//Database.executeBatch(new dm\_QuoteMatchedBatch(),100);

//Validate Quote alll- non error

//Validate for Matched Records

public class dm\_QuoteMatchedBatch implements Database.Batchable<sObject>,Database.Stateful {

//Validation

//Compare the Total

public Map<SBQQ\_\_Quote\_\_c,Decimal> quoteToProdAmount = new Map<SBQQ\_\_Quote\_\_c,Decimal>();

public database.querylocator start(Database.BatchableContext BC)

{

String queryCPQQuote = 'Select Id,Name,SBQQ\_\_Source\_\_c,SBQQ\_\_BillingFrequency\_\_c,SBQQ\_\_Opportunity2\_\_c,SBQQ\_\_Opportunity2\_\_r.Name,SBQQ\_\_Opportunity2\_\_r.StageName,SBQQ\_\_Account\_\_c,'

+'SBQQ\_\_NetAmount\_\_c ,SBQQ\_\_SubscriptionTerm\_\_c,SBQQ\_\_StartDate\_\_c,SBQQ\_\_EndDate\_\_c from SBQQ\_\_Quote\_\_c where '

+' DM\_Quote\_\_c = true and SBQQ\_\_Primary\_\_c = true and Dm\_Check\_\_c = true';//No Manual fix

// + ' AND DM\_ErrorQuote\_\_c = true and Manual\_Fix\_\_c = false and SBQQ\_\_Primary\_\_c = true';//Manual Fix//';//and DM\_ErrorQuote\_\_c = false';//AND Manual\_Fix\_\_c = true';//DM\_ErrorQuote\_\_c = true and Manual\_Fix\_\_c = true and SBQQ\_\_Source\_\_c != null

String queryCPQQuoteSource = 'Select Id,Name,SBQQ\_\_Source\_\_c,SBQQ\_\_BillingFrequency\_\_c,SBQQ\_\_Opportunity2\_\_c,SBQQ\_\_Opportunity2\_\_r.Name,SBQQ\_\_Opportunity2\_\_r.StageName,SBQQ\_\_Account\_\_c,'

+'SBQQ\_\_NetAmount\_\_c ,SBQQ\_\_SubscriptionTerm\_\_c,SBQQ\_\_StartDate\_\_c,SBQQ\_\_EndDate\_\_c from SBQQ\_\_Quote\_\_c where '

+' DM\_Quote\_\_c = true and SBQQ\_\_Primary\_\_c = true and SBQQ\_\_Source\_\_c != null';

return Database.getQueryLocator(queryCPQQuote);//queryCPQQuoteSource

}

public void execute(Database.BatchableContext BC,List<SBQQ\_\_Quote\_\_c> quotes)

{

//-----------------Final Validation Quote and QL validation - Amount

verifyMigratedQuoteAmount(quotes,quoteToProdAmount);

//Final With Cloned Quotes

//verifyMigratedQuoteAmountClonedQuotes(quotes,quoteToProdAmount);

}

public static void verifyMigratedQuoteAmount(List<SBQQ\_\_Quote\_\_c> quotes,Map<SBQQ\_\_Quote\_\_c,Decimal> quoteToProdAmount )

{

//Get the Quote amount

//Get the Opp Prod Base - dm Prod = true Quote Amount

//And Map it.

Map<Id,SBQQ\_\_Quote\_\_c> quoteMap = new Map<Id,SBQQ\_\_Quote\_\_c>(quotes);

//Prod Total

Map<Id,Decimal> prdOppProdTotal = new Map<Id,Decimal>();

//Data Migration Quotes

Map<Id,Decimal> dmQuoteTotal = new Map<Id,Decimal>();

Map<Id,Id> quoteOldNewMap = new Map<Id,Id>();

List<dm\_OpportunityProducts\_\_c> prodOppProds = [Select Id,TOTALPRICE\_\_c,Quote\_Temp\_\_c from dm\_OpportunityProducts\_\_c where

dm\_Product\_Name\_New\_\_c != null

and dm\_ProductId\_New\_\_c != null

//and From\_Prod\_\_c=true

and Quote\_Temp\_\_c in :quoteMap.keySet()];//quoteMap.keySet()];

//Prod Net Total for Quotes.

for(dm\_OpportunityProducts\_\_c oppProd : prodOppProds)

{

if(prdOppProdTotal.containsKey(oppProd.Quote\_Temp\_\_c))

{

Decimal amount = prdOppProdTotal.get(oppProd.Quote\_Temp\_\_c) + oppProd.TOTALPRICE\_\_c;

prdOppProdTotal.put(oppProd.Quote\_Temp\_\_c,amount);

}

else {

prdOppProdTotal.put(oppProd.Quote\_Temp\_\_c,oppProd.TOTALPRICE\_\_c);

//system.debug('Total price-->'+oppProd.TOTALPRICE\_\_c);

}}

//Quote Net Total

for(SBQQ\_\_Quote\_\_c q : quoteMap.Values())

{

if(dmQuoteTotal.containsKey(q.Id))//q.Id))

{

Decimal amount = dmQuoteTotal.get(q.Id);//q.Id);

dmQuoteTotal.put(q.Id,amount);//q.Id,amount);

}

else {

dmQuoteTotal.put(q.Id,q.SBQQ\_\_NetAmount\_\_c);//q.Id,q.SBQQ\_\_NetAmount\_\_c);

// system.debug('Net Amount-->'+q.SBQQ\_\_NetAmount\_\_c);

}}

for(SBQQ\_\_Quote\_\_c q : quoteMap.Values())

{

if(dmQuoteTotal.containsKey(q.Id) && prdOppProdTotal.containsKey(q.Id))

{

// DIff

//if(dmQuoteTotal.get(q.Id) != prdOppProdTotal.get(q.Id))

if(dmQuoteTotal.get(q.Id) == prdOppProdTotal.get(q.Id))

//if(dmQuoteTotal.get(q.SBQQ\_\_Source\_\_c) == prdOppProdTotal.get(q.SBQQ\_\_Source\_\_c))

{ if(! quoteToProdAmount.containsKey(q))

quoteToProdAmount.put(q,prdOppProdTotal.get(q.Id));}//q.Id));}

}

}

//sendEmailWithAttachment(quoteToProdAmount);

}

/\* public static void verifyMigratedQuoteAmountClonedQuotes(List<SBQQ\_\_Quote\_\_c> quotes,Map<SBQQ\_\_Quote\_\_c,Decimal> quoteToProdAmount )

{

//Get the Quote amount

//Get the Opp Prod Base - dm Prod = true Quote Amount

//And Map it.

Map<Id,SBQQ\_\_Quote\_\_c> quoteMap = new Map<Id,SBQQ\_\_Quote\_\_c>(quotes);

//Prod Total

Map<Id,Decimal> prdOppProdTotal = new Map<Id,Decimal>();

//Data Migration Quotes

Map<Id,Decimal> dmQuoteTotal = new Map<Id,Decimal>();

Map<Id,Id> quoteOldNewMap = new Map<Id,Id>();

for(SBQQ\_\_Quote\_\_c q : quotes)

{

if(q.SBQQ\_\_Source\_\_c != null)

quoteOldNewMap.put(q.SBQQ\_\_Source\_\_c,q.Id);

}

List<dm\_OpportunityProducts\_\_c> prodOppProds = [Select Id,TOTALPRICE\_\_c,Quote\_Temp\_\_c from dm\_OpportunityProducts\_\_c where

dm\_Product\_Name\_New\_\_c != null

and dm\_ProductId\_New\_\_c != null

//and From\_Prod\_\_c=true

and Quote\_Temp\_\_c in :quoteOldNewMap.keySet()];//quoteMap.keySet()];

//Prod Net Total for Quotes.

for(dm\_OpportunityProducts\_\_c oppProd : prodOppProds)

{

if(prdOppProdTotal.containsKey(oppProd.Quote\_Temp\_\_c))

{

Decimal amount = prdOppProdTotal.get(oppProd.Quote\_Temp\_\_c) + oppProd.TOTALPRICE\_\_c;

prdOppProdTotal.put(oppProd.Quote\_Temp\_\_c,amount);

}

else prdOppProdTotal.put(oppProd.Quote\_Temp\_\_c,oppProd.TOTALPRICE\_\_c);

}

//Quote Net Total

for(SBQQ\_\_Quote\_\_c q : quoteMap.Values())

{

if(dmQuoteTotal.containsKey(q.SBQQ\_\_Source\_\_c))//q.Id))

{

Decimal amount = dmQuoteTotal.get(q.SBQQ\_\_Source\_\_c);//q.Id);

dmQuoteTotal.put(q.SBQQ\_\_Source\_\_c,amount);//q.Id,amount);

}

else dmQuoteTotal.put(q.SBQQ\_\_Source\_\_c,q.SBQQ\_\_NetAmount\_\_c);//q.Id,q.SBQQ\_\_NetAmount\_\_c);

}

for(SBQQ\_\_Quote\_\_c q : quoteMap.Values())

{

if(dmQuoteTotal.containsKey(q.SBQQ\_\_Source\_\_c) && prdOppProdTotal.containsKey(q.SBQQ\_\_Source\_\_c))// q.Id) && prdOppProdTotal.containsKey(q.Id))

{

// DIff

// if(dmQuoteTotal.get(q.Id) != prdOppProdTotal.get(q.Id))

//if(dmQuoteTotal.get(q.Id) == prdOppProdTotal.get(q.Id))

// if(dmQuoteTotal.get(q.SBQQ\_\_Source\_\_c) != prdOppProdTotal.get(q.SBQQ\_\_Source\_\_c))

// {

if(! quoteToProdAmount.containsKey(q))

quoteToProdAmount.put(q,prdOppProdTotal.get(q.SBQQ\_\_Source\_\_c));}//q.Id));}

// }

}

//sendEmailWithAttachment(quoteToProdAmount);

}

\*/

//Validate Amount

public static void sendEmailWithAttachment(Map<SBQQ\_\_Quote\_\_c,Decimal> quoteToProdAmount)

{

String subject = 'SandBox Quote Amount with Prod Amount';

String body = 'Amount Validation';

// Define the email

Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage();

// Define email file attachment list

Messaging.EmailFileAttachment[] emailAttList = new List<Messaging.EmailFileAttachment>();

// Creating the CSV file

if(! quoteToProdAmount.isEmpty()){

// Creating the CSV file for error

String finalstr = 'Quote Id,Name,OpportunityId,Opportunity Name,Opportunity Stage,Quote Total Amount,Prod Total,Quote Term,'

+ 'Start\_Date\_\_c, End\_Date\_\_c\n';

String attName = 'Quote\_WithMatchAmount\_' + system.now().format('YYYYMMDDhhmm') + '.csv';

for(SBQQ\_\_Quote\_\_c q : quoteToProdAmount.keySet()){

string recordString = q.Id+'","'+q.Name+'","'+q.SBQQ\_\_Opportunity2\_\_c+'","'+q.SBQQ\_\_Opportunity2\_\_r.Name+'","'+q.SBQQ\_\_Opportunity2\_\_r.StageName+'","'+q.SBQQ\_\_NetAmount\_\_c +'","'+ quoteToProdAmount.get(q)+'","'+q.SBQQ\_\_SubscriptionTerm\_\_c +'","'+

q.SBQQ\_\_StartDate\_\_c+'","'+q.SBQQ\_\_EndDate\_\_c+'"\n';

finalstr = finalstr +recordString;

}

// Create the email attachment

Messaging.EmailFileAttachment efa = new Messaging.EmailFileAttachment();

efa.setFileName(attName);

efa.setBody(Blob.valueOf(finalstr));

emailAttList.add(efa);

}

// Sets the paramaters of the email

email.setSubject( subject );

//Need to get the List

List<User> userList = new List<User>();

List<String> toAddresses = new List<String>();

toAddresses.add('revathi.mohan@nteligroup.com');//-------------------TO DO

toAddresses.add('anand.kumaraswamy@nteligroup.com');

toAddresses.add('nageswaran.athithya@nteligroup.com');//-------------------TO DO

toAddresses.add('dipa.rathod@nteligroup.com');

toAddresses.add('swetha.kb@nteligroup.com');

toAddresses.add('charumathi@nteligroup.com');

email.setToAddresses( toAddresses);

email.setPlainTextBody( body );

email.setFileAttachments(emailAttList);

// Sends the email

Messaging.SendEmailResult [] r =

Messaging.sendEmail(new Messaging.SingleEmailMessage[] {email});

}

// The batch job finishes

public void finish(Database.BatchableContext bc){

//Error

if(quoteToProdAmount.size()>0)

sendEmailWithAttachment(quoteToProdAmount);}

}

@isTest

public class dm\_QuoteMatchedBatchTest {

public static decimal NetAmount;

public static decimal TotalPrice;

@testsetup

static void setup(){

//Account Creation

Account objAccount = TestDataUtility.createAccount(

'Test Account'

);

insert objAccount;

List<Account> accId = [Select ID from Account where Name = 'Test Account' Limit 1];

// Opportunity Creation

Opportunity objOpportunity = TestDataUtility.createOpportunity(

'Test Opportunity',

objAccount.Id,

'Closed Won',

Date.Today().Adddays(30),

'New',

Test.getStandardPricebookId()

);

insert objOpportunity;

// Opportunity Creation

Opportunity objOpportunity1 = TestDataUtility.createOpportunity(

'Test Opportunity',

objAccount.Id,

'Closed Won',

Date.Today().Adddays(30),

'New',

Test.getStandardPricebookId()

);

insert objOpportunity1;

List<Opportunity> oppId = [Select ID from Opportunity where Name = 'Test Opportunity' Limit 1];

// Product Creation

Product2 objproduct = new Product2();

objProduct.Name='Insights Value Assurance-CPQ';

objProduct.ProductCode='Insights Value Assurance';

objProduct.Product\_Category\_\_c='Services';

objProduct.NetSuite\_Product\_Internal\_ID\_\_c='237';

objProduct.SBQQ\_\_PricingMethod\_\_c='List';

objProduct.SBQQ\_\_ChargeType\_\_c= 'One-Time';

objProduct.Family='Professional Services';

objProduct.PC\_1\_\_c='Services';

objProduct.PC\_2\_\_c='Services: Insights';

objProduct.PC\_3\_\_c='Insights Value Assurance';

objProduct.SBQQ\_\_AssetConversion\_\_c='One per unit';

objProduct.SBQQ\_\_QuantityEditable\_\_c=True;

objProduct.IsActive= True;

insert objProduct;

// PriceBookEntry Creation

PriceBookEntry objPriceBookEntry = TestDataUtility.createpriceBookEntry(

Test.getStandardPricebookId(),

objProduct.Id,

2000.00,

True

);

insert objPriceBookEntry;

// Quote Creation

SBQQ\_\_Quote\_\_c objQuote = new SBQQ\_\_Quote\_\_c();

objQuote.Amendment\_Partial\_Segment\_\_c=FALSE;

objQuote.AVA\_SFCPQ\_\_AutomaticValidateAddress\_\_c=FALSE;

objQuote.AVA\_SFCPQ\_\_Is\_Seller\_Importer\_Of\_Record\_\_c=FALSE;

objQuote.AVA\_SFCPQ\_\_Quote\_AutomateTaxCalculation\_\_c=TRUE;

objQuote.Billing\_Day\_Of\_Month\_\_c='24';

objQuote.Contract\_dm\_\_c='8003r0000010xlWAAQ';

objQuote.CreatedById='00578000000yvlZAAQ';

objQuote.Custom\_Billing\_Frequency\_Payment\_\_c=FALSE;

objQuote.Custom\_Payment\_\_c=FALSE;

objQuote.Dm\_Checkbox1\_\_c=FALSE;

objQuote.DM\_ErrorQuote\_\_c=FALSE;

objQuote.dm\_quote\_auto\_calc\_\_c=FALSE;

objQuote.DM\_Quote\_\_c=TRUE;

objQuote.dm\_verified\_\_c=FALSE;

objQuote.ExtendContractEnd\_Date\_\_c=FALSE;

objQuote.Has\_License\_Product\_\_c=FALSE;

objQuote.Is\_End\_Date\_less\_than\_Start\_Date\_\_c=FALSE;

objQuote.Manual\_Fix\_\_c=FALSE;

objQuote.SBQQ\_\_Account\_\_c=objAccount.id;

objQuote.SBQQ\_\_BillingFrequency\_\_c='Annual';

objQuote.SBQQ\_\_ContractingMethod\_\_c='By Subscription End Date';

objQuote.SBQQ\_\_EndDate\_\_c=Date.newInstance(2023,06,29);

objQuote.SBQQ\_\_ExpirationDate\_\_c=Date.newInstance(2023,05,28);

objQuote.SBQQ\_\_LineItemsGrouped\_\_c=FALSE;

objQuote.SBQQ\_\_LineItemsPrinted\_\_c=TRUE;

objQuote.SBQQ\_\_Opportunity2\_\_c=objOpportunity.id;

objQuote.SBQQ\_\_OrderByQuoteLineGroup\_\_c=FALSE;

objQuote.SBQQ\_\_Ordered\_\_c=TRUE;

objQuote.SBQQ\_\_PaperSize\_\_c='Default';

objQuote.SBQQ\_\_PaymentTerms\_\_c='Due Upon Receipt';

objQuote.SBQQ\_\_Primary\_\_c=TRUE;

objQuote.SBQQ\_\_StartDate\_\_c=Date.newInstance(2021,05,24);

objQuote.SBQQ\_\_Status\_\_c='Draft';

objQuote.SBQQ\_\_SubscriptionTerm\_\_c=26;

objQuote.SBQQ\_\_Type\_\_c='Quote';

insert objQuote;

List<SBQQ\_\_Quote\_\_c> QList = [Select Id,DM\_Quote\_\_c,SBQQ\_\_NetAmount\_\_c from SBQQ\_\_Quote\_\_c Where DM\_Quote\_\_c =True Limit 1];

NetAmount = QList[0].SBQQ\_\_NetAmount\_\_c ;

// QuoteLine Creation

SBQQ\_\_QuoteLine\_\_c objQLine = new SBQQ\_\_QuoteLine\_\_c();

objQLine.DM\_QuoteLine\_\_c=TRUE;

objQLine.List\_Price\_\_c=2000;

objQLine.PC\_1\_\_c='Services';

objQLine.PC\_2\_\_c='Services: Insights';

objQLine.PC\_3\_\_c='Insights Value Assurance';

objQLine.Quote\_Group\_\_c='1';

objQLine.SBQQ\_\_ChargeType\_\_c='One-Time';

objQLine.SBQQ\_\_CustomerPrice\_\_c=2000;

objQLine.SBQQ\_\_ListPrice\_\_c=2000;

objQLine.SBQQ\_\_NetPrice\_\_c=2000;

objQLine.SBQQ\_\_Number\_\_c=1;

objQLine.SBQQ\_\_Optional\_\_c=FALSE;

objQLine.SBQQ\_\_PartnerPrice\_\_c=2000;

objQLine.SBQQ\_\_PricebookEntryId\_\_c='01u78000001JyUeAAK';

objQLine.SBQQ\_\_PriceEditable\_\_c=FALSE;

objQLine.SBQQ\_\_PricingMethodEditable\_\_c=FALSE;

objQLine.SBQQ\_\_PricingMethod\_\_c='List';

objQLine.SBQQ\_\_Product\_\_c=objProduct.Id;

objQLine.SBQQ\_\_ProratedListPrice\_\_c=2000;

objQLine.SBQQ\_\_ProratedPrice\_\_c=2000;

objQLine.SBQQ\_\_ProrateMultiplier\_\_c=1;

objQLine.SBQQ\_\_Quantity\_\_c=1;

objQLine.SBQQ\_\_Quote\_\_c=objQuote.Id;

objQLine.SBQQ\_\_RegularPrice\_\_c=2000;

objQLine.SBQQ\_\_Renewal\_\_c=FALSE;

objQLine.SBQQ\_\_SpecialPriceType\_\_c='Custom';

objQLine.SBQQ\_\_SpecialPrice\_\_c=2000;

objQLine.Year\_1\_Product\_\_c=TRUE;

objQLine.SBQQ\_\_StartDate\_\_c=Date.newInstance(2021,05,24);

objQLine.SBQQ\_\_EndDate\_\_c=Date.newInstance(2023,06,29);

insert objQLine;

//Contract

Contract objContract = new Contract(

Status = 'Draft',

StartDate = objQuote.SBQQ\_\_StartDate\_\_c,

EndDate = objQuote.SBQQ\_\_EndDate\_\_c,

ContractTerm = 24,

AccountId = objQuote.SBQQ\_\_Account\_\_c, //objAccount.Id,

SBQQ\_\_Opportunity\_\_c = objQuote.SBQQ\_\_Opportunity2\_\_c, //objOpportunity.Id,

SBQQ\_\_Quote\_\_c = objQuote.Id

);

objContract.Name = 'Test';

insert objContract;

// Order Creation

Order objOrder = TestDataUtility.createOrder(

objAccount.id,

'New',

'Draft',

Date.newInstance(2021,05,24),

Date.newInstance(2023,06,29),

'1',

'Net 15',

objQuote.Id,

objOpportunity.Id

);

insert objOrder;

//dm\_OpportunityProducts\_\_c Creation

dm\_OpportunityProducts\_\_c objDmopppro= new dm\_OpportunityProducts\_\_c();

objDmopppro.TOTALPRICE\_\_c = 2000;

objDmopppro.Quote\_Temp\_\_c = objQuote.Id;

objDmopppro.dm\_Product\_Name\_New\_\_c=objProduct.Name;

objDmopppro.dm\_ProductId\_New\_\_c=objProduct.Id;

insert objDmopppro;

}

@isTest

public static void executeTest(){

test.startTest();

dm\_QuoteMatchedBatch bc = new dm\_QuoteMatchedBatch();

Id batchId = Database.executeBatch(bc);

test.stopTest();

}

}