**Question:**

*A company would like to view the sum of all order amounts placed by a customer (account) by each month for the current year.*

*The sum needs to be calculated as soon as a new order is created save it to the account object in a custom long text field.*

*Note: EffectiveDate, TotalAmount on orders can be used for doing this calculation.*

**Answer:**

***Requirement: - A company would like to view the sum of all order amounts placed by a customer (account) by each month for the current year.***

We have many ways to achieve this functionality but salesforce encourage us to minimize the code that’s why I have created standard Order Report. The report properties are as below

* Group Rows:
  + Order Start Date (EffectiveDate)
  + Account Name
* Filters:
  + Filter by Order Start Date
    - Date: - Order Start Date
    - Range: - Current FY
* Columns:
  + Order Start Date (EffectiveDate) :- Group Date By – Calendar Month
  + Account Name
  + Order Amount (Total Amount)
  + Detail Row: - True
  + Sub Total: - True



Report image with sample data

***Requirement: - The sum needs to be calculated as soon as a new order is created save it to the account object in a custom long text field.***

***Note: EffectiveDate, TotalAmount on orders can be used for doing this calculation.***

**Answer:**

To Achieve above requirement, I have created Triggers and it’s Handler class and Test class with 100% code coverage.

|  |  |  |
| --- | --- | --- |
| **Object** | **Trigger** | **Handler Class** |
| OrderItem | RollupOrderLine | RollupOrderLineHelper |
| Order | RollupOrder | RollupOrderHandler |
| Account | SumOrders | SumOrdersHandler |

**RollupOrderLine.apxt**

trigger RollupOrderLine on OrderItem (after insert, after update, after delete, after undelete) {

RollupOrderLineHelper.rollupOrderLine(Trigger.old,Trigger.new); //Call Handler Method

}

**RollupOrderLineHelper.apxc**

public class RollupOrderLineHelper {

public static void rollupOrderLine(list<OrderItem> OrderlineListOld,list<OrderItem> OrderlineListNew){

Set<Id> orderIds = new Set<Id>(); // Store Set of Order Id

if(OrderlineListOld != null && OrderlineListOld.size() > 0) {

for(OrderItem record: OrderlineListOld) {

orderIds.add(record.OrderId); // Adding Ids to Set

}

}

if(OrderlineListNew != null && OrderlineListNew.size() > 0) {

for(OrderItem record: OrderlineListNew) {

orderIds.add(record.OrderId); // Adding Ids to set

}

}

if(OrderlineListNew.size() > 0){

update [SELECT Id FROM Order WHERE Id IN :orderIds]; // Update Order

}

}

}

**RollupOrder.apxt**

trigger RollupOrder on Order (after insert, after update, after delete, after undelete) {

RollupOrderHandler.rollupOrder(Trigger.old, Trigger.new); // Call Handler Method

}

**RollupOrderHandler.apxc**

public class RollupOrderHandler {

public static void rollupOrder(list<Order> orderListOld,list<Order> orderListNew){

Set<Id> accountIds = new Set<Id>(); // To store set of Account Id

if(orderListOld != null && orderListOld.size() > 0) {

for(Order record: orderListOld) {

accountIds.add(record.AccountId); // Add Account Id to set

}

}

if(orderListNew != null && orderListNew.size() > 0) {

for(Order record: orderListNew) {

accountIds.add(record.AccountId); // Add Account Id to set

}

}

If(accountIds.size() > 0){

update [SELECT Id FROM Account WHERE Id IN :accountIds];

}

}

}

**SumOrders.apxt**

trigger SumOrders on Account (before update) {

SumOrdersHandler.sumOrders(Trigger.new, Trigger.newmap); // Call Handler Method

}

**SumOrdersHandler.apxc**

public class SumOrdersHandler {

Public static void sumOrders(list<Account> accList,Map<Id,Account> accMap){

for(Account record: accList) {

record.Grand\_Total\_Order\_Amount\_\_c = null; // Setting Grand Total Amount to null

}

for(AggregateResult result: [SELECT SUM(TotalAmount) Amt, AccountId Id FROM Order WHERE AccountId IN :accList GROUP BY AccountId]) {

accMap.get((Id)result.get('Id')).Grand\_Total\_Order\_Amount\_\_c = string.valueof(result.get('Amt')); // Assign Sum of all order Amount to Grand Total Order Amount

}

}

}

**RollupTotalOrderAmountTest.apxc**

@isTest

public class RollupTotalOrderAmountTest {

@isTest

Public static void sumOrderTest(){

Account acc = new Account();

acc.Name = 'Test Account';

insert acc;

Contract cont = new Contract();

cont.Name = 'Test Contract';

cont.AccountId = acc.Id;

cont.Status = 'Draft';

insert cont;

Product2 p = new Product2();

p.Name = ' Test Product ';

p.Description='Test Product';

p.productCode = 'TP01';

p.isActive = true;

insert p;

Id pricebookId = Test.getStandardPricebookId();

PricebookEntry standardPrice = new PricebookEntry();

standardPrice.Pricebook2Id = pricebookId;

standardPrice.Product2Id = p.Id;

standardPrice.UnitPrice = 1;

standardPrice.IsActive = true;

standardPrice.UseStandardPrice = false;

insert standardPrice ;

Order o = new Order();

o.Name = 'Test Order';

o.Status = 'Draft';

o.EffectiveDate = system.today();

o.EndDate = system.today() + 4;

o.AccountId = acc.id;

o.Pricebook2Id = pricebookId ;

insert o;

OrderItem i = new OrderItem();

i.OrderId = o.id;

i.Quantity = 1;

i.UnitPrice = 1000;

i.Product2id = p.id;

i.PricebookEntryId=standardPrice.id;

insert i;

}

}