VISUALFORCE ASSIGNMENT

1. **Create visualforce pages for Class and Student objects and override standard “New” and “Edit” buttons.**

```
<apex:page standardController="Student__c">
   <apex:form >
        <apex:pageBlock title="Student Form">
        <apex:pageBlockSection columns="1">
            <apex:inputField value="{! Student__c.Name }"/>
            <apex:inputField value="{! Student__c.FirstName__c }"/>
            <apex:inputField value="{! Student__c.LastName__c }"/>
            <apex:inputField value="{! Student__c.CLass__c }"/>
            <apex:inputField value="{! Student__c.Dob__c }"/>
            <apex:inputField value="{! Student__c.sex__c }"/>
         </apex:pageBlockSection>
         <apex:pageBlockButtons >
            <apex:commandButton action="{! save }" value="Save" />
         </apex:pageBlockButtons>
      </apex:pageBlock>
   </apex:form>
</apex:page>
```

```
<apex:page standardController="Class__c">
   <apex:form >
        <apex:pageBlock title="CLass Form">
        <apex:pageBlockSection columns="1">
            <apex:inputField value="{! Class__c.Name }"/>
            <apex:inputField value="{! Class__c.classTeacher__c }"/>
            <apex:inputField value="{! Class__c.custom_status__c }"/>
         </apex:pageBlockSection>
         <apex:pageBlockButtons >
            <apex:commandButton action="{! save }" value="Save" />
         </apex:pageBlockButtons>
      </apex:pageBlock>
   </apex:form>
</apex:page>
```

2. **Create a Page Where some filters in BilingCity, BillingState, BillingCountry of Account and click on Search button for displaying first 10 Accounts at a time and provide Pagination for (Previous, Next. First and Last)**

//Visualforce Page

```
<apex:page controller="SearchAccountDetails" action="{!searchAcc}" >
   <apex:form >
      <apex:pageBlock id="thePb" title="Account Details To Search">
         <apex:pageblockSection id="thepbs">
            <apex:inputField value="{!acc.Name}" required="false"
id="accName"/>
            <apex:inputfield value="{!acc.BillingCity}"/>
            <apex:inputfield value="{!acc.BillingState}"/>
            <apex:inputfield value="{!acc.BillingCountry}"/>
         </apex:pageblockSection>
         <apex:pageblockButtons location="bottom">
            <apex:commandButton value="Search" action="{!searchAcc}" />
         </apex:pageblockButtons>
      </apex:pageBlock>

      <apex:pageBlock title="Account Details" id="noRec" rendered="{! IF(
accountList != null && accountList.size ==0 , true, false)}" >
         <apex:outputPanel >
            <h1>No Records Found </h1>
         </apex:outputPanel>
      </apex:pageBlock>


      <apex:pageBlock title="Account Details" id="details" rendered="{! IF(
accountList != null && accountList.size >0, true, false)}" >

         <apex:pageBlockTable value="{!accountList}" var="a">
            <apex:column headerValue="Account Name">
               <apex:outputLink target="_blank"
value="/{!a.id}">{!a.Name}</apex:outputLink>
            </apex:column>
            <apex:column value="{!a.accountNumber}" headerValue="Account
Number"/>
```

```
                <apex:column value="{!a.Industry}" headerValue="Industry"/>
                <apex:column value="{!a.AnnualRevenue}" headerValue="Annual
Revenue"/>
                <apex:column value="{!a.Phone}" headerValue="Phone"/>
                <apex:column value="{!a.website}" headerValue="Web"/>
            </apex:pageBlockTable>
            <apex:commandButton value="First" rerender="details"
action="{!FirstPage}" disabled="{!prev}" />
            <apex:commandButton value="Previous" rerender="details"
action="{!previous}" disabled="{!prev}" />
            <apex:commandButton value="Next" rerender="details" action="{!next}"
disabled="{!nxt}" />
            <apex:commandButton value="Last Page" rerender="details"
action="{!LastPage}" disabled="{!nxt}" />
        </apex:pageBlock>

    </apex:form>
</apex:page>
```

//Apex Class

```
public with sharing class SearchAccountDetails {
    public Account acc{get;set;}
    public List<Account> accountList {get;set;}
    private integer totalRecs = 0;
    private integer OffsetSize = 0;
    private integer LimitSize= 10;

    List<string> conditions = new List<string>();

    public SearchAccountDetails()
    {
        acc = new Account();
    }

    public void searchAcc()
    {
        if(accountList !=null && accountList.size()>0)
        {
            accountList=null;
        }
        searchAccounts();
        conditions.clear();
    }


    public Void searchAccounts(){

        if(accountList != null && !accountList.isEmpty()){
            accountList.clear();
        }
        String strQuery ='SELECT
Id,Name,AccountNumber,CreatedDate,Phone,Website,Industry,AnnualRevenue
From Account';

        if(acc.Name !=null && acc.Name !=''){
            conditions.add('Name Like \'%' +acc.Name +'%\' ');
        }
        if(acc.BillingCity !=null && acc.BillingCity !=''){
            conditions.add('BillingCity Like\'%' +acc.AccountNumber +'%\' ');
        }
        if(acc.BillingState !=null && acc.BillingState !=''){
            conditions.add('BillingState Like\'%' +acc.AccountNumber +'%\' ');
        }
        if(acc.BillingCountry !=null && acc.BillingCountry !=''){
            conditions.add('BillingCountry Like\'%' +acc.AccountNumber +'%\' ');
        }

        if (conditions.size() > 0) {
            strQuery += '  WHERE ' + conditions[0];
            for (Integer i = 1; i < conditions.size(); i++)
                strQuery += '  AND ' + conditions[i];
        }

        if(totalRecs !=null && totalRecs ==0){
            List<Account> accTemp = Database.query(strQuery);
            totalRecs = (accTemp !=null &&accTemp.size()>0)?accTemp.size():0;
        }
        strQuery += ' ORDER BY Name  ASC LIMIT :LimitSize OFFSET
:offSetSize';
        accountList = Database.query(strQuery);
    }

    public void FirstPage()
    {
        OffsetSize = 0;
        searchAccounts();
    }
    public void previous()
    {
        OffsetSize = (OffsetSize-LimitSize);
        searchAccounts();
    }
    public void next()
    {
        OffsetSize = OffsetSize + LimitSize;
        searchAccounts();
    }
    public void LastPage()
    {
        OffsetSize = totalrecs - math.mod(totalRecs,LimitSize);
        searchAccounts();
    }
    public boolean getprev()
    {

        if(OffsetSize == 0){
            return true;
        }
        else {
            return false;
        }
    }
    public boolean getnxt()
    {
        if((OffsetSize + LimitSize) > totalRecs){

            return true;
        }
        else {
            return false;
        }
    }
}
```

3. **As you have created two new fields (BillToContact and Manager) on Opportunity previously. Now the requirement is to select BillToContact using Custom LookUp (This lookup displays a list of Contacts related to that Manager on Opportunity).**

   //Visualforce Page

```
<apex:page controller="ManagerContactController">
    <apex:form >
          <apex:pageBlock title="Contacts Related to Manager">
           <apex:pageBlockButtons location="bottom">
               <apex:commandButton value="Save" action="{! save}"/>
               <apex:commandButton value="Clear" action="{!clear}"/>
               <apex:commandButton value="Cancel" action="{!cancel}"/>
           </apex:pageBlockButtons>
           <apex:pageBlockSection >
               <apex:pageBlockTable value="{!Contacts}" var="c">
                   <apex:column >
                       <apex:selectRadio value="{!ConId}" >
                           <apex:selectOption itemValue="{!c.Id}" itemlabel="{!c.name}" />
                       </apex:selectRadio>
                   </apex:column>
               </apex:pageBlockTable>
           </apex:pageBlockSection>
        </apex:pageBlock>
    </apex:form>
</apex:page>
```

   //Apex Class

```
public with sharing class ManagerContactController {
    public List<contact> contacts{get;set;}
    public string oppId;
    public string conId{get;set;}
    Opportunity o;

    public ManagerContactController(){
        oppId = System.currentPageReference().getParameters().get('ID');
        o = [SELECT name, Manager__c FROM opportunity WHERE id =: oppId];
        Id i = o.Manager__c;
        contacts = [SELECT id, name FROM contact WHERE accountId =: i];
    }

    public PageReference clear() {
        conId = null;
        o.BillToContact__c = conId;
        update o;
        PageReference pg = new PageReference('/'+oppid);
        pg.setRedirect(true);
        return pg;
    }
    public PageReference save() {
        if(ConId == null) {
                                return Null;
        }
        else {
            o.BillToContact__c = ConId;
            update o;
            PageReference pg = new PageReference('/'+oppid);
            pg.setRedirect(true);
            return pg;
        }
    }
    public PageReference cancel() {

                     PageReference pg = new PageReference('/'+oppid);
        pg.setRedirect(true);
        return pg;
    }
}
```

4. **Create a Formula Link Field named “Generate PDF”.**

//Page to prevent “Too many nested getContent calls” error

```
<apex:page standardController="Student__c" action="{!saveAttach}"
extensions="StudentDetails" renderAs="pdf">
</apex:page>
```

//Visualforce Page

```
<apex:page standardController="Student__c">
   <apex:form >
      <apex:pageBlock title="Student Details">
         <apex:pageBlockSection columns="1">
            <apex:outputText >Name: {!Student__c.Name}</apex:outputText>
            <apex:outputText >Age: {!Student__c.Age__c}</apex:outputText>
            <apex:outputText >Class:
{!Student__c.Class__r.name}</apex:outputText>
            <apex:outputText >Sex: {!Student__c.Sex__c}</apex:outputText>
         </apex:pageBlockSection>
      </apex:pageBlock>
   </apex:form>
</apex:page>
```

//Apex Page

```
public class StudentDetails {
   public Student__c student{get;set;}
   public Id id;

   public StudentDetails(ApexPages.StandardController controller){
      id = apexpages.currentPage().getParameters().get('ID');
   }
   public PageReference saveAttach(){
      PageReference pdf = Page.StudentDetails;
      pdf.getParameters().put('id',id);
      List<Attachment> att=[Select id,name from Attachment where parentId=:id ];
      if(att.size()>0)
         Delete att;

      // create the new attachment
      Attachment attach = new Attachment();
```

```
        // the contents of the attachment from the pdf
        Blob body;

        try {
          // returns the output of the page as a PDF
              body = pdf.getContentAsPDF();
        }
        catch (VisualforceException e) {
              body = Blob.valueOf('Some Text');
        }

        attach.Body = body;
        // add the user entered name
        attach.Name = 'details.pdf';
        attach.IsPrivate = false;
        // attach the pdf to the account
        attach.ParentId = id;
        insert attach;

        // send the user to the account to view results
        return new PageReference('/'+id);

    }

}
```

5. **Create a page which shows output as a JSON formatted string. (We can use this mechanism when we send response to any service as a JSON)**

//Visualforce Page

```
<apex:page controller="CreateJson"  contentType="application/x-JavaScript;
charset=utf-8" showHeader="false" standardStylesheets="false" sidebar="false">
   <apex:form >
         {!jsonStr}
   </apex:form>
</apex:page>
```

//Apex Class
```
public class CreateJson {
   public String jsonStr {get;set;}

        public CreateJson() {
                jsonStr = prepareData();

        }

   private string prepareData(){
                List<Account> accounts = [SELECT ID,NAME,Phone,
AnnualRevenue FROM Account];
                return JSON.serialize(accounts);
        }
}
```

6. **Create 2 record types (TGT and PGT) in the Teacher (Contact) table and on the detail page show a bar as a header containing "Record Type Value" in a bar.**

//Visualforce Page
```
<apex:page standardController="contact" extensions="TgtPgt">
        <apex:form >
                <apex:pageBlock title="Teacher Detail">
         <apex:pageBlockButtons location="top">
                <apex:commandButton action="{! save}" value="Save"/>
            <apex:commandButton action="{! clone}" value="Clone"/>
            <apex:commandButton action="{! edit}" value="Edit"/>
         </apex:pageBlockButtons>
                        <div style="background-color:Blue;height:25px">
                      <center>
                   <h2 style="color:white;font-size:150%"><apex:outputField
value="{!con.RecordType.Name}"/></h2>
                </center>
                </div>
         <apex:pageBlockSection columns="2">
            <apex:outputText >Contact Owner:
{!contact.owner.Name}</apex:outputText>
            <apex:outputText >Name: {!contact.name}</apex:outputText>
```

```
                <apex:outputText >Account Name:
{!contact.account.Name}</apex:outputText>
                <apex:outputText >Title: {!contact.title}</apex:outputText>
                <apex:outputText >Department:
{!contact.department}</apex:outputText>
                <apex:outputText >Phone: {!contact.phone}</apex:outputText>
                <apex:outputText >Home Phone:
{!contact.homePhone}</apex:outputText>
                <apex:outputText >Mobile: {!contact.mobilePhone}</apex:outputText>
                <apex:outputText >Other Phone:
{!contact.otherPhone}</apex:outputText>
                <apex:outputText >Fax: {!contact.fax}</apex:outputText>
            </apex:pageBlockSection>
              </apex:pageBlock>
          </apex:form>
</apex:page>
```

//Apex Class

```
public class TgtPgt {
          Public id contactID;
    Public contact con{get;set;}

    public TgtPgt(ApexPages.StandardController controller) {
                    if(ApexPages.currentPage().getParameters().get('id') != null) {
                              contactID =
ApexPages.currentPage().getParameters().get('id');
          if(contactID != null)
                    con = [select id,RecordType.Name,
name,account.name,OtherPhone, mobilePhone, owner.name, HomePhone,
Phone, title, department, fax from contact where id =:contactId];
          }
          }
}
```

7. **Create a Visualforce Page named manageClass , on this page show list of available classes with Edit and Delete Link. When will the user click on Edit a small Area displayed just below the same page with some fields (4-5 fields). Users can save and return back to the same Page. (You can use ajax functionality for the same).**

   //Visualforce Page

```
<apex:page controller="ManageClassController">
    <apex:form id="form" >
        <apex:pageBlock title="Classes">
            <apex:pageMessages ></apex:pageMessages>
            <apex:pageBlockTable value="{!classes}" var="row">
                <apex:column >
                    <apex:outputLink title=""
value="/{!row.id}/e?retURL=/apex/{!$CurrentPage.Name}"
style="font-weight:bold">Edit</apex:outputLink>&nbsp;|&nbsp;
                    <apex:commandLink action="{!DeleteClass}" reRender="form"
value="Delete">
                        <apex:param name="classid" value="{!row.Id}"
assignTo="{!SelectedClassId}"/>
                    </apex:commandLink>
                </apex:column>
                <apex:column value="{!row.Name}"/>
               </apex:pageBlockTable>
        </apex:pageBlock>
           </apex:form>
</apex:page>
```

   //Apex Class

```
public class ManageClassController {
    public List< Class__c > classes { get; set; }
    public string SelectedClassId { get; set; }

    public ManageClassController() {
        LoadData();
    }

    private void LoadData() {
        classes = [Select id, name from Class__c limit 20];
    }
    public void DeleteClass()
    {
        if (SelectedClassId == null) {
          return;
        }

        Class__c tobeDeleted = null;
        for(Class__c cls : classes)
            if (cls.Id == SelectedClassId) {
                tobeDeleted = cls;
                break;
            }

        if (tobeDeleted != null) {
            Delete tobeDeleted;
        }

        LoadData();
    }
}
```