```
/*** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** ***/
/* PROGRAM:  PDP Submission
/* PURPOSE:   Retrieve PDP information
/* PROGRAMMER: John Carroll
/* DATE: 12/15/2020
/* OUTPUT FILES: i 04229500PDP Texas AM University Central Texas Cohort TermYYYY.txt
/*     i 04229500PDP Texas AM University Central Texas Course TermYYYY.txt
/*     i 04229500PDP Texas AM University Central Texas Financial Aid TermYYYY.txt
/*     Output data for TermYYYY.xlsx
/* REFERENCES: PDP-Submission-Guide sept2020.pdf
/*** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** ***/

%Let Term = ('201508', '201601', '201606'); /*Term of report*/
%Let OUTPUT_Term = Summer2016; /*Used in output file.*/
/*Used for Working papers. This should match terms used above in Term. Pulls data from CBMs to compare with PDP data from banner.*/
%Let Report_Terms = ((STU_RPT_YEAR = 2015 and STU_RPT_Sem = 1)or(STU_RPT_YEAR = 2016 and STU_RPT_Sem = 2)or(STU_RPT_YEAR = 2016 and STU_RPT_Sem = 3));
%Let Path = S:\Projects\IRA21_005 National Student Clearinghouse Postsecondary Data Partnership\Output; /*location of output files*/
%let Today = %sysfunc(today());

/*Country Codes used to Identify out of United States countries. For list of codes see PDP-Submission-Guide sept2020*/
%Let Country_Code = ('AX', 'AD', 'AE', 'AF', 'AG', 'AI', 'AL', 'AM', 'AN', 'AO', 'AQ', 'AR', 'AS', 'AT', 'AU', 'AW', 'AZ', 'BA', 'BB', 'BD', 'BE', 'BF',
 'BG', 'BH', 'BI', 'BJ', 'BL', 'BM', 'BN', 'BO', 'BQ', 'BR', 'BS', 'BT', 'BV', 'BW', 'BY', 'BZ', 'CA', 'CC', 'CD', 'CF', 'CG', 'CH',
 'CI', 'CK', 'CL', 'CM', 'CN', 'CO', 'CR', 'CU', 'CV', 'CW', 'CX', 'CY', 'CZ', 'DE', 'DJ', 'DK', 'DM', 'DO', 'DZ', 'EC', 'EE', 'EG',
 'EH', 'ER', 'ES', 'ET', 'FI', 'FJ', 'FK', 'FM', 'FO', 'FR', 'GA', 'GB', 'GD', 'GE', 'GF', 'GG', 'GH', 'GI', 'GL', 'GM', 'GN', 'GP',
 'GQ', 'GR', 'GS', 'GT', 'GU', 'GW', 'GY', 'HK', 'HM', 'HN', 'HR', 'HT', 'HU', 'ID', 'IE', 'IL', 'IM', 'IN', 'IO', 'IQ', 'IR', 'IS',
 'IT', 'JE', 'JM', 'JO', 'JP', 'KE', 'KG', 'KH', 'KI', 'KM', 'KN', 'KP', 'KR', 'KW', 'KY', 'KZ', 'LA', 'LB', 'LC', 'LI', 'LK', 'LR',
 'LS', 'LT', 'LU', 'LV', 'LY', 'MA', 'MC', 'MD', 'ME', 'MF', 'MG', 'MH', 'MK', 'ML', 'MM', 'MN', 'MO', 'MP', 'MQ', 'MR', 'MS', 'MT',
 'MU', 'MV', 'MW', 'MX', 'MY', 'MZ', 'NA', 'NC', 'NE', 'NF', 'NG', 'NI', 'NL', 'NO', 'NP', 'NR', 'NU', 'NZ', 'OM', 'PA', 'PE', 'PF',
 'PG', 'PH', 'PK', 'PL', 'PM', 'PN', 'PR', 'PS', 'PT', 'PW', 'PY', 'QA', 'RE', 'RO', 'RS', 'RU', 'RW', 'SA', 'SB', 'SC', 'SD', 'SE',
 'SG', 'SH', 'SI', 'SJ', 'SK', 'SL', 'SM', 'SN', 'SO', 'SR', 'SS', 'ST', 'SV', 'SX', 'SY', 'SZ', 'TC', 'TD', 'TF', 'TG', 'TH', 'TJ',
 'TK', 'TL', 'TM', 'TN', 'TO', 'TP', 'TR', 'TT', 'TV', 'TW', 'TZ', 'UA', 'UG', 'UM', 'US', 'UY', 'UZ', 'VA', 'VC', 'VE', 'VG', 'VI',
 'VN', 'VU', 'WF', 'WS', 'YE', 'YT', 'YU', 'ZA', 'ZM', 'ZW');

/*State Codes used to Identify United States states. For list of codes see PDP-Submission-Guide sept2020*/
%Let State_Code = ('AA', 'AB', 'AE', 'AK', 'AL', 'AP', 'AR', 'AS', 'AZ', 'BC', 'CA', 'CN', 'CO', 'CT', 'CZ', 'DC', 'DE', 'FC', 'FL', 'FM', 'FO', 'GA',
 'GU', 'HI', 'IA', 'ID', 'IL', 'IN', 'IQ', 'KS', 'KY', 'LA', 'MA', 'MB', 'MD', 'ME', 'MI', 'MN', 'MO', 'MP', 'MS', 'MT', 'MX', 'MX',
 'NB', 'NC', 'ND', 'NE', 'NF', 'NH', 'NJ', 'NL', 'NM', 'NR', 'NS', 'NT', 'NU', 'NV', 'NY', 'OH', 'OK', 'ON', 'OR', 'PA', 'PE', 'PQ',
 'PR', 'PW', 'QC', 'RI', 'SC', 'SD', 'SK', 'TN', 'TT', 'TX', 'UK', 'UT', 'VA', 'VI', 'VT', 'WA', 'WI', 'WV', 'WY', 'YT');

data person;
 input Program $ credential $ credential_provider $;
 datalines;
BSWSWK Y Other
BSNNURS Y Other
MEDMHCH Y Other
BAASBMGT Y Other
BAASBUOC Y Other
BAASBUS Y Other
BAASIT Y Other
BBAACC Y Other
BBAADMS Y Other
BBACIS Y Other
BBAECO Y Other
BBAFIN Y Other
BBAHRMG Y Other
BBAINTB Y Other
BBAMGMT Y Other
```

```
BBAMKTG Y Other
BSACC Y Other
BSADMS Y Other
BSAPSC Y Other
BSBA Y Other
BSCIS Y Other
BSCS Y Other
BSECO Y Other
BSFIN Y Other
BSMGMT Y Other
MBABA Y Other
MSACC Y Other
MSHRMG Y Other
MSISYS Y Other
MSMGLD Y Other
MSMGMT Y Other
MSOPL Y Other
RBAASBMGT Y Other
;

/*Cumulative GPA by term and level*/

Proc sql;
create table tbl_GPA as
 Select c.id,
  a.SHRTGPA_PIDM, /*Selection of fields from table a*/
  a.term_code,
  a.Level,
  sum(b.THOURS) as THOURS, /*Cumulative of Transfer hours from table b, grouping as per the definition defined in outer query group by statement and having clause*/
  sum(b.IHOURS) as IHOURS, /*Cumulative of Institutional hours from table b, grouping as per the definition defined in outer query group by statement and having clause*/
  sum(b.OHOURS) as OHOURS, /*Cumulative of Overall hours from table b, grouping as per the definition defined in outer query group by statement and having clause*/
  sum(b.TPOINTS) as TPOINTS, /*Cumulative of TPOINTS calculated in table b, grouping as per the definition defined in outer query group by statement and having clause*/
  sum(b.IPOINTS) as IPOINTS, /*Cumulative of IPOINTS calculated in table b, grouping as per the definition defined in outer query group by statement and having clause*/
  sum(b.OPOINTS) as OPOINTS, /*Cumulative of OPOINTS calculated in table b, grouping as per the definition defined in outer query group by statement and having clause*/
 (Calculated TPOINTS / CALCULATED THOURS) as TGPA, /*Calculating Transfer GPA using formula*/
  (Calculated IPOINTS / CALCULATED IHOURS) as IGPA, /*Calculating Transfer GPA using formula*/
  (Calculated OPOINTS / CALCULATED OHOURS) as OGPA  /*Calculating Transfer GPA using formula*/

 From (select distinct    /*Begin: Subquery to create table a*/
  SHRTGPA_PIDM,
  SHRTGPA_TERM_CODE as term_code,
  SHRTGPA_LEVL_CODE as Level from saturn.shrtgpa) as a  /*End: Subquery to create table a*/
 left join (select SHRTGPA_PIDM, SHRTGPA_TERM_CODE as term_code, SHRTGPA_LEVL_CODE as Level, /*Begin: Subquery to create table b*/
      SUM(CASE WHEN SHRTGPA_GPA_TYPE_IND = 'T' then SHRTGPA_GPA_HOURS else 0 end) as THOURS, /*Summing Transfer GPA hours as per Group defined in the grouping fields in subquery*/
      SUM(CASE WHEN SHRTGPA_GPA_TYPE_IND = 'I' then SHRTGPA_GPA_HOURS else 0 end) as IHOURS, /*Summing Institutional GPA hours as per Group defined in the grouping fields in
subquery*/
      SUM(CASE WHEN SHRTGPA_GPA_TYPE_IND = 'T' then SHRTGPA_QUALITY_POINTS else 0 end) as TPOINTS, /*Summing Institutional GPA hours as per Group defined in the grouping fields in
subquery*/
      SUM(CASE WHEN SHRTGPA_GPA_TYPE_IND = 'I' then SHRTGPA_QUALITY_POINTS else 0 end) as IPOINTS, /*Summing Transfer GPA hours as per Group defined in the grouping fields in
subquery*/
      SUM(CALCULATED THOURS, CALCULATED IHOURS) as OHOURS,
      SUM(CALCULATED TPOINTS, CALCULATED IPOINTS) as OPOINTS
      from saturn.shrtgpa
      group by SHRTGPA_PIDM, SHRTGPA_TERM_CODE, SHRTGPA_LEVL_CODE)
    as b on a.SHRTGPA_PIDM = b.SHRTGPA_PIDM     /*End: Subquery to create table b*/
     and a.TERM_CODE ge b.TERM_CODE       /* Note: Check if this is equal sign or ge*/
     and a.LEVEL = b.LEVEL
```

```
 left join edw.pidm_to_id as c on a.SHRTGPA_PIDM = c.pidm /*Cross list of pidms and ids*/
 group by c.id, a.SHRTGPA_PIDM, a.TERM_CODE, a.LEVEL
 having abs(Input(a.TERM_CODE, 6.)  /*having clause will subset observations of the group by obeying the specified condition*/
    - input(b.TERM_CODE, 6.)) = min(abs(input(a.TERM_CODE, 6.) - input(b.TERM_CODE, 6.)))  /* Cumulative is performed based on this condition, which is starting term in the
group*/
 order by a.SHRTGPA_PIDM, a.TERM_CODE, a.LEVEL
; quit;

/* Student Financial Aid */
proc sql;
create table M014_2 as
 select  Distinct
   c.ID,
   h.STUDENT_LEVEL as Level_Cde 'Level Code',
   h.STUDENT_LEVEL_DESC as Level 'Level',
   a.RPRATRM_PIDM as PIDM Label 'PIDM',
   a.RPRATRM_PERIOD as Term Label 'Term',
   a.RPRATRM_FUND_CODE as Fund_Code 'Fund Code', /*Added 7-13-2020*/
   case when substr(a.RPRATRM_PERIOD, 5, 6) = '08' then Catx('-', input(substr(a.RPRATRM_PERIOD, 1, 4), 4.), input(substr(a.RPRATRM_PERIOD, 1, 4), 4.)+1)
    else Catx('-', input(substr(a.RPRATRM_PERIOD, 1, 4), 4.)-1, input(substr(a.RPRATRM_PERIOD, 1, 4), 4.)) end as Aid_Year,
    f.RTVFTYP_DESC as Type label 'Type',
   g.RTVFSRC_DESC as Source label 'Source',
   a.RPRATRM_ACCEPT_AMT as Amount label 'Amount',
   case when d.RFRASPC_NA_REQD_IND = "Y" then 'Need-Based'
    else 'Merit-Based' end as Need Label 'Need'
 from faismgr.rpratrm as a
 left join faismgr.rFrbase as b on a.RPRATRM_FUND_CODE = b.RFRBASE_FUND_CODE
 left join FAISMGR.RFRASPC as d on a.RPRATRM_FUND_CODE = d.RFRASPC_FUND_CODE and a.RPRATRM_AIDY_CODE = d.RFRASPC_AIDY_CODE
 left join FAISMGR.RTVFTYP as f on b.RFRBASE_FTYP_CODE = f.RTVFTYP_CODE
 left join FAISMGR.RTVFSRC as g on b.RFRBASE_FSRC_CODE = g.RTVFSRC_CODE
 Left Join edw.PIDM_TO_ID as c on a.RPRATRM_PIDM = c.PIDM
 Left Join (select  Distinct
      ID,
      ACADEMIC_PERIOD,
      STUDENT_LEVEL,
      STUDENT_LEVEL_DESC
    From ODSMGR.ACADEMIC_STUDY
    Where PRIMARY_PROGRAM_IND = "Y") as h on c.Id = h.ID and a.RPRATRM_PERIOD = h.ACADEMIC_PERIOD
 where a.RPRATRM_ACCEPT_AMT > 0;
quit;

Proc SQL;
create table M014_1 as
 select Distinct
   c.ID,
   h.STUDENT_LEVEL as Level_Cde 'Level Code',
   h.STUDENT_LEVEL_DESC as Level 'Level',
   b.TBRACCD_PIDM as PIDM Label 'PIDM',
   b.TBRACCD_TERM_CODE as Term Label 'Term',
   a.TBBDETC_DETAIL_CODE as Fund_Code 'Fund Codes', /*added 7-13-2020*/
   case when substr(b.TBRACCD_TERM_CODE, 5, 6) = '08' then Catx('-', input(substr(b.TBRACCD_TERM_CODE, 1, 4), 4.), input(substr(b.TBRACCD_TERM_CODE, 1, 4), 4.)+1)
     else Catx('-', input(substr(b.TBRACCD_TERM_CODE, 1, 4), 4.)-1, input(substr(b.TBRACCD_TERM_CODE, 1, 4), 4.)) end as Aid_Year,
   case when TBBDETC_DCAT_CODE = 'EXM' or TBBDETC_DETAIL_CODE = '30042' then 'Exemption'
     when TBBDETC_DETAIL_CODE in ('3000', '3002', '3004', '3006', '3008', '3014', '3026', '3038', '3044') then 'Military'
     else 'Waiver' end as Exemption_Waiver,
    a.TBBDETC_DESC as Exemption_Waiver_Desc label 'Exemption_Waiver_Desc',
    'Merit-Based' as Need,
```

```
   Sum(b.TBRACCD_AMOUNT) as Amount
 from (select TBBDETC_DETAIL_CODE,
     TBBDETC_DESC,
     TBBDETC_DCAT_CODE
   from TAISMGR.TBBDETC
   where TBBDETC_DESC contains ')-W' or TBBDETC_DESC contains 'Waiver' or TBBDETC_DCAT_CODE = 'EXM'
     or TBBDETC_DETAIL_CODE in ('3000', '3002', '3004', '3006', '3008', '3014', '3026', '3038', '3042', '3044') ) as a
 left join TAISMGR.TBRACCD as b on a.TBBDETC_DETAIL_CODE = b.TBRACCD_DETAIL_CODE
 Left Join edw.PIDM_TO_ID as c on b.TBRACCD_PIDM = c.PIDM
 Left Join (select  Distinct
      ID,
      ACADEMIC_PERIOD,
      STUDENT_LEVEL,
      STUDENT_LEVEL_DESC
    From ODSMGR.ACADEMIC_STUDY
    Where PRIMARY_PROGRAM_IND = "Y") as h on c.Id = h.ID and b.TBRACCD_TERM_CODE = h.ACADEMIC_PERIOD
 where PIDM Ne .
 group by b.TBRACCD_PIDM, b.TBRACCD_TERM_CODE, Exemption_Waiver, Exemption_Waiver_Desc
 Having amount >=1;
Quit;

Data M014; set M014_2 M014_1; Run;

Data Tasklist; /*Create Table for all Tasks to be inserted into*/ Format Report $20. Student $20. Task $2000. Ref $20.; Run;

Proc SQL; /*List of all undergraduate Students and calculate the term they first started taking courses with us. This This is the base list for the cohort report.*/
 Create Table Cohort as
  Select Distinct
    PERSON_UID,
    ID,
    ACADEMIC_PERIOD as Cohort_Period,
    ACADEMIC_YEAR_DESC as Cohort_Desc,
    SubStr(ACADEMIC_YEAR_DESC, 1, 4)||'-'||SubStr(ACADEMIC_YEAR_DESC, 8, 2) as Cohort /*Cohort created to match PDP requirements.*/
  From ODSMGR.STUDENT_COURSE
  Where  INSTITUTION_COURSE_IND = 'Y' and
    COURSE_LEVEL NE 'GR'
  Group by ID
  Having ACADEMIC_PERIOD = min(ACADEMIC_PERIOD)
 ; Quit;

Proc SQL; /*List of all undergraduate Students who took a courses during the term. This is the base list for the course report.*/
 Create Table Students as
  Select Distinct
    ID,
    ACADEMIC_PERIOD,
    ACADEMIC_YEAR_DESC,
    SubStr(ACADEMIC_YEAR_DESC, 1, 4)||'-'||SubStr(ACADEMIC_YEAR_DESC, 8, 2) as Academic_Year
  From ODSMGR.STUDENT_COURSE
  Where ACADEMIC_PERIOD in &Term and
    INSTITUTION_COURSE_IND = 'Y' and
    COURSE_LEVEL NE 'GR'
 ; Quit;

Proc SQL; /*Identifies students receiving Pell*/
 Create table Pell as
  Select  Distinct
    ID,
```

```
    Term,
    'Y' as Pell
  From M014
  Where Fund_Code = 'FPELL'
  Group by ID,
     term;
Quit;

Proc SQL; /*Returns ISIR data for all students by aid Year. one row per person per aid year*/
 Create Table ISIR as
  Select Distinct
    ID,
    Substr(AID_YEAR_DESC, 1, 9) as Aid_Year,
    'Y' as ISIR,
    ADJUSTED_GROSS_INCOME,
    HOUSING,
    FM_INAS,
    DEPENDENCY_INDEPEND,
    HOUSING,
    Housing_Desc,
    Put(PELL_EFC, 8.) as PELL_EFC,
    Put(FAMILY_SIZE, 8.) as NoDepdents

  From ODSMGR.NEED_ANALYSIS
  Where CURRENT_RECORD_IND = 'Y'
  Group By ID, AID_YEAR_DESC;
Quit;

Proc SQL; /*Total tuition students was charged per year. One Row Per Student Per Academic Year*/
 Create Table Tuition as
  Select Distinct
    ID,
    ACADEMIC_YEAR_DESC,
    Sum(AMOUNT) as total_Tutition
  from ODSMGR.RECEIVABLE_ACCOUNT_DETAIL
  Where DETAIL_CODE_TYPE = 'C'
  Group by ID,
     ACADEMIC_YEAR_DESC
; Quit;

data Races; /*Get race information and format for report. One row per person */
 Length Race $2.;
 set GENERAL.GORPRAC;
 by GORPRAC_PIDM;
 length Race_new $200;
 Race = 'UK';
 if GORPRAC_RACE_CDE = 'WH' then Race = 'W';
 if GORPRAC_RACE_CDE = 'AS' then Race = 'AN';
 if GORPRAC_RACE_CDE = 'BL' then Race = 'B';
 if GORPRAC_RACE_CDE = 'HA' then Race = 'HP';
 if GORPRAC_RACE_CDE = 'IN' then Race = 'IA';

 retain Race_new;
 Race_new=ifc(first.GORPRAC_PIDM, RACE, catx('|', Race_new, RACE));
 if last.GORPRAC_PIDM then output;
 Keep GORPRAC_PIDM
   Race_new;
```

```
run;

/*1 = No High School, 3 = Some High School, no diploma, 4 = High School diploma or GED, 6 = Some college, 7 Associate/two-year degree
 8 = Bachelor's/four-year degree, 13 = Graduate/Professional degree
 0 Unknown or not applicable
 - These data were sourced from ApplyTexas prior to omitting the item from the report in Fall 2018*/
Proc SQL;
 Create Table First_Gen as
  select Distinct
    ID,
    person_uid,
    max(case when application_info1 in ('01', '03', '04') and application_info2 in ('01', '03', '04') then 1
       when application_info1 in ('06') or application_info2 in ('06') then 2
       when application_info1 in ('07') or application_info2 in ('07') then 3
       when application_info1 in ('08', '13') or application_info2 in ('08', '13') then 4
       else 0 end) as first_gen
  from odsmgr.admissions_application
  group by person_uid
; Quit;

Proc SQL; /*Get Identifying information for student. One row per student per semester*/
 Create Table Identifying_Information as
  Select Distinct
    b.ID,
    a.PERSON_UID,
    a.Cohort_Period,
    SubStr(b.ACADEMIC_YEAR_DESC, 1, 9) as Aid_Year,
    i.STUDENT_LEVEL,
a.Cohort,
Case When SubStr(a.Cohort_Period, 5, 2) = '01' then 'Spring'
   When SubStr(a.Cohort_Period, 5, 2) = '06' then 'Summer'
  When SubStr(a.Cohort_Period, 5, 2) = '08' then 'Fall'
  Else 'X' end as Cohort_Term,
put(Datepart(c.STVTERM_START_DATE), YYMMDDn8.) as CohortTermBeginDate,
Put(Datepart(c.STVTERM_END_DATE), YYMMDDn8.) as CohortTermEndDate,
Case When substr(d.TAX_ID, 1, 1) in ('9') then ''
  Else d.TAX_ID end as SSN 'SSN',
CAse When substr(d.TAX_ID, 1, 1) in ('9') then d.TAX_ID
  Else '' end as ITIN,
b.ID as Student_ID,
d.FIRST_NAME,
d.MIDDLE_NAME,
d.LAST_NAME,
d.NAME_SUFFIX, /*Used in Course Data not in Cohort Data*/
Case When Length(k.STREET_LINE1) < 2 then 'UK'
  Else Coalesce(Strip(k.STREET_LINE1), 'UK') end as STREET_LINE1 length = 30,
k.STREET_LINE2 length = 30,
coalesce(k.City, 'UK') as City,
coalesce(k.STATE_PROVINCE, 'UK') as State,
k.POSTAL_CODE,
Case When k.NATION in &Country_Code then k.NATION
  When k.STATE_PROVINCE in &State_Code then 'US'
  Else 'UK' end as Country,
put(Datepart(d.BIRTH_DATE), YYMMDDn8.) as BirthDate,
d.PHONE_NUMBER_COMBINED, /*Used in Course Data not in Cohort Data*/
d.EMAIL_ADDRESS, /*Used in Course Data not in Cohort Data*/
```

```
  Case When d.HISPANIC_LATINO_ETHNICITY_IND = 'Y' then 'H'
    When d.HISPANIC_LATINO_ETHNICITY_IND = 'N' then 'N'
    Else 'UK' end as Ethnicity,
  Coalesce(e.Race_new, "UK") as Race,
  'OPEID' as Institution_ID_Type,
  '04229500' as Institution_ID,
  j.credential,
  j.credential_provider

From Cohort as a
Left Join ODSMGR.STUDENT as b on a.PERSON_UID = b.PERSON_UID and a.Cohort_Period = b.ACADEMIC_PERIOD
Left Join SATURN.STVTERM as c on a.Cohort_Period = c.STVTERM_CODE
Left Join ODSMGR.PERSON as d on b.ID = d.ID
Left Join Races as e on a.PERSON_UID = e.GORPRAC_PIDM
Left Join (Select  Distinct
     PERSON_UID,
     ACADEMIC_PERIOD,
     STUDENT_LEVEL,
     PROGRAM
   From ODSMGR.ACADEMIC_STUDY
   Where PRIMARY_PROGRAM_IND = 'Y') as i on a.PERSON_UID = i.PERSON_UID and a.Cohort_Period = i.ACADEMIC_PERIOD
Left Join tbl_GPA as h on d.ID = h.Id and a.Cohort_Period = h.Term_code and i.STUDENT_LEVEL = h.LEVEL
Left Join Program_Accreditation as j on i.PROGRAM = j.PROGRAM
Left Join (Select Distinct
     id,
     Tranwrd(STREET_LINE1, ', ', ' ') as STREET_LINE1 Length = 30,
     Tranwrd(STREET_LINE2, ', ', ' ') as STREET_LINE2 Length = 30,
     Tranwrd(CITY, ', ', ' ') as City Length = 20,
     STATE_PROVINCE,
     POSTAL_CODE,
     NATION
    From (select *,
        Case When PREFERRED_ADDRESS_IND = "Y" then 1 Else 0 End as Pref_Ind
      from ODSMGR.ADDRESS m
   Where m.address_status_ind is null
     and datepart(m.address_start_date) <= "&SYSDATE9"d <= coalesce(datepart(m.address_end_date), '31DEC2099'd)
     Group by id
     Having Address_Number = max(address_Number))
    Where address_type in ('MA', 'PA')
    Group by Id
    Having Pref_Ind = Max(Pref_Ind)) as K on a.ID = k.ID
  Where b.ID is not missing and i.Student_Level = 'UG'
; Quit;

Proc SQL; /*Get students High school information one row per person */
 Create Table High_School_Information as
  Select Distinct
    ID,
    Case when SECONDARY_DIPLOMA in ('DS', 'ST') then 'D' Else '' end as HS_Completion_Status,
    put(Datepart(SECONDARY_SCHOOL_GRAD_DATE), YYMMDDn8.) as HS_Completion_Year,
    case when SCHOOL_GPA > '4.00' Then '' else SCHOOL_GPA end as HS_Unweighted_GPA 'HS_Unweighted_GPA',
    '' as HS_Weighted_GPA
  From ODSMGR.PREVIOUS_EDUCATION
  Where INSTITUTION_TYPE_DESC = 'High School' and REQUIREMENT = 'HST1' and SECONDARY_DIPLOMA is not missing
; Quit;

Proc SQL; /*Get student enrollment and GPA data for when students started at institution. one row per person*/
```

```
Create Table Enrollment_Info as
 Select Distinct
   a.ID,
   a.SHRTGPA_PIDM,
   a.term_code,
   Case when b.first_gen = 1 then 'N'
     when b.first_gen = 2 then 'P'
     when b.first_gen = 3 then 'A'
     when b.first_gen = 4 then 'B'
     else '' end as First_Gen,
   '' as Dual_and_Summer_Enrollment,
   Case When STUDENT_POPULATION = 'N' then 'F'
     Else 'T' end as Enrollment_Type,
   Coalesce(a.Thours, 0) as T_Hours 'Number of College Credits Attempted to Transfer',
   Coalesce(a.Thours, 0) as Hours_Accepted,
   Case When d.SZRTXSI_MATH_OBLIG_MET = 'Y' then 'C'
     When d.SZRTXSI_MATH_OBLIG_MET = 'N' then d.SZRTXSI_MATH_OBLIG_MET
     Else 'UK' end as Math,
   Case When d.SZRTXSI_WRTG_OBLIG_MET = 'Y' then 'C'
     When d.SZRTXSI_WRTG_OBLIG_MET = 'N' then d.SZRTXSI_WRTG_OBLIG_MET
     Else 'UK' end as ENGLISH,
   'N' as G_Math,
   'N' as G_English

 From tbl_GPA as a
 Left Join First_gen as b on b.ID = a.ID
 Left Join ODSMGR.ACADEMIC_STUDY as c on c.ID = a.ID and c.ACADEMIC_PERIOD = a.Term_Code
 Left Join (Select * from TXCNMGR.SZRTXSI group by SZRTXSI_PIDM having SZRTXSI_SEQ_NO = min(SZRTXSI_SEQ_NO)) as d on a.SHRTGPA_PIDM = d.SZRTXSI_PIDM
 Where Level = 'UG' and PRIMARY_PROGRAM_IND = "Y"
; Quit;

Proc SQL; /*Get academic term information. one row per person, per term.*/
 Create Table Academic_Term_Information as
  Select Distinct
    a.ID,
    a.ACADEMIC_PERIOD,
    a.ACADEMIC_STUDY_VALUE as STUDENT_LEVEL,

    /*Academic Term Information*/
    'NA' as CompleteDevMath,
    'NA' as CompleteDevEnglish,
    'N' as TTransferIntent,
    Case when AWARD_CATEGORY = '24' then 'B'
      when AWARD_CATEGORY = '42' then 'M'
      Else 'UK' end as DegreeTypeSought,
    Put(a.GPA, 8.2) as Sem_GPA,
    Put(c.OGPA, 8.2) as Overall_GPA
  From ODSMGR.GPA_BY_TERM as a
  Left Join ODSMGR.ACADEMIC_STUDY as b on a.id = b.ID and a.Academic_Period = b.Academic_Period and PRIMARY_PROGRAM_IND = 'Y'
  Left Join tbl_GPA as c on a.ID = c.ID and a.ACADEMIC_PERIOD = c.Term_Code and a.ACADEMIC_STUDY_VALUE = c.Level
  Where a.ACADEMIC_STUDY_VALUE = 'UG' and a.GPA_TYPE = 'I'
; Quit;

Proc SQL; /*Get course data for the courses student attended during given period*/
 Create Table Course_Information as
  Select Distinct
    a.ID,
```

```
a.ACADEMIC_PERIOD,
a.ACADEMIC_YEAR_DESC,
Case when f.STUDENT_CLASSIFICATION = 'GR' then 'GR' Else 'UG' end as Student_Level,
SubStr(a.ACADEMIC_YEAR_DESC, 1, 4)||'-'||SubStr(a.ACADEMIC_YEAR_DESC, 8, 2) as Academic_Year,
Case When Substr(a.ACADEMIC_PERIOD, 5, 2) = '08' then 'Fall'
  When Substr(a.ACADEMIC_PERIOD, 5, 2) = '01' then 'Spring'
  When Substr(a.ACADEMIC_PERIOD, 5, 2) = '06' then 'Summer'
  Else "UK" end as Term,
Compress(a.SUBJECT) as Subject,
a.COURSE_NUMBER,
a.COURSE_REFERENCE_NUMBER,
a.COURSE_TITLE_SHORT,
c.COURSE_TEXT_NARRATIVE format = $255.,
Catx(".", Substr(c.PROGRAM_CLASSIFICATION, 1, 2), Substr(c.PROGRAM_CLASSIFICATION, 3, 4)) as PROGRAM_CLASSIFICATION,
Case When a.COURSE_LEVEL = 'UG' then 'CU'
  When a.COURSE_LEVEL = 'GR' then 'CG'
  Else 'O' end as Course_Type,
'NA' as GateWayCourse,
'N' as CoRequisite,
Put(Datepart(a.START_DATE), YYMMDDn8.) as CourseBeginDate,
Put(Datepart(a.END_DATE), YYMMDDn8.) as CourseEndDate,
Case When a.FINAL_GRADE in ('A') then '4'
  When a.FINAL_GRADE in ('B') then '3'
  When a.FINAL_GRADE in ('C') then '2'
  When a.FINAL_GRADE in ('D') then '1'
  When a.FINAL_GRADE in ('F', 'FN', 'FS') then '0'
  When a.FINAL_GRADE in ('I', 'P', 'W') then a.FINAL_GRADE
  When a.FINAL_GRADE in ('AU') then 'A'
  When a.FINAL_GRADE in ('NP', 'U') then 'F'
  When a.FINAL_GRADE in ('IP') then 'I'
  When a.FINAL_GRADE in ('N', 'NG') then 'O'
  When a.FINAL_GRADE in ('S') then 'P'
  When a.FINAL_GRADE in ('Q', 'WF') then 'W'
  Else 'O' end as Grade,
 Put(a.CREDITS_ATTEMPTED, 8.2) as CREDITS_ATTEMPTED,
 Put(a.CREDITS_EARNED, 8.2) as CREDITS_EARNED,
 Case when a.INSTRUCTION_METHOD = '1' then 'F'
  when a.INSTRUCTION_METHOD = '2' then 'O'
  Else 'H' end as Delivery_Method,
 'N' as Core_Course,
 '' as Core_Course_Type,
 '' as Core_Competency_Completed,
 put(d.OHOURS, 8.) as OHOURS,
 '1' as Purpose_of_Course_Exchange,
 'N' as Cert_Endorsed,
 '' as Cert_Industry,
 Put(Datepart(a.FINAL_GRADE_DATE), YYMMDDn8.) as Grade_Effective_Date,
 'OPEID' as DGI_ID_TYPE,
 '04229500' as DGI_ID,
 a.ID as DGI_Stu_ID

From ODSMGR.STUDENT_COURSE As a
Left Join ODSMGR.COURSE_CATALOG as c on a.COURSE_IDENTIFICATION = c.COURSE_IDENTIFICATION and a.ACADEMIC_PERIOD = c.ACADEMIC_PERIOD
Left Join tbl_GPA as d on a.ID = d.ID and a.ACADEMIC_PERIOD = d.Term_Code
Left Join ODSMGR.STUDENT as f on a.id = f.id and a.ACADEMIC_PERIOD = f.ACADEMIC_PERIOD
Where a.INSTITUTION_COURSE_IND = 'Y' and a.COURSE_LEVEL = 'UG' and a.ACADEMIC_PERIOD in &Term
; Quit;
```

```
Proc SQL; /*Get FASFA data for student. One row per person per aid year*/
 Create Table FASFA_Information as
  Select a.ID,
    a.Aid_Year,

    SubStr(a.Aid_Year, 1, 4)||'-'||SubStr(a.Aid_Year, 8, 2) as Academic_Year,
    Coalesce(a.ISIR, 'N') as Applied_Aid,
    a.DEPENDENCY_INDEPEND,
    Put(coalesce(a.ADJUSTED_GROSS_INCOME, 0), 8.) as ADJUSTED_GROSS_INCOME,
    a.Housing,
    '0' as Room,
    '0' as Board,
    '0' as Books,
    '0' as Other_exp,
    a.PELL_EFC,
    Case When e.MARITAL_STATUS = 'S' then '1'
      When e.MARITAL_STATUS = 'M' then '2'
      When e.MARITAL_STATUS = 'P' then '3'
      When e.MARITAL_STATUS in ('D', 'W') then '4'
      Else '' end as Marital_Status,
    a.NoDepdents
  From ISIR as a
  Left Join ODSMGR.PERSON as e on a.id = e.id
; Quit;

Proc SQL; /*Get grants and load data for each student by academic year.*/
 Create Table Grant_Loans as
  Select ID,
Level_Cde,
AID_YEar,
Sum(Case When Fund_Code = 'FSEOG' then Amount Else 0 end) as SEOG,
Sum(Case When Fund_Code in ('FTEACG', 'FTEACU') then Amount Else 0 end) as TEACH,
Sum(Case When Exemption_Waiver = 'Military' then Amount Else 0 end) as Military,
Sum(Case When Source = 'Federal' and Type = 'Grant' and FUND_CODE not in ('FSEOG', 'FTEACG', 'FTEACU', 'FPELL') then Amount Else 0 end) as FGO,
Sum(Case When Fund_Code = 'FPELL' then Amount Else 0 end) as Pell,
Sum(Case When Source = 'State' and Type = 'Grant' and Need = 'Need-Based' then Amount Else 0 end) as SGN,
Sum(Case When Source = 'State' and Type = 'Grant' and Need = 'Merit-Based' then Amount Else 0 end) as SGM,
Sum(Case When Source = 'Institutional' and Type = 'Grant' and Need = 'Need-Based' then Amount Else 0 end) as IGN,
0 as INST_Grant_Employ,
Sum(Case When Source = 'Institutional' and Type = 'Grant' and Need = 'Merit-Based' then Amount Else 0 end) as IGM,
0 as INST_Grant_Mil,
0 as INST_Grant_Other,
0 as Grant_Other,
Sum(Case When Source = 'Federal' and Type = 'Loan' and FUND_CODE not in ('FPLUS', 'FGPLS', 'FGPLSS', 'ZPLUS', 'ZPLSM') then Amount Else 0 end) as FLO,
Sum(Case When Source = 'State' and Type = 'Loan' then Amount Else 0 end) as SL,
Sum(Case When Source = 'Institutional' and Type = 'Loan' then Amount Else 0 end) as IL,
Sum(Case When FUND_CODE in ('FPLUS', 'FGPLS', 'FGPLSS', 'ZPLUS', 'ZPLSM') then Amount Else 0 end) as PLUS,
Sum(Case When Source = 'External' and Type = 'Loan' then Amount Else 0 end) as EL,
Sum(Case When Source = 'Federal' and Type = 'Work' then Amount Else 0 end) as FW,
Sum(Case When Source = 'State' and Type = 'Work' then Amount Else 0 end) as SW,
Sum(Case When Source = 'Institutional' and Type = 'Work' then Amount Else 0 end) as IW,
Sum(Amount) as Total_Aid,
Case When Sum(Amount) - calculated SEOG - calculated TEACH - calculated Military - calculated FGO - calculated Pell - calculated SGN - calculated SGM - calculated IGN
 - calculated IGM - calculated FLO - calculated SL - calculated PLUS - calculated EL - calculated FW - calculated SW - calculated IW > 0.00 then
Sum(Amount) - calculated SEOG - calculated TEACH - calculated Military - calculated FGO - calculated Pell - calculated SGN - calculated SGM - calculated IGN
 - calculated IGM - calculated FLO - calculated SL - calculated PLUS - calculated EL - calculated FW - calculated SW - calculated IW
```

```
     Else 0 end as Other
  From M014
  Where Level_Cde = 'UG'
  Group by ID,
     Level_Cde,
     AID_YEar
; Quit;

/*** Cohort Report ***/
Proc SQL; /*Combine sections to create Cohort Data*/
 Create Table Cohort_Data_File as
  Select Distinct
    a.ID,
   /*Identifying Information*/
    'D1' as CH1,
    b.Cohort,
    b.Cohort_Term,
    b.CohortTermBeginDate,
    b.CohortTermEndDate,
    b.SSN,
    b.ITIN,
    b.Student_ID,
    b.FIRST_NAME,
    b.MIDDLE_NAME,
    b.LAST_NAME,
    Coalesce(b.STREET_LINE1, 'UK') as STREET_LINE1,
    b.STREET_LINE2,
    Coalesce(b.City, 'UK') as City,
    Coalesce(b.State, 'UK') as State,
    b.POSTAL_CODE,
    b.Country,
    b.BirthDate,
    b.Ethnicity,
    b.Race,
    b.Institution_ID_Type,
    b.Institution_ID,

   /*High School Information*/

    c.HS_Completion_Status,
    c.HS_Completion_Year,
    c.HS_Unweighted_GPA,
    c.HS_Weighted_GPA,

   /*Enrollment Information*/

    d.First_Gen,
    d.Dual_and_Summer_Enrollment,
    d.Enrollment_Type,
    d.T_Hours,
    d.Hours_Accepted,
    d.Math,
    d.English,
    d.G_Math,
    d.G_English

  From Students as a
```

```
  Left Join Identifying_Information as b on a.ID = b.ID
  Left Join High_School_Information as c on a.ID = c.ID
  Left Join Enrollment_Info as d on a.ID = d.ID and b.Cohort_Period = d.term_code
  Where b.Cohort is not missing and b.Cohort_Period in &Term
; Quit;

/**************** Cohort Task list Start ****************/
Proc SQL;
 Insert into Tasklist
  Select 'Cohort' as Report, ID as Student, 'Student is missing required field Cohort' as task, 'Cohort.2.1' as Ref
  From Cohort_Data_File
  Where Cohort is missing;
 Insert into Tasklist
  Select 'Cohort' as Report, ID as Student, 'Student is missing required field Cohort Term Begin Date' as task, 'Cohort.4.1' as Ref
  From Cohort_Data_File
  Where CohortTermBeginDate is missing;
 Insert into Tasklist
  Select 'Cohort' as Report, ID as Student, 'Student is missing required field Cohort Term End Date' as task, 'Cohort.5.1' as Ref
  From Cohort_Data_File
  Where CohortTermEndDate is missing;
 Insert into Tasklist
  Select 'Cohort' as Report, ID as Student, 'SSN cannot be blank if Student ID is blank' as task, 'Cohort.6.1' as Ref
  From Cohort_Data_File
  Where SSN is missing and Student_ID is Missing;
 Insert into Tasklist
  Select 'Cohort' as Report, ID as Student, 'Student is missing required field First Name' as task, 'Cohort.9.1' as Ref
  From Cohort_Data_File
  Where First_Name is Missing;
 Insert into Tasklist
  Select 'Cohort' as Report, ID as Student, 'Student is missing required field Last Name' as task, 'Cohort.11.1' as Ref
  From Cohort_Data_File
  Where Last_Name is Missing;
Insert into Tasklist
  Select 'Cohort' as Report, ID as Student, 'Student is missing required field Street Line 1' as task, 'Cohort.12.1' as Ref
  From Cohort_Data_File
  Where STREET_LINE1 is missing;
Insert into Tasklist
  Select 'Cohort' as Report, ID as Student, 'Student is missing required field City' as task, 'Cohort.14.1' as Ref
  From Cohort_Data_File
  Where City is missing;
Insert into Tasklist
  Select 'Cohort' as Report, ID as Student, 'Student is missing required field State' as task, 'Cohort.15.1' as Ref
  From Cohort_Data_File
  Where State is missing;
Insert into Tasklist
  Select 'Cohort' as Report, ID as Student, 'Student is missing required field Country' as task, 'Cohort.17.1' as Ref
  From Cohort_Data_File
  Where Country is missing;
Insert into Tasklist
  Select 'Cohort' as Report, ID as Student, 'Student is missing required field Date of Birth' as task, 'Cohort.18.1' as Ref
  From Cohort_Data_File
  Where BirthDate is missing;
Insert into Tasklist
  Select 'Cohort' as Report, ID as Student, 'Student is missing required field Ethnicity' as task, 'Cohort.19.1' as Ref
  From Cohort_Data_File
  Where Ethnicity is missing;
Insert into Tasklist
```

```
 Select 'Cohort' as Report, ID as Student, 'Student is missing required field Race' as task, 'Cohort.20.1' as Ref
 From Cohort_Data_File
 Where Race is missing;
Insert into Tasklist
 Select 'Cohort' as Report, ID as Student, 'Student is missing required field Institution ID Type' as task, 'Cohort.21.1' as Ref
 From Cohort_Data_File
 Where Institution_ID_Type is missing;
Insert into Tasklist
 Select 'Cohort' as Report, ID as Student, 'Student is missing required field Institution ID' as task, 'Cohort.22.1' as Ref
 From Cohort_Data_File
 Where Institution_ID is missing;
Insert into Tasklist
 Select 'Cohort' as Report, ID as Student, 'Student is missing required field Enrollment Type' as task, 'Cohort.29.1' as Ref
 From Cohort_Data_File
 Where Enrollment_Type is missing;
Insert into Tasklist
 Select 'Cohort' as Report, ID as Student, 'Student is missing required field Math Placement' as task, 'Cohort.32.1' as Ref
 From Cohort_Data_File
 Where Math is missing;
Insert into Tasklist
 Select 'Cohort' as Report, ID as Student, 'Student is missing required field English Placement' as task, 'Cohort.33.1' as Ref
 From Cohort_Data_File
 Where English is missing;
Insert into Tasklist
 Select 'Cohort' as Report, ID as Student, 'Student is missing required field Gateway Math Status' as task, 'Cohort.34.1' as Ref
 From Cohort_Data_File
 Where G_Math is missing;
Insert into Tasklist
 Select 'Cohort' as Report, ID as Student, 'Student is missing required field Gateway English Status' as task, 'Cohort.35.1' as Ref
  From Cohort_Data_File
  Where G_English is missing;
Quit;

/**************** Task list End ****************/

/**************** Export Cohort Data into Spreadsheet for review ****************/
Proc Export data = Cohort_Data_File outfile = "&Path\Ouput data for &OUTPUT_Term..XLSX" label DBMS = xlsx Replace;
 Sheet = "Cohort";
Run;

Data Cohort_Output; /*Cohort report header*/
 Length output $2000.;
 Output = 'DCE01, 10073232, 042295, 00, , , , , '||put("&SYSDATE"d, YYMMDDn8.)||", , &OUTPUT_Term, , , , ";
Run;

Proc SQL;
 Insert into Cohort_Output
  Values('CH1, Cohort, Cohort Term, Cohort Term Begin Date, Cohort Term End Date, SSN, ITIN, Student ID, First Name, Middle Name, Last Name, Street Line 1, Street Line 2,
    City, State, Zip/Postal Code, Country, Date of Birth, Ethnicity, Race, Institution ID Type, Institution ID, HS Completion Status, HS Completion Year, HS Unweighted GPA,
    HS Weighted GPA, First Gen, Dual and Summer Enrollment, Enrollment Type, Number of College Credits Attempted to Transfer, Number of College Transfer Credits Accepted,
    Math Placement, English Placement, Gateway Math Status, Gateway English Status')
; Quit;

Proc SQL; /*Cohort Report Body*/
 Insert into Cohort_Output
  Select Strip(CH1)||", "||
    Strip(Cohort)||", "||
```

```
Strip(Cohort_Term)||", "||
strip(CohortTermBeginDate)||", "||
Strip(CohortTermEndDate)||", "||
Strip(SSN)||", "||
Strip(ITIN)||", "||
Strip(Student_ID)||", "||
Strip(FIRST_NAME)||", "||
Strip(MIDDLE_NAME)||", "||
Strip(LAST_NAME)||", "||
Strip(STREET_LINE1)||", "||
Strip(STREET_LINE2)||", "||
Strip(City)||", "||
Strip(State)||", "||
Strip(POSTAL_CODE)||", "||
Strip(Country)||", "||
strip(BirthDate)||", "||
Strip(Ethnicity)||", "||
Strip(Race)||", "||
Strip(Institution_ID_Type)||", "||
Strip(Institution_ID)||", "||
Strip(HS_Completion_Status)||", "||
Strip(HS_Completion_Year)||", "||
Strip(HS_Unweighted_GPA)||", "||
Strip(HS_Weighted_GPA)||", "||
Strip(First_Gen)||", "||
Strip(Dual_and_Summer_Enrollment)||", "||
Strip(Enrollment_Type)||", "||
strip(put(T_Hours, 8.))||", "||
Strip(put(Hours_Accepted, 8.))||", "||
    Strip(Math)||", "||
    Strip(English)||", "||
    Strip(G_Math)||", "||
    Strip(G_English) as Output
  From Cohort_Data_File;
Quit;

Proc SQL; /*Cohort report footer*/
 Insert into Cohort_Output
  Select "T1, "||strip(Put(Count(*)+3, 8.))||", " as Output
  From Cohort_Data_File;
Quit;

Proc Export data = Cohort_Output outfile = "&Path\i_04229500PDP_Texas_AM_University_Central_Texas_Cohort_&OUTPUT_Term..Txt" label DBMS = Tab Replace; putnames = No;
Run;

Proc SQL; /*Combine sections to create Course Data*/
 Create Table Course_Data_File as
  Select Distinct
    a.ID,
    e.ACADEMIC_PERIOD,
    Catx('-', e.SUBJECT, e.COURSE_NUMBER, e.COURSE_REFERENCE_NUMBER) as Course,

   /*Identifying Information*/

    'D1' as CH1,
    b.Cohort,
    b.Cohort_Term,
```

```
/*Course Data*/
 e.Academic_Year,
 e.Term,
/*End Course Data*/
 b.Institution_ID_Type,
 b.Institution_ID,
 b.SSN,
 b.ITIN,
 b.Student_ID,
 b.FIRST_NAME,
 b.MIDDLE_NAME,
 b.LAST_NAME,
 b.NAME_SUFFIX,
 Coalesce(b.STREET_LINE1, 'UK') as Street_Line1,
 b.STREET_LINE2,
 Coalesce(b.City, 'UK') as City,
 Coalesce(b.State, 'UK')As State,
 b.POSTAL_CODE,
 b.Country,
 b.BirthDate,
 '' as PHONE_NUMBER_COMBINED,
/*Pell*/
 coalesce(c.Pell, 'N') as Pell,
/*End Pell*/
 b.EMAIL_ADDRESS,

/*Academic Term Information*/

 d.CompleteDevMath,
 d.CompleteDevEnglish,
 d.TTransferIntent,
 d.DegreeTypeSought,
 Coalesce(d.Sem_GPA, '0') as Sem_GPA,
 Case when d.Overall_GPA = '.' then '0'
   Else coalesce(d.Overall_GPA, '0') end as Overall_GPA,

/*Course Information*/
 e.SUBJECT,
 e.COURSE_NUMBER,
 e.COURSE_REFERENCE_NUMBER,
 Tranwrd(e.COURSE_TITLE_SHORT, ', ', ' ') as COURSE_TITLE_SHORT,
 Tranwrd(e.COURSE_TEXT_NARRATIVE, ', ', ' ') as COURSE_TEXT_NARRATIVE,
 e.PROGRAM_CLASSIFICATION,
 e.Course_Type,
 'NA' as GateWayCourse,
 'N' as CoRequisite,
 e.CourseBeginDate,
 e.CourseEndDate,
 e.Grade,
 e.CREDITS_ATTEMPTED,
 e.CREDITS_EARNED,
 e.Delivery_Method,
 e.Core_Course,
 e.Core_Course_Type,
 e.Core_Competency_Completed,
 e.OHOURS,
 e.Purpose_of_Course_Exchange,
```

```
    Coalesce(b.credential, e.Cert_Endorsed) as Cert_Endorsed,
    Coalesce(b.credential_provider, e.Cert_Industry) as Cert_Industry,
    e.Grade_Effective_Date,
    e.DGI_ID_TYPE,
    e.DGI_ID,
    e.DGI_Stu_ID
  From Students as a
  Left Join Identifying_Information as b on a.ID = b.ID
  Left Join Pell as c on a.id = c.id and a.ACADEMIC_PERIOD = c.Term
  Left Join Academic_Term_Information as d on a.ID = d.ID and a.ACADEMIC_PERIOD = d.ACADEMIC_PERIOD
  Left join Course_Information as e on a.ID = e.ID and a.ACADEMIC_PERIOD = e.ACADEMIC_PERIOD
  Where b.Cohort is not missing and a.ACADEMIC_PERIOD in &Term
; Quit;

/***************** Course Task list Start ****************/
Proc SQL;
 Create Table DegreeSoughtTest as
  Select Sum(Case When DegreeTypeSought = 'B' then 1 Else 0 End) as Bach,
    Sum(Case When DegreeTypeSought = 'M' then 1 Else 0 End) as Mast,
    Sum(Case When DegreeTypeSought = 'UK' then 1 Else 0 End) as UNKNW,
    Count(*) as total
  From Course_Data_File;
Quit;

Proc SQL;
 Insert into Tasklist
  Select 'Course' as Report, ID as Student, 'Student is missing required field CH1' as task, 'Course.1.1' as Ref
  From Course_Data_File
  Where CH1 is missing;
Insert into Tasklist
 Select 'Course' as Report, ID as Student, 'Student is missing required field Cohort' as task, 'Course.2.1' as Ref
 From Course_Data_File
 Where Cohort is missing;
Insert into Tasklist
 Select 'Course' as Report, ID as Student, 'Student is missing required field Cohort Term' as task, 'Course.3.1' as Ref
 From Course_Data_File
 Where Cohort_Term is missing;
Insert into Tasklist
 Select 'Course' as Report, ID as Student, 'Student is missing required field Academic Year' as task, 'Course.4.1' as Ref
 From Course_Data_File
 Where Academic_Year is missing;
Insert into Tasklist
 Select 'Course' as Report, ID as Student, 'Student is missing required field Term' as task, 'Course.5.1' as Ref
 From Course_Data_File
 Where Term is missing;
Insert into Tasklist
 Select 'Course' as Report, ID as Student, 'Student is missing required field Insitution ID Type' as task, 'Course.6.1' as Ref
 From Course_Data_File
 Where Institution_ID_Type is missing;
Insert into Tasklist
 Select 'Course' as Report, ID as Student, 'Student is missing required field Insitution ID' as task, 'Course.7.1' as Ref
 From Course_Data_File
 Where Institution_ID is missing;
Insert into Tasklist
 Select 'Course' as Report, ID as Student, 'Student must have either an SSN or Student ID' as task, 'Course.8.1' as Ref
 From Course_Data_File
 Where SSN is missing and Student_ID is missing;
```

```
Insert into Tasklist
 Select 'Course' as Report, ID as Student, 'Student is missing required field First Name' as task, 'Course.11.1' as Ref
 From Course_Data_File
 Where First_Name is missing;
Insert into Tasklist
 Select 'Course' as Report, ID as Student, 'Student is missing required field Last Name' as task, 'Course.13.1' as Ref
 From Course_Data_File
 Where Last_Name is missing;
Insert into Tasklist
 Select 'Course' as Report, ID as Student, 'Student is missing required field Current Street 1' as task, 'Course.15.1' as Ref
 From Course_Data_File
 Where Street_Line1 is missing;
Insert into Tasklist
 Select 'Course' as Report, ID as Student, 'Student is missing required field Current City' as task, 'Course.17.1' as Ref
 From Course_Data_File
 Where City is missing;
Insert into Tasklist
 Select 'Course' as Report, ID as Student, 'Student is missing required field Current State' as task, 'Course.18.1' as Ref
 From Course_Data_File
 Where State is missing;
Insert into Tasklist
 Select 'Course' as Report, ID as Student, 'Student is missing required field Current Country' as task, 'Course.20.1' as Ref
 From Course_Data_File
 Where Country is missing;
Insert into Tasklist
 Select 'Course' as Report, ID as Student, 'Student is missing required field Date of Birth' as task, 'Course.21.1' as Ref
 From Course_Data_File
 Where BirthDate is missing;
Insert into Tasklist
 Select 'Course' as Report, ID as Student, 'Student is missing required field Pell' as task, 'Course.23.1' as Ref
 From Course_Data_File
 Where Pell is missing;
Insert into Tasklist
 Select 'Course' as Report, ID as Student, 'Student is missing required field CompleteDevMath' as task, 'Course.25.1' as Ref
 From Course_Data_File
 Where CompleteDevMath is missing;
Insert into Tasklist
 Select 'Course' as Report, ID as Student, 'Student is missing required field CompleteDevMath' as task, 'Course.26.1' as Ref
 From Course_Data_File
 Where CompleteDevEnglish is missing;
Insert into Tasklist
 Select 'Course' as Report, ID as Student, 'Student is missing required field Degree Type Sought' as task, 'Course.28.1' as Ref
 From Course_Data_File
 Where DegreeTypeSought is missing;
Insert into Tasklist
 Select 'Course' as Report, ID as Student, 'Student is missing required field Semester/Session GPA' as task, 'Course.29.1' as Ref
 From Course_Data_File
 Where Sem_GPA is missing;
Insert into Tasklist
 Select 'Course' as Report, ID as Student, 'Student is missing required field Overall GPA' as task, 'Course.30.1' as Ref
 From Course_Data_File
 Where Overall_GPA is missing;
Insert into Tasklist
 Select 'Course' as Report, ID as Student, 'Student is missing required field Course Prefix' as task, 'Course.31.1' as Ref
 From Course_Data_File
 Where Subject is missing;
Insert into Tasklist
```

```
 Select 'Course' as Report, ID as Student, 'Student is missing required field Course Number' as task, 'Course.32.1' as Ref
 From Course_Data_File
 Where Course_Number is missing;
Insert into Tasklist
 Select 'Course' as Report, ID as Student, 'Student is missing required field Course Number' as task, 'Course.33.1' as Ref
 From Course_Data_File
 Where COURSE_REFERENCE_NUMBER is missing;
Insert into Tasklist
 Select 'Course' as Report, ID as Student, 'Student is missing required field Course Name' as task, 'Course.34.1' as Ref
 From Course_Data_File
 Where COURSE_TITLE_SHORT is missing;
Insert into Tasklist
 Select 'Course' as Report, ID as Student, 'Course is listed as being able to exchange for reverse transfer and requires Course Description' as task, 'Course.35.1' as Ref
 From Course_Data_File
 Where COURSE_TEXT_NARRATIVE is missing and Purpose_of_Course_Exchange = '1';
Insert into Tasklist
 Select 'Course' as Report, ID as Student, 'Student is missing required field Course Classification of Instructional Programs (CIP) code' as task, 'Course.36.1' as Ref
 From Course_Data_File
 Where PROGRAM_CLASSIFICATION is missing;
Insert into Tasklist
 Select 'Course' as Report, ID as Student, 'Student is missing required field Course Type' as task, 'Course.37.1' as Ref
 From Course_Data_File
 Where Course_Type is missing;
Insert into Tasklist
 Select 'Course' as Report, ID as Student, 'Student is missing required field Math Or English Gateway' as task, 'Course.38.1' as Ref
 From Course_Data_File
 Where GateWayCourse is missing;
Insert into Tasklist
 Select 'Course' as Report, ID as Student, 'Student is missing required field Course Begin Date' as task, 'Course.40.1' as Ref
 From Course_Data_File
 Where CourseBeginDate is missing;
Insert into Tasklist
 Select 'Course' as Report, ID as Student, 'Student is missing required field Course Begin Date' as task, 'Course.41.1' as Ref
 From Course_Data_File
 Where CourseEndDate is missing;
Insert into Tasklist
 Select 'Course' as Report, ID as Student, 'Student is missing required field Grade' as task, 'Course.42.1' as Ref
 From Course_Data_File
 Where Grade is missing;
Insert into Tasklist
 Select 'Course' as Report, ID as Student, 'Student is missing required field Number of Credits Attempted' as task, 'Course.43.1' as Ref
 From Course_Data_File
 Where CREDITS_ATTEMPTED is missing;
Insert into Tasklist
 Select 'Course' as Report, ID as Student, 'Student is missing required field Number of Credits Earned' as task, 'Course.44.1' as Ref
 From Course_Data_File
 Where CREDITS_EARNED is missing;
Insert into Tasklist
 Select 'Course' as Report, ID as Student, 'Course is identified as core course and must have competency area' as task, 'Course.47.1' as Ref
 From Course_Data_File
 Where Core_Course_Type is missing and Core_Course = "Y";
Insert into Tasklist
 Select 'Course' as Report, ID as Student, 'Course is listed as being able to exchange for reverse transfer and requires total combined earned and transferred credit' as task,
'Course.49.1' as Ref
 From Course_Data_File
 Where OHOURS is missing and Purpose_of_Course_Exchange = '1';
Insert into Tasklist
```

```
  Select 'Course' as Report, ID as Student, 'Student is missing required field Purpose of Course Exchange' as task, 'Course.50.1' as Ref
  From Course_Data_File
  Where Purpose_of_Course_Exchange is missing;
 Insert into Tasklist
  Select 'Course' as Report, ID as Student, 'Student is missing required field Purpose of DGI Institution ID Type' as task, 'Course.54.1' as Ref
  From Course_Data_File
  Where DGI_ID_TYPE is missing;
 Insert into Tasklist
  Select 'Course' as Report, ID as Student, 'Student is missing required field Purpose of DGI Institution ID' as task, 'Course.55.1' as Ref
  From Course_Data_File
  Where DGI_ID is missing;
 Insert into Tasklist
  Select 'Course' as Report, 'Report' as Student, 'The number of course records that contain a Degree Type Sought value of Unknown (UK) is greater than 0.' as task, 'Course.56.1'
as Ref
  From DegreeSoughtTest
  Where UNKNW > 0;
 Insert into Tasklist
  Select 'Course' as Report, 'Report' as Student, 'The number of course records that contain a Degree Type Sought value of Unknown (UK) exceeds the expected threshold.' as task,
'Course.57.1' as Ref
  From DegreeSoughtTest
  Where UNKNW/total > .01;
Quit;

/*************** Course Task list End ***************/

/*************** Export Course Data into Spreadsheet for review ***************/
Proc Export data = Course_Data_File outfile = "&Path\Ouput data for &OUTPUT_Term..XLSX" label DBMS = xlsx Replace;
 Sheet = "Course";
Run;

Data Course_Output; /*Course Report header*/
 Length output $2000.;
 Output = "DCE02, 10073232, 042295, 00, , , , , "||put("&SYSDATE"d, YYMMDDn8.)||", , &OUTPUT_Term, , , , ";
Run;

PRoc SQL;
 Insert into Course_Output
  Values('CH1, Cohort, Cohort Term, Academic Year, Term, Institution ID Type, Institution ID, SSN, ITIN, Student ID, First Name, Middle Name,
     Last Name, Suffix, Current Street 1, Current Street 2, Current City, Current State, Current Zip/Postal Code, Current Country,
     Date of Birth, Student Phone Number, Pell Recipient, Student Email, CompleteDevMath, CompleteDevEnglish, TransferIntent,
     Degree Type Sought, Semester/Session GPA, Overall GPA, Course Prefix, Course Number, Section ID, Course Name,
     Course Description, Course CIP, Course Type, MathOrEnglishGateway, Co-requisite Course, Course Begin Date, Course End Date,
     Grade, Number of Credits Attempted, Number of Credits Earned, Delivery Method, Core Course, Core Course Type,
     Core Competency Completed, Total Combined Earned and Transferred Credits, Purpose of Course Exchange,
     Certification Endorsed Curriculum/Program, Certificate Endorsing Industry, Grade Effective Date, DGI Institution ID Type,
     DGI Institution ID, DGI Student ID')
; Quit;

Proc SQL; /*Course report Body*/
 Insert into Course_Output
  Select strip(CH1)||", "||
    Strip(Cohort)||", "||
    Strip(Cohort_Term)||", "||
    Strip(Academic_Year)||", "||
    Strip(Term)||", "||
    Strip(Institution_ID_Type)||", "||
    Strip(Institution_ID)||", "||
```

```
Strip(SSN)||", "||
Strip(ITIN)||", "||
Strip(Student_ID)||", "||
Strip(FIRST_NAME)||", "||
Strip(MIDDLE_NAME)||", "||
Strip(LAST_NAME)||", "||
Strip(NAME_SUFFIX)||", "||
Strip(STREET_LINE1)||", "||
Strip(STREET_LINE2)||", "||
Strip(City)||", "||
Strip(State)||", "||
Strip(POSTAL_CODE)||", "||
Strip(Country)||", "||
STRIP(BirthDate)||", "||
Strip(PHONE_NUMBER_COMBINED)||", "||
Strip(Pell)||", "||
Strip(EMAIL_ADDRESS)||", "||
Strip(CompleteDevMath)||", "||
Strip(CompleteDevEnglish)||", "||
Strip(TTransferIntent)||", "||
Strip(DegreeTypeSought)||", "||
Strip(Sem_GPA)||", "||
Strip(Overall_GPA)||", "||
Strip(SUBJECT)||", "||
Strip(COURSE_NUMBER)||", "||
Strip(COURSE_REFERENCE_NUMBER)||", "||
Strip(COURSE_TITLE_SHORT)||", "||
Strip(COURSE_TEXT_NARRATIVE)||", "||
Strip(PROGRAM_CLASSIFICATION)||", "||
    Strip(Course_Type)||", "||
    Strip(GateWayCourse)||", "||
    Strip(CoRequisite)||", "||
    Strip(CourseBeginDate)||", "||
    Strip(CourseEndDate)||", "||
    Strip(Grade)||", "||
    Strip(CREDITS_ATTEMPTED)||", "||
    Strip(CREDITS_EARNED)||", "||
    Strip(Delivery_Method)||", "||
    Strip(Core_Course)||", "||
    Strip(Core_Course_Type)||", "||
    Strip(Core_Competency_Completed)||", "||
    Strip(OHOURS)||", "||
    Strip(Purpose_of_Course_Exchange)||", "||
    Strip(Cert_Endorsed)||", "||
    Strip(Cert_Industry)||", "||
    Strip(Grade_Effective_Date)||", "||
    Strip(DGI_ID_TYPE)||", "||
    Strip(DGI_ID)||", "||
    Strip(DGI_Stu_ID) as Output
  From Course_Data_File;
Quit;

Proc SQL; /*Course Report Footer*/
 Insert into Course_Output
  Select "T1, "||strip(Put(Count(*)+3, 8.))||", " as Output
  From Course_Data_File;
Quit;
```

```
data _null_; /*Export text file*/
 file "&Path\i_04229500PDP_Texas_AM_University_Central_Texas_Course_&OUTPUT_Term..Txt";
 set Course_Output;
 put outPut ~ ; *variable to be exported;
run;

Proc SQL; /*Combine sections to create Financial Aid Data*/
 Create Table Financial_Aid_Data_File as
  Select Distinct
    a.ID,

   /*Identifying Information*/

    'D1' as CH1,
    b.Cohort,
    b.Cohort_Term,
    a.Academic_Year,
    b.Institution_ID_Type,
    b.Institution_ID,
    b.SSN,
    b.ITIN,
    b.Student_ID,
    b.FIRST_NAME,
    b.MIDDLE_NAME,
    b.LAST_NAME,
    b.NAME_SUFFIX,
    b.STREET_LINE1,
    b.STREET_LINE2,
    b.City,
    b.State,
    b.POSTAL_CODE,
    b.Country,
    b.BirthDate,

  /*FASFA Data*/

    coalesce(c.Applied_Aid, 'N') as Applied_Aid,
    c.DEPENDENCY_INDEPEND,
    Case When Strip(c.ADJUSTED_GROSS_INCOME) < '0' then '0'
      Else Coalesce(Strip(c.ADJUSTED_GROSS_INCOME), '0') end as ADJUSTED_GROSS_INCOME,
    coalesce(Put(e.total_Tutition, 8.), '0') as total_Tutition,
    Coalesce(c.Housing, '3') as Housing,
    Coalesce(strip(c.Room), '0') as Room,
    coalesce(Strip(c.Board), '0') as Board,
    Coalesce(Strip(c.Books), '0') as Books,
    Coalesce(Strip(c.Other_exp), '0') as Other_Exp,
    Coalesce(Strip(c.PELL_EFC), '0') as PELL_EFC,
    c.Marital_Status,
    Case When c.NoDepdents = '.' then '0' end as NoDepdents,

  /*Grant and Loan Information*/

    Put(Coalesce(d.SEOG, 0), 8.) as SEOG,
    Put(Coalesce(d.TEACH, 0), 8.) as TEACH,
    Put(Coalesce(d.Military, 0), 8.) as Military,
    Put(Coalesce(d.FGO, 0), 8.) as FGO,
```

```
    Put(Coalesce(d.Pell, 0), 8.) as Pell,
    Put(Coalesce(d.SGN, 0), 8.) as SGN,
    Put(Coalesce(d.SGM, 0), 8.) as SGM,
    Put(Coalesce(d.IGN, 0), 8.) as IGN,
    Put(Coalesce(d.INST_Grant_Employ, 0), 8.) as INST_Grant_Employ,
    put(Coalesce(d.IGM, 0), 8.) as IGM,
    Put(Coalesce(d.INST_Grant_Mil, 0), 8.) as INST_Grant_Mil,
    Put(Coalesce(d.INST_Grant_Other, 0), 8.) as INST_Grant_Other,
    put(Coalesce(d.Grant_Other, 0), 8.) as Grant_Other,
    put(Coalesce(d.FLO, 0), 8.) as FLO,
    Put(Coalesce(d.SL, 0), 8.) as SL,
    Put(Coalesce(d.IL, 0), 8.) as IL,
    Put(Coalesce(d.PLUS, 0), 8.) as PLUS,
    Put(Coalesce(d.EL, 0), 8.) as EL,
    Put(Coalesce(d.FW, 0), 8.) as FW,
    Put(Coalesce(d.SW, 0), 8.) as SW,
    Put(Coalesce(d.IW, 0), 8.) as IW,
    Put(Coalesce(d.Other, 0), 8.) as Other
  From Students as a
  Left Join Identifying_Information as b on a.ID = b.ID
  Left Join FASFA_Information as c on a.id = c.id and a.ACADEMIC_YEAR_DESC = c.Aid_Year
  Left Join Grant_Loans as d on a.id = d.id and a.ACADEMIC_YEAR_DESC = d.Aid_Year
  Left Join Tuition as e on a.id = e.id and a.ACADEMIC_YEAR_DESC = e.ACADEMIC_YEAR_DESC
  Where b.Cohort is not missing
; Quit;

/**************** Financial Aid Task list Start ****************/
Proc SQL;
 Insert into Tasklist
 Select 'FinAid' as Report, ID as Student, 'Student is missing required field CH1' as task, 'FinAid.1.1' as Ref
 From Financial_Aid_Data_File
 Where CH1 is missing;
Insert into Tasklist
 Select 'FinAid' as Report, ID as Student, 'Student is missing required field Cohort' as task, 'FinAid.2.1' as Ref
 From Financial_Aid_Data_File
 Where Cohort is missing;
Insert into Tasklist
 Select 'FinAid' as Report, ID as Student, 'Student is missing required field Cohort Term' as task, 'FinAid.3.1' as Ref
 From Financial_Aid_Data_File
 Where Cohort_Term is missing;
Insert into Tasklist
 Select 'FinAid' as Report, ID as Student, 'Student is missing required field Academic Year' as task, 'FinAid.4.1' as Ref
 From Financial_Aid_Data_File
 Where Academic_Year is missing;
Insert into Tasklist
 Select 'FinAid' as Report, ID as Student, 'Student is missing required field Institution ID Type' as task, 'FinAid.5.1' as Ref
 From Financial_Aid_Data_File
 Where Institution_ID_Type is missing;
Insert into Tasklist
 Select 'FinAid' as Report, ID as Student, 'Student is missing required field Institution ID' as task, 'FinAid.6.1' as Ref
 From Financial_Aid_Data_File
 Where Institution_ID is missing;
Insert into Tasklist
 Select 'FinAid' as Report, ID as Student, 'Student must have either SSN or Student ID' as task, 'FinAid.7.1' as Ref
 From Financial_Aid_Data_File
 Where SSN is missing and Student_ID is missing;
Insert into Tasklist
```

```
 Select 'FinAid' as Report, ID as Student, 'Student is missing required field First Name' as task, 'FinAid.10.1' as Ref
 From Financial_Aid_Data_File
 Where First_Name is missing;
Insert into Tasklist
 Select 'FinAid' as Report, ID as Student, 'Student is missing required field Last Name' as task, 'FinAid.12.1' as Ref
 From Financial_Aid_Data_File
 Where Last_Name is missing;
Insert into Tasklist
 Select 'FinAid' as Report, ID as Student, 'Student is missing required field Street Line 1' as task, 'FinAid.14.1' as Ref
 From Financial_Aid_Data_File
 Where Street_Line1 is missing;
Insert into Tasklist
 Select 'FinAid' as Report, ID as Student, 'Student is missing required field City' as task, 'FinAid.16.1' as Ref
 From Financial_Aid_Data_File
 Where City is missing;
Insert into Tasklist
 Select 'FinAid' as Report, ID as Student, 'Student is missing required field State' as task, 'FinAid.17.1' as Ref
 From Financial_Aid_Data_File
 Where State is missing;
Insert into Tasklist
 Select 'FinAid' as Report, ID as Student, 'Student is missing required field Country' as task, 'FinAid.19.1' as Ref
 From Financial_Aid_Data_File
 Where Country is missing;
Insert into Tasklist
 Select 'FinAid' as Report, ID as Student, 'Student is missing required field Birth Day' as task, 'FinAid.20.1' as Ref
 From Financial_Aid_Data_File
 Where BirthDate is missing;
Insert into Tasklist
 Select 'FinAid' as Report, ID as Student, 'Student is missing required field Applied Aid' as task, 'FinAid.21.1' as Ref
 From Financial_Aid_Data_File
 Where Applied_Aid is missing;
Insert into Tasklist
 Select 'FinAid' as Report, ID as Student, 'Student is identified as having applied aid and must have a dependency status' as task, 'FinAid.22.1' as Ref
 From Financial_Aid_Data_File
 Where DEPENDENCY_INDEPEND is missing and Applied_Aid = 'Y';
Insert into Tasklist
 Select 'FinAid' as Report, ID as Student, 'Student is identified as having applied aid and must have an income' as task, 'FinAid.23.1' as Ref
 From Financial_Aid_Data_File
 Where ADJUSTED_GROSS_INCOME is missing and Applied_Aid = 'Y';
Insert into Tasklist
 Select 'FinAid' as Report, ID as Student, 'Student is missing required field Tutition' as task, 'FinAid.24.1' as Ref
 From Financial_Aid_Data_File
 Where total_Tutition is missing;
Insert into Tasklist
 Select 'FinAid' as Report, ID as Student, 'Student is missing required field Type of Housing' as task, 'FinAid.25.1' as Ref
 From Financial_Aid_Data_File
 Where Housing is missing;
Insert into Tasklist
 Select 'FinAid' as Report, ID as Student, 'Student is missing required field Room Charges' as task, 'FinAid.26.1' as Ref
 From Financial_Aid_Data_File
 Where Room is missing;
Insert into Tasklist
 Select 'FinAid' as Report, ID as Student, 'Student is missing required field Board Charges' as task, 'FinAid.27.1' as Ref
 From Financial_Aid_Data_File
 Where Board is missing;
Insert into Tasklist
 Select 'FinAid' as Report, ID as Student, 'Student is missing required field Books' as task, 'FinAid.28.1' as Ref
```

```
  From Financial_Aid_Data_File
  Where Books is missing;
 Insert into Tasklist
  Select 'FinAid' as Report, ID as Student, 'Student is missing required field Other Expense' as task, 'FinAid.29.1' as Ref
  From Financial_Aid_Data_File
  Where Other_Exp is missing;

/*Add Grant and loans task list.*/
Quit;

/**************** Financial Aid Task list End ****************/

/**************** Export Financial Aid Data into Spreadsheet for review ****************/
Proc Export data = Financial_Aid_Data_File outfile = "&Path\Ouput data for &OUTPUT_Term..XLSX" label DBMS = xlsx Replace;
 Sheet = "Financial Aid";
Run;

Data Financial_Aid_Output; /*Financial Aid Header*/
 Length output $2000.;
 Output = "DCE03, 10073232, 042295, 00, , , , , "||put("&SYSDATE"d, YYMMDDn8.)||", , &OUTPUT_Term, , , , ";
Run;

PRoc SQL;
 Insert into Financial_Aid_Output
  Values('CH1, Cohort, Cohort Term, Academic Year, Institution ID Type, Institution ID, SSN, ITIN, Student ID, First Name,
     Middle Name, Last Name, Suffix, Street Line 1, Street Line 2, City, State, Zip/Postal Code, Country, Date of Birth,
     Applied Aid, Depend, Income, Tuition, Type of Housing, Room Charges, Board Charges, Books, Other Expense, EFC,
     Marital Status, Number of Dependents, SEOG, TEACH, Veteran and Military, Other Federal Grant, Pell Amount,
     State Grant Need Based, State Grant Non Need Based, Institution Grant Need Based,
     Institution Grant Employer Aid, Institution Grant Merit, Institution Grant Military or Veteran,
     Other Institution Grant Non Need Based, Other Grant, Federal Loan, State Loan, Institution Loan, Parent PLUS,
     Other Loan, Federal Work, State Work, Inst Work, Other Aid')
; Quit;

Proc SQL; /*Financial Aid Body*/
 Insert into Financial_Aid_Output
  Select strip(CH1)||", "||
    Strip(Cohort)||", "||
    Strip(Cohort_Term)||", "||
    Strip(Academic_Year)||", "||
    Strip(Institution_ID_Type)||", "||
    Strip(Institution_ID)||", "||
    Strip(SSN)||", "||
    Strip(ITIN)||", "||
    Strip(Student_ID)||", "||
    Strip(FIRST_NAME)||", "||
    Strip(MIDDLE_NAME)||", "||
    Strip(LAST_NAME)||", "||
    Strip(NAME_SUFFIX)||", "||
    Strip(STREET_LINE1)||", "||
    Strip(STREET_LINE2)||", "||
    Strip(City)||", "||
    Strip(State)||", "||
    Strip(POSTAL_CODE)||", "||
    Strip(Country)||", "||
    Strip(BirthDate)||", "||
    Strip(Applied_Aid)||", "||
```

```
Strip(DEPENDENCY_INDEPEND)||", "||
Strip(ADJUSTED_GROSS_INCOME)||", "||
Strip(total_Tutition)||", "||
Strip(Housing)||", "||
Strip(Room)||", "||
Strip(Board)||", "||
Strip(Books)||", "||
Strip(Other_exp)||", "||
Strip(PELL_EFC)||", "||
Strip(Marital_Status)||", "||
Strip(NoDepdents)||", "||
Strip(SEOG)||", "||
Strip(TEACH)||", "||
Strip(Military)||", "||
Strip(FGO)||", "||
Strip(Pell)||", "||
Strip(SGN)||", "||
Strip(SGM)||", "||
Strip(IGN)||", "||
Strip(INST_Grant_Employ)||", "||
Strip(IGM)||", "||
Strip(INST_Grant_Mil)||", "||
Strip(INST_Grant_Other)||", "||
Strip(Grant_Other)||", "||
Strip(FLO)||", "||
Strip(SL)||", "||
Strip(IL)||", "||
Strip(PLUS)||", "||
Strip(EL)||", "||
    Strip(FW)||", "||
    Strip(SW)||", "||
    Strip(IW)||", "||
    Strip(Other)

     as Output
  From Financial_Aid_Data_File;
Quit;

Proc SQL; /*Financial Aid Footer*/
 Insert into Financial_Aid_Output
  Select "T1, "||strip(Put(Count(*)+3, 8.))||", " as Output
  From Financial_Aid_Data_File;
Quit;

Proc Export data = Financial_Aid_Output outfile = "&Path\i_04229500PDP_Texas_AM_University_Central_Texas_Financial_Aid_&OUTPUT_Term..Txt" label DBMS = Tab Replace; putnames = No;
Run;

/*Remove any blank lines from Task list*/
Proc SQL; Create Table Tasklist as Select * From Tasklist Where Report is not missing; Quit;

/*Export Export Tasklist into Spreadsheet for review*/
Proc export data=Tasklist dbms=xlsx outfile="&Path\Ouput data for &OUTPUT_Term..XLSX" replace;
 Sheet = "Task List";
run;

Proc SQL;
 Create Table Courses_test as
```

```
  Select Distinct
    stu_subj,
    stu_crse,
    Substr(STU_CRN, 1, 5) as STU_CRN,
    Catx("-", stu_subj, stu_crse, Substr(STU_CRN, 1, 5)) as Course,
    COunt(Stu_CBMID) as Class_Count
  From EDW.CBM00S
  Where &Report_Terms
  Group by stu_subj, stu_crse, STU_CRN, Course
  Order by stu_subj, stu_crse, STU_CRN, Course
 ; Quit;

/*Export Export Tasklist into Spreadsheet for review*/
Proc export data = Courses_test dbms=xlsx outfile="&Path\Ouput data for &OUTPUT_Term..XLSX" replace; Sheet = "CBM00S Course Compare"; run;

Proc SQL; Create Table Cohort_Test as Select Distinct ID From EDW.CBM0E1 Where STU_CLASS in ('1', '2', '3', '4', '5') and STU_TRNS_FTIC NE . and &Report_terms ; Quit;

/*Export Export Tasklist into Spreadsheet for review*/
Proc export data = Cohort_Test dbms=xlsx outfile="&Path\Ouput data for &OUTPUT_Term..XLSX" replace; Sheet = "CBM0E1 ID Compare"; run;
```