1.

trigger updatePhone on Account (before insert) {

for(Account acc: Trigger.New){

if(acc.Phone!=null)

acc.phone='+91'+acc.Phone;

}

}

2.

trigger updateType on Account (before insert,before update) {

for(Account ac:Trigger.new){

if(ac.Type=='Prospect')

ac.Status \_\_c='Active';

else

ac.Status \_\_c='Inactive';

if(ac.Status \_\_c=='Active' && ac.Active\_\_c!=true)

ac.Active\_\_c.addError('Active field should be checked);

else if(ac.Status \_\_c=='Inactive' && ac.Active\_\_c!=false)

ac.Active\_\_c.addError('Active field should be Unchecked);

}

}

3.

trigger createContact on Account (after insert, after update){

// we have to insert conatcts based on the count on Account's No of contact field, so we need a list of contacts

List<contact> contlist=new List<contact>();

for(Account a:Trigger.new){

integer i=0; //contact number

if(a.No\_of\_Contact\_\_c!=null){

// based on no of contact field we will create contact

if((trigger.isUpdate && a.No\_of\_Contact\_\_c != Trigger.oldMap.get(a.id).No\_of\_Contact\_\_c) || trigger.isInsert){

/\* here if account record is being inserted then we will need to create contact or we are updating no of contact field on the account \*/

while(i<a.No\_of\_Contact\_\_c){

contact c=new contact();

c.LastName='Cont Of '+a.Name+'-'+i;

c.AccountId=a.id;

contlist.add(c);

i++;

}

}

}

}

if(contlist.size()>0)

insert contlist;

}

4.

If we will code it using for loop and list:

trigger saveAccountForEmailTrigger on Lead(before insert){

List<Account> acList = new List<Account>();

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

for(Lead ld : Trigger.New){

if(ld.email!=null){

emailList.add(ld.email);

}

}

if(emailList.size()>0){

acList = [SELECT name, email\_\_c FROM Account WHERE email\_\_c IN : emailList];

}

if(acList.size()>0){

for(Lead ld : Trigger.New){

for(Account acc : acList){

if(ld.email == acc.email\_\_c){

acc.Name = ld.LastName;

}

}

}

}

else{

for(Lead ld : Trigger.New){

Account acc = new Account();

acc.Name = ld.LastName;

acc.email\_\_c = ld.email;

acList.add(acc);

}

}

if(upsertlist.size()>0){

upsert upsertlist;

}

}

5.

trigger emailAcc on Lead (before insert) {

set <string> emailset=new set <string>();

for(lead ld: Trigger.New){

if(ld.email!=null){

emailset.add(ld.email);

}

}

Map<String,Account> emailmap=new Map<String,Account>();

if(emailset!=null && emailset.size()>0){

list<Account> aclist=[Select Id,name,email\_\_c from Account where email\_\_c in:emailset];

if(aclist.size()>0){

for(Account a: aclist)

emailmap.put(a.email\_\_c,a);

}

}

list<Account> upsertlist=new List<Account>();

for(lead ld: Trigger.New){

Account acc = new Account();

if(emailmap.size()>0 && emailmap.containsKey(ld.email)){

acc = emailmap.get(ld.email);

acc.name=ld.LastName;

upsertlist.add(acc);

}else{

acc.Name= ld.LastName;

acc.email\_\_c = ld.email;

upsertlist.add(acc);

}

}

if(upsertlist.size()>0){

upsert upsertlist;

}

}

6.

trigger PopulateArea on Account(before insert,after delete){

if(trigger.isInsert){

Set<id> strNew = new Set<id>();

for(Account ac:trigger.new)

strNew.add(ac.Area\_\_c);

Map<id,Area\_\_c> mapArea = new Map<id,Area\_\_c>([Select Name,AreaCode\_\_c,AreaCount\_\_c,(select id, Area\_Count\_\_c from Accounts\_\_r Order By createdDate DESC LIMIT 1) from Area\_\_c where id IN: strNew]);

Map<id,Integer> mapCount = new Map<id,Integer>();

/\* In the above section, we are operating upon insertion.

First, we added the id of the Account in the set that we have just inserted then queried.

The fields of the Area and the related Account in the inner query to work upon these fields. We retrieve the values and opt for the last created account so that we have the last counter value to increase it.

\*/

for(Account act:trigger.new){

if(mapArea.containsKey(act.Area\_\_c)){

Integer count = 0, arCount=0;

if(mapArea.get(act.Area\_\_c).Accounts\_\_r.size()>0)

arCount =Integer.valueOf(mapArea.get(act.Area\_\_c).Accounts\_\_r[0].Area\_Count\_\_c.split('-')[1]);

if(mapCount.containsKey(act.Area\_\_c)){

count = arCount + 1 + apCount.get(act.Area\_\_c);

Integer mpcnt=mapCount.get(act.Area\_\_c)+1;

mapCount.put(act.Area\_\_c,mpcnt);

}

else{

count = arCount + 1;

mapCount.put(act.Area\_\_c,1);

}

if(count<10)

act.Area\_Count\_\_c = mapArea.get(act.Area\_\_c).AreaCode\_\_c + '-00' + count;

else if(count>=10 && count<100)

act.Area\_Count\_\_c = mapArea.get(act.Area\_\_c).AreaCode\_\_c + '-0' + count;

else if(count>100)

act.Area\_Count\_\_c = mapArea.get(act.Area\_\_c).AreaCode\_\_c + '-' + count;

}

}

/\*

In the above code, we check if the map contains that area, then increase the countMap by 1 to store the count of the Area and increment it .and put that counter value on the map. In the next logic, we have just checked if it crosses the limited value to a particular range then the counter will change.

\*/

for(Area\_\_c ar: mapArea.values()){

if(mapCount.containsKey(ar.id))

ar.AreaCount\_\_c += mapCount.get(ar.id);

}

update mapArea.values();

}

/\*

Then, at last, we put the Area\_Count value in the field from the map and update it.

\*/

if(trigger.isDelete){

Set<id> strOld = new Set<id>();

Map<id,Area\_\_c> mapDel = new Map<id,Area\_\_c>();

Map<id,Integer> mapDelCount = new Map<id,Integer>();

for(Account acct : trigger.old)

strOld.add(acct.Area\_\_c);

List<Area\_\_c> arr = [select Name,AreaCode\_\_c,AreaCount\_\_c from Area\_\_c where id IN: strOld];

for(Area\_\_c area : arr)

mapDel.put(area.id, area);

for(Account acct : trigger.old){

if(mapDel.containsKey(acct.Area\_\_c)){

if(mapDelCount.containsKey(acct.Area\_\_c)){

Integer countDel=mapDelCount.get(acct.Area\_\_c)+1;

mapDelCount.put(acct.Area\_\_c,countDel);

}

else

mapDelCount.put(acct.Area\_\_c,1);

}

}

for(Area\_\_c ar: mapDel.values()){

if(mapDelCount.containsKey(ar.id))

ar.AreaCount\_\_c -= mapDelCount.get(ar.id);

}

update mapDel.values();

}

}

7.

trigger LeaveDaysCount on Leave\_Application\_\_c (before insert,before update) {  
 Integer count=0;  
 set<Date> holidays=new set<date>();  
 for(Holiday\_\_c c:[SELECT date\_\_c from Holiday\_\_c]){  
 holidays.add(c.date\_\_c);  
 }  
 For(Leave\_Application\_\_c l:Trigger.New){  
 for(Date d=l.FromDate\_\_c; d<=l.ToDate\_\_c; d=d+1){  
 Datetime dt=date.newInstance(d.year(), d.month(), d.day());  
 if(dt.format('EEEE')!='Sunday' && dt.format('EEEE')!='Saturday' && !holidays.contains(d)){  
 count++;  
 }   
 }  
 l.CountDays\_\_c=count;  
 }  
}

8.

trigger SingleTrigger on Lead (after insert, after update, before insert, before update)

{

//helper class name

TriggerHelper class1 = new TriggerHelper();

if(trigger.isAfter && trigger.isInsert)

{

class1.Method1(trigger.New, trigger.newMap);

}

if(trigger.isBefore && trigger.isInsert)

{

//sequence which method 1st

class1.Method2(trigger.New, trigger.newMap);

class1.Method3(trigger.New, trigger.newMap);

}

if(trigger.isBefore && trigger.isUpdate)

{

class1.Method4(trigger.New, trigger.newMap,trigger.Old,trigger.oldMap);

}

if(trigger.isAfter && trigger.isUpdate)

{

class1.Method5(trigger.New, trigger.newMap,trigger.Old,trigger.oldMap);

}

}

public class TriggerHelper {

public void Method1(list<Lead> triggerNew,map<Id,Lead> triggerNewmap){

---

}

public void Method2(list<Lead> triggerNew,map<Id,Lead> triggerNewmap){

---

}

public void Method3(list<Lead> triggerNew,map<Id,Lead> triggerNewmap){

---

}

public void Method4(list<Lead> triggerNew,map<Id,Lead> triggerNewmap, list<Lead> triggerOld,map<Id,Lead> triggerOldmap){

---

}

public void Method5(list<Lead> triggerNew,map<Id,Lead> triggerNewmap, list<Lead> triggerOld,map<Id,Lead> triggerOldmap){

---

}

}