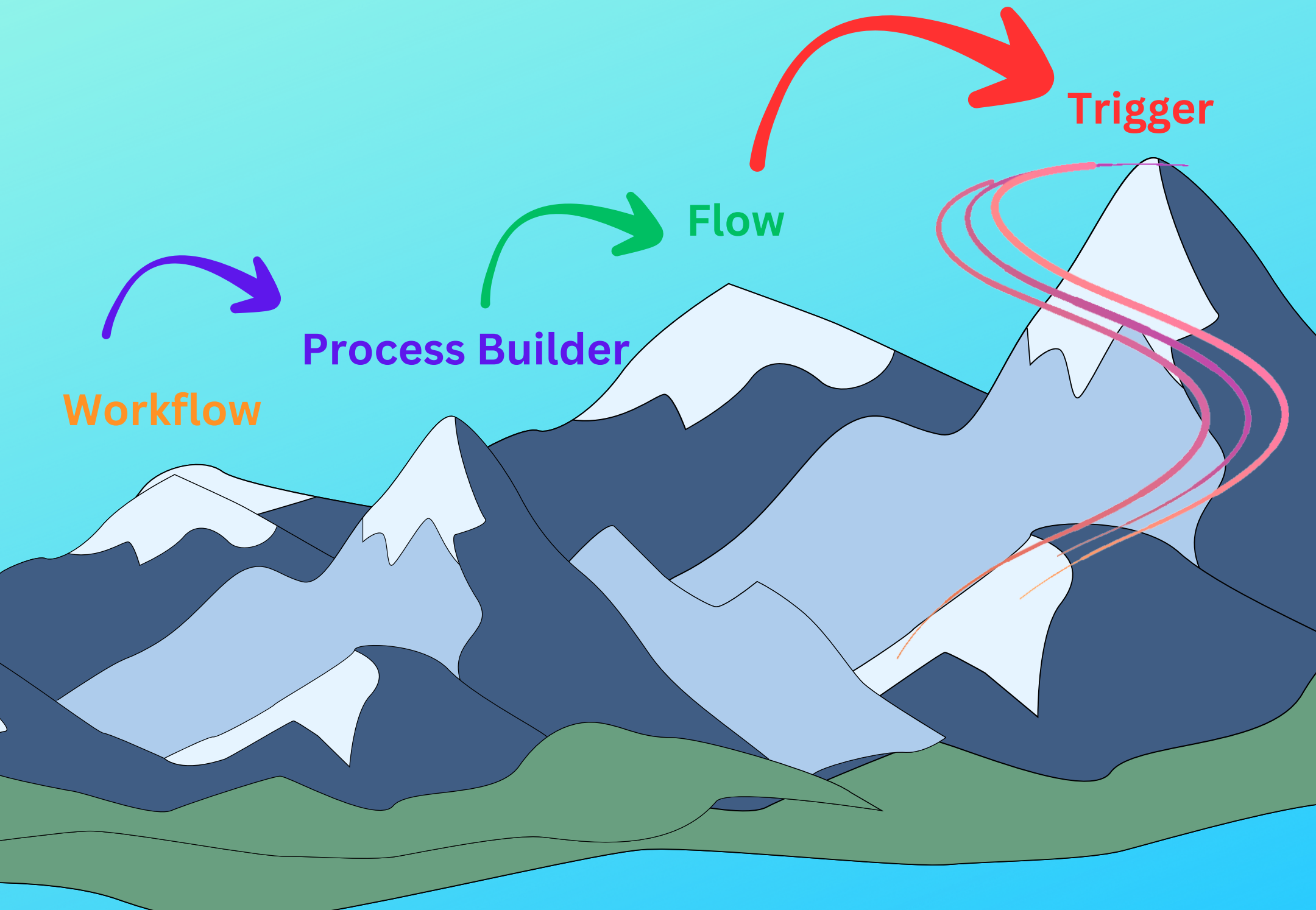
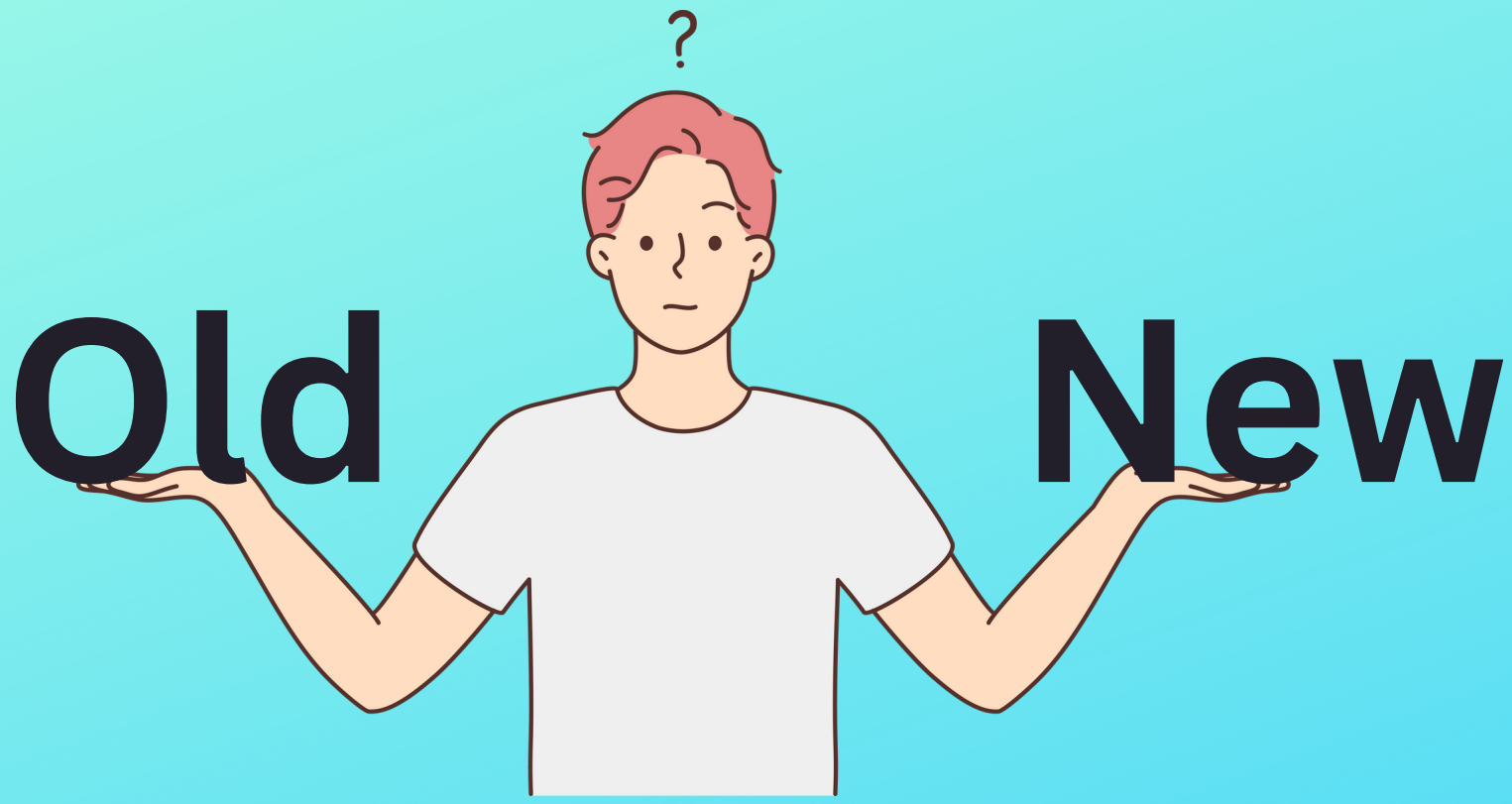
If you know these trigger scenarios then you can write any Apex Trigger flawlessly.

"Remember, it is not necessary to write everything in the trigger. You can use workflow, process builder & flow also."



1.Compare old & new field values.

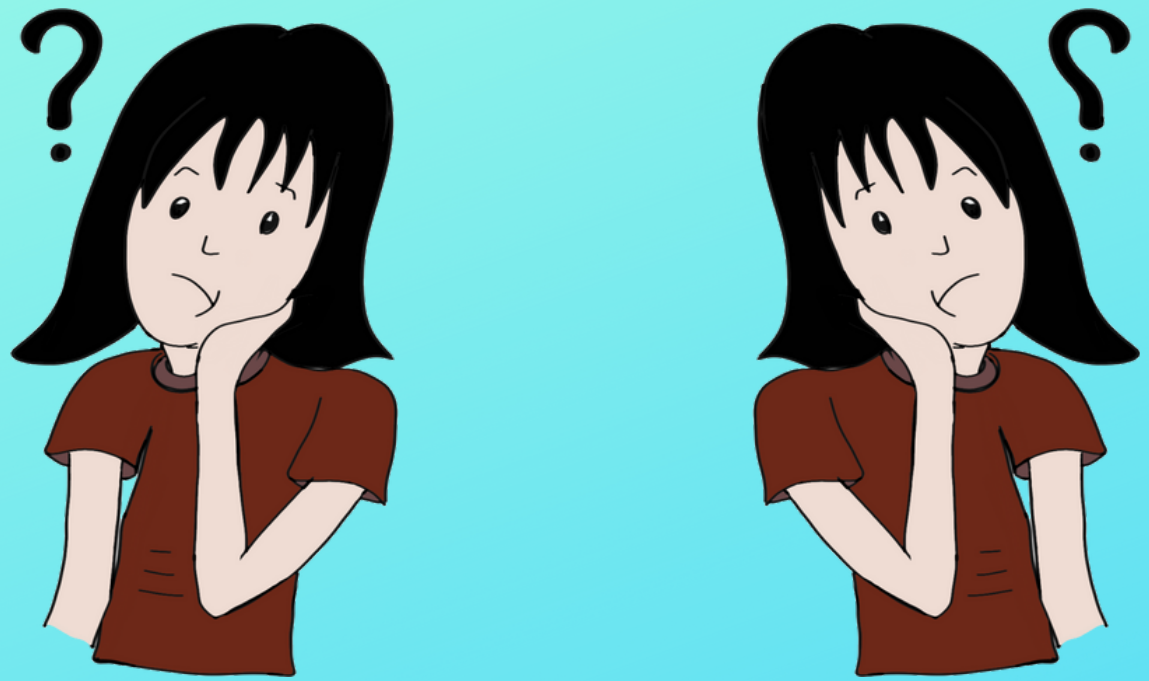



```
trigger CompareOldWithNewValue on Contact (before update) {
  for(Contact con : trigger.new){
    if(trigger.oldMap.get(con.id).email != con.email ){
      // Perform any action.....
    }
  }
}
```

2.Data validation in before trigger.

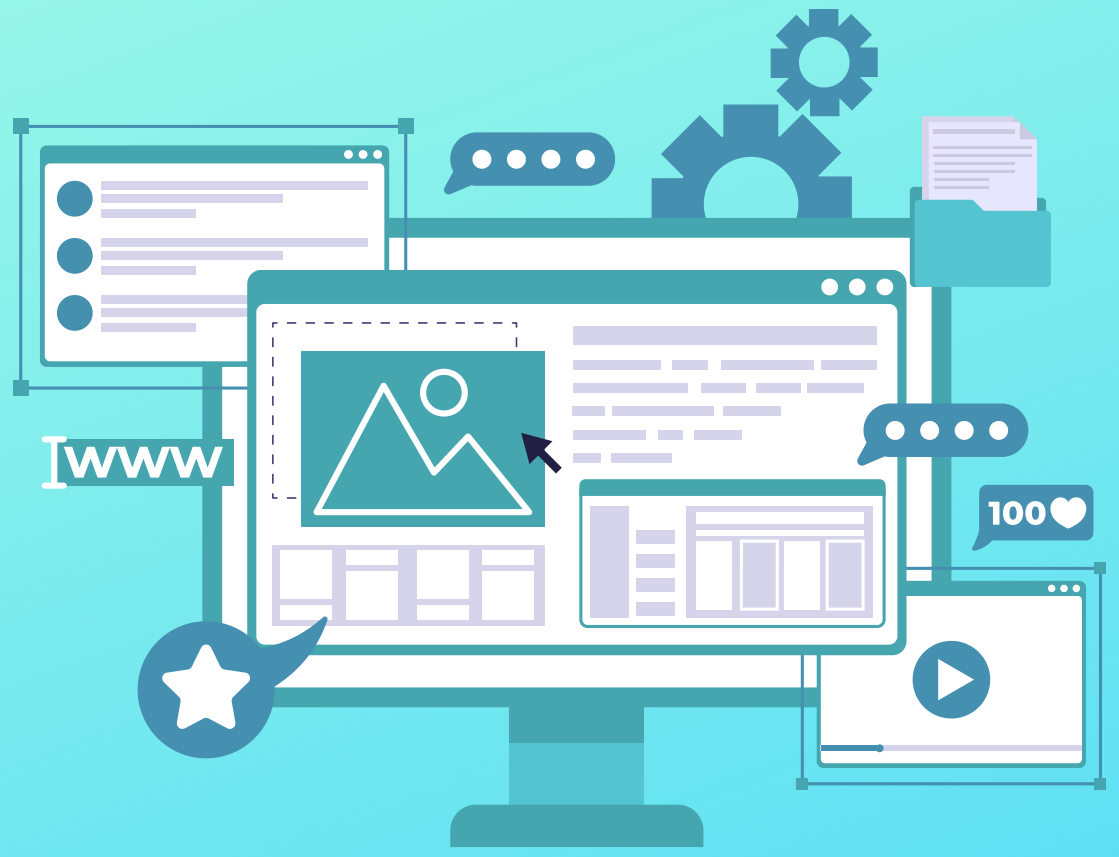
```
trigger CompareOldWithNewValue on Contact (before update) {
  for(Contact con : trigger.new){
    if(con.email == null){
        con.addError('Email Addres is required field.');
    }
  }
}
```

3.Check duplicate records to avoid storing similar records.



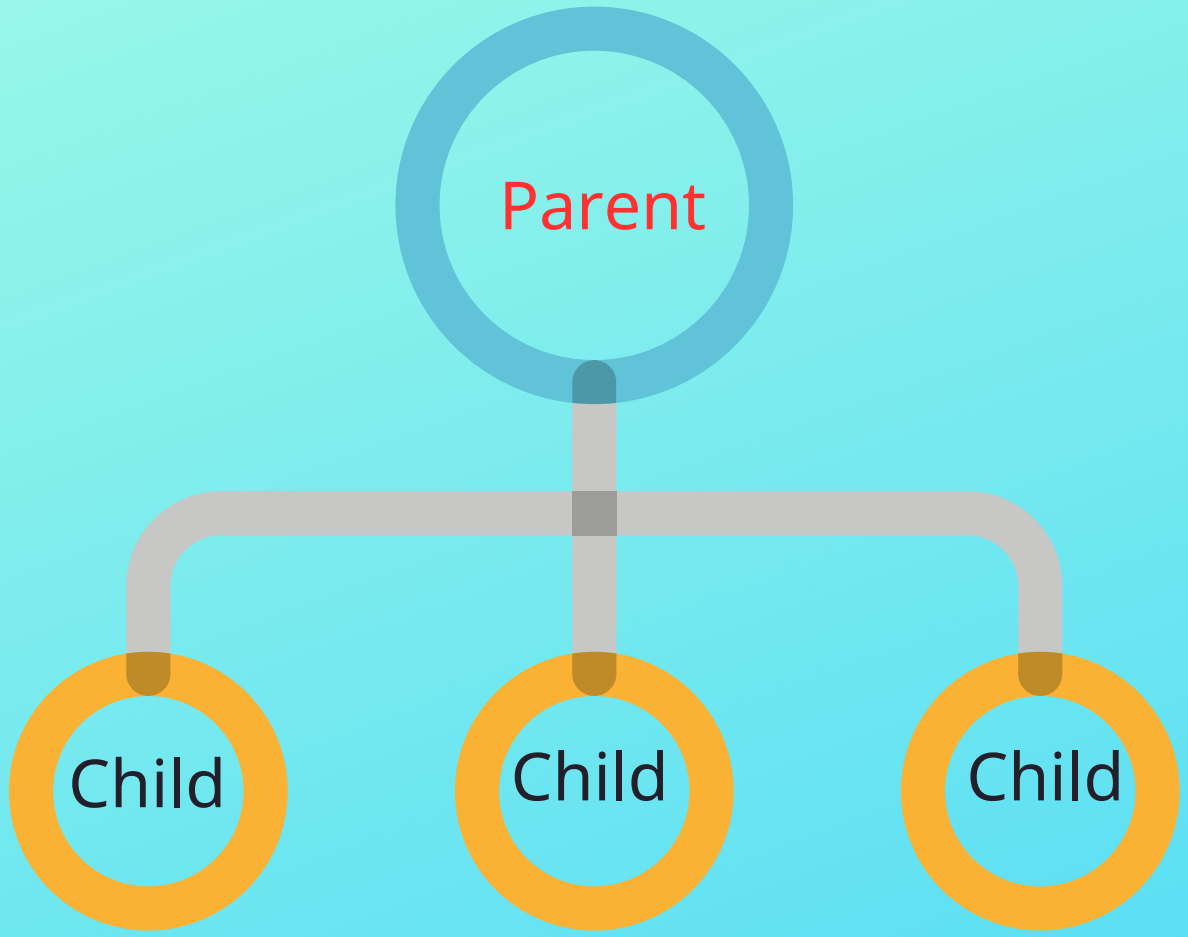
```
trigger TriggerScenarios on Account (before insert) {

  set<String> companyName = new set<String>();
  for(Account newAcc : trigger.new){
    companyName.add(newAcc.Name);
  }
  Set<Account> duplicatedAccounts = new Set<Account>([SELECT Id,Name FROM Account WHERE Name
Like :companyName]);
  if(duplicatedAccounts.size() > 0){
    for(Account newAcc : trigger.new){
      newAcc.addError('Account Already Present In The System. Please Search Account Name in the org');
    }
    }
}
```

4.Assign Record Type based on business logic.



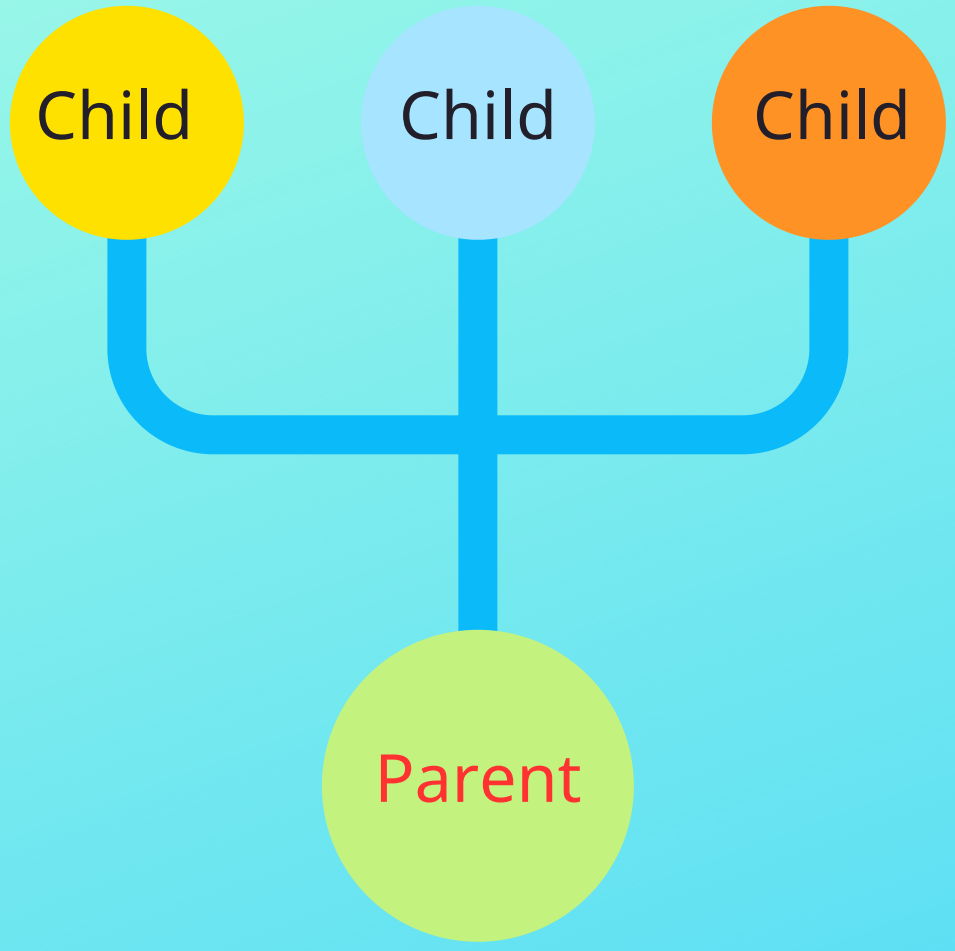

```
trigger AssignRecortType on Opportunity (After insert,before update) {

    Map<String,Id> rTypeMap = new Map<String,Id>();
    for(RecordType rt: [SELECT Id,developerName FROM RecordType WHERE sobjectType ='Opportunity']){
        rTypeMap.put(rt.developerName,rt.Id);
    }

    for(Opportunity opp: trigger.new){
        if(opp.StageName =='Negotiation/Review'){
            //Change the record type to readonly
            opp.RecordTypeId = rTypeMap.get('ReadOnly');
        }
    }
}
```

5.Update child records from parent record.



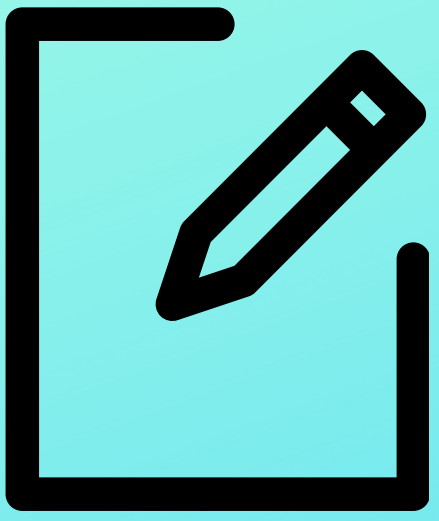

```
trigger updateChildRecord on Account (after update) {

    set<Id> accId = new set<Id>();
    for(Account acc:Trigger.new){
        Account oldAccount = trigger.oldMap.get(acc.id);
        boolean isChange = (acc.BillingCity != oldAccount.BillingCity) || (acc.BillingCountry != oldAccount.BillingCountry);
        if(isChange){
            accId.add(acc.id);
        }
    }

    if(accId.size() > 0){
        list<contact> lstContact = new list<contact>([SELECT Id,OtherCity,OtherCountry,AccountId FROM Contact WHERE AccountId IN : accId]);
        //Update all the records
        if(lstContact.size() > 0){
            for(contact con :lstContact){
                //Get Parent
                Account updtAccount = trigger.newMap.get(con.AccountId);
                //update childs
                 con.OtherCity = updtAccount.BillingCity;
                 con.OtherCountry = updtAccount.BillingCountry;
            }
            update lstContact;
        }
    }
}
```

6.Update parent record from child records.




```
trigger updateParentRecord on contact (after insert) {

    List<Account> parentRec = new List<Account>();
    for(contact con:trigger.new){
        if(con.Phone != null){
            account acc = new account(Id=con.AccountId);
            acc.sfc17__Status__c = true;
            parentRec.add(acc);
        }
    }
    if(parentRec.size() > 0){
        update parentRec;
    }
}
```

7.Create or delete child records from parent record.

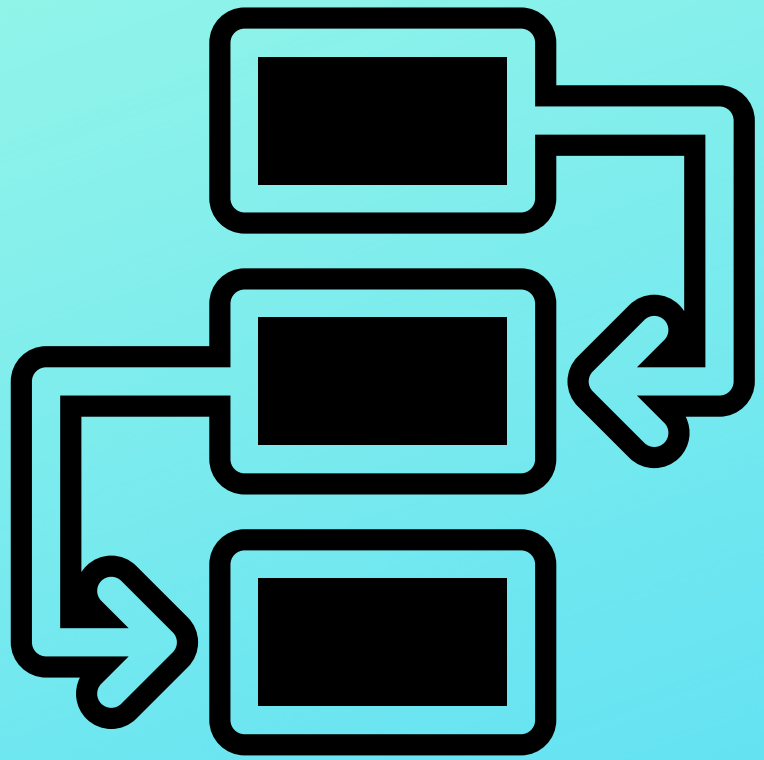



```
1trigger CreateOrDeleteChild on Account (after insert,after update,before delete){

    List<Opportunity> lstOpp = new List<Opportunity>();
    If(trigger.isInsert || trigger.isUpdate){
        Map<Id,Account> accWithOpp = new Map<Id,Account>([SELECT Id,(SELECT Id FROM Opportunities) FROM Account Where Id IN :trigger.new]);
        for(Account acc: trigger.new){
            if(accWithOpp.get(acc.id).Opportunities.size() == 0){
                lstOpp.add(new Opportunity(Name = acc.Name + ' Opportunity',
                                           AccountId = acc.Id,
                                           Type = 'New Customer',
                                           RecordTypeId ='01201000000nfl5QAA',
                                           Amount = 150,
                                           StageName ='Qualification',
                                           CloseDate = System.today().addMonths(1) ));
            }
        }
        if(lstOpp.size() > 0){
            insert lstOpp; // insert Opportunity.
        }
    }
    //Don't Delete account if there is any opportunity
    if(trigger.isDelete){
        for(Account acc:[SELECT Id,(SELECT Id FROM Opportunities) FROM Account Where Id IN :trigger.old]){
            trigger.oldMap.get(acc.Id).addError('Can not delete account with opportunity');
        }
    }
}
```

8.If deleting or restoring parent record then child records should also perform same action (In lookup relationship).

```
1trigger DeleteRelatedList on Contact (before delete,after undelete) {

  List<Scen1__c> lstchildsDelete = new List<Scen1__c>();
  set<id>  parentId = new set<id>();

  If (trigger.isDelete){

    for(Contact co:trigger.old){
      parentId.add(co.id);
    }

    List<Scen1__c> lstChilds = new List<Scen1__c>([Select Id from sfc17__Scen1__c where ContactDetail__c IN :parentId]);
    If(lstChilds.size() > 0) {
      Delete lstChilds;
    }
  }

  if(trigger.isUndelete){
    set<Id> parentRecord = new set<Id>(Trigger.newMap.keyset());
    List<Scen1__c> restoreChilds = new List<Scen1__c>([Select Id From sfc17__Scen1__c Where  ContactDetail__c IN :parentRecord ALL ROWS]);
    if(restoreChilds.size() > 0){
      undelete restoreChilds;
    }
  }
}
```

9.Rollup summary calculation if two objects have lookup relationship.



```
trigger rollUpSummaryField on Opportunity (after insert,after update,after delete,after undelete) {

    set<Id> accIds = new set<Id>();
    if(trigger.isInsert || trigger.isUndelete){
        for(Opportunity opp:trigger.new){
            accIds.add(opp.AccountId);
        }
    }

    if(trigger.isUpdate){
        for(Opportunity opp:trigger.new){
            if(opp.Amount != trigger.OldMap.get(opp.Id).Amount){
                accIds.add(opp.AccountId);
            }
        }
    }

    if(trigger.isDelete){
        for(Opportunity opp:trigger.old){
            accIds.add(opp.AccountId);
        }
    }

    if(!accIds.IsEmpty()){
        Map<Id,Double> mapOfSumAmount = new Map<Id,Double>();
        AggregateResult[] sumOfAmount =[SELECT SUM(Amount)sumAmount, AccountId FROM opportunity WHERE AccountId IN :accIds group by AccountId];
        for(AggregateResult agg: sumOfAmount){
            Id AccId = (Id)agg.get('AccountId');
            double totalOppAmount = (double)agg.get('sumAmount');
            mapOfSumAmount.put(AccId,totalOppAmount);
        }

        List<Account> lstAcc = new List<Account>([SELECT Id,sfc17__Sum_Amount__c FROM Account WHERE Id IN:accIds]);
        if(!lstAcc.isEmpty()){
            List<Account> updateAcc = new List<Account>();
            for(Account acc:lstAcc){
                if(mapOfSumAmount.containsKey(acc.Id)){
                    acc.sfc17__Sum_Amount__c = mapOfSumAmount.get(acc.Id);
                }else{
                    acc.sfc17__Sum_Amount__c = 0;
                }
            }
            update lstAcc;
        }
    }
}
```

10. Trigger logic that create activities, apex sharing & add user in teams member etc.

"Follow the Trigger Best Practices. If you know well about Map & list usage then you can write trigger logic easily."

LIST

When should I use list ?

SET

Should I use set or list ? Can we perform DML in set elements?

MAP

Can map manage child record of parent record? Is it possible to use map inside map ? or List inside Map ?