```sql
USE WideWorldImporters

/* Оптимизируем запрос
Цель: Используем все свои полученные знания для оптимизации сложного запроса.
Вариант 2. Оптимизируйте запрос по БД WorldWideImporters.
Приложите текст запроса со статистиками по времени и операциям ввода вывода,
  опишите кратко ход рассуждений при оптимизации.
Используем DMV, хинты и все прочее для сложных случаев */


SET STATISTICS IO, TIME ON
-- исходный запрос
Select
    ord.CustomerID,
    det.StockItemID,
    SUM(det.UnitPrice),
    SUM(det.Quantity),
    COUNT(ord.OrderID)
FROM Sales.Orders AS ord
    JOIN Sales.OrderLines AS det
        ON det.OrderID = ord.OrderID
    JOIN Sales.Invoices AS Inv
        ON Inv.OrderID = ord.OrderID
    JOIN Sales.CustomerTransactions AS Trans
        ON Trans.InvoiceID = Inv.InvoiceID
    JOIN Warehouse.StockItemTransactions AS ItemTrans
        ON ItemTrans.StockItemID = det.StockItemID
WHERE Inv.BillToCustomerID != ord.CustomerID
    AND (Select
            SupplierId
        FROM Warehouse.StockItems AS It
            Where It.StockItemID = det.StockItemID) = 12
    AND (SELECT
            SUM(Total.UnitPrice*Total.Quantity)
        FROM Sales.OrderLines AS Total
        Join Sales.Orders AS ordTotal
            On ordTotal.OrderID = Total.OrderID
        WHERE ordTotal.CustomerID = Inv.CustomerID) > 250000
    AND DATEDIFF(dd, Inv.InvoiceDate, ord.OrderDate) = 0
GROUP BY ord.CustomerID, det.StockItemID
ORDER BY ord.CustomerID, det.StockItemID;


/* Результат оптимизации представлен ниже. Что сделано:
    2.1. В оригинальном запросе присутствуют две выборки с соединением из таблиц
      Sales.OrderLines и Sales.Orders (в инструкциях FROM и WHERE)
        Чтобы исключить повторную выборку, вынесем ее в CTE CteCust с оконной
          функцией для последующей проверки условия SUM
          (Total.UnitPrice*Total.Quantity) > 250000
    2.2. Во второй CTE CteInv добавлено соединение с таблицей Invoices и условия по
       полям таблицы Invoices. Вторая CTE предназначена больше для улучшения
      читаемости кода,
        в оптимизацию она вклада не вносит
    2.3. Анализ промежуточного плана запроса (см. файл HW26_Key_Lookup.sqlplan)
      показал, что наиболее "тяжелый" оператор - Key Lookup по таблице
```

```
      Sales.Invoices
        Чтобы исключить Key Lookup, сделаем индекс с включением необходимых полей
          для выборки: */
    CREATE NONCLUSTERED INDEX [FK_Sales_Invoices_OrderID_Inc] ON [Sales].[Invoices]
    (
        [OrderID] ASC
    ) INCLUDE ([BillToCustomerID], InvoiceDate) WITH (PAD_INDEX = OFF,
      STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF,
      ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [USERDATA]
    GO

    /* 2.4  Анализируем пакетный план запроса из двух запросов (исходного и
      оптимизированного) (HW26.sqlplan) и видим что оптимизированный запрос
      составляет 25% стоимости.
        Статистика (см. файл HW26 Stats.pdf) показывает значительное уменьшение
          логических чтений (1544 vs 118926), сканирований (28 vs 15869 ).
        Результаты запросов совпадают по содержанию и количеству строк (3619)
        Есть основание считать, что оптимизация состоялась.
        */
-- оптимизированный запрос
With CteCust AS
(SELECT
    CustomerID, OrderDate,
    StockItemID,
    UnitPrice,
    Quantity,
    OrderID, detSum
 FROM (Select
        ord.CustomerID,
        ord.OrderDate,
        det.StockItemID,
        det.UnitPrice,
        det.Quantity,
        ord.OrderID,
        SUM(det.UnitPrice*det.Quantity) OVER (PARTITION BY ord.CustomerId) AS
          detSum
    FROM Sales.Orders AS ord
    JOIN Sales.OrderLines AS det
        ON det.OrderID = ord.OrderID
        )  AS o
Where o.detSum > 250000
    AND (Select
            SupplierId
        FROM Warehouse.StockItems AS It
            Where It.StockItemID = o.StockItemID) = 12
),
CteInv As (Select det.CustomerID, OrderDate,
    StockItemID,
    UnitPrice,
    Quantity,
    det.OrderID,
    Inv.InvoiceID
    FROM CteCust AS det
    JOIN Sales.Invoices AS Inv
        ON Inv.OrderID = det.OrderID
WHERE
```

```
Inv.BillToCustomerID != det.CustomerID
    AND
    DATEDIFF(dd, Inv.InvoiceDate, det.OrderDate)  = 0
)

Select  det.CustomerID,
    det.StockItemID,
    SUM(det.UnitPrice),
    SUM(det.Quantity),
    COUNT(det.OrderID)
FROM CteInv as det
    INNER JOIN Sales.CustomerTransactions AS Trans
        ON Trans.InvoiceID = det.InvoiceID
    INNER JOIN Warehouse.StockItemTransactions AS ItemTrans
        ON ItemTrans.StockItemID = det.StockItemID
GROUP BY det.CustomerID, det.StockItemID
ORDER BY det.CustomerID, det.StockItemID;
```