Create a trigger after creation of Account record to query list of all Contacts associated with the triggering Accounts

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

        for(Account a : Trigger.new){

                List<Contact> conList = [SELECT LastName FROM Contact WHERE AccountId IN
                :Trigger.new];

        }

}
```

Everytime Opp is closed won, trigger should fire and do the following:

1) Create a new shipment record
2) Shipment name should be same as Opp name
3) Delivery date in Shipment record should be same as Close date of the Opportunity
4) Update the opportunity description field with the unique id of shipment once it is created

```
Trigger OppUpdated ( before update){

        List<Shipment> shipList = new List<Shipment>();

        For(Integer i = 0; i < Trigger.new.size(); i++){

                If(Trigger.new[i].StageName != Trigger.old[i].StageName && Trigger.new[i].StageName
== 'Closed Won'){

                        Shipment ship = new Shipment(Name = opp.Name, Delivery_Date__c =
                        opp.CloseDate);

                }

        }

}
```

On creation on new Opportunity, create a new Contact with the same account name which is in Opportunity

```
trigger ContactAccountOpportunity on Opportunity(after insert){
 List<Contact> cList = new List<Contact>();
   for(Opportunity o : Trigger.new){
      Contact c = new Contact();
       c.AccountId = o.AccountId;
       c.FirstName = 'Opportunity';
       c.LastName = 'Owner';
       cList.add(c);
   }
   insert cList;
}
```

On creation of an account with Industry = Other, it should throw error

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

            for(Account a : Trigger.new){

                        if(a.Industry == ‘Other’){

                                    a.addError(‘Please choose Industry from the list’);

                        }

            }

}
```

No new opportunity should be created if the Account Industry is Other

Approach 1)

```
trigger OtherOppNotAllowed on Opportunity(before insert){
   for(Opportunity o: Trigger.new){
       String s = o.Account.Industry;
       if(s == 'Other'){
          o.addError('Opp should not be created for Other Account');
       }
   }
}
```

Approach 2)

```
trigger OtherOppNotAllowed on Opportunity(before insert){

   for(Opportunity o: Trigger.new){

       Id aid = o.Account.Id;

       Account a = [SELECT Name, Id FROM Account WHERE Id = :aid];

        if(a.Industry == 'Other'){

   o.addError(‘Other Opp not allowed’);

       }
```

```
    }

}
```

If Opportunity created with Account ABC, add Description in Account ABC ‘Opportunity.Name created on this account’

```
trigger UpdateAccountOnOppCreation on Opportunity(before insert){
    for(Opportunity o : Trigger.new){
        o.Account.Description = o.Name+' Opp created';
    }
}
```

If Customer Status field of Customer object changes from any other state to inactive, create a new Invoice

Approach 1 – using traditional For loop, Trigger.new[] and Trigger.old[]

```
trigger CustomerStatusChanged on APEX_Customer__c(before update){
    List<APEX_Invoice__c> invList = new List<APEX_Invoice__c>();
    for(Integer i=0; i<Trigger.new.size(); i++){
        if(Trigger.new[i].APEX_Customer_Status__c == 'active' && Trigger.old[i].APEX_Customer_Status__c != 'active')
        {
            APEX_Invoice__c inv = new APEX_Invoice__c(APEX_Status__c = 'Pending', APEX_Customer__c = Trigger.
new[i].Id);
            invList.add(inv);
        }
    }
    insert invList;
}
```

Approach 2 – using list for loop, Trigger.new and Trigger.oldMap

```
trigger CustomerStatusChanged on APEX_Customer__c(before update){

    List<APEX_Invoice__c> invList = new List<APEX_Invoice__c>();

    for(APEX_Customer__c c : Trigger.new){

        APEX_Customer__c oldCustomer = Trigger.oldMap.get(c.Id);

        if(c.APEX_Customer_Status__c == 'active' &&

           oldCustomer.APEX_Customer_Status__c != 'active' )

           {
```

```
            APEX_Invoice__c inv = new APEX_Invoice__c(APEX_Status__c = 'Pending', APEX_Customer__c = c.Id);
            invList.add(inv);
        }
    }
    insert invList;
}
```

If new Contact created without Account and update its description as neglected Contact

```
trigger NeglectedContact on Contact (before insert) {
    for(Contact c: Trigger.new){
        if(String.isBlank(c.AccountId)){
            c.Description = 'Neglected Contact';
        }
    }
}
```

New Account should not be created with same name

Approach 1)

```
trigger NoDupeAccount on Account (before insert) {
    List<Account> accList = [SELECT Name FROM Account];
    for(Account a1: Trigger.new){
        for(Account a2: accList){
            if(a1.Name == a2.Name){
                a1.addError('Duplicate Account not allowed');
            }
        }
    }
}
```

Approach 2)

```
trigger NoDupeAccount on Account (before insert) {
    List<Account> accList;
```

```
    for(Account a1: Trigger.new){
        accList = [SELECT Name FROM Account WHERE Name = :a1.Name];
        if(accList.size() > 0){
            a1.addError('Duplicate account not allowed');
        }
    }
}
```

If a user creates opps in a day worth more than $ 50,000, throw error

Approach one:

```
trigger Opportunity_perDayLimit on Opportunity(before insert, before update){
    List<Opportunity> oppList = [SELECT Amount FROM Opportunity WHERE CreatedDate = TODAY and CreatedById = :UserInfo.getUserID()];
    Decimal amount = 0;
    for(Opportunity o : oppList){
        try{
            amount += o.Amount;
            }catch(Exception e){}
    }

    for(Opportunity o : Trigger.new){
        try{
            if(amount + o.Amount > 50000)
                o.addError('You cannot create opportunities exceeding $50,000 in a day');
            }catch(Exception e){}
    }
  }
```

Approach 2) Using helper class

Helper class:

```
public class OppLimitHelper{
  static Decimal amount = 0;
  public static void helper(List<Opportunity> triggerNew){
```

```
        for(Opportunity o : [SELECT Amount FROM Opportunity WHERE CreatedDate = TODAY and CreatedById = :U
serInfo.getUserId()]){
            try{
                amount += o.Amount;
                }catch(Exception e){}
        }

        for(Opportunity o : triggerNew){
            try{
                amount += o.Amount;
                if(amount > 100000){
                    o.addError('Your daily opportunity limit has exceeded $100,000');
                    }
                    }catch(Exception e){}
            }
        }

    }
```

```
Trigger: trigger Opportunity_perDayLimit on Opportunity(before insert, before update){
        OppLimitHelper.helper(Trigger.new);
    }
```

If Account has no Opportunity, update its description as neglected Account

If Contact is created, automatically create an Account

Throw error if same record is being modified by multiple user

Create a button to close won opp. When an opp is being closed won, pop up message, do you want to close won the opp, if clicked ok, change it to closed won.

If opp is closed won, pop up message saying opp is closed won