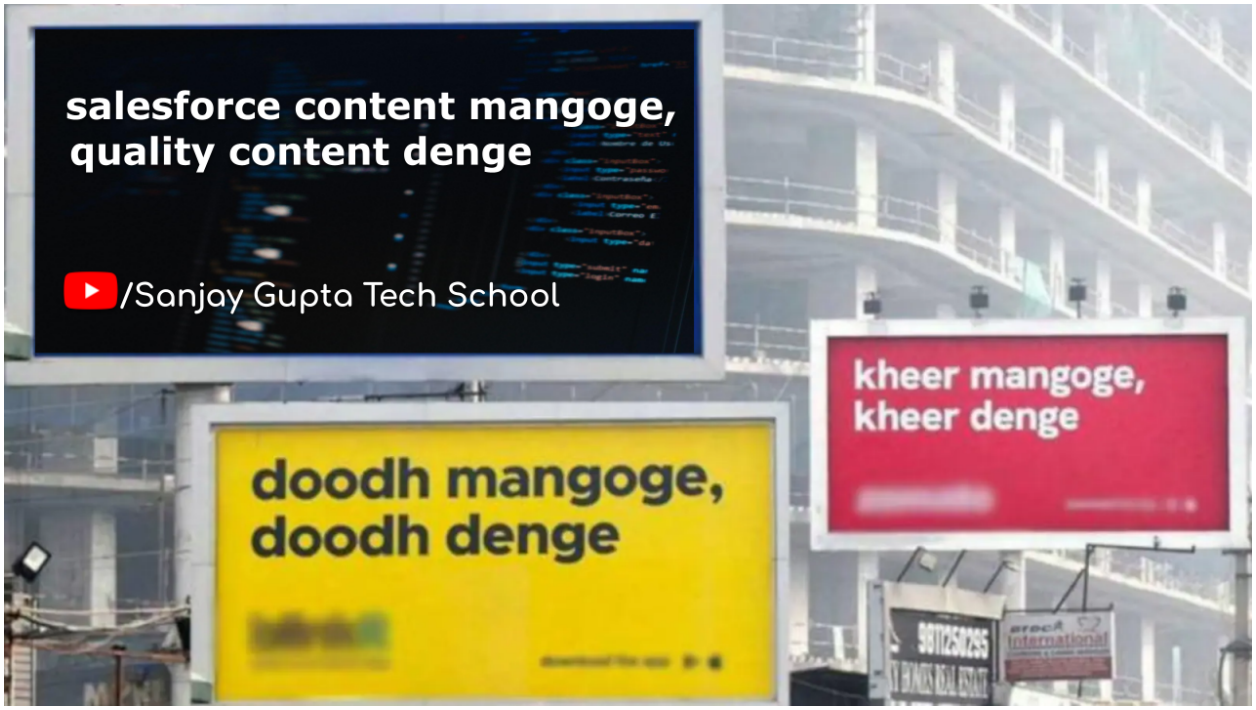

Solved

Apex Scenarios

(Test Your Apex Skills)

Sanjay Gupta Tech School

https://studysalesforce.com

1. Query 5 Account records into a list. Now iterate on a list and create a set of account Ids and display values of the set.

Version 1:

```
public class ApexDemo01 {

    public static void listToSetUsingLoop( ){
        List<Account> accList = New List<Account>( );
        Set<id> idset= New Set<id>();

        accList = [SELECT Id FROM Account Limit 5];

        for (Account act : accList){
            idSet.add(act.Id);
        }

        for (Id i : idset){
            System.debug('Account Id: ' + i);
        }
    }
```

Version 2:

```
public class ApexDemo01 {

    public static void listToSetUsingLoop( ){
        List<Account> accList = New List<Account>( );
        Set<id> idset= New Set<id>();

        for (Account act : [SELECT Id FROM Account Limit 5]){
            idSet.add(act.Id);
        }

        for (Id i : idset){
            System.debug('Account Id: ' + i);
        }
    }
```

2. Query 5 Account records into a list. Now iterate on a list and create a map where key will be account Id and value will be Account sObject (using loop) and display values of map.

Version 1:

```
public class ApexDemo02 {

        public static void listToMapUsingLoop( ){
            List<Account> accList= New List<Account>();
            Map<Id, Account> accMap= New Map<Id, Account>();

            accList=[SELECT Id FROM Account Limit 5];

            for (Account act:accList){
               accMap.put(act.id, act);
            }

            System.debug(accMap);                    //Displays key as well as values of Map

            for (id k : accmap.keySet()){
               System.debug(accmap.get(k));          //Displays only values of Map
            }
        }
}
```

Version 2:

```
public class ApexDemo02 {

        public static void listToMapUsingLoop( ){
            List<Account> accList= New List<Account>();
            Map<Id, Account> accMap= New Map<Id, Account>();

            for (Account act:[SELECT Id FROM Account Limit 5]){
               accMap.put(act.id, act);
            }

            System.debug(accMap);                    //Displays key as well as values of Map

            for (id k : accmap.keySet()){
               System.debug(accmap.get(k));          //Displays only values of Map
            }
        }
}
```

3. Query 5 Account records into a list. Now create a map where key will be account Id and value will Account sObject (without using loop) and display values of map.

Version 1:

```
public class ApexDemo03 {

    public static void listToMapWithoutUsingLoop( ){
        List<Account> accList= New List<Account>();
        accList=[SELECT Id FROM Account Limit 5];

        Map<Id, Account> accMap= New Map<Id, Account>(accList);

        System.debug(accMap);                       //Displays key as well as values of Map

        for (id k : accmap.keySet()){
            System.debug(accMap.get(k));            //Displays only values of Map
        }
    }
}
```

Version 2:

```
public class ApexDemo03 {

    public static void listToMapWithoutUsingLoop( ){

        Map<Id, Account> accMap= New Map<Id, Account>([SELECT Id, FROM Account Limit 5]);

        System.debug(accMap);                       //Displays key as well as values of Map

        for (id k : accmap.keySet()){
            System.debug(accMap.get(k));            //Displays only values of Map
        }
    }
}
```

4. Query 5 Account records into a list. Now create a set of Account ID and Map <Id, Account> using a list. Now iterate on set and check whether set values match with map key, if yes then display map value.

Version 1:

```
public class ApexDemo04 {

    public static void listToSetAndMap( ){
        List<Account> accList= New List<Account>();
        Set<id> idSet= New Set<id>();

        accList=[SELECT Id FROM Account Limit 5];

        Map<id, Account> accMap= New Map<id, Account>(accList);          //List to Map

        for(Account acc : acclist){
            idSet.add(acc.id);                          //List to Set
        }

        for (Id i: idSet){
            if(idset.contains(i)){
                System.debug(accMap.get(i));
            }
        }
    }
}
```

Version 2:

```
public class ApexDemo04 {

    public static void listToSetAndMap( ){
        List<Account> accList= New List<Account>();
        Set<id> idSet= New Set<id>();

        accList=[SELECT Id FROM Account Limit 5];

        Map<id, Account> accMap= New Map<id, Account>( );

        for(Account acc : acclist){
            idSet.add(acc.id);                          //List to Set
            accMap.put(acc.id, acc);                    //List to Map
        }

        for (Id i: idSet){
```

```
            if(idset.contains(i)){
                System.debug(accMap.get(i));
            }
        }
    }
}
```

5. Query This/Last Week created Accounts with related contacts.

```
public class ApexDemo05 {

    public static void accountsWithContacts( ){
        List<Account> accList= new List<Account>( );

        accList=[SELECT Id, Name, (SELECT Id, Firstname FROM Contacts) FROM Account
WHERE CreatedDate = LAST_WEEK];

        for (account acc : accList){
            System.debug('Account Name: ' + acc.Name);
            System.debug('Total related contacts : ' + acc.Contacts.size( ));
                    for(Contact con : acc.Contacts){
                            System.debug('Contact First Name : ' + con.FirstName);
                    }
            }
    }
}
```

6. Query This/Last Week created contacts with related account Name and Phone.

```
public class ApexDemo06 {

    public Static Void contactsWithRelatedAcc(){
        List<Contact> conList =New List<Contact>();
        conList =[SELECT Id, Name, Account.Name, Account.Phone FROM Contact WHERE
CreatedDate = Last_week] ;
        for(Contact con: conList){
            System.debug('Contact Name= '+con.name
                    + ',   Account Name = ' +con.Account.name
                    +',  Phone Number ='  +con.Account.phone);
        }
    }
}
```

7. Query Account (Id, Name, Phone) with Related Opportunities (Name, CloseDate, Stage) where Account Phone != null. Sort the records in both Asc or Desc Order based on CreatedDate.

Version 1: Ascending Order

```
public class ApexDemo07 {

    public static void accountWithRelatedOpp(){
        List<Account> accList= new List<Account>();
        accList=[SELECT Id, Name, Phone,
                 (SELECT Name, CloseDate, StageName FROM Opportunities)
                 FROM Account WHERE Phone!=null ORDER BY CreatedDate ASC];

        for(Account acc: accList){
            System.debug('Account Name: '+acc.Name);
            for (Opportunity opp: acc.Opportunities){
                System.debug('Opp Name: '+opp.Name
                             +', Opp Stage: '+opp.StageName);
            }
        }
    }
}
```

Version 2: Descending Order

```
public class ApexDemo07 {

    public static void accountWithRelatedOpp(){
        List<Account> accList= new List<Account>();
        accList=[SELECT Id, Name, Phone,
                 (SELECT Name, CloseDate, StageName FROM Opportunities)
                 FROM Account WHERE Phone!=null ORDER BY CreatedDate DESC];

        for(Account acc: accList){
            System.debug('Account Name: '+acc.Name);
            for (Opportunity opp: acc.Opportunities){
                System.debug('Opp Name: '+opp.Name
                             +', Opp Stage: '+opp.StageName);
            }
        }
    }
}
```

8. Query Account along with related contacts where Account phone!=null. Now display results through system.debug( ) in following ways: Account Name has #ofcontacts related.

```
public class ApexDemo08 {

 public static void accountWithRelatedcontacts(){
       List<Account> accList= new List<Account>();
       accList=[SELECT Id, Name, Phone,
                (SELECT Id, Name From Contacts)
                From Account where Phone!=null];

       for(Account acc: accList){
          System.debug('Account Name= ' +acc.Name
                       +' and No of Contacts= '+acc.Contacts.size());
       }
   }
}
```

9. Query Account along with related opportunities where Account phone!=null. Now display results through system.debug( ) in following ways:
“Account Name has TotalAmountOnRelatedOpp worth Opportunities”.

```
public class ApexDemo09 {

    public static void oppRelatedToAccountAndTotalAmount(){
       List<Account> accList= new List<Account>([SELECT Id, Name, (SELECT Name, Amount
FROM Opportunities) FROM Account WHERE Phone!=Null ]);

       for (Account acc : accList){
          Decimal TotalAmount = 0;
          for(Opportunity Opp: acc.Opportunities){
             if(opp.Amount!=Null){
                TotalAmount+=opp.Amount;
             }
          }
          System.debug('Account Name:' + acc.name
                       +', Number of Opportunities ' + acc.Opportunities.size()
                       +',  Total Amount:'+TotalAmount);
       }
   }
}
```

10. Create 2 Account records with two separate insert statements. Then create 2 Account records with a single insert statement using List.

Version 1: Accounts with separate insert statements: [Not Suggested as number of DML statements will be increased and governor limit will hit.]

```
public class ApexDemo10 {

  public static void createAccts(){

        Account acc1= new Account();
        acc1.Name= 'Sanjay Gupta Tech School';
        acc1.Phone='190014';
        insert acc1;

        Account acc2= new Account();
        acc2.Name= 'Binge Code School';
        acc2.Phone='190014';
        insert acc2;
      }
}
```

Version 2: Accounts with single insert statements: [Use this as part of Best Practice.]

```
public class ApexDemo10 {

  public static void createAccts(){
      List<Account> accList = New List<Account>();
        Account acc1= new Account();
        acc1.Name= 'Sanjay Gupta Tech School';
        acc1.Phone='190014';
        accList.add(acc1);

        Account acc2= new Account();
        acc2.Name= 'Binge Code School';
        acc2.Phone='190014';
        accList.add(acc2);

        if(!accList.isEmpty()){
            insert accList;
        }
    }
}
```

11. Insert 5 Accounts with different account Names and Phone =’123456’. [Hint: Loop]

```
public class ApexDemo11 {

        public static void accountRecordsWithInsert(){
                List<Account> accList = new List<Account>();
                for (Integer i=1; i<=5; i++){
                        Account acc= new Account();
                        acc.name= 'Sanjay Gupta Tech Schooll' + i;
                        acc.Phone='123456';
                        accList.add(acc);
                 }
                if(accList.size()>0){
                        insert accList;
                }
          }
}
```

12. Insert 5 Opportunities with different Opportunity Names, CloseDate = Today’s Date and Stage = ‘ Prospecting. [Hint: Loop]

```
public class ApexDemo12 {

    public static void oppInsertWithLoop1(){
        List<Opportunity> oppList= New  List<Opportunity>();
        for (integer i=1; i<=5; i++){
            Opportunity opp= new Opportunity();
            opp.Name = ‘Sanjay Gupta Tech School’ +i;
            opp.CloseDate = System.today();
            opp.StageName= 'Prospecting';
            oppList.add(opp);
        }
        if(oppList.size()>0){
            Insert oppList;
        }
    }
```

13. Insert an Account along with 1 related Contact.

```
public class ApexDemo13 {

 public static void insertRelatedContacts(){
        Account acc= New Account();
        acc.Name = 'Sanjay Gupta Tech School';
        acc.Phone = '123456';
        Insert acc;

        if(acc.Id! = null){
            Contact Con = New Contact();
            con.Firstname = ‘Sanjay’;
            con.LastName= ‘Gupta’;
            con.AccountId=acc.Id;
            Insert con;
        }
}
```

14. Insert an Account along with 1 related Contact and 1 related Opportunity.

```
public class ApexDemo14 {
    public static void insertRelatedConAndOpp(){
        Account acc= New Account();
        acc.Name = 'Sanjay Gupta Tech School';
        acc.Phone = '123456';
        Insert acc;
        if(acc.Id! = null){
            Contact Con = New Contact();
            con.FirstName = ‘Sanjay’;
            con.LastName= ‘Gupta’;
            con.AccountId=acc.Id;
            Insert con;
        }
        if(acc.Id!=null){
            Opportunity opp= New Opportunity();
            opp.name = ‘Sanjay Gupta Tech School’;
            opp.closeDate =System.today( ) + 7;
            opp.StageName='Prospecting';
            opp.AccountId=acc.Id;
            Insert opp;
        }
    }
}
```

15. Insert 5 Accounts with at least 1 related Opportunity.

```
public class ApexDemo15 {
  public static void insertAccountWithOpps(){
        List<Account> accList = New List <Account>();
        for (integer i=1; i<=5; i++){
            Account acc= New Account();
            acc.Name='Sanjay Gupta Tech School '+i;
            acc.Phone ='123456';
            accList.add(acc);
        }
        if(!accList.isEmpty()){
            Insert accList;
        }

        List<Opportunity> oppList= New  List<Opportunity>();
        for(Account acc: accList){
            Opportunity opp= new Opportunity();
            opp.Name = acc.Name;
            opp.CloseDate = System.today( );
            opp.StageName= 'Prospecting';
            opp.AccountId = Opp.Id;
            oppList.add(opp);
        }
        if(!oppList.isEmpty()){
            Insert oppList;
        }
    }
}
```

16. Insert 5 Accounts and a Contact & a Opportunity related to each account.

```
public class ApexDemo16 {
    public static void insertAccountWithConAndOpp( ){
        List<Account> accList = New List <Account>();
        for (integer i=1; i<=5; i++){
            Account acc= New Account();
            acc.Name='Sanjay Gupta Tech School '+i;
            acc.Phone ='123456';
            accList.add(acc);
        }
        if(!accList.isEmpty()){
            Insert accList;
        }
```

```
        List<Opportunity> oppList= New  List<Opportunity>();
        for(Account acc: accList){
            Opportunity opp= new Opportunity();
            opp.Name = acc.Name;
            opp.CloseDate = System.today();
            opp.StageName= 'Prospecting';
            opp.AccountId = Opp.Id;
            oppList.add(opp);
        }
        if(!oppList.isEmpty()){
            Insert oppList;
        }

        List<Contact> conList= New  List<Contact>();
        for(Account acc: accList){
            Contact con= new Contact();
            con.Firstname = ‘Sanjay’;
            con.LastName=’Gupta’;
            con.AccountId=acc.Id;
            conList.add(con);
        }
        if(!conList.isEmpty()){
            Insert conList;
        }
    }
}
```

17 Query Accounts where Phone =’123456’ and update Phone to ’654321’.

```
public class ApexDemo17 {
    public static void updateAccountPhoneNo(){
        List<Account> accList = New List <Account>();
        accList=[SELECT ID, Name, Phone FROM Account WHERE Phone = '123456'];
        if(!accList.isEmpty()){
            for(Account acc: accList){
                acc.phone='654321';
            }
        }
        if(!accList.isEmpty()){
            update accList;
        }
    }
}
```

18 Query Contacts where Email is null and update their email with 'sanjaygupta.techschool@gmail.com'.

```
public class ApexDemo18 {

    public static void updateContactEmail(){
        List<Contact> conList = New List<Contact>();
        conList=[SELECT Id, Name, Phone FROM Contact WHERE Email = Null];
        if(!conList.IsEmpty()){
            for(Contact con: conList){
                con.Email='sanjaygupta.techschool@gmail.com';
            }
        }
        if(!conList.IsEmpty()){
            update conList;
        }
    }
}
```

19 Query Accounts where Phone =654321 and update Phone to 123456. Also create 5 new account records. [Hint : Use Upsert]

```
public class ApexDemo19 {

    public static void updateAccountPhNoAndNewAcs(){
        List<Account> accList = New List <Account>();
        accList=[SELECT ID, Name, Phone FROM Account WHERE Phone = '123456'];
        if(!accList.isEmpty()){
            for(Account acc: accList){
                acc.phone='654321';
            }
        }
        for (integer i=1; i<=5; i++){
            Account acc= New Account();
            acc.name='Sanjay Gupta Tech School'+i;
            acc.Phone ='1234567890';
            accList.add(acc);
        }
        if(!accList.isEmpty()){
            upsert accList;
        }
    }
}
```

20 Query Opportunities where created date is last month and updated their stage to closed won. Also create 5 new opportunities. [Hint: Use Upsert]

```
public class ApexDemo20 {

    public static void oppCreatedLastMonth(){
        List<Opportunity> oppList= New  List<Opportunity>();
        oppList=[SELECT Id, Name, StageName FROM Opportunity WHERE CreatedDate =
Last_Month];
       if(!oppList.IsEmpty()) {
           for(Opportunity opp: oppList){
                    opp.StageName='Closed Won';
           }
       }

       for (integer i=1; i<=5; i++){
           Opportunity opp = new Opportunity();
           opp.Name = 'Sanjay Gupta Tech School ' +i;
           opp.CloseDate = System.today()+7;
           opp.StageName= 'Prospecting';
           oppList.add(opp);
       }

        if(!oppList.IsEmpty()) {
           Upsert oppList;
        }
    }
}
```

21 Delete account records where Phone =’123456’. Verify whether records are deleted or not through Recycle Bin.

```
public class ApexDemo21 {

    public static void deleteAccountRecords(){
        List<Account> accList = New List <Account>();
        accList=[SELECT ID, Name, Phone FROM Account WHERE Phone = '123456'];

        if(!accList.IsEmpty()) {
                    delete accList;
        }
    }
}
```

22 Undelete account records deleted in previous point. Check recycle bin is not having those records once undeleted and they will be under the accounts tab in the app.

```
public class ApexDemo22 {

    public static void undeleteAccountRecords(){
        List<Account> accList = New List <Account>();
        accList=[SELECT ID, NameFROM Account WHERE Phone = '123456' All ROWS ];

            if(!accList.IsEmpty()) {
                    undelete accList;
            }
    }
}
```

23 Query Accounts created last week. Now Insert one opportunity under each account.

```
public class ApexDemo23 {

    public static void oppForAccountCreatedLastWeek(){
        List<Account> accList = New List <Account>();
        List<Opportunity> oppList = New  List<Opportunity>();

        accList = [SELECT Id, Name, Phone FROM Account Where CreatedDate = Last_Week ];

        if(accList.Size()>0){
            for(Account acc : accList){
                Opportunity opp= New Opportunity();
                opp.Name = acc.Name;
                opp.Closedate = System.today()+7;
                opp.StageName = 'Prospecting';
                opp.AccountId = acc.Id;
                oppList.add(opp);
            }
        }
        if(oppList.size()>0){
            insert oppList;
        }
    }
}
```

24 Query Accounts created in the last 7 days. Now update their phone number with '12345678'.

```
public class ApexDemo24 {

    public static void updatePhoneForAccountCreatedLastWeek(){
        List<Account> accList = New List <Account>();

        accList=[SELECT Id, Phone FROM Account WHERE CreatedDate=Last_Week ];
        if(!accList.isEmpty()){
            for (Account acc : accList){
                acc.Phone='1234567890';
            }
        }
        if(!accList.isEmpty()){
            update accList;
        }
    }
}
```

25 Query Opportunities created in the last 7 days. If the opportunity stage is closed won then update description as 'Opportunity is Closed Won', in case of closed lost update description as 'Opportunity is Closed Lost' otherwise 'Opportunity is Open'.

```
public class ApexDemo25 {

    public static void updateOppDescForLastWeek(){
        List<Opportunity> oppList = New  List<Opportunity>();

        oppList=[SELECT Id,StageName FROM Opportunity WHERE CreatedDate = Last_Week ];

        If(!oppList.isEmpty()){
            for(Opportunity opp : oppList){
                if(opp.StageName =='Closed Won'){
                    opp.Description = 'Opportunity is Closed Won';
                }
                else if(opp.StageName =='Closed Lost'){
                    opp.Description = 'Opportunity is Closed Lost';
                }else{
                        opp.Description = 'Opportunity is Open';
                }
            }
        }
```

```
        if(!oppList.isEmpty()){
            update oppList;
        }
    }
}
```

26. Query Accounts with related contacts created in the last one month. Populated 'Total Contacts' Field on Account by counting number of contacts associated with Account. If no contact then populate 0.

```
public class ApexDemo26 {

    public static void populateTotalContactsOnAccount(){
        List<Account> accList = New List <Account>();
        accList=[SELECT ID, Name, Total_Contacts__c,
                    (SELECT Id, FirstName, LastName FROM Contacts)
                    FROM Account WHERE CreatedDate=Last_Week ];

        if(accList.size()!=null){
            for (Account acc : accList){
                acc.Total_Contacts__c=acc.Contacts.Size();
            }
        }
        if(!accList.isEmpty()){
            update accList;
        }
    }
}
```

27 Query Contacts created last week and if Mobile phone is not populated on contact then copy Phone of Related Account. [Child-Parent SOQL]

```
public class ApexDemo27 {

    public static void updatePhoneOnConFromAccount(){
        List<Contact> conList = New List<Contact>();

        conList=[SELECT Id, Phone, Account.Phone FROM Contact WHERE AccountId!=Null];
        if(!conList.IsEmpty()){
            for(Contact con : conList){
                if(con.Phone==null){
                    con.phone=con.Account.Phone;
                }
```

```
        }
      }
      if(!conList.isEmpty()){
        update conList;
      }
    }
}
```

28 Create a Custom Object named “Employee”. Create Name, Salary, Phone, Lookup (Account) fields on Employee Object. Create a Total Salary field on Account. Now Query Accounts created this year along with related Employees. Populate total salary of all related employees on account. [Parent-Child SOQL]

```
public class ApexDemo28 {

    public Static Void empTotalSalaryOnAccount(){
      List<Account> accList = New List<Account>();

      accList=[SELECT Id, Name, Total_Salary__c,
                (SELECT Name, Total_Salary__c FROM Employees__r )
                FROM Account WHERE CreatedDate = Last_Year];

      if(!accList.isEmpty()){
        for(Account acc : accList) {
          Decimal TotalSalary = 0;
          for(Employee__c emp : acc.employees__r){
            TotalSalary = TotalSalary + emp.Total_Salary__c;
          }
          acc.Total_Salary__c = TotalSalary;
        }
      }
      if(accList.Size() > 0){
        update accList;
      }
    }
}
```

29 Query Employees created this year and if Phone is not populated on Employee record then copy Phone of Related Account. [Child-Parent SOQL]

```
public class ApexDemo29 {

    public static void populateAccountPhoneOnEmp(){
        List<Employee__c> empList = New List<Employee__c>();

        empList = [SELECT Name, Phone__c, Employee_Account__r.Phone
                    FROM Employee__c WHERE CreatedDate = Last_Year];
        if(!empList.isEmpty()){
            for (Employee__c emp : empList){
                if(emp.Phone__c == NULL){
                    emp.Phone__c = emp.Employee_Account__r.Phone;
                }
            }
            if(!empList.isEmpty()){
                update empList;
            }
        }
    }
}
```

30 Create a lookup to Employee on Contact object. Query Contacts created last week and if Mobile phone is not populated on contact then copy Phone of Related Employee.

```
public class ApexDemo30 {

    public static void copyEmpPhonetoContact(){
        List<Contact> conList = New List<Contact>();
        conList=[SELECT Id, Phone, Employee__r.Phone__c FROM Contact WHERE
CreatedDate=This_Year];
        if(!conList.isEmpty()){
            for(Contact con : conList){
                if(con.Phone==null){
                      con.Phone=con.Employee__r.Phone__c;
                }
            }
        }
        if(conList!=null){
            update conList;
        }
    }
}
```

31 Insert 200 Account records and make sure the Governor limit should not hit. Then delete those records as well.

```
public class ApexDemo31 {

  public static void bulkAccCreation(){
        List<Account> accList = New List <Account>();
        for (integer i=1; i<=200; i++){
            Account acc= New Account();
            acc.Name='Sanjay Gupta Tech School '+i;
            acc.Phone ='123456';
            accList.add(acc);
        }
        if(!accList.isEmpty()){
            Insert accList;
        }

        accList = [SELECT Id FROM Account WHERE CreatedDate = System.Today( )];
        if(!accList.isEmpty()){
            delete accList;
        }
    }
}
```

32 Delete opportunities whose stage is closed lost and created last month.

```
public class ApexDemo32 {

  public static void deleteOpps(){
          List<Opportunity> oppList = new List<Opportunity>( );

          oppList = [SELECT Id FROM Opportunity
                              WHERE StageName = 'Closed Lost'
                              && CreatedDate = LAST_MONTH];

          if(!oppList.isEmpty( )){
            delete oppLIst;
          }
     }
}
```