A **Sales** table contains the following columns:

* OrderID (unique identifier for each order)
* TotalAmount (the total value of an order before any discounts)
* DiscountPercentage (the discount applied on the order)
* Quantity (number of products in the order)
* Assume the following data in the Sales table:

| **OrderID** | **TotalAmount** | **DiscountPercentage** | **Quantity** |
| --- | --- | --- | --- |
| 1 | 100.00 | 10 | 2 |
| 2 | 250.00 | 15 | 1 |
| 3 | 350.00 | 5 | 3 |

Your task is to:

1. **Find the absolute value of the discount** (for cases where negative discounts might have been incorrectly entered).
2. **Round the total order amount** to two decimal places after applying the discount.
3. **Round up the total amount** to the nearest whole number for processing purposes (for billing or tax purposes).
4. **Round down the total amount** to the nearest whole number for reporting purposes.
5. **Calculate the total amount after applying a specific discount percentage.**
6. **Find the square root of the total order amount** to determine a customer loyalty score

**1. Absolute Value (ABS):**

You may want to ensure that the discount is always a positive value, even if entered incorrectly as a negative number.

SELECT

OrderID,

DiscountPercentage,

ABS(DiscountPercentage) AS CorrectedDiscountPercentage

FROM

Sales;

**Explanation**: The ABS() function ensures the discount percentage is always positive, which is essential for financial data integrity.

| OrderID | DiscountPercentage | CorrectedDiscountPercentage |

|---------|--------------------|-----------------------------|

| 1 | 10 | 10 |

| 2 | 15 | 15 |

| 3 | 5 | 5 |

**2. Rounding to Two Decimal Places (ROUND):**

After applying the discount, round the final total to two decimal places.

SELECT

OrderID,

TotalAmount,

DiscountPercentage,

ROUND(TotalAmount \* (1 - DiscountPercentage / 100), 2) AS FinalAmount

FROM

Sales;

**Explanation**: The ROUND() function is used to ensure the final amount is rounded to two decimal places for proper billing.

| OrderID | TotalAmount | DiscountPercentage | FinalAmount |

|---------|-------------|--------------------|-------------|

| 1 | 100.00 | 10 | 90.00 |

| 2 | 250.00 | 15 | 212.50 |

| 3 | 350.00 | 5 | 332.50 |

**3. Ceiling Function (CEILING):**

In some cases, you might need to round up the order amount to the nearest whole number (e.g., for tax calculation or billing purposes).

SELECT

OrderID,

TotalAmount,

CEILING(TotalAmount \* (1 - DiscountPercentage / 100)) AS RoundedUpAmount

FROM

Sales;

SELECT

OrderID,

TotalAmount,

CEILING(TotalAmount \* (1 - DiscountPercentage / 100)) AS RoundedUpAmount

FROM

Sales;

**Explanation**: The CEILING() function ensures that any fractional amounts are rounded up to the next whole number, useful for rounding tax or processing fees.

| OrderID | TotalAmount | RoundedUpAmount |

|---------|-------------|-----------------|

| 1 | 100.00 | 91 |

| 2 | 250.00 | 213 |

| 3 | 350.00 | 333 |

**4. Floor Function (FLOOR):**

For reporting purposes, you may want to round down the final amount to the nearest whole number (e.g., rounding for reporting purposes or budget analysis).

SELECT

OrderID,

TotalAmount,

FLOOR(TotalAmount \* (1 - DiscountPercentage / 100)) AS RoundedDownAmount

FROM

Sales;

**Explanation**: The FLOOR() function rounds down the order amount to the nearest integer, which can be useful for simplified reporting or budget purposes.

| OrderID | TotalAmount | RoundedDownAmount |

|---------|-------------|-------------------|

| 1 | 100.00 | 90 |

| 2 | 250.00 | 212 |

| 3 | 350.00 | 332 |

**5. Power Function (POWER):**

You might want to calculate a reward based on the total order amount, like a loyalty points multiplier. For example, you can square the total order amount to reward customers based on their spending.

SELECT

OrderID,

TotalAmount,

POWER(TotalAmount, 2) AS LoyaltyPoints

FROM

Sales;

**Explanation**: The POWER() function calculates the total order amount squared, which might be used in a hypothetical loyalty program to reward higher spending customers.

| OrderID | TotalAmount | LoyaltyPoints |

|---------|-------------|---------------|

| 1 | 100.00 | 10000 |

| 2 | 250.00 | 62500 |

| 3 | 350.00 | 122500 |

**6. Square Root Function (SQRT):**

You can also use the square root of the total order amount to determine a customer loyalty score, assuming this is part of a more complex customer loyalty model.

SELECT

OrderID,

TotalAmount,

SQRT(TotalAmount) AS CustomerLoyaltyScore

FROM

Sales;

**Explanation**: The SQRT() function returns the square root of the total order amount, which could be part of a more advanced model for customer loyalty, where higher total amounts are rewarded with increased loyalty points.

| OrderID | TotalAmount | CustomerLoyaltyScore |

|---------|-------------|----------------------|

| 1 | 100.00 | 10 |

| 2 | 250.00 | 15.81 |

| 3 | 350.00 | 18.71 |