```
libname temp '/<FILE PATH FOR WORKING DIRECTORY>/';
libname myfolder '/<FILE PATH FOR FINAL DATASET>/';

options errors= 2;
options nocenter ps=52 errors=1 compress=yes obs=max;

** directory with performance and acquisition .txt files **;
%let directory = /<INSERT FILE(S) LOCATION HERE>/;

*******************************************************************;
** file headers - variables beginning in 'x' will be reformatted **;
*******************************************************************;
%let acq_head =
loan_id :$12.
orig_chn :$1.
seller :$80.
orig_rt
orig_amt
orig_trm
x_orig_date :$7.
x_first_pay :$7.
oltv
ocltv
num_bo
dti
cscore_b
fthb_flg :$1.
purpose :$1.
prop_typ :$2.
num_unit
occ_stat :$1.
state :$2.
zip_3 :$5.
mi_pct
x_prod_type :$3.
cscore_c
;

%let perf_head =
loan_id :$12.
x_period :$10.
y_servicer :$80.
y_curr_rte
y_act_upb
x_loan_age
x_rem_mths
x_adj_rem_months
x_maturity_date :$7.
msa :$5.
x_dlq_status :$3.
y_mod_ind :$1.
z_zb_code :$2.
x_zb_date :$7.
```

```
x_lpi_dte :$10.
x_fcc_dte :$10.
x_disp_dte :$10.
fcc_cost
pp_cost
ar_cost
ie_cost
tax_cost
ns_procs
ce_procs
rmw_procs
o_procs
;

%macro lppub (quarter);
****************************************;
** upload and format acquisition files **;
****************************************;

data acq (drop = x_orig_date x_first_pay x_prod_type);
  infile "&directory.Acquisition_&quarter..txt" delimiter = '|' missover dsd lrecl=32767;
  input &acq_head;

   *converting date formats from mm/yyyy to sas date;
   format orig_dte frst_pay mmddyy8.;
   orig_dte = mdy(input(substr(x_orig_date,1,2),2.),1,input(substr(x_orig_date,4,4),4.));
   frst_pay = mdy(input(substr(x_first_pay,1,2),2.),1,input(substr(x_first_pay,4,4),4.));

run;

proc sort data = acq; by loan_id; run;

*************************************;
** upload and format activity files **;
*************************************;

data temp (drop = x_zb_date x_period x_maturity_date x_adj_rem_months x_rem_mths x_loan_age
                  x_dlq_status x_lpi_dte x_fcc_dte x_disp_dte);
  infile "&directory.Performance_&quarter..txt" delimiter = '|' missover dsd lrecl=32767;
  input &perf_head;

  *converting dates from mm/yyyy to sas date;
  format y_act_date z_zb_date lpi_dte fcc_dte disp_dte mmddyy8.;
  z_zb_date = mdy(input(substr(x_zb_date,1,2),2.),1,input(substr(x_zb_date,4,4),4.));

  *converting dates from mm/dd/yyyy to sas date;
  y_act_date = mdy(input(substr(x_period,1,2),2.),1,input(substr(x_period,7,4),4.));
  lpi_dte = mdy(input(substr(x_lpi_dte,1,2),2.),1,input(substr(x_lpi_dte,7,4),4.));
  fcc_dte = mdy(input(substr(x_fcc_dte,1,2),2.),1,input(substr(x_fcc_dte,7,4),4.));
  disp_dte = mdy(input(substr(x_disp_dte,1,2),2.),1,input(substr(x_disp_dte,7,4),4.));

  *to convert delinquency status from character to number, we set 'X' values to '999';
  if x_dlq_status = 'X' then y_dlq_stat = 999;
  else y_dlq_stat = x_dlq_status*1;
```

```
run;

*sorting loans by activity date to keep in chronological order;
proc sort data = temp; by loan_id y_act_date; run;

*****************************************************************************************************
**********************;
**---------------------- retaining elements from activity files required for static data set ----------------------**;
*****************************************************************************************************
**********************;

data act (drop = y_mod_ind y_servicer y_num_periods
        rename =( y_act_upb = last_upb
                  y_dlq_stat = z_last_status
                  y_curr_rte = last_rt
                  y_act_date = last_activity_date ));
  set temp;
  by loan_id y_act_date;

  length servicer $80;
  retain servicer;
  retain y_num_periods f180_dte fce_dte f180_upb fce_upb mod_flag fmod_dte z_num_periods_180
z_num_periods_ce z_num_periods_lqd;
  format f180_dte fce_dte fmod_dte mmddyy8.;

  if first.loan_id then do;

     servicer = y_servicer;

     y_num_periods = 1;
     y_prev_upb = .;

    if 999 > y_dlq_stat >= 6 then f180_dte = y_act_date; else f180_dte = .;
    if 999 > y_dlq_stat >= 6 then f180_upb = y_act_upb;  else f180_upb= .;
    if 999 > y_dlq_stat >= 6 then z_num_periods_180 = y_num_periods; else z_num_periods_180 = .;

    if 999 > y_dlq_stat >= 6 and z_zb_code in ('03','09') then fce_dte = y_act_date; else fce_dte = .;
    if 999 > y_dlq_stat >= 6 and z_zb_code in ('03','09') then fce_upb = y_act_upb;  else fce_upb= .;
    if 999 > y_dlq_stat >= 6 and z_zb_code in ('03','09') then z_num_periods_ce = y_num_periods; else
z_num_periods_ce = .;

    if y_mod_ind = 'Y' then mod_flag = 1; else mod_flag = 0;
    if y_mod_ind = 'Y' then fmod_dte = y_act_date; else fmod_dte = .;

    *helps in capturing last upb of quick to liquidate loans*;
    if z_zb_code in ('01','03','06','09') then z_num_periods_lqd = y_num_periods; else z_num_periods_lqd = .;

  end;
  else do;

     *servicer field will capture the current servicer*;
     if y_servicer ne '' then servicer = y_servicer;
```

```
      y_num_periods = y_num_periods + 1;

      y_prev_upb = lag(y_act_upb);

      *capturing the last upb for zero balance loans;
      if y_act_upb <= 0 and z_zb_code in ('01','03','06','09') then y_act_upb = y_prev_upb;
      if z_zb_code in ('01','03','06','09') then z_num_periods_lqd = y_num_periods;

      *performance flags*;
      if 999 > y_dlq_stat >= 6 and f180_dte = . then do;
        f180_dte = y_act_date;
        f180_upb = y_act_upb;
        z_num_periods_180 = y_num_periods;
      end;
      if (999 > y_dlq_stat >= 6 or z_zb_code in ('03','09')) and fce_dte = . then do;
        fce_dte = y_act_date;
        fce_upb = y_act_upb;
        z_num_periods_ce = y_num_periods;
      end;

      if y_mod_ind = 'Y' and mod_flag = 0 then do;
        mod_flag = 1;
        fmod_dte = y_act_date;
      end;

   end;

   if last.loan_id;

run;



   ***********************************;
   ** merge to create combined dataset **;
   ***********************************;
data temp.comb_&quarter (drop = z_zb_code z_zb_date z_last_status z_num_periods_180 z_num_periods_ce
z_num_periods_lqd);
  merge acq   (in=a)
        act   (in=b);
  by loan_id;
  if a;

  ** correcting the null ce values on early dlq loans upb values **;
  if 0 < z_num_periods_180 <= 8 then f180_upb = orig_amt;
  if 0 < z_num_periods_ce  <= 8 then fce_upb  = orig_amt;
  if 0 < z_num_periods_lqd <= 8 and  last_upb <= 0 then last_upb = orig_amt;

  ** minimum credit score **;
  cscore_mn = min(cscore_b,cscore_c);

  **setting mipct equal to 0 when missing like freddie**;
  mi_pct = max(mi_pct,0);
  ** origination home value **;
  orig_val = orig_amt/(oltv/100);

  format last_dte date9.;
  if disp_dte ne . then last_dte = disp_dte;
  else if z_zb_date ne . then last_dte = z_zb_date;
  else last_dte = last_activity_date;

  *last status;
  length last_stat $1;
  if z_zb_code = '09'            then last_stat = "F";
  else if z_zb_code = '03'       then last_stat = "S";
  else if z_zb_code = '06'       then last_stat = "R";
  else if z_zb_code = '01'       then last_stat = "P";
  else if 999 > z_last_status > 9 then last_stat = "9";
  else if 9 >=  z_last_status > 0 then last_stat = put(z_last_status,1.);
  else if z_last_status = 0      then last_stat = "C";
  else                                last_stat = 'X';

  if last_stat in ("F","S") and disp_dte ^=. then do;

    int_cost = intck("month",lpi_dte,last_dte)*(((last_rt/100)-0.0035)/12)*last_upb;
    net_loss = sum(last_upb,fcc_cost,pp_cost,ar_cost,ie_cost,tax_cost,int_cost,-1*ns_procs,-1*ce_procs,-1*rmw_procs,-
1*o_procs);
    net_sev = (net_loss/last_upb);

    if int_cost=. then int_cost=0;
    if net_loss=. then net_loss=0;
    if fcc_cost=. then fcc_cost=0;
    if pp_cost=. then pp_cost=0;
    if ar_cost=. then ar_cost=0;
    if ie_cost=. then ie_cost=0;
    if tax_cost=. then tax_cost=0;
    if ns_procs=. then ns_procs=0;
    if ce_procs=. then ce_procs=0;
    if rmw_procs=. then rmw_procs=0;
    if o_procs=. then o_procs=0;

   tot_exp=sum(fcc_cost,pp_cost,ar_cost,ie_cost,tax_cost);
   tot_cost=sum(int_cost,tot_exp,last_upb);
   tot_proc=sum(ns_procs,ce_procs,rmw_procs,o_procs);

  end;

run;

%mend;

%LPPUB(2000Q1);
%LPPUB(2000Q2);
%LPPUB(2000Q3);
%LPPUB(2000Q4);

%LPPUB(2001Q1);
```

```
%LPPUB(2001Q2);
%LPPUB(2001Q3);
%LPPUB(2001Q4);

%LPPUB(2002Q1);
%LPPUB(2002Q2);
%LPPUB(2002Q3);
%LPPUB(2002Q4);

%LPPUB(2003Q1);
%LPPUB(2003Q2);
%LPPUB(2003Q3);
%LPPUB(2003Q4);

%LPPUB(2004Q1);
%LPPUB(2004Q2);
%LPPUB(2004Q3);
%LPPUB(2004Q4);

%LPPUB(2005Q1);
%LPPUB(2005Q2);
%LPPUB(2005Q3);
%LPPUB(2005Q4);

%LPPUB(2006Q1);
%LPPUB(2006Q2);
%LPPUB(2006Q3);
%LPPUB(2006Q4);

%LPPUB(2007Q1);
%LPPUB(2007Q2);
%LPPUB(2007Q3);
%LPPUB(2007Q4);

%LPPUB(2008Q1);
%LPPUB(2008Q2);
%LPPUB(2008Q3);
%LPPUB(2008Q4);

%LPPUB(2009Q1);
%LPPUB(2009Q2);
%LPPUB(2009Q3);
%LPPUB(2009Q4);

%LPPUB(2010Q1);
%LPPUB(2010Q2);
%LPPUB(2010Q3);
%LPPUB(2010Q4);

%LPPUB(2011Q1);
%LPPUB(2011Q2);
%LPPUB(2011Q3);
%LPPUB(2011Q4);
```

```
%LPPUB(2012Q1);
%LPPUB(2012Q2);
%LPPUB(2012Q3);
%LPPUB(2012Q4);

%LPPUB(2013Q1);
%LPPUB(2013Q2);
%LPPUB(2013Q3);
%LPPUB(2013Q4);

%LPPUB(2014Q1);
%LPPUB(2014Q2);

data temp.combined_data;
   set temp.comb_2000q1 temp.comb_2000q2 temp.comb_2000q3 temp.comb_2000q4
       temp.comb_2001q1 temp.comb_2001q2 temp.comb_2001q3 temp.comb_2001q4
       temp.comb_2002q1 temp.comb_2002q2 temp.comb_2002q3 temp.comb_2002q4
       temp.comb_2003q1 temp.comb_2003q2 temp.comb_2003q3 temp.comb_2003q4
       temp.comb_2004q1 temp.comb_2004q2 temp.comb_2004q3 temp.comb_2004q4
       temp.comb_2005q1 temp.comb_2005q2 temp.comb_2005q3 temp.comb_2005q4
       temp.comb_2006q1 temp.comb_2006q2 temp.comb_2006q3 temp.comb_2006q4
       temp.comb_2007q1 temp.comb_2007q2 temp.comb_2007q3 temp.comb_2007q4
       temp.comb_2008q1 temp.comb_2008q2 temp.comb_2008q3 temp.comb_2008q4
       temp.comb_2009q1 temp.comb_2009q2 temp.comb_2009q3 temp.comb_2009q4
       temp.comb_2010q1 temp.comb_2010q2 temp.comb_2010q3 temp.comb_2010q4
       temp.comb_2011q1 temp.comb_2011q2 temp.comb_2011q3 temp.comb_2011q4
       temp.comb_2012q1 temp.comb_2012q2 temp.comb_2012q3 temp.comb_2012q4
       temp.comb_2013q1 temp.comb_2013q2 temp.comb_2013q3 temp.comb_2013q4
       temp.comb_2014q1 temp.comb_2014q2;
run;

  proc sort data = temp.combined_data; by loan_id; run;

***********************************;
********start of analysis********;
***********************************;

** create dataset for summary tables **;

data myfolder.combined_data (drop = comb);
        set temp.combined_data (rename = (ocltv = comb));

** acquisition table flags **;
  *setting null values for combined ltv to original ltv;
if comb = . then ocltv = oltv; else ocltv = comb;
  *building flags for second lien, non owner occupied, refinanced, and mortgage insured properties;
if ocltv > oltv   then seclien = 1; else seclien = 0;
if occ_stat in ("I","S") then nonown  = 1; else nonown  = 0;
if purpose in ("C","R","U")  then refis = 1; else refis = 0;
if mi_pct > 0     then mi      = 1; else mi        = 0;

** terminal counts **;
if last_stat in ("C","1","2","3","4","5","6","7","8","9") then active_cnt = 1; else active_cnt = 0;
if last_stat in ("C","1","2","3","4","5","6","7","8","9") then active_upb = last_upb; else active_upb = 0;
```

```
  active_upb_mil = active_upb/1000000;
if last_stat = "P" then prepaid_cnt = 1; else prepaid_cnt = 0;
if last_stat = "F" then reo_cnt = 1; else reo_cnt = 0;
if last_stat = "S" then alt_cnt = 1; else alt_cnt = 0;
if last_stat = "R" then repurch_cnt = 1; else repurch_cnt = 0;

** reo completion flag **;
  *building a flag to designate loans we consider part of our defaulted loan population;
if last_stat in ("F","S") and disp_dte ne . then compl_flg = 1; else complt_flg= 0;
  *default upb is the final reported upb for loans in our defaulted loan population;
if complt_flg = 1 then default_upb = last_upb; else default_upb = 0;

** performance rates **;
  *calculating the portion of our originations that defaults, and the portion of originations that we lose due to default;
default_rt = default_upb/orig_amt;
if complt_flg = 1 then nloss_rt = net_loss/orig_amt; else nloss_rt = 0;

** loss components & net severity **;
  *calculating costs and proceeds due to default relative to the upb for those defaulted loans;
if complt_flg = 1 then do;
  int_cost1 = int_cost/default_upb;
  tot_exp1 = tot_exp/default_upb;
  fcc_cost1 = fcc_cost/default_upb;
  pp_cost1 = pp_cost/default_upb;
  ar_cost1 = ar_cost/default_upb;
  ie_cost1 = ie_cost/default_upb;
  tax_cost1 = tax_cost/default_upb;
  tot_cost1 = tot_cost/default_upb;
  ns_procs1 = ns_procs/default_upb;
  ce_procs1 = ce_procs/default_upb;
  rmw_procs1 = rmw_procs/default_upb;
  o_procs1 = o_procs/default_upb;
  tot_proc1 = tot_proc/default_upb;
end;

 ** refinance type/occupancy counts **;
 if occ_stat = 'I' then inv_cnt = 1; else inv_cnt = 0;
 if occ_stat = 'P' then pri_cnt = 1; else pri_cnt = 0;
 if occ_stat = 'S' then sec_cnt = 1; else sec_cnt = 0;
 if purpose = 'C' then co_cnt = 1; else co_cnt = 0;
 if purpose = 'P' then pur_cnt = 1; else pur_cnt = 0;
 if purpose = 'R' then rt_cnt = 1; else rt_cnt = 0;
 if purpose = 'U' then u_cnt = 1; else u_cnt = 0;

 ** credit score buckets **;
 if 0 < cscore_mn < 620 then cscorebkt='[0-620)';
 if 620 <= cscore_mn < 660 then cscorebkt='[620-660)';
 if 660 <= cscore_mn < 700 then cscorebkt='[660-700)';
 if 700 <= cscore_mn < 740 then cscorebkt='[700-740)';
 if 740 <= cscore_mn < 780 then cscorebkt='[740-780)';
 if 780 <= cscore_mn then cscorebkt='[780+)';
   *building a variable to sum loans in each credit bucket;
    count=1;
```

```
run;


* opening the excel document that we will write acquisition, performance, and historical loss statistics to;
   * file will save to the same folder as the sas code;

ods tagsets.excelxp file = "./lppub loss sas summary tables.xls"
                    style=seaside
                    options ( fittopage       = 'yes'
                              pages_fitwidth  = '1'
                              pages_fitheight = '1'
                              autofit_height  = 'yes'
                              );

* building a tab with loan counts by refinance purpose;
ods tagsets.excelxp options(sheet_interval = 'none' sheet_name='vint.refi.counts');
proc tabulate data = myfolder.combined_data missing;
      class orig_dte;
      format orig_dte year.;
      var co_cnt
          pur_cnt
          rt_cnt
          u_cnt;
      tables
         (co_cnt='CASHOUT REFI'
          pur_cnt='PMM'
          rt_cnt='RATE/TERM REFI'
          u_cnt = 'UNKNOWN REFI')*sum=''
          n='sum', (orig_dte='' all);
run;

* building a tab with loan counts by occupancy type;
ods tagsets.excelxp options(sheet_interval = 'none' sheet_name='vint.occ.counts');
proc tabulate data = myfolder.combined_data missing;
      class orig_dte;
      format orig_dte year.;
      var inv_cnt
          pri_cnt
          sec_cnt;
      tables
         (inv_cnt='INVESTOR'
          pri_cnt='PRIMARY RES'
          sec_cnt='SECOND HOME')*sum=''
          n='sum', (orig_dte='' all);
run;

* building a tab with frequencies by last status;
ods tagsets.excelxp options(sheet_interval = 'none' sheet_name='vint.last_stat.counts');
proc freq data = myfolder.combined_data;
 tables last_stat;
run;

*building summary statistics for credit score, oltv, and origination upb;
ods tagsets.excelxp options(sheet_interval = 'none' sheet_name='summary.stats');
```

```
proc means data = myfolder.combined_data
                        min p25 p50 mean p75 max nmiss;
 var cscore_mn oltv orig_amt;
run;

* building a tab with counts by fico bucket and vintage;
ods tagsets.excelxp options(sheet_interval = 'none' sheet_name='vint.fico.counts');
proc tabulate data = myfolder.combined_data missing;
 class orig_dte cscorebkt;
 format orig_dte year.;
 var count;
 table cscorebkt*count=''*sum='', (orig_dte='' all);
run;

* building the acquisition statistics tab of the excel document;
ods tagsets.excelxp options(sheet_interval = 'none' sheet_name='aqsn. stats');
proc tabulate data = myfolder.combined_data missing;
        class orig_dte;
        format orig_dte year.;
        var orig_amt
            orig_amt_bn;
        var cscore_b
                cscore_c
                oltv
                ocltv
                dti
                orig_rt/weight=orig_amt;
         table (orig_dte='' all), n='loan count'
                              orig_amt='total orig. upb'*sum=''
                              orig_amt='avg. orig upb($)'*mean
                              cscore_b*mean = 'borrower credit score'
                              cscore_c*mean = 'co-borrower credit score'
                              oltv*mean = 'ltv ratio'
                              ocltv*mean = 'cltv ratio'
                              dti*mean = 'dti'
                              orig_rt*mean = 'note rate';

run;

 * building the performance statistics tab of the excel document to present loan counts, rates, and dollar amounts of
different performance outcomes;
ods tagsets.excelxp options(sheet_interval = 'none' sheet_name='perf.stat.counts');
proc tabulate data = myfolder.combined_data missing;
        class orig_dte;
        format orig_dte year.;
        var orig_amt
            active_cnt
            active_upb
            prepaid_cnt
            reo_cnt
            alt_cnt
            repurch_cnt
            default_upb
            mod_flag;
        var default_rt
            nloss_rt / weight=orig_amt;

         table (orig_dte='' all), n='loan count'
                                 orig_amt='total orig. upb'*sum=''
                                 active_cnt='loan count (active)'
                                 active_upb='active upb'
                                (prepaid_cnt='prepaid'
                                 repurch_cnt='repurchased'
                                 alt_cnt='alternative disposition'
                                 reo_cnt='reo disposition'
                                 mod_flag='modified')
                                 default_upb='default upb'*sum=''
                                 nloss_rt='net loss rate'*mean=''*f=percent10.5;

run;

 * building the historical loss statistics tab of the excel document to present cost, proceed, and loss amounts by vintage;
ods tagsets.excelxp options(sheet_interval = 'none' sheet_name='historical net loss by vintage');
proc tabulate data = myfolder.combined_data (where = (complt_flg = 1)) missing;
        class orig_dte;
        format orig_dte year.;
        var ar_cost1
            ie_cost1
            pp_cost1
            fcc_cost1
            tax_cost1
            tot_exp1
            tot_cost1
            int_cost1
            tot_proc1
            ns_procs1
            ce_procs1
            rmw_procs1
            o_procs1
            net_sev/ weight=default_upb;
        var default_upb
            net_loss;
        tables
            n='loan count'*f=comma10.
            default_upb='default upb ($m)'*sum=''
           (int_cost1='delinquent interest'
            tot_exp1='total liquidation exp.'
            fcc_cost1='foreclosure'
            pp_cost1='property preservation'
            ar_cost1='asset recovery'
            ie_cost1='misc. holding expenses'
            tax_cost1='associated taxes'
            tot_cost1='total costs'
            ns_procs1='net sales proceeds'
            ce_procs1='credit enhancement'
            rmw_procs1='repurchase/make whole'
            o_procs1='other proceeds'
            tot_proc1='total proceeds'
```

```
          net_sev='severity')*mean=''*f=percent10.5
          net_loss='total net loss ($m)'*sum='',(orig_dte='' all);
run;

proc tabulate data = myfolder.combined_data missing;
       class orig_dte;
       format orig_dte year.;
       var default_rt/ weight=orig_amt;
       tables
          default_rt='default rate'*f=percent10.5*mean='',(orig_dte='' all);
run;

 * building the historical loss statistics tab of the excel document to present cost, proceed, and loss amounts by vintage;
ods tagsets.excelxp options(sheet_interval = 'none' sheet_name='historical net loss by disp dt');
proc tabulate data = myfolder.combined_data (where = (complt_flg = 1)) missing;
       class disp_dte;
       format disp_dte year.;
       var ar_cost1
          ie_cost1
          pp_cost1
          fcc_cost1
          tax_cost1
          tot_exp1
          tot_cost1
          int_cost1
          tot_proc1
          ns_procs1
          ce_procs1
          rmw_procs1
          o_procs1
          net_sev/ weight=default_upb;
       var default_upb
          net_loss;
       tables
          n='loan count'*f=comma10.
          default_upb='default upb'*sum=''
         (int_cost1='delinquent interest'
          tot_exp1='total liquidation exp.'
          fcc_cost1='foreclosure'
          pp_cost1='property preservation'
          ar_cost1='asset recovery'
          ie_cost1='misc. holding expenses'
          tax_cost1='associated taxes'
          tot_cost1='total costs'
          ns_procs1='net sales proceeds'
          ce_procs1='credit enhancement'
          rmw_procs1='repurchase/make whole'
          o_procs1='other proceeds'
          tot_proc1='total proceeds'
          net_sev='severity')*mean=''*f=percent10.5
          net_loss='total net loss'*sum='',(disp_dte='' all);
run;

proc tabulate data = myfolder.combined_data missing;
```

```
        class last_dte;
        format last_dte year.;
        var default_rt/ weight=orig_amt;
        tables
            default_rt='default rate'*mean=''*f=percent10.5,(last_dte='' all);
run;
```