[Saturday 9:41 PM] Christiana, Vethanbu Libertin

```
public class RestrictPreviousStage {
public static void restrict(List ListNew ,ListListOld){
String objectName = 'Order';
String fieldName ='Status';
String newValue;
String oldValue;
Schema.SObjectType s = Schema.getGlobalDescribe().get(objectName) ;
Schema.DescribeSObjectResult r = s.getDescribe() ;
Map fields = r.fields.getMap() ;
Schema.DescribeFieldResult fieldResult = fields.get(fieldName).getDescribe();
List ple = fieldResult.getPicklistValues();
//for( Schema.PicklistEntry pickListVal : ple){
//// System.debug(pickListVal.getLabel() +' '+pickListVal.getValue());
List st = new List();
Set str= new Set();
for(Order orderNew:ListNew)
{
newValue = orderNew.Status;
System.debug('Current value' + newValue);
}
for(Order orderOld:ListOld)
{
oldValue = orderOld.Status;
System.debug('old value' + oldValue);
}
for( Schema.PicklistEntry pickListVal : ple)
{
str.add(pickListVal.getValue());
}
st.addAll(str);
if(st.indexOf(newValue) Trigger.new[0].addError('Could not change the stage from' + oldValue + 'to'
+ newValue);
}
}
}
```

[Saturday 9:42 PM] Christiana, Vethanbu Libertin

```
trigger RestrictTrigger on Order (before update) {
if(Trigger.isBefore && Trigger.isUpdate){
RestrictPreviousStage.restrict(Trigger.new , Trigger.old);
}
}
```

[Saturday 9:43 PM] Christiana, Vethanbu Libertin

```
public class InvoiceHandler {
public static void afterInsert(List<Invoice__c> inList){

for(Invoice__c invoice: inList){
List<Order> LstOrder = [Select Status from Order where Id=:invoice.Order__c];
String newStatus ='Invoice Generated';
LstOrder[0].Status=newStatus;
update LstOrder;


}
}
public static void afterUpdateForPaymentRecieved(List<Invoice__c> inList){

for(Invoice__c invoice: inList){
if(invoice.Invoice_Payment_Received__c==true){
System.debug(invoice.Invoice_Payment_Received__c);
List<Order> LstOrder = [Select Status from Order where Id=:invoice.Order__c];
String newStatus ='Payment Received';
LstOrder[0].Status=newStatus;
update LstOrder;
}
}
}
public static void afterUpdateForDeliveryInPlan(List<Invoice__c> inList){

for(Invoice__c invoice: inList){
if(invoice.Status_of_Delivery__c=='Delivery In Plan'){
System.debug(invoice.Status_of_Delivery__c);
List<Order> LstOrder = [Select Status from Order where Id=:invoice.Order__c];
String newStatus ='Delivery In Plan';
LstOrder[0].Status=newStatus;
update LstOrder;
}
}
}
public static void afterUpdateForDelivered(List<Invoice__c> inList){

for(Invoice__c invoice: inList){
if(invoice.Status_of_Delivery__c=='Delivered'){
System.debug(invoice.Status_of_Delivery__c);
List<Order> LstOrder = [Select Status from Order where Id=:invoice.Order__c];
String newStatus ='Delivered';
```

```
LstOrder[0].Status=newStatus;
update LstOrder;
}
}
}
public static void afterUpdateForCancelled(List<Invoice__c> inList){

for(Invoice__c invoice: inList){
if(invoice.Status_of_Delivery__c=='Cancelled' && invoice.Order_Remark__c!=null){
System.debug(invoice.Status_of_Delivery__c);
List<Order> LstOrder = [Select Status from Order where Id=:invoice.Order__c];
String newStatus ='Cancelled';
LstOrder[0].Status=newStatus;
update LstOrder;
}
else if(invoice.Status_of_Delivery__c=='Cancelled' && invoice.Order_Remark__c==null){
invoice.addError('Order Remarks filed is mandatary, Please enter the remarks');
}
}
}
}
```

[Saturday 9:43 PM] Christiana, Vethanbu Libertin

[03/12/2022 14:25] Srivastava, Nilesh

```
public class RestrictPreviousStage {
    public static void restrict(List<Order> ListNew ,List<Order>ListOld){
        String objectName = 'Order';
        String fieldName ='Status';
        String newValue;
        String oldValue;
        Schema.SObjectType s = Schema.getGlobalDescribe().get(objectName) ;
        Schema.DescribeSObjectResult r = s.getDescribe() ;
        Map<String,Schema.SObjectField> fields = r.fields.getMap() ;
        Schema.DescribeFieldResult fieldResult = fields.get(fieldName).getDescribe();
        List<Schema.PicklistEntry> ple = fieldResult.getPicklistValues();
        //for( Schema.PicklistEntry pickListVal : ple){
        //// System.debug(pickListVal.getLabel() +' '+pickListVal.getValue());
        List <String> st = new List<String>();
        Set<String> str= new Set<String>();
        for(Order orderNew:ListNew)
        {
            newValue = orderNew.Status;
```

```
            System.debug('Current value' +  newValue);
        }
        for(Order orderOld:ListOld)
        {
            oldValue = orderOld.Status;
            System.debug('old value' +  oldValue);
        }
        for( Schema.PicklistEntry pickListVal : ple)
        {
            str.add(pickListVal.getValue());
        }
        st.addAll(str);
        if(st.indexOf(newValue)<st.indexOf(oldValue)){
            Trigger.new[0].addError('Could not change the stage from' + oldValue + 'to' + newValue);
        }
    }
}
```

[03/12/2022 14:25] Srivastava, Nilesh

```
trigger RestrictTrigger on Order (before update) {
    if(Trigger.isBefore && Trigger.isUpdate){
        RestrictPreviousStage.restrict(Trigger.new , Trigger.old);
    }
}
```

[03/12/2022 14:25] Srivastava, Nilesh

```
//////////////////////////////////////////////////
```

[03/12/2022 14:25] Srivastava, Nilesh

```
public class InvoiceHandler {
    public static void afterInsert(List<Invoice__c> inList){

        for(Invoice__c invoice: inList){
         List<Order> LstOrder = [Select Status from Order where Id=:invoice.Order__c];
              String newStatus ='Invoice Generated';
              LstOrder[0].Status=newStatus;
              update LstOrder;
```

```
  }
  }
public static void afterUpdateForPaymentRecieved(List<Invoice__c> inList){

    for(Invoice__c invoice: inList){
        if(invoice.Invoice_Payment_Received__c==true){
            System.debug(invoice.Invoice_Payment_Received__c);
            List<Order> LstOrder = [Select Status from Order where Id=:invoice.Order__c];
            String newStatus ='Payment Received';
            LstOrder[0].Status=newStatus;
            update LstOrder;
       }
    }
}
public static void afterUpdateForDeliveryInPlan(List<Invoice__c> inList){

    for(Invoice__c invoice: inList){
        if(invoice.Status_of_Delivery__c=='Delivery In Plan'){
            System.debug(invoice.Status_of_Delivery__c);
           List<Order> LstOrder = [Select Status from Order where Id=:invoice.Order__c];
            String newStatus ='Delivery In Plan';
            LstOrder[0].Status=newStatus;
            update LstOrder;
       }
    }
}
public static void afterUpdateForDelivered(List<Invoice__c> inList){

    for(Invoice__c invoice: inList){
        if(invoice.Status_of_Delivery__c=='Delivered'){
            System.debug(invoice.Status_of_Delivery__c);
           List<Order> LstOrder = [Select Status from Order where Id=:invoice.Order__c];
            String newStatus ='Delivered';
            LstOrder[0].Status=newStatus;
            update LstOrder;
       }
    }
}
public static void afterUpdateForCancelled(List<Invoice__c> inList){

    for(Invoice__c invoice: inList){
        if(invoice.Status_of_Delivery__c=='Cancelled' && invoice.Order_Remark__c!=null){
            System.debug(invoice.Status_of_Delivery__c);
           List<Order> LstOrder = [Select Status from Order where Id=:invoice.Order__c];
```

```
                String newStatus ='Cancelled';
                LstOrder[0].Status=newStatus;
                update LstOrder;
            }
             else if(invoice.Status_of_Delivery__c=='Cancelled' && invoice.Order_Remark__c==null){
                    invoice.addError('Order Remarks filed is mandatary, Please enter the remarks');
            }
        }
    }
}
```

[03/12/2022 14:26] Srivastava, Nilesh

```
trigger InvoiceTrigger on Invoice__c (after insert , after update) {
    if(Trigger.isInsert && Trigger.isAfter) {
        InvoiceHandler.afterInsert(Trigger.new);
    }
    if(Trigger.isUpdate && Trigger.isAfter) {
        InvoiceHandler.afterUpdateForPaymentRecieved(Trigger.new);
        InvoiceHandler.afterUpdateForDeliveryInPlan(Trigger.new);
        InvoiceHandler.afterUpdateForDelivered(Trigger.new);
        InvoiceHandler.afterUpdateForCancelled(Trigger.new);
    }



}
```

[Saturday 9:44 PM] Christiana, Vethanbu Libertin

```
trigger InvoiceTrigger on Invoice__c (after insert , after update) {
if(Trigger.isInsert && Trigger.isAfter) {
InvoiceHandler.afterInsert(Trigger.new);
}
if(Trigger.isUpdate && Trigger.isAfter) {
InvoiceHandler.afterUpdateForPaymentRecieved(Trigger.new);
InvoiceHandler.afterUpdateForDeliveryInPlan(Trigger.new);
InvoiceHandler.afterUpdateForDelivered(Trigger.new);
InvoiceHandler.afterUpdateForCancelled(Trigger.new);
}


}
```

[Saturday 9:55 PM] Christiana, Vethanbu Libertin

```
<template>

<lightning-card title = "Search by Product name" icon-name = "custom:custom63">

<div class = "slds-m-around_medium">

<lightning-input placeholder="Enter Product Name" type = "search" onchange={handleKeyChange}
class = "slds-m-bottom_small" label = "Search" >

</lightning-input>

<lightning-input type="text" placeholder="Add Quantity" label ="Add quantity" value={quantity}
variant="label-hidden"></lightning-input>

<template if:true = {Products}>

<div style="height: 300px;">

<lightning-datatable key-field="Id"

data={Products}

columns={columns}

onrowaction={callRowAction}>

</lightning-datatable>

</div>

</template>

<template if:true = {error}>

{error}>

</template>

</div>

</lightning-card>

<lightning-card title = "Search by Brand" icon-name = "custom:custom63">
```

```
<div class = "slds-m-around_medium">

<lightning-input placeholder="Enter Product Brand " type = "search" onchange
={handleKeyChangeBrand} class = "slds-m-bottom_small" label = "Search" >

</lightning-input>

<lightning-input type="text" placeholder="Add Quantity" label="Add quantity" value={quantity1}
variant="label-hidden">

</lightning-input>

<template if:true = {Products1}>

<div style="height: 300px;">

<lightning-datatable key-field="Id"

data={Products1}

columns={columns}

onrowaction={callRowActionBrand}>

</lightning-datatable>

</div>

</template>

<template if:true = {error}>

{error}>

</template>

</div>

</lightning-card>

<lightning-card title = "Search by MRP" icon-name = "custom:custom63">

<div class = "slds-m-around_medium">

<lightning-input placeholder="Enter Product MRP " type = "search" onchange
```

```
={handleKeyChangeMRP} class = "slds-m-bottom_small" label = "Search" >

</lightning-input>

<lightning-input type="text" placeholder="Add Quantity" label="Add quantity" value={quantity2}
variant="label-hidden"></lightning-input>

<template if:true = {Products2}>

<div style="height: 300px;">

<lightning-datatable key-field="Id"

data={Products2}

columns={columns}

onrowaction={callRowActionMRP}>

</lightning-datatable>

</div>

</template>

<template if:true = {error}>

{error}>

</template>

</div>

</lightning-card>

<lightning-card title="Your cart items" icon-name = "custom:custom63">

<lightning-button onclick={handleclick} value="My Cart" label="My Cart"></lightning-button>

<lightning-datatable

key-field="id"

data={data}

columns={columns}
```

onrowselection={handleRowSelection}>

</lightning-datatable>

</lightning-card>

</template>

[Saturday 9:57 PM] Christiana, Vethanbu Libertin

```
import { api, LightningElement, track } from 'lwc';

import retriveProducts from '@salesforce/apex/home.retriveProducts';

import getProducts from '@salesforce/apex/home.getProducts';

import buyProducts from '@salesforce/apex/home.buyProducts';

import { NavigationMixin } from 'lightning/navigation';



const COLUMNS = [

{ label: 'Id', fieldName: 'Id' },

{ label: 'Name', fieldName: 'Name' },

{ label: 'Product Code', fieldName: 'ProductCode' },

{ label: 'Brand', fieldName: 'Brand__c' },

{ label: 'MRP', fieldName: 'MRP__c' },

{ type: "button", typeAttributes: {

label: 'View',

name: 'View',

title: 'View',
```

```
disabled: false,

value: 'view',

iconPosition: 'left'

} },

{ type: "button", typeAttributes: {

label: 'Edit',

name: 'Edit',

title: 'Edit',

disabled: false,

value: 'edit',

iconPosition: 'left'

} } ,

{ type: "button", typeAttributes: {

label: '+',

name: 'Increment',

title: 'Increment',

disabled: false,

value: 'increment',

iconPosition: 'left'



} }

];
```

```
export default class home extends NavigationMixin(LightningElement) {

Products1;

Products;

Products2;

error;

columns = COLUMNS;

@track quantity = 0;

@track quantity1 = 0;

@track quantity2 = 0;

@api selectedAccountlist=[];

@track data = [];

searchKey;

searchKeyb;

searchKeyg;



handleclick(){

var el = this.template.querySelector('lightning-datatable');

console.log(el);

var selected = el.getSelectedRows();

console.log(selected);

}
```

```
handleRowSelection(event){
const selectedRows = event.detail.selectedRows;
}

handleKeyChange( event ) {
this.searchKey = event.target.value;
if ( this.searchKey ) {
retriveProducts( { searchKey:this.searchKey } )
.then(result => {
this.Products = result;
})
.catch(error => {
this.error = error;
});
} else
this.Products = undefined;
}
handleKeyChangeBrand( event ){
this.searchKeyg = event.target.value;

if ( this.searchKeyg ) {
```

```
getProducts( { searchKeyg:this.searchKeyg } )
.then(result => {
this.Products1 = result;
})
.catch(error => {
this.error = error;
});
} else
this.Products1= undefined;
}

handleKeyChangeMRP( event ) {
this.searchKeyb = event.target.value;

if ( this.searchKeyb ) {
buyProducts( { searchKeyb:this.searchKeyb } )
.then(result => {
this.Products2 = result;
})
.catch(error => {
this.error = error;
});
```

```
} else
this.Products2 = undefined;
}
callRowAction( event ) {
const recId = event.detail.row.Id;
const actionName = event.detail.action.name;
if ( actionName === 'Edit' ) {
this[NavigationMixin.Navigate]({
type: 'standard__recordPage',
attributes: {
recordId: recId,
objectApiName: 'Product2',
actionName: 'edit'
}
})
} else if ( actionName === 'View') {
this[NavigationMixin.Navigate]({
type: 'standard__recordPage',
attributes: {
recordId: recId,
objectApiName: 'Product2',
actionName: 'view'
}
```

```
})
}
else if(actionName === 'Increment') {
this.quantity++;


}


}
callRowActionBrand( event ) {
const recId = event.detail.row.Id;
const actionName = event.detail.action.name;
if ( actionName === 'Edit' ) {
this[NavigationMixin.Navigate]({
type: 'standard__recordPage',
attributes: {
recordId: recId,
objectApiName: 'Product2',
actionName: 'edit'
}
})
} else if ( actionName === 'View') {
this[NavigationMixin.Navigate]({
```

```
type: 'standard__recordPage',
attributes: {
recordId: recId,
objectApiName: 'Product2',
actionName: 'view'
}
})
}
else if(actionName === 'Increment') {
this.quantity1++;



}



}
callRowActionMRP( event ) {
const recId = event.detail.row.Id;
const actionName = event.detail.action.name;
if ( actionName === 'Edit' ) {
this[NavigationMixin.Navigate]({
type: 'standard__recordPage',
attributes: {
recordId: recId,
```

```
objectApiName: 'Product2',
actionName: 'edit'
}
})
} else if ( actionName === 'View') {
this[NavigationMixin.Navigate]({
type: 'standard__recordPage',
attributes: {
recordId: recId,
objectApiName: 'Product2',
actionName: 'view'
}
})
}
else if(actionName === 'Increment') {
this.quantity2++;



}



}
}
```

[Saturday 9:58 PM] Christiana, Vethanbu Libertin

```
public with sharing class home {

@AuraEnabled( cacheable = true )

public static List< Product2 > retriveProducts( String searchKey ) {

String strKey = '%' + searchKey + '%';

List<Product2> lstProd = [SELECT Id, Name, ProductCode,Brand__c,Stock_Quantity__c,MRP__c FROM
Product2 WHERE Name LIKE :strKey LIMIT 5];

return lstProd;

}

@AuraEnabled( cacheable = true )

public static List< Product2 > getProducts( String searchKeyg ) {

String strKeyg = '%' + searchKeyg + '%';

List<Product2> lstProd1 = [SELECT Id, Name, ProductCode,Brand__c,Stock_Quantity__c,MRP__c
FROM Product2 WHERE Brand__c LIKE :strKeyg];

return lstProd1;

}

@AuraEnabled( cacheable = true )

public static List< Product2 > buyProducts( String searchKeyb) {

String strkeyb = '%' + searchKeyb + '%';

List<Product2> lstProd2 = [SELECT Id, Name, ProductCode,Brand__c,Stock_Quantity__c,MRP__c
FROM Product2 ORDER BY MRP__c ASC NULLS LAST];

return lstProd2;

}




}
```