Triggers Scenario based Interview Questions

Q.Which trigger event allows a developer to update fields in the Trigger.new list without using an additional DML statement? Choose two answers:

1. **Before insert**.
2. **Before update.**
3. After update.
4. After insert.

Q.A developer wrote a workflow e-mail alert on case creation so that an e-mail is sent to the case owner-manager when a case is created. When will the e-mail be sent?

5. After committing to the database.
6. Before trigger execution.
7. After Trigger execution.
8. **Before committing to the database.**

Q 3. In which order does Salesforce execute events upon saving a record?

9. **Before Triggers; Validation Rules; After Triggers; Assignment Rules; Workflow Rules; Commit.**
10. Validation Rules**;** Before Triggers; After Triggers; Workflow Rules; Assignment Rules; Commit
11. Before Triggers; Validation Rules; After Triggers; Workflow Rules; Assignment Rules; Commit
12. Validation Rules; Before Triggers; After Triggers; Assignment Rules; Workflow Rules; Commit.

Q.Whenever a record is inserted into the account automatically inserted into the contact.

```
trigger SCENARIO1 on Account (after insert) {

list c=new list();

for(account a:trigger.new)

{

contact b=new contact();

b.LastName=a.Name;
b.AccountId=a.Id;
c.add(b);
}
insert c;
}
```

Q. Whenever a record is inserted to the contact automatically inserted into the account.

```
trigger scenario2 on Contact (after insert) {
if(Recursive.flag)
{
Recursive.flag=false;
lista=new list();
for(contact c:trigger.new)
{
account a1=new account();
a1.Phone=c.Phone;
a1.Name=c.LastName;
a.add(a1);
}
insert a;
}
```

Q.How to avoid the recursive triggers in Salesforce?

```
public class Recursive {
public static boolean flag=true;
}
```

Q.Whenever a created opportunity object record, update total opportunities and total amount in the account object.

```
trigger scenario24 on Opportunity (after insert) {
setids=new set();
for(Opportunity op:trigger.new)
{

ids.add(op.accountid);

}

listac=[select Total_opportunities__c,Total_Amount__c,(select
id,Amount from Opportunities ) from account where id=:ids];

for(account a:ac)

{

a.Total_opportunities__c=a.opportunities.size();

decimal sum=0;

for(opportunity p:a.opportunities)

{

sum=sum+p.amount;

}

a.Total_Amount__c=sum;

}

update ac;

}
```

Q.On contact object, whenever department equal to cse automatically insert email on the email field.

```
trigger scenario4 on Contact (before insert) {

for(contact c:trigger.new)

{

if(c.Department=='CSE')

{

c.Email='testemail@gmail.com';

}

}

}
```

Q. Whenever we modifying the inputout__c object doctor name automatically updates on the droppoff__c object text field on the relationship.

```
trigger SCENARIO32 on Inputout__c (after update) {

listd=[select id,name,Text__c from Dropoff1__c where
Text__c='naveen'];

string name;

for(Inputout__c c:trigger.new)

{

name=c.Doctor_Name__c;

}

for(Dropoff1__c dp:d)

{

dp.Text__c=name;

}

update d;

}
```

Q.Limit reached the records.

```
trigger SCENARIO6 on Account (before insert,before update) {

integer count=0;

lista=[select id,name from account where createddate=today or
lastmodifieddate=today];

for(account ac:trigger.new)

{

count=a.size();

ac.NumberofLocations__c=count;

if(count>2)

{

ac.adderror('reached limit today');

}

}
}
```

Q. Can not insert/update/delete that user account object records

```
trigger scenario30 on Account (before insert,before update,before
delete) {

user u=[select id,name from user where
username='testemail@gmail.com'];

if(u.id==userinfo.getUserId())

{

if(trigger.isdelete)

{

for(account a:trigger.old)

{

a.adderror('cant delete record');

}

}

if(trigger.isupdate)

{

for(account b:trigger.new)

{

b.adderror('can not update');

}

}

if(trigger.isinsert)

{

for(account c:trigger.new)

{

c.adderror('can not insert');

}

}
}

}
```

Q.Already existing records display an error message.

```
trigger scenario8 on Contact (before insert) {

listst=new list();

for(contact c:trigger.new)

{

list<contact>a=[select id,name,Email,lastname from contact where
Email=:c.Email];

if(a.size()>0)

{

c.Email.adderror('already existing');

}

}

}
```

Q.For loop without query.

```
trigger duplicatetrigger on Inputout__c (before insert) {

sets=new set();

for(Inputout__c op:trigger.new)

{

s.add(op.Doctor_Name__c);

}

listd=[select id,Doctor_Name__c from Inputout__c where
Doctor_Name__c=:s];

setdupids=new set();

for(Inputout__c don:d)

{

dupids.add(don.Doctor_Name__c);

}

for(Inputout__c c:trigger.new)
```

```
{
if(c.Doctor_Name__c!=null)
{
if(dupids.contains(c.Doctor_Name__c))
{
c.Doctor_Name__c.adderror('already existing record');
}
}
}
```

Q. Count of related contacts and accounts field display size

```
public class rollupsummery {
public static void increment(list<contact>con)
{
set<id>ids=new set<id>();
for(contact c:con)
{
ids.add(c.accountid);
}
list<account>a=[select id,name,NumberOfEmployees,(select
id,lastname from contacts) from account where id=:ids];
for(account ac:a)
{
ac.NumberOfEmployees=ac.contacts.size();
}
update a;
}
trigger:
trigger scenario11 on Contact (after insert) {
rollupsummery.increment(trigger.new); }
```

Q.Whenever opportunity stagename =closedwon automatically update the account field rating=hot.

```
trigger scenario12 on Opportunity (after insert,after update) {

setids=new set();

listac=new list();

for(opportunity op:trigger.new)

{

ids.add(op.AccountId);

ac=[select id,name,rating from account where id=:ids];

if(op.StageName=='Closed won')

{

for(account a:ac)

{

a.Rating='hot';

}

update ac;

}

}

}
```

Q. Whenever account name is “LearnSalesforce” automatically update the contact all lastnames.

```
trigger scenario13 on Account (after update) {

string names;

list<contact>c=[select id,lastname,firstname from contact where
lastname=:names ];

for(account a:trigger.new)

{

names=a.name;

}

for(contact con:c)
{
con.lastname=names;
}
update c;
}
```

Q.When ever an opportunity is created then record amount field is calculated by account total field

```
trigger scenario21 on Opportunity (after insert,after update,after
delete) {
setids=new set();
mapopp=new map();
Decimal oldVal;
Decimal newVal;
if(trigger.isinsert)
{
for(opportunity op:trigger.new)
{
ids.add(op.AccountId);
opp.put(op.AccountId, op);
}
list<account> acc=[select id,Total_Amount__c from account where
id=:ids];
for(account a:acc)
{
if(a.Total_Amount__c==null )
{
a.Total_Amount__c=opp.get(a.Id).amount;
}
else
{
```

```
a.Total_Amount__c= a.Total_Amount__c+opp.get(a.Id).amount;
}
}
update acc;
}
if(trigger.isUpdate)
{
for(opportunity op:trigger.new)
{
ids.add(op.AccountId);
opp.put(op.AccountId, op);
newVal=op.Amount;
}
for(Opportunity  ops:trigger.old){
oldVal=ops.Amount;
}
list<account> acc=[select id,Total_Amount__c from account where
id=:ids];
for(account a:acc)
{
if(a.Total_Amount__c==null )
{
a.Total_Amount__c=opp.get(a.Id).amount;
}
else
{
a.Total_Amount__c= a.Total_Amount__c+opp.get(a.Id).amount-oldVal;
}
}
update acc;
```

```
}

}

}
```

Q.Whenever we create a lead object then automatically converted to account , contact, opportunity.

```
trigger scenario19 on Lead (after insert) {

listacc=new list();

listcon=new list();

listop=new list();

for(lead l:trigger.new)

{

account a=new account();

a.Name=l.lastname;

a.Phone=l.Phone;

acc.add(a);

contact c=new contact();

c.LastName=l.Name;

con.add(c);

opportunity o=new opportunity();

o.Amount=l.AnnualRevenue;

o.CloseDate=system.today();

o.StageName='closed won';

op.add(o);

}

insert acc;

insert con;

insert op;

}
```

Q. Whenever we create a contact then automatically update opportunity fields.

```
trigger scenario17 on Contact (after insert) {

listop=[select id,name,stagename,Description,amount from
opportunity limit 50];

for(contact c:trigger.new){

for(opportunity o:op)

{

if(o.amount<5000||o.Amount==null)

{

o.amount=5000;

o.Name=o.Name+’Mr’;

o.StageName=’prospecting’;

}

else{

o.Amount=o.Amount+1000;

o.Name=o.Name+’Dr’;

}

update o;

}

}

}
```

Q.What is Action poller in Salesforce.

```
public class actionpoller1 {

public datetime dateandtime{get;set;}

public void datetimemethod()

{

dateandtime=system.now();
```

Visualforce page :

```
<<a href="https://www.crmsalesforcetraining.com/salesforce-apex-training-everything-about-salesforce-apex/" data-internallinksmanager029f6b8e52c="7" title="Apex Programming" target="_blank" rel="noopener">apex</a>:page controller=”actionpoller1”>

<apex:form>

<apex:pageBlock id=”pb”>

<apex:actionPoller action=”{!datetimemethod}” reRender=”pb” interval=”5″ />

time:{!dateandtime}

refresh 5 mins

</apex:pageBlock>

</apex:form>

</apex:page>
```

Q.What is Action: status

```
public class actionstatus {

Integer count = 0;

public PageReference incrementCounter() {

count++;

return null;

}

public Integer getCount() {

return count;

}

}
```

Visualforce page:

```
<apex:page controller=”actionstatus”>

<apex:form >

<apex:outputpanel id=”counter”>

<apex:outputText value=”Click Me!: {!count}”/>
```

```
<apex:actionSupport event="onclick" action="{!incrementCounter}"
rerender="counter" status="counterStatus"/>

</apex:outputpanel>

<apex:actionStatus id="counterStatus" startText=" (processing…)"
stopText="(completed)"/>

</apex:form>

</apex:page>
```

Q.What are Aggregate functions in Salesforce?

```
public class aggregatefunctions {

public listac{get;set;}

public integer count{get;set;}

public decimal sum{get;set;}

public decimal min{get;set;}

public decimal max{get;set;}

public aggregatefunctions()

{

ac= [select id,name,AnnualRevenue from account where
name=:'naveen'];

count=[select count() from account];

aggregateresult res=[select
sum(AnnualRevenue)sumt,min(AnnualRevenue)mint,max(AnnualRevenue)max
t from account ];

sum=(decimal)res.get('sumt');

min=(decimal)res.get('mint');

max=(decimal)res.get('maxt');

}
```

Visualforce page:

```
<apex:page controller="aggregatefunctions" >

<apex:form >

<apex:pageBlock >

<apex:pageBlockSection >
```

```
<apex:pageBlockTable value="{!ac}" var="n">
<apex:column value="{!n.name}"/>
<apex:column value="{!n.AnnualRevenue}"/>
</apex:pageBlockTable>
</apex:pageBlockSection>
{!sum}
{!min}
{!max}
</apex:pageBlock>
</apex:form>
</apex:page>
```

Q.Sending email outbound email message

```
public class emailprogramme1 {
public void myemails()
{
messaging.SingleEmailMessage m1=new messaging.SingleEmailMessage();
string[] toadd=new string[]{'naveengorentla1@gmail.com'};
string[] tocc=new string[]{'naveen123sfdc@gmail.com'};
m1.setToAddresses(toadd);
m1.setCcAddresses(tocc);
m1.setSubject('accenture');
m1.setPlainTextBody('this is interview call letter');
messaging.email[] m2=new messaging.Email[]{m1};
messaging.sendEmail(m2);
}
}
```

Executions:

```
emailprogramme1 v=new emailprogramme1();
v.myemails();
```

Q.Outbound message pdf file.

```
public class emailprogramme2 {

public void emailbody()

{

messaging.SingleEmailMessage m1=new messaging.SingleEmailMessage();

string[] toadd=new string[]{'naveengorentla1@gmail.com'};

m1.setToAddresses(toadd);

m1.setSubject('Pdf file');

m1.setPlainTextBody('THIS IS BILL OF TELEPHONE');

messaging.EmailFileAttachment m2=new
messaging.EmailFileAttachment();

pagereference p=page.page1;

blob body=p.getContentAsPDF();

m2.setBody(body);

m2.setFileName('jan-feb-march');

messaging.EmailFileAttachment[] eft1=new
messaging.EmailFileAttachment[]{m2};

m1.setFileAttachments(eft1);

messaging.Email[] m3=new messaging.Email[]{m1};

messaging.sendEmail(m3);

}
```

Output :

```
emailprogramme2 v=new emailprogramme2();

v.emailbody();
```

Q.Sending email template to the outbound message

```
public class emailprogramme3 {

public void emailmethod()

{

messaging.SingleEmailMessage m1=new messaging.SingleEmailMessage();

emailtemplate et=[select id from emailtemplate where name='doctor'
];
m1.setTemplateId(et.Id);

contact c=[select id,lastname,phone from contact where
phone='999'];

m1.setTargetObjectId(c.id);

Inputout__c D=[select id,Doctor_Name__c from Inputout__c limit 1];

m1.setWhatId(D.Id);

messaging.Email[] m2=new messaging.Email[]{m1};

messaging.sendEmail(m2);

}

}
```

output:

```
emailprogramme3 v=new emailprogramme3();

v.emailmethod();
```

Q.Sending email trigger.

```
trigger sendingmailtrigger on Inputout__c (before insert) {

for(Inputout__c i:trigger.new)

{

if(i.Check_box__c==true)

{

messaging.SingleEmailMessage m1=new messaging.SingleEmailMessage();

string[] toadd =new string[]{'naveengorentla1@gmail.com'};

m1.setToAddresses(toadd);

m1.setSubject('accenture');

m1.setPlainTextBody('this is interview call letter');

messaging.Email[] mail1=new messaging.Email[]{m1};

messaging.sendEmail(mail1);
```

Q. Create “Sales Rep” field with the data type(Text) on the Account object When we create the Account record, the Account owners will be automatically added to the sales rep field. When we update the Account owner of the record, then also the Sales Rep will be automatically updated.

Trigger :

```
trigger UpdateSalesRep on Account(Before insert, Before update)

{

set<Id> setAccowner = new.set<Id>();

 for(Account Acc : trigger.New)

 {

  setAccowner.add(Acc.ownerId);

  }

Map<Id, user> usermap = new Map<Id, user>([SELECT Name FROM
user WHERE Id IN: setAccowner]);

for(Account Acc : Trigger.New)

{

  user usr = usermap.get(Acc.ownerId);

  Acc.sales_rep__c = usr.Name;

   }

 }

}
```

Explanation :

- In the above trigger, as we want to fire the trigger before creating any Account record hence we used before insert trigger on Account object. Similarly as per the second requirement i.e. for update event hence we are writing before update trigger on Account object.

```
trigger UpdateSalesRep on Account(Before insert, Before update)
```

- We used set to store Id’s as we don’t want duplicate Id’s of AccountOwner. Hence we created an instance of that as setAccowner.

- Then we used for loop for every new account hence we used trigger.new inside the loop. Inside the loop, we are adding account ownerId in a set.
- Then we have created an instance of map usermap, which will store key=Id and value=user along with one query

```
SELECT Name FROM user WHERE Id IN: setAccowner
```

which will give you the Name of the user whose id is equal to Account OwnerId.

- Then in the next for loop, we created an instance for the user as usr. Which will be equal to the Account ownerId which we will get from the map i.e. from Acc.ownerId.
- And at the end, we are saying sales_rep__c from an Account which will be equal to the user’s name i.e. usr.Name.

Q.Create a field called “Contact Relationship” checkbox on the contact object and create the object called “Contact Relationship” which is a related list to the Contacts (Lookup Relationship).

Now build a logic when we create contact by checking the Contact Relationship checkbox, then contact Relationship will be created automatically for that contact.

Trigger

```
trigger CreateCRonContactCreation on Contact(after insert)

{

if(trigger.isAfter)

{

 if(trigger.isInsert)

 {

   ContactMasterHandler ConIns = New ContactMasterHandler();

   ConIns.createContactRelationshipByContact(trigger.new);

 }

}
```

Apex Class :

```
public class ContactMasterhandler

{

 public void createContactRelationShipByContact(List<contact>
lstCon)

  {

  List<Contact_Relationship__c> conlist = new
List<Contact_Relationship>();

for(Contact newcon : lstCon)

{

  if(newcon.Contact_Relationship__c == true)

   {

    Contact_Relationship__c CR = new Contact_Relationship__c();

    CR.Name = newcon.LastName;

    CR.Contact__c = newConts.id;

    ConList.add(CR);

    }

   }

  insert ConList;

 }

}
```

Before understanding the structure of the above scenario, I know it's a somewhat different example where we create an apex class and just call the method wherever necessary. Which is makes your development easy. Also, it's very easy to understand for the developer as well. Where your actual logic is written inside apex class and you and use its methods wherever you want. The advantage of doing this is, you can use the same apex class in different triggers as well for the same logic. No need to rewrite it.

- As per the requirement, we are operating on the trigger when the user submits the record that means we need to use as after insert trigger. Then when we need to check more about the event, hence we used a trigger.isAfter and trigger.isInsert to fire the trigger on only these particular events.

- Then we created an instance of apex class ContactMasterhandler as ConIns.Then we called a createContactRelationshipByContact method from apex class where actual logic is written.
- Let’s move towards the apex method createContactRelationshipByContact where we are passing a list of contact in lstCon instance coming from new newly created contact.
- Then we have created an instance of List of Contact_Relationship__c as conlist which will store a list of Contact_Relationship__c records.
- After that, we created for loop for all the newly created contact in salesforce.
- Then if newcon.Contact_Relationship__c which is a field from newly created contact. if it is true then it processes further and acts.
- Then we created an instance for Contact_Relationship__c object as CR.
- Then to create a new record for CR we are giving the Name same as LastName of the newly created contact. Also, it’s Contact is equal to the id of newly created contact.
- Then we added CR in list ConList and by Insert we successfully inserted Contact_Relationship__c record in salesforce.

Q. When we change the owner of the Contact Relationship, then the owner name will be automatically populated in the Contact Relationship, then the owner name will be automatically populated in the Contact Relationship Name Field.

Trigger :

```
trigger ContactRelationshipMasterTrigger on
Contact_Relationship__c(before update)

{

if(trigger.isBefore)

{

  if(trigger.isUpdate)

 {
   updateCROwnerName conReload = New UpdateCROwnerName();
   conReload.updateCRNameByOwner(trigger.New);
   }
 }
}
```

Apex Class :

```
public class updateCROwnerName
{
public void updateCRNameByOwner(List<Contact_Relationship__c >
con_rel)
{
 Map<Id, String> map_id_Name = new Map<Id,String>();
 Set<Id> IdSet = new Set<Id>();
for(Contact_Relationship__c ListRecs : con_rel )
{
  IdSet.add(ListRecs.ownerId);
 }
List<user> u =[SELECT Id, Name FROM user WHERE Id IN : IdSet];
for(user list_users : u)
{
 map_id_Name.put(list_users.Id, list_users.Name);
 }
 if(u! = null && u.size()>0)
{
 for(Contact_Relationship__c ListRecs : con_rel)
  {
    if(ListRecs.ownerId != null)
    {
      ListRecs.Name = map_id_Name.get(ListRecs.ownerId);
```

```
                }
            }
        }
    }
}
```

- As per the requirement, before performing the update operation we have to do some changes in Contact_Relationship__c object. Hence we created a trigger before the update.
- Then we check for a particular operation to fire the trigger hence we checked whether the event is trigger.isBefore and trigger.isUpdate. Then we simply called the apex class method by creating an instance of a class conReload.
- Inside method, Created a Map with a map_id_Name instance which contains key=Id and value=String as well as Set with IdSet instance which contains Id’s set.
- Then in the first for loop, we set the ListRecs.ownerId i.e. ownerId of contact relationship record.
- Then we fetch the users from a query and store their Id and Name in the map. Hence our map now contains userId and userName.
- Then we have null check if a user is there in the list then it will go inside for loop. Where if Contact_Relationship__c ownerId is not equal to null, then Contact Relationship Name is equal to the name of the user.