) AS number_of_rows
42
43 UNION ALL
44
45 SELECT
46   'Employees' AS table_name,
47   (SELECT COUNT(*) FROM pragma_table_info('Employees')) AS number_of_attributes,
48   (SELECT COUNT(*) FROM employees) AS number_of_rows
49
50 UNION ALL
```

```
50 SELECT
51   'Offices' AS table_name,
52   (SELECT COUNT(*) FROM pragma_table_info('Offices')) AS number_of_attributes,
53   (SELECT COUNT(*) FROM offices) AS number_of_rows;
```

2. Which Products Should We Order More of or Less of?

Low Stock:

```

1 SELECT productCode,
2     ROUND(SUM(quantityOrdered) * 1.0 / (SELECT quantityInStock
3 FROM products
4 WHERE orderdetails.productCode = products.productCode), 2) AS low_stock
5 FROM orderdetails
6 GROUP BY productCode
7 ORDER BY low_stock DESC
8 LIMIT 10;

```

Priority for Restocking:

```

1  WITH
2
3  low_stock_table AS (
4  SELECT productCode,
5         ROUND(SUM(quantityOrdered) * 1.0/(SELECT quantityInStock
6         FROM products p
7         WHERE od.productCode = p.productCode), 2) AS low_stock
8  FROM orderdetails od
9  GROUP BY productCode
10 ORDER BY low_stock DESC
11 LIMIT 10
12 ),
13
14 products_to_restock AS (
15 SELECT productCode,
16        SUM(quantityOrdered * priceEach) AS prod_perf
17 FROM orderdetails od
18 WHERE productCode IN (SELECT productCode
19        FROM low_stock_table)
20 GROUP BY productCode
21 ORDER BY prod_perf DESC
22 LIMIT 10
23 )
24
25 SELECT productName, productLine
26 FROM products AS p
27 WHERE productCode IN (SELECT productCode
28        FROM products_to_restock);
29

```

3. How Should We Match Marketing and Communication Strategies to Customer Behavior?

Profit Calculation:

```
1 SELECT orders.customerNumber,  
2    SUM(quantityOrdered * (priceEach - buyPrice)) AS profit  
3 FROM orders  
4 JOIN orderdetails ON orders.orderNumber = orderdetails.orderNumber  
5 JOIN products ON orderdetails.productCode = products.productCode  
6 GROUP BY orders.customerNumber  
7 ORDER BY profit DESC;  
8  
9
```

Customer Segmentation:

```
1 WITH customer_profit AS (  
2   SELECT  
3     orders.customerNumber,  
4     SUM(quantityOrdered * (priceEach - buyPrice)) AS profit  
5   FROM orders  
6   JOIN orderdetails ON orders.orderNumber = orderdetails.orderNumber  
7   JOIN products ON orderdetails.productCode = products.productCode  
8   GROUP BY orders.customerNumber  
9 ),  
10 ranked_customers AS (  
11   SELECT  
12     customerNumber,  
13     profit,  
14     NTILE(10) OVER (ORDER BY profit DESC) AS percentile_rank -- Divide into 10 groups  
15   FROM customer_profit  
16 )  
17 SELECT  
18   customerNumber,  
19   profit,  
20   CASE  
21     WHEN percentile_rank = 1 THEN 'VIP' -- Top 10%  
22     WHEN percentile_rank BETWEEN 2 AND 5 THEN 'Engaged' -- Next 40%  
23     ELSE 'Less-Engaged' -- Bottom 50%  
24   END AS customer_segment  
25 FROM ranked_customers  
26 ORDER BY profit DESC;  
27  
28
```

Top 5 VIP Customers:

```
1 WITH profit_cte AS (  
2   SELECT o.customerNumber, SUM(quantityOrdered * (priceEach - buyPrice)) AS profit  
3   FROM products p  
4   JOIN orderdetails od  
5     ON p.productCode = od.productCode  
6   JOIN orders o  
7     ON o.orderNumber = od.orderNumber  
8   GROUP BY o.customerNumber  
9 )  
10  
11 SELECT c.contactLastName, c.contactFirstName, c.city, c.country, profit  
12 FROM customers c  
13 JOIN profit_cte  
14   ON c.customerNumber = profit_cte.customerNumber  
15 ORDER BY profit DESC  
16 LIMIT 5;  
17  
18
```

Top 5 less engaged Customers:

```
1 WITH profit_cte AS (  
2   SELECT o.customerNumber, SUM(quantityOrdered * (priceEach - buyPrice)) AS profit  
3   FROM products p  
4   JOIN orderdetails od  
5     ON p.productCode = od.productCode  
6   JOIN orders o  
7     ON o.orderNumber = od.orderNumber  
8   GROUP BY o.customerNumber  
9 )  
10  
11 SELECT c.contactLastName, c.contactFirstName, c.city, c.country, profit  
12 FROM customers c  
13 JOIN profit_cte  
14   ON c.customerNumber = profit_cte.customerNumber  
15 ORDER BY profit ASC  
16 LIMIT 5;  
17  
18
```

The number of new customers arriving each month to check if it's worth spending money on acquiring new customers:

```
1 WITH
2
3 payment_with_year_month_table AS (
4 SELECT *,
5     CAST(SUBSTR(paymentDate, 1,4) AS INTEGER)*100 +
6     CAST(SUBSTR(paymentDate, 6,7) AS INTEGER) AS year_month
7 FROM payments p
8 ),
9 customers_by_month_table AS (
10 SELECT pl.year_month, COUNT(*) AS number_of_customers, SUM(pl.amount) AS total
11 FROM payment_with_year_month_table pl
12 GROUP BY pl.year_month
13 ),
14
15 new_customers_by_month_table AS (
16 SELECT pl.year_month,
17     COUNT(DISTINCT customerNumber) AS number_of_new_customers,
18     SUM(pl.amount) AS new_customer_total,
19     (SELECT number_of_customers
20      FROM customers_by_month_table c
21      WHERE c.year_month = pl.year_month) AS number_of_customers,
22     (SELECT total
23      FROM customers_by_month_table c
24      WHERE c.year_month = pl.year_month) AS total
25 FROM payment_with_year_month_table pl
26 WHERE pl.customerNumber NOT IN (SELECT customerNumber
27                                FROM payment_with_year_month_table p2
28                                WHERE p2.year_month < pl.year_month)
29 GROUP BY pl.year_month
30 )
31
32 SELECT year_month,
33     ROUND(number_of_new_customers*100/number_of_customers,1) AS
34     number_of_new_customers_props,
35     ROUND(new_customer_total*100/total,1) AS new_customers_total_props
36 FROM new_customers_by_month_table;
```

The Customer Lifetime Value (LTV), which represents the average amount of money a customer generates. We can then determine how much we can spend on marketing:

```
1 WITH customer_ltv AS (  
2   SELECT orders.customerNumber,  
3   SUM(orderdetails.quantityOrdered * (orderdetails.priceEach - products.buyPrice))  
   AS total_profit_per_customer  
4   FROM orderdetails  
5   JOIN orders ON orders.orderNumber = orderdetails.orderNumber  
6   JOIN products ON orderdetails.productCode = products.productCode  
7   GROUP BY orders.customerNumber  
8 )  
9  
10 SELECT ROUND((SUM (total_profit_per_customer)) / COUNT(customerNumber), 2)  
   AS Customer_Lifetime_Value  
11 FROM customer_ltv;
```