1. *Let's explore the dataset:*

```sql
SELECT
  'Customers' AS table_name,
  (SELECT COUNT(*) FROM pragma_table_info('Customers')) AS number_of_attributes,
  (SELECT COUNT(*) FROM customers) AS number_of_rows

UNION ALL

SELECT
  'Products' AS table_name,
  (SELECT COUNT(*) FROM pragma_table_info('Products')) AS number_of_attributes,
  (SELECT COUNT(*) FROM products) AS number_of_rows

UNION ALL

SELECT
  'ProductLines' AS table_name,
  (SELECT COUNT(*) FROM pragma_table_info('ProductLines')) AS
number_of_attributes,
  (SELECT COUNT(*) FROM productlines) AS number_of_rows

UNION ALL

SELECT
  'Orders' AS table_name,
  (SELECT COUNT(*) FROM pragma_table_info('Orders')) AS number_of_attributes,
  (SELECT COUNT(*) FROM orders) AS number_of_rows

UNION ALL

SELECT
  'OrderDetails' AS table_name,
  (SELECT COUNT(*) FROM pragma_table_info('OrderDetails')) AS
number_of_attributes,
  (SELECT COUNT(*) FROM orderdetails) AS number_of_rows

UNION ALL

SELECT
  'Payments' AS table_name,
  (SELECT COUNT(*) FROM pragma_table_info('Payments')) AS number_of_attributes,
  (SELECT COUNT(*) FROM payments) AS number_of_rows

UNION ALL

SELECT
  'Employees' AS table_name,
  (SELECT COUNT(*) FROM pragma_table_info('Employees')) AS number_of_attributes,
  (SELECT COUNT(*) FROM employees) AS number_of_rows

UNION ALL
```

```sql
SELECT
  'Offices' AS table_name,
  (SELECT COUNT(*) FROM pragma_table_info('Offices')) AS number_of_attributes,
  (SELECT COUNT(*) FROM offices) AS number_of_rows;
```

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

Low Stock:

```sql
SELECT productCode,
    ROUND(SUM(quantityOrdered) * 1.0 / (SELECT quantityInStock
FROM products
WHERE orderdetails.productCode = products.productCode), 2) AS low_stock
 FROM orderdetails
 GROUP BY productCode
 ORDER BY low_stock DESC
 LIMIT 10;
```

Priority for Restocking:

```sql
WITH

low_stock_table AS (
SELECT productCode,
    ROUND(SUM(quantityOrdered) * 1.0/(SELECT quantityInStock
                        FROM products p
                        WHERE od.productCode = p.productCode), 2) AS low_stock
 FROM orderdetails od
 GROUP BY productCode
 ORDER BY low_stock DESC
 LIMIT 10
),

products_to_restock AS (
SELECT productCode,
    SUM(quantityOrdered * priceEach) AS prod_perf
 FROM orderdetails od
 WHERE productCode IN (SELECT productCode
            FROM low_stock_table)
 GROUP BY productCode
 ORDER BY prod_perf DESC
 LIMIT 10
)

SELECT productName, productLine
 FROM products AS p
 WHERE productCode IN (SELECT productCode
            FROM products_to_restock);

```

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

Profit Calculation:

```sql
SELECT orders.customerNumber,
    SUM(quantityOrdered * (priceEach -  buyPrice)) AS profit
FROM orders
JOIN orderdetails ON orders.orderNumber = orderdetails.orderNumber
JOIN products ON orderdetails.productCode = products.productCode
GROUP BY orders.customerNumber
ORDER BY profit DESC;


```

Customer Segmentation:

```sql
WITH customer_profit AS (
 SELECT
   orders.customerNumber,
   SUM(quantityOrdered * (priceEach - buyPrice)) AS profit
 FROM orders
 JOIN orderdetails ON orders.orderNumber = orderdetails.orderNumber
 JOIN products ON orderdetails.productCode = products.productCode
 GROUP BY orders.customerNumber
),
ranked_customers AS (
 SELECT
   customerNumber,
   profit,
   NTILE(10) OVER (ORDER BY profit DESC) AS percentile_rank -- Divide into 10 groups
 FROM customer_profit
)
SELECT
  customerNumber,
  profit,
  CASE
    WHEN percentile_rank = 1 THEN 'VIP'          -- Top 10%
    WHEN percentile_rank BETWEEN 2 AND 5 THEN 'Engaged' -- Next 40%
    ELSE 'Less-Engaged'                  -- Bottom 50%
  END AS customer_segment
FROM ranked_customers
ORDER BY profit DESC;


```

Top 5 VIP Customers:

```
WITH profit_cte AS (
  SELECT o.customerNumber, SUM(quantityOrdered * (priceEach - buyPrice)) AS profit
  FROM products p
  JOIN orderdetails od
     ON p.productCode = od.productCode
  JOIN orders o
     ON o.orderNumber = od.orderNumber
  GROUP BY o.customerNumber
  )

SELECT c.contactLastName, c.contactFirstName, c.city, c.country, profit
FROM customers c
JOIN profit_cte
  ON c.customerNumber = profit_cte.customerNumber
ORDER BY profit DESC
LIMIT 5;


```

Top 5 less engaged Customers:

```
WITH profit_cte AS (
  SELECT o.customerNumber, SUM(quantityOrdered * (priceEach - buyPrice)) AS profit
  FROM products p
  JOIN orderdetails od
     ON p.productCode = od.productCode
  JOIN orders o
     ON o.orderNumber = od.orderNumber
  GROUP BY o.customerNumber
  )

SELECT c.contactLastName, c.contactFirstName, c.city, c.country, profit
FROM customers c
JOIN profit_cte
  ON c.customerNumber = profit_cte.customerNumber
ORDER BY profit ASC
LIMIT 5;


```

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

```sql
WITH

payment_with_year_month_table AS (
SELECT *,
    CAST(SUBSTR(paymentDate, 1,4) AS INTEGER)*100 +
CAST(SUBSTR(paymentDate, 6,7) AS INTEGER) AS year_month
 FROM payments p
),

customers_by_month_table AS (
SELECT p1.year_month, COUNT(*) AS number_of_customers, SUM(p1.amount) AS total
 FROM payment_with_year_month_table p1
 GROUP BY p1.year_month
),

new_customers_by_month_table AS (
SELECT p1.year_month,
    COUNT(DISTINCT customerNumber) AS number_of_new_customers,
    SUM(p1.amount) AS new_customer_total,
    (SELECT number_of_customers
      FROM customers_by_month_table c
     WHERE c.year_month = p1.year_month) AS number_of_customers,
    (SELECT total
      FROM customers_by_month_table c
     WHERE c.year_month = p1.year_month) AS total
 FROM payment_with_year_month_table p1
WHERE p1.customerNumber NOT IN (SELECT customerNumber
                    FROM payment_with_year_month_table p2
                   WHERE p2.year_month < p1.year_month)
GROUP BY p1.year_month
)

SELECT year_month,
    ROUND(number_of_new_customers*100/number_of_customers,1) AS
number_of_new_customers_props,
    ROUND(new_customer_total*100/total,1) AS new_customers_total_props
 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;
```