Merging and Querying SQL Tables

1. Overview

This process involves merging four tables: `Customers`, `InvoiceItems`, `Invoices`, and `Products` into a single, combined table named `SalesSummary`. This allows for easier querying and reporting on sales data, including customer information, product details, and invoice-related details.

2. Table Descriptions and Relationships

2.1 `Customers` Table

- **Description**: Stores customer information.
- **Key Columns**:
    - `CustomerID`: Unique identifier for each customer.
    - `Country`: Customer's country.

2.2 `Invoices` Table

- **Description**: Stores information about invoices.
- **Key Columns**:
    - `InvoiceNo`: Unique identifier for each invoice.
    - `CustomerID`: Links to the `Customers` table.
    - `InvoiceDate`: Date and time when the invoice was created.

2.3 `InvoiceItems` Table

- **Description**: Contains individual items included in an invoice.
- **Key Columns**:
    - `ID`: Unique identifier for each row.
    - `InvoiceNo`: Links to the `Invoices` table.
    - `StockCode`: Links to the `Products` table.
    - `Quantity`: Quantity of the product.
    - `TotalPrice`: Total price for the specific item.

2.4 `Products` Table

- **Description**: Stores product details.
- **Key Columns**:
    - `StockCode`: Unique identifier for each product.
    - `Description`: Description of the product.
    - `UnitPrice`: Price per unit of the product.

3. Process of Merging Tables

3.1 Create the Combined Table

A new table named `SalesSummary` will be created to combine data from all four tables.

SQL Command:

```
CREATE TABLE SalesSummary (
    CustomerID INT,
    Country VARCHAR(100),
    InvoiceNo INT,
    InvoiceDate DATETIME,
    StockCode VARCHAR(100),
    Description VARCHAR(255),
    UnitPrice DECIMAL(10, 2),
    Quantity INT,
    TotalPrice DECIMAL(10, 2),
    TotalItemPrice AS (Quantity * UnitPrice)  -- Calculated field for total item price
);
```

3.2 Insert Data into the Combined Table

Data is inserted into `SalesSummary` by joining the `Customers`, `Invoices`, `InvoiceItems`, and `Products` tables.

```
INSERT INTO SalesSummary (CustomerID, Country, InvoiceNo, InvoiceDate, StockCode,
Description, UnitPrice, Quantity, TotalPrice)
SELECT
    c.CustomerID,
    c.Country,
    i.InvoiceNo,
    i.InvoiceDate,
    ii.StockCode,
    p.Description,
    p.UnitPrice,
    ii.Quantity,
    ii.TotalPrice
FROM
    dbo.Customers AS c
JOIN
    dbo.Invoices AS i
    ON c.CustomerID = i.CustomerID
JOIN
    dbo.InvoiceItems AS ii
    ON i.InvoiceNo = ii.InvoiceNo
JOIN
    dbo.Products AS p
    ON ii.StockCode = p.StockCode;
```

3.3 Data Example

```
SELECT TOP 10 *
FROM dbo.SalesSummary;
```

CustomerID	Country	InvoiceNo	InvoiceDate	StockCode	Description	UnitPrice	Quantity	TotalPrice	TotalItemPrice
13426	United Kingdom	579439	2011-11-29 13:29:00.000	20914	SET/5 RED RETROSPOT LID GLASS BOWLS	2.55	32	81.60	81.60

all

4. Example Queries

Once the data is loaded into the SalesSummary table, the following types of queries can be run for reporting and analysis.

4.1 Total Sales per Customer

This query calculates the total sales per customer, grouped by their `CustomerID` and `Country`.

```
SELECT CustomerID, Country, SUM(TotalItemPrice) AS TotalSales
FROM SalesSummary
GROUP BY CustomerID, Country;
```

Total_Sales_per_Customer

4.2 Top-Selling Products

This query retrieves the top-selling products based on the total quantity sold.

```
SELECT StockCode, Description, SUM(Quantity) AS TotalQuantitySold

FROM SalesSummary
GROUP BY StockCode, Description
ORDER BY TotalQuantitySold DESC;
```

Top-Selling-Products

4.3 Invoices within a Specific Date Range

This query shows all invoices issued between two specific dates, along with the total amount for each invoice.

```
SELECT InvoiceNo, InvoiceDate, CustomerID, Country, SUM(TotalItemPrice) AS
TotalInvoiceAmount
FROM SalesSummary
WHERE InvoiceDate BETWEEN '2011-11-01' AND '2011-11-30'
GROUP BY InvoiceNo, InvoiceDate, CustomerID, Country;
```

Invoices_within_a_Specific_Date Range

5. Key Considerations

- **Data Integrity**: Ensure the correct relationships between tables are maintained using primary and foreign keys.
- **Indexes**: Consider adding indexes to the `SalesSummary` table on frequently queried fields such as `CustomerID`, `InvoiceNo`, or `StockCode` to improve performance.
- **Rounding Issues**: Double-check calculations, especially `TotalItemPrice`, to ensure consistency across different systems and database environments.

6. Conclusion

By combining the four tables into a single, unified `SalesSummary` table, the process of querying sales, customer information, and product data becomes much easier. The new table structure supports various business queries such as customer sales reporting, product analysis, and invoice tracking.