1. **Auto-update the Account's LastActivityDate when a new Task is created**
2. **Auto-update the Account's Total revenue when a new Opp is Closed**

**Object:** Task **Type of Trigger:** After Insert

apex

Copy code

trigger UpdateAccountLastActivityDate on Task (after insert) {

Set<Id> accountIds = new Set<Id>();

for (Task task : Trigger.new) {

if (task.WhatId != null && task.WhatId.getSObjectType() == Account.SObjectType) {

accountIds.add(task.WhatId);

}

}

if (!accountIds.isEmpty()) {

List<Account> accounts = [SELECT Id, LastActivityDate FROM Account WHERE Id IN :accountIds];

for (Account acc : accounts) {

acc.LastActivityDate = System.today();

}

update accounts;

}

}

**2. Validate that no two contacts have the same email within an Account**

**Object:** Contact **Type of Trigger:** Before Insert

apex

Copy code

trigger ValidateContactEmail on Contact (before insert) {

Map<Id, Set<String>> accountEmailMap = new Map<Id, Set<String>>();

for (Contact con : Trigger.new) {

if (con.AccountId != null && con.Email != null) {

if (!accountEmailMap.containsKey(con.AccountId)) {

accountEmailMap.put(con.AccountId, new Set<String>());

}

accountEmailMap.get(con.AccountId).add(con.Email);

}

}

List<Contact> existingContacts = [SELECT AccountId, Email FROM Contact WHERE AccountId IN :accountEmailMap.keySet() AND Email IN :accountEmailMap.values()];

for (Contact existing : existingContacts) {

if (accountEmailMap.get(existing.AccountId).contains(existing.Email)) {

for (Contact con : Trigger.new) {

if (con.Email == existing.Email && con.AccountId == existing.AccountId) {

con.addError('Email address already exists for this Account.');

}

}

}

}

}

**3. Prevent Opportunity stage change if there are open Tasks**

**Object:** Opportunity **Type of Trigger:** Before Update

apex

Copy code

trigger PreventStageChange on Opportunity (before update) {

Set<Id> oppIds = new Set<Id>();

for (Opportunity opp : Trigger.new) {

Opportunity oldOpp = Trigger.oldMap.get(opp.Id);

if (oldOpp.StageName != opp.StageName) {

oppIds.add(opp.Id);

}

}

List<Task> openTasks = [SELECT Id, WhatId FROM Task WHERE Status != 'Completed' AND WhatId IN :oppIds];

if (!openTasks.isEmpty()) {

for (Opportunity opp : Trigger.new) {

if (oppIds.contains(opp.Id)) {

opp.addError('Cannot change stage while there are open tasks.');

}

}

}

}

**4. Auto-create a new Case when a high-value Opportunity is closed**

**Object:** Opportunity **Type of Trigger:** After Update

apex

Copy code

trigger CreateCaseOnOpportunityClose on Opportunity (after update) {

List<Case> casesToCreate = new List<Case>();

for (Opportunity opp : Trigger.new) {

Opportunity oldOpp = Trigger.oldMap.get(opp.Id);

if (oldOpp.StageName != 'Closed Won' && opp.StageName == 'Closed Won' && opp.Amount > 50000) {

Case newCase = new Case();

newCase.Subject = 'Follow-up on high-value Opportunity';

newCase.AccountId = opp.AccountId;

newCase.Description = 'A high-value opportunity has been closed.';

newCase.Origin = 'Web';

casesToCreate.add(newCase);

}

}

if (!casesToCreate.isEmpty()) {

insert casesToCreate;

}

}

**5. Populate the Lead source on Opportunity from related Account**

**Object:** Opportunity **Type of Trigger:** Before Insert

apex

Copy code

trigger PopulateLeadSourceOnOpportunity on Opportunity (before insert) {

Set<Id> accountIds = new Set<Id>();

for (Opportunity opp : Trigger.new) {

if (opp.AccountId != null) {

accountIds.add(opp.AccountId);

}

}

Map<Id, Account> accountMap = new Map<Id, Account>([SELECT Id, LeadSource FROM Account WHERE Id IN :accountIds]);

for (Opportunity opp : Trigger.new) {

if (opp.AccountId != null && accountMap.containsKey(opp.AccountId)) {

opp.LeadSource = accountMap.get(opp.AccountId).LeadSource;

}

}

}

**6. Update Opportunity close date based on related Quote close date**

**Object:** Quote **Type of Trigger:** After Update

apex

Copy code

trigger UpdateOpportunityCloseDate on Quote (after update) {

Set<Id> oppIds = new Set<Id>();

for (Quote quote : Trigger.new) {

if (quote.OpportunityId != null) {

oppIds.add(quote.OpportunityId);

}

}

Map<Id, Opportunity> oppMap = new Map<Id, Opportunity>([SELECT Id, CloseDate FROM Opportunity WHERE Id IN :oppIds]);

for (Quote quote : Trigger.new) {

if (quote.OpportunityId != null && oppMap.containsKey(quote.OpportunityId)) {

Opportunity opp = oppMap.get(quote.OpportunityId);

opp.CloseDate = quote.ExpirationDate;

}

}

if (!oppMap.isEmpty()) {

update oppMap.values();

}

}

**7. Ensure that a Contact's email is updated if related Lead's email changes**

**Object:** Lead **Type of Trigger:** After Update

apex

Copy code

trigger UpdateContactEmail on Lead (after update) {

Set<Id> leadIds = new Set<Id>();

for (Lead lead : Trigger.new) {

if (lead.ConvertedContactId != null) {

leadIds.add(lead.Id);

}

}

Map<Id, Lead> leadMap = new Map<Id, Lead>([SELECT Id, ConvertedContactId, Email FROM Lead WHERE Id IN :leadIds]);

List<Contact> contactsToUpdate = new List<Contact>();

for (Lead lead : Trigger.new) {

if (lead.ConvertedContactId != null && leadMap.containsKey(lead.Id)) {

Lead oldLead = Trigger.oldMap.get(lead.Id);

if (lead.Email != oldLead.Email) {

Contact contact = new Contact(Id = lead.ConvertedContactId, Email = lead.Email);

contactsToUpdate.add(contact);

}

}

}

if (!contactsToUpdate.isEmpty()) {

update contactsToUpdate;

}

}

**8. Prevent deletion of Accounts with active Opportunities**

**Object:** Account **Type of Trigger:** Before Delete

apex

Copy code

trigger PreventAccountDeletion on Account (before delete) {

Set<Id> accountIds = Trigger.oldMap.keySet();

List<Opportunity> activeOpportunities = [SELECT Id, AccountId FROM Opportunity WHERE AccountId IN :accountIds AND IsClosed = false];

if (!activeOpportunities.isEmpty()) {

for (Account acc : Trigger.old) {

for (Opportunity opp : activeOpportunities) {

if (opp.AccountId == acc.Id) {

acc.addError('Cannot delete Account with active Opportunities.');

}

}

}

}

}

**9. Copy Billing Address from Account to related Contacts**

**Object:** Contact **Type of Trigger:** Before Insert

apex

Copy code

trigger CopyBillingAddress on Contact (before insert) {

Set<Id> accountIds = new Set<Id>();

for (Contact con : Trigger.new) {

if (con.AccountId != null) {

accountIds.add(con.AccountId);

}

}

Map<Id, Account> accountMap = new Map<Id, Account>([SELECT Id, BillingStreet, BillingCity, BillingState, BillingPostalCode, BillingCountry FROM Account WHERE Id IN :accountIds]);

for (Contact con : Trigger.new) {

if (con.AccountId != null && accountMap.containsKey(con.AccountId)) {

Account acc = accountMap.get(con.AccountId);

con.MailingStreet = acc.BillingStreet;

con.MailingCity = acc.BillingCity;

con.MailingState = acc.BillingState;

con.MailingPostalCode = acc.BillingPostalCode;

con.MailingCountry = acc.BillingCountry;

}

}

}

**10. Calculate and update Total Revenue on Account when new Opportunity closes**

**Object:** Opportunity **Type of Trigger:** After Update

apex

Copy code

trigger UpdateTotalRevenue on Opportunity (after update) {

Set<Id> accountIds = new Set<Id>();

for (Opportunity opp : Trigger.new) {

if (opp.StageName == 'Closed Won' && Trigger.oldMap.get(opp.Id).StageName != 'Closed Won') {

accountIds.add(opp.AccountId);

}

}

Map<Id, Account> accountMap = new Map<Id, Account>([SELECT Id, Total\_Revenue\_\_c FROM Account WHERE Id IN :accountIds]);

for (Opportunity opp : Trigger.new) {

if (opp.StageName == 'Closed Won' && accountMap.containsKey(opp.AccountId)) {

Account acc = accountMap.get(opp.AccountId);

acc.Total\_Revenue\_\_c += opp.Amount;

}

}

if (!accountMap.isEmpty()) {

update accountMap.values();

}

}

**11. Send email notification when a high-priority Case is created**

**Object:** Case **Type of Trigger:** After Insert

apex

Copy code

trigger SendEmailOnHighPriorityCase on Case (after insert) {

List<Messaging.SingleEmailMessage> emails = new List<Messaging.SingleEmailMessage>();

for (Case c : Trigger.new) {

if (c.Priority == 'High') {

Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage();

email.setToAddresses(new String[] {'support@company.com'});

email.setSubject('High Priority Case Created: ' + c.CaseNumber);

email.setPlainTextBody('A new high priority case has been created. Case Number: ' + c.CaseNumber);

emails.add(email);

}

}

if (!emails.isEmpty()) {

Messaging.sendEmail(emails);

}

}

**12. Auto-update the Account's industry based on the Opportunity's product**

**Object:** Opportunity **Type of Trigger:** After Insert

apex

Copy code

trigger UpdateAccountIndustry on Opportunity (after insert) {

Set<Id> accountIds = new Set<Id>();

for (Opportunity opp : Trigger.new) {

if (opp.AccountId != null) {

accountIds.add(opp.AccountId);

}

}

Map<Id, Account> accountMap = new Map<Id, Account>([SELECT Id, Industry FROM Account WHERE Id IN :accountIds]);

for (Opportunity opp : Trigger.new) {

if (opp.AccountId != null && accountMap.containsKey(opp.AccountId)) {

Account acc = accountMap.get(opp.AccountId);

acc.Industry = opp.Product\_Family\_\_c; // Assuming Product\_Family\_\_c is the custom field on Opportunity

}

}

if (!accountMap.isEmpty()) {

update accountMap.values();

}

}

**13. Prevent Contact deletion if the Contact is a Primary Contact on any Account**

**Object:** Contact **Type of Trigger:** Before Delete

apex

Copy code

trigger PreventPrimaryContactDeletion on Contact (before delete) {

Set<Id> contactIds = Trigger.oldMap.keySet();

List<Account> accounts = [SELECT Id FROM Account WHERE Primary\_Contact\_\_c IN :contactIds];

if (!accounts.isEmpty()) {

for (Contact con : Trigger.old) {

for (Account acc : accounts) {

if (acc.Primary\_Contact\_\_c == con.Id) {

con.addError('Cannot delete Primary Contact for an Account.');

}

}

}

}

}

**14. Auto-close a Case when related Opportunity is closed**

**Object:** Opportunity **Type of Trigger:** After Update

apex

Copy code

trigger AutoCloseCases on Opportunity (after update) {

Set<Id> oppIds = new Set<Id>();

for (Opportunity opp : Trigger.new) {

if (opp.StageName == 'Closed Won' && Trigger.oldMap.get(opp.Id).StageName != 'Closed Won') {

oppIds.add(opp.Id);

}

}

List<Case> casesToClose = [SELECT Id, Status FROM Case WHERE OpportunityId IN :oppIds AND Status != 'Closed'];

for (Case c : casesToClose) {

c.Status = 'Closed';

}

if (!casesToClose.isEmpty()) {

update casesToClose;

}

}

**15. Ensure Account's Annual Revenue is not negative**

**Object:** Account **Type of Trigger:** Before Insert/Update

apex

Copy code

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

for (Account acc : Trigger.new) {

if (acc.AnnualRevenue < 0) {

acc.addError('Annual Revenue cannot be negative.');

}

}

}

**16. Create a follow-up Task when a new Lead is created**

**Object:** Lead **Type of Trigger:** After Insert

apex

Copy code

trigger CreateFollowUpTask on Lead (after insert) {

List<Task> tasks = new List<Task>();

for (Lead lead : Trigger.new) {

Task task = new Task();

task.Subject = 'Follow-up with Lead';

task.WhatId = lead.Id;

task.OwnerId = lead.OwnerId;

task.ActivityDate = Date.today() + 7;

tasks.add(task);

}

if (!tasks.isEmpty()) {

insert tasks;

}

}

**17. Calculate and update Total Opportunity Amount on Account**

**Object:** Opportunity **Type of Trigger:** After Insert/Update/Delete

apex

Copy code

trigger UpdateTotalOpportunityAmount on Opportunity (after insert, after update, after delete) {

Set<Id> accountIds = new Set<Id>();

for (Opportunity opp : Trigger.new) {

accountIds.add(opp.AccountId);

}

for (Opportunity opp : Trigger.old) {

accountIds.add(opp.AccountId);

}

Map<Id, Decimal> accountTotalMap = new Map<Id, Decimal>();

for (AggregateResult ar : [SELECT AccountId, SUM(Amount) total FROM Opportunity WHERE AccountId IN :accountIds AND IsClosed = true GROUP BY AccountId]) {

accountTotalMap.put((Id)ar.get('AccountId'), (Decimal)ar.get('total'));

}

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

for (Id accountId : accountIds) {

Account acc = new Account(Id = accountId);

acc.Total\_Opportunity\_Amount\_\_c = accountTotalMap.containsKey(accountId) ? accountTotalMap.get(accountId) : 0;

accountsToUpdate.add(acc);

}

if (!accountsToUpdate.isEmpty()) {

update accountsToUpdate;

}

}