**1. Select All Records from OnlineRetail Table**

SELECT \*

FROM **OnlineRetail**;

**Explanation:**

* **SELECT \***: Retrieves all columns from the specified table.
* **FROM OnlineRetail**: Specifies the table OnlineRetail.
* **Purpose**: To view all data available in the OnlineRetail table.

**2. Update Null stockcode Values to 0**

UPDATE OnlineRetail

SET stockcode = IFNULL(stockcode, 0);

**Explanation:**

* **UPDATE OnlineRetail**: Indicates that we're modifying data in the OnlineRetail table.
* **SET stockcode = IFNULL(stockcode, 0)**: Replaces NULL values in stockcode with 0.
* **Purpose**: To ensure that stockcode has no NULL values, which can prevent issues in calculations or joins.

**3. Calculate Total Cost for Each Record**

SELECT UnitPrice \* Quantity AS TotalCost

FROM OnlineRetail;

**Explanation:**

* **UnitPrice \* Quantity**: Multiplies the unit price by the quantity sold.
* **AS TotalCost**: Aliases the calculated value as TotalCost for readability.
* **Purpose**: To calculate the total cost for each transaction in the OnlineRetail table.

**4. Add totalcost Column to OnlineRetail Table**

ALTER TABLE OnlineRetail

ADD COLUMN totalcost FLOAT;

**Explanation:**

* **ALTER TABLE OnlineRetail**: Modifies the structure of the OnlineRetail table.
* **ADD COLUMN totalcost FLOAT**: Adds a new column named totalcost with a data type of FLOAT.
* **Purpose**: To store the total cost for each transaction directly in the table.

**5. Update totalcost Column with Calculated Values**

UPDATE OnlineRetail

SET totalcost = UnitPrice \* Quantity;

**Explanation:**

* **UPDATE OnlineRetail**: Indicates that we're modifying data in the OnlineRetail table.
* **SET totalcost = UnitPrice \* Quantity**: Calculates the total cost and stores it in the totalcost column.
* **Purpose**: To populate the new totalcost column with the correct values for each record.

**6. Add refunditem Column to OnlineRetail Table**

ALTER TABLE OnlineRetail

ADD COLUMN refunditem INT;

**Explanation:**

* **ADD COLUMN refunditem INT**: Adds a new column named refunditem with a data type of INT.
* **Purpose**: To flag or store information about refunded items within the OnlineRetail table.

**7. Update refunditem for Negative Quantities**

UPDATE OnlineRetail

SET refunditem = Quantity

WHERE Quantity < 0;

**Explanation:**

* **SET refunditem = Quantity**: Assigns the value of Quantity to refunditem.
* **WHERE Quantity < 0**: Applies the update only to records where the Quantity is negative, indicating a refund.
* **Purpose**: To identify refunded items by storing negative quantities in the refunditem column.

**8. Create Refund Table**

CREATE TABLE Refund (

InvoiceRefundNo FLOAT,

ItemNo FLOAT,

Description VARCHAR(255),

Quantity INT,

DateRefund DATETIME,

UnitPrice FLOAT,

CustomerID DECIMAL(10, 0),

Total FLOAT

);

**Explanation:**

* **CREATE TABLE Refund**: Creates a new table named Refund.
* **Column Definitions**: Specifies the columns and their data types for the Refund table.
* **Purpose**: To have a separate table that specifically holds information about refunded transactions.

**9. Select Records with Negative Quantities**

SELECT InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, totalcost

FROM OnlineRetail

WHERE Quantity < 0;

**Explanation:**

* **SELECT ...**: Retrieves specific columns from the OnlineRetail table.
* **WHERE Quantity < 0**: Filters records to include only those with negative quantities (refunds).
* **Purpose**: To extract all refunded transactions for further processing or analysis.

**10. Insert Refunded Transactions into Refund Table**

INSERT INTO Refund (InvoiceRefundNo, ItemNo, Description, Quantity, DateRefund, UnitPrice, CustomerID, Total)

SELECT InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, totalcost

FROM OnlineRetail

WHERE Quantity < 0;

**Explanation:**

* **INSERT INTO Refund (...) SELECT ...**: Inserts data into the Refund table based on a SELECT query.
* **Mapping Columns**: Columns from OnlineRetail are mapped to the corresponding columns in Refund.
* **Purpose**: To move refund data into the Refund table for dedicated storage and analysis.

**11. Adjust Total and Quantity in Refund Table**

UPDATE Refund

SET Total = Total \* -1;

UPDATE Refund

SET Quantity = Quantity \* -1;

**Explanation:**

* **Total = Total \* -1**: Converts negative totals to positive values.
* **Quantity = Quantity \* -1**: Converts negative quantities to positive numbers.
* **Purpose**: To standardize the data by making quantities and totals positive, representing the actual amounts refunded.

**12. Convert InvoiceDate to String Format**

UPDATE OnlineRetail

SET InvoiceDate = DATE\_FORMAT(InvoiceDate, '%Y-%m-%d %H:%i:%s');

**Explanation:**

* **DATE\_FORMAT(InvoiceDate, '%Y-%m-%d %H:%i:%s')**: Formats the InvoiceDate as a string in the specified format.
* **Purpose**: To store InvoiceDate as a string. Note that changing date columns to strings can limit date-related functionalities.

**13. Delete Refunded Transactions from OnlineRetail**

DELETE FROM OnlineRetail

WHERE Quantity < 0;

**Explanation:**

* **DELETE FROM OnlineRetail**: Removes records from the OnlineRetail table.
* **WHERE Quantity < 0**: Targets only records with negative quantities (refunds).
* **Purpose**: To remove refunded transactions from the main sales table after they've been moved to the Refund table.

**14. Delete Zero Total Refunds from Refund Table**

DELETE FROM Refund

WHERE Total = 0;

**Explanation:**

* **WHERE Total = 0**: Filters out any refund records where the total amount is zero.
* **Purpose**: To clean up the Refund table by removing invalid or unnecessary records.

**15. Select Refunds with Specific Descriptions**

SELECT \*

FROM Refund

WHERE Description LIKE 'white hanging heart%';

**Explanation:**

* **LIKE 'white hanging heart%'**: Uses a wildcard to find descriptions that start with 'white hanging heart'.
* **Purpose**: To retrieve all refund records for items whose descriptions match a specific pattern.

**16. Create Top10Item View**

CREATE VIEW Top10Item AS

SELECT \*

FROM OnlineRetail

ORDER BY totalcost DESC

LIMIT 10;

**Explanation:**

* **CREATE VIEW Top10Item**: Defines a new view named Top10Item.
* **ORDER BY totalcost DESC LIMIT 10**: Orders the records by totalcost in descending order and limits the result to the top 10 records.
* **Purpose**: To create a quick reference for the top 10 items based on total cost.

**17. Delete Specific Records from OnlineRetail**

DELETE FROM OnlineRetail

WHERE Description LIKE 'amazon fee';

**Explanation:**

* **WHERE Description LIKE 'amazon fee'**: Filters records where the description matches 'amazon fee'.
* **Purpose**: To remove non-sales or fee-related records from the OnlineRetail table.

**18. Delete Unwanted Records from Refund**

DELETE FROM Refund

WHERE Description LIKE 'amazon fee'

OR Description LIKE 'Discount'

OR Description LIKE 'Manual'

OR Description LIKE 'bank cha%'

OR Description LIKE 'sam%'

OR Description LIKE 'pos%';

**Explanation:**

* **OR**: Combines multiple conditions to filter records that match any of the specified patterns.
* **LIKE 'bank cha%'**: Uses a wildcard to match any description starting with 'bank cha'.
* **Purpose**: To clean the Refund table by removing records that are not actual product refunds or are irrelevant.

**19. Create Top10Refund View**

CREATE VIEW Top10Refund AS

SELECT \*

FROM Refund

ORDER BY Total DESC

LIMIT 10;

**Explanation:**

* **CREATE VIEW Top10Refund**: Defines a new view named Top10Refund.
* **Purpose**: To provide quick access to the top 10 highest refund transactions.

**20. Create Revenue View**

CREATE VIEW Revenue AS

SELECT SUM(totalcost) AS Revenue

FROM OnlineRetail;

**Explanation:**

* **SUM(totalcost)**: Calculates the total revenue by summing up the totalcost of all transactions.
* **Purpose**: To have a view that displays the total revenue generated from sales.

**21. Create TotalRefund View**

CREATE VIEW TotalRefund AS

SELECT SUM(Total) AS TotalRefund

FROM Refund;

**Explanation:**

* **SUM(Total)**: Calculates the total amount refunded.
* **Purpose**: To easily access the total refunds issued.

**22. Create Profit View**

CREATE VIEW Profit AS

SELECT

(SELECT SUM(totalcost) FROM OnlineRetail) - (SELECT SUM(Total) FROM Refund) AS Profit;

**Explanation:**

* **Subqueries**: Calculates total revenue and total refunds separately.
* **AS Profit**: Aliases the result as Profit.
* **Purpose**: To compute the net profit by subtracting total refunds from total sales revenue.

**23. Delete Records with Null Descriptions**

DELETE FROM OnlineRetail

WHERE Description IS NULL;

**Explanation:**

* **WHERE Description IS NULL**: Targets records where the Description field is empty.
* **Purpose**: To clean the data by removing incomplete or invalid records.

**24. Switch to Portfolio Database**

USE Portfolio;

**Explanation:**

* **USE Portfolio**: Changes the context to the Portfolio database.
* **Purpose**: To execute subsequent commands within the Portfolio database.

**25. Select Distinct Customer IDs from Refund**

SELECT DISTINCT CustomerID

FROM Refund;

**Explanation:**

* **SELECT DISTINCT CustomerID**: Retrieves unique customer IDs from the Refund table.
* **Purpose**: To get a list of all customers who have received refunds