Ecommerce/Retail sales Analysis

1. **Find Customers Who Purchased Exactly Two Different Products in a Single Month**

Tables: Orders (order_id, customer_id, product_id, order_date)

```
SELECT customer_id
FROM orders
GROUP BY customer_id
Having DISTINCT_COUNT(product_id) = 2
```

2. **Identify Customers Who Haven't Made Any Purchase in the Last 6 Months**

Tables: Customers (customer_id, customer_name), Orders (order_id, customer_id, order_date)

```
SELECT customer_name
FROM customers
WHERE customer_id
NOT IN(
        SELECT customer_id
        FROM orders
        WHERE o.order_date ⩾ (o.order_date - INTERVAL '6 months')
      )
```

3. **List the Top 3 Products per Category Based on Revenue**

Tables: Products (product_id, category), Sales (sale_id, product_id, amount)

```
WITH temptable as
(
SELECT p.category, p.product_id, SUM(s.amount),
RANK() OVER(PARTITION BY p.category ORDER BY sum(s.amount) DESC) as rnk
FROM products p
LEFT JOIN sales s ON s.product_id = p.product_id
GROUP BY p.category, p.product_id
)

SELECT * FROM temptable WHERE rnk⩽3
```

4. **Calculate the Percentage of Orders Delivered Later Than Expected**

Tables: Orders (order_id, order_date, expected_delivery_date, actual_delivery_date)

In order to filter late orders from on time/early delivery I have created two categories

1. Late delivery
2. Ontime delivery

If further analysis on early deliveries need to be analysed then it can extended just by adding an additional case statement with condition

(actual_delivery_date - expected_delivery_date) < 0 as 'early_delivery') and replace(actual_delivery_date - expected_delivery_date)<=0 as <0

```
With
temptable AS
    (
      SELECT order_id,
       CASE
            WHEN (actual_delivery_date -  expected_delivery_date)> 0 THEN
            'late_delivery'
            WHEN (actual_delivery_date -  expected_delivery_date)⩽0 THEN
            'ontime_delivery'
       END AS order_status

    )
Total_orders AS
(
SELECT count(order_status)
FROM temptable
)
SELECT order_status, count(order_status)/total_orders as relative
proportion
FROM temptable
GROUP BY order_status
```

5. **Calculate Each Customer's Lifetime Spending (Customer Lifetime Value)**
Tables: Customers (customer_id), Orders (order_id, customer_id, order_date, amount)

Customer Lifetime Value (CLV) - a metric used by companies to estimate total value customer will generate for a business from their first purchase to last.

Assuming the data available in the dataset as total lifetime of customer I have calculated CLV as

```
SELECT customer_id, SUM(amount)
FROM orders
GROUP BY customer_id
```

6. **Find Employees Who Have Changed Departments More Than Twice**
Tables: Employee_Dept_History (employee_id, department, start_date, end_date)

I have considered **an assumption** which is important considering the information given in the question i.e., **end_date for the current role is considered as NULL** which improves query to filter data that an employee has already worked

```
SELECT employee_id
FROM employee_dept_history
WHERE enddate is NOT NULL
GROUP BY employee_id
HAVING COUNT(department)>2
```

7. **Find Days Where Total Sales Decreased More Than 20% Compared to the Previous Day**
Tables: Daily_Sales (date, total_sales_amount)
I have considered an additional condition to handle errors. Assumption for the condition is sales of a previous day is not NULL which is important to consider during division

```
SELECT date, (total_sales_amount/LAG(total_sales) OVER(ORDER BY date)) as
sales_proportion
FROM  daily sales
WHERE  LAG(total_sales) OVER (ORDER BY date) is NOT NULL AND
(total_sales_amount/LAG(total_sales) OVER(ORDER BY date) ) < 0.8
```

8. **Identify Users Who Made Their First Purchase During a Promotional Campaign**
Tables: Users (user_id), Orders (order_id, user_id, order_date, promo_applied)

For this query I have assumed promo_applied column is having values as
‘yes’ - where promo is applied
‘no’ - where promo is not applied

```
WITH temptable as
(
SELECT user_id, order_id, promo_applied
ROW_NUMBER( ) OVER(PARTITION BY user_id ORDER BY order_date ASC) as rnk
FROM orders
)

SELECT user_id
FROM temptable
WHERE rnk=1 and promo_applied = ‘yes’
```

9. **List Products That Have Never Been Out of Stock**
Tables: Products (product_id, name), Inventory (product_id, inventory_date, stock_available)

```
SELECT product_id, name
FROM products
WHERE product_id   NOT IN(
                 SELECT product_id
                 FROM inventory
                 WHERE stock_available =0
                 )
```

✅ Compare Average Order Values for New vs Returning Customers
Tables: Orders (order_id, customer_id, order_amount, order_date)

Here the main goal is to find average order value of new and returning customers. However, it can't be find directly as there is direct way to group new and returning customers

First step is to differentiate between new customers and returning customers. To find this we need a flag such as number of purchases

Condition assumed:
New customer - number of purchases = 1
Returning customer - number of purchases > 1

```
With cte as(
SELECT customer_id, order_amount, COUNT(order_id) OVER(partition by
customer_id) as order_count
FROM orders)

Temptable as(
SELECT customer_id,
CASE
      WHEN order_count = 1 THEN 'new customer'
      WHEN order_count > 1 THEN 'returning customer'
 END AS customer_type
FROM cte
)

SELECT customer_type , AVG(order_amount) as average_order_value
FROM temptable
GROUP BY customer_type
```