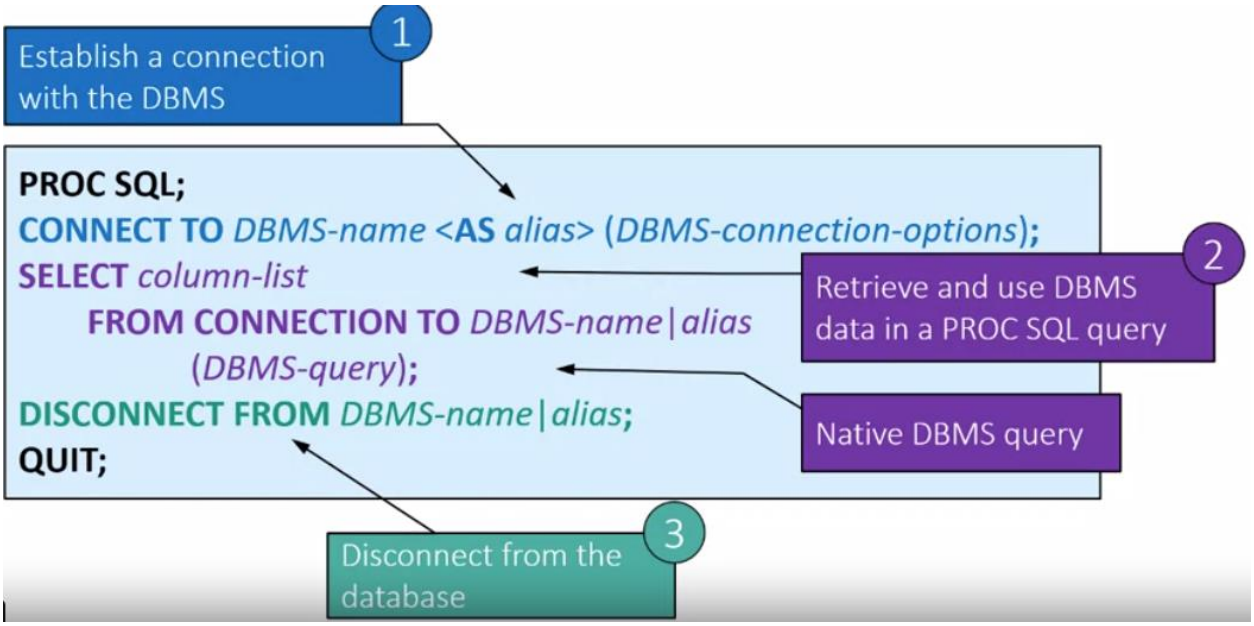
SAS Advanced Programmer

SAP1 W6 SAS/ACCESS Technology and SQL Pass-Through Facility

/* Defines the path to your data and assigns the libref. */

%let path=~/ESQ1M6;

libname sq "&path/data";




```
proc sql;
connect to oracle(user=sas_user pw=sastest
                  path=localhost);
select UserID, Income format=dollar16., State
    from connection to oracle
       (select UserID, Income, State
            from customer
            where Income is not null
            order by Income desc
            fetch first 10 rows only);
disconnect from oracle;
quit;
```

```
proc sql;
connect to oracle(user=sas_user pw=sastest
                path=localhost);
create table work.totalcustomer as
select UserID, Income format=dollar16., State
   from connection to oracle
   ...query...

*************************************************************;
*  Using an SQL Pass-Through Query                          *;
*************************************************************;
*  Syntax                                                   *;
*                                                           *;
*  PROC SQL;                                                *;
*    CONNECT TO DBMS-name <AS alias>                        *;
*             (DBMS-connection-options);                    *;
*    SELECT col-name, col-name                              *;
*       FROM CONNECTION TO DBMS-name|alias                  *;
*        (DBMS-query);                                      *;
*    DISCONNECT FROM DBMS-name|alias;                       *;
*  QUIT;                                                    *;
*************************************************************;

*************************************************************;
*  Demo                                                     *;
*  1) Open the s107d01.sas program in the demos folder      *;
*    and find the Demo section. If you have not already     *;
*    done so, run the libname.sas program to define the     *;
*    PATH macro variable.                                   *;
```

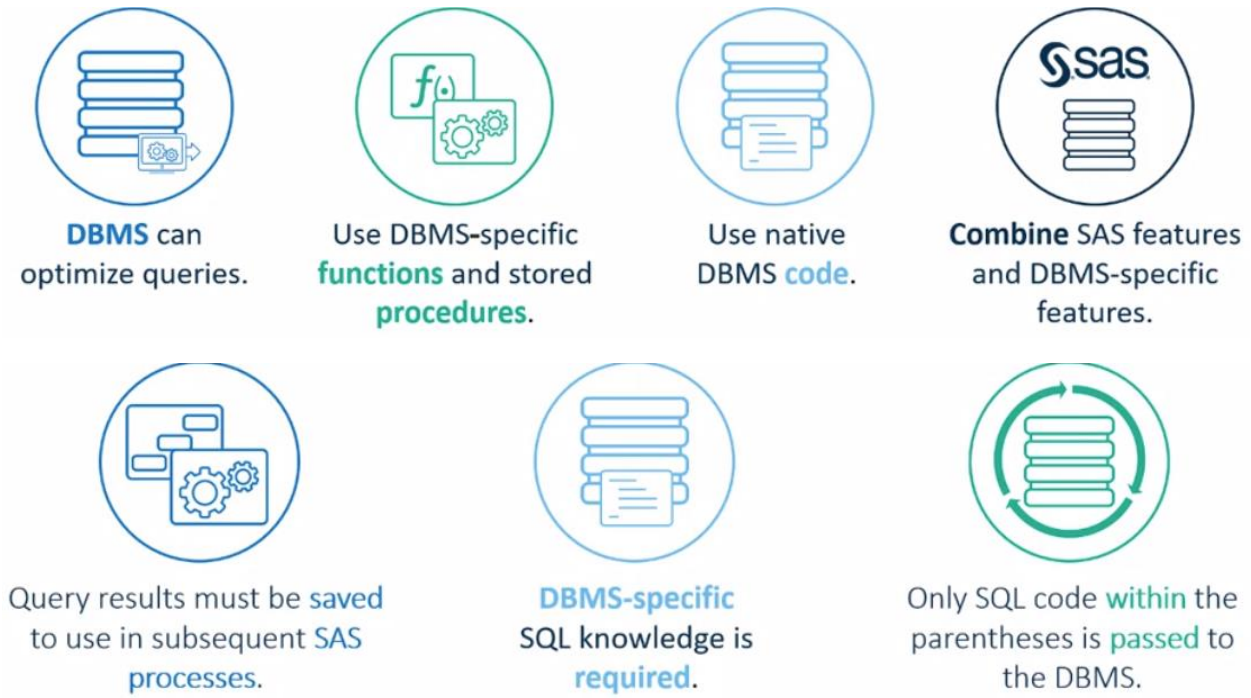
```
*  2) Run the Microsoft Access query. Discuss the syntax  *;
*     error.                                  *;
*  3) Add a CONNECT TO PCFILES statement above the SELECT *;
*     statement. After PCFILES, add the PATH= and        *;
*     DBPASSWORD=SASTEST options in parentheses. End the  *;
*     statement with a semicolon.                        *;
*  4) Add the SELECT statement after the CONNECT TO      *;
*     statement, and select all columns using an         *;
*     asterisk. Add the FROM CONNECTION TO component and  *;
*     reference the pcfiles database. Enclose the        *;
*     original Microsoft Access query in parentheses.     *;
*  5) Add the DISCONNECT FROM statement to disconnect     *;
*     from the pcfiles database.                         *;
*  6) Run the SQL pass-through query and view the        *;
*     results.                                  *;
*  7) In the SELECT statement, remove the asterisk and    *;
*     add the columns UserID, Income, and State. Format   *;
*     the Income column using the DOLLAR16. format. Run   *;
*     the query and view the results.                    *;
************************************************************;

************************************************************;
* Run the following native Microsoft Access Query        *;
************************************************************;
proc sql;
connect to pcfiles(path="&path/database/SQL_DB.accdb" dbpassword=sastest);
select * from connection to pcfiles
        (select top 10, UserID, Income, State
        from customer
```

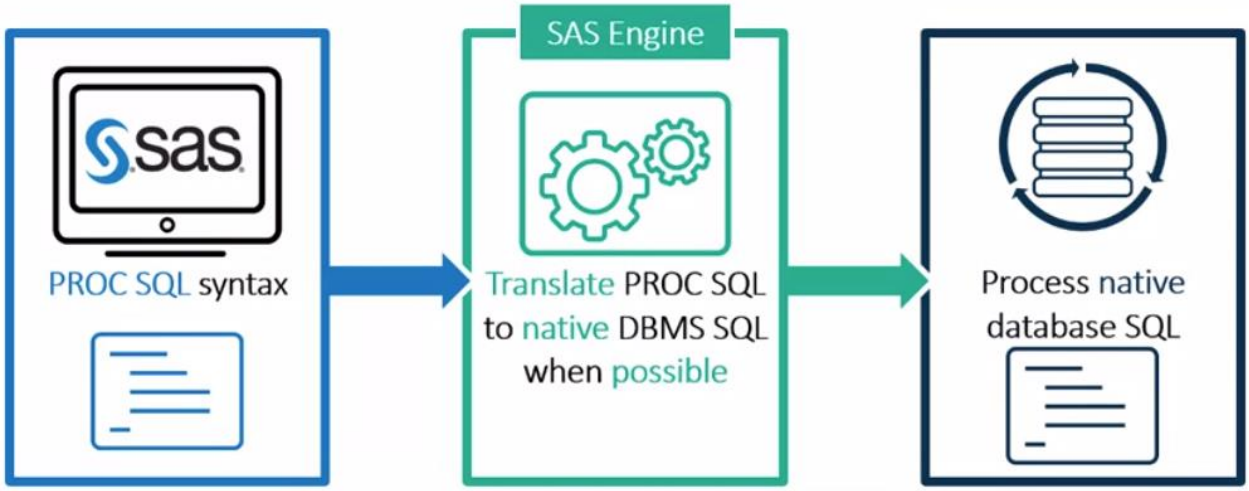
```
        order by Income desc);
disconnect from pcfiles;
quit;
```

UserID	Income	State
marremartinez6531@ismissing.com	229306	IL
kimlihuffman843@fakeemail.com	228165	NY
heacamoredock827@invalid.com	220092.2	NY
heljaboone8613@invalid.com	188953.8	NY
jambeerskin7521@fakeemail.com	172372.7	LA
aranihale6320@ismissing.com	169312.6	CA
terlowhite687@invalid.com	165776.3	FL
barmablanton521@n/a.com	150512.8	CT
wilhoflores5912@notreal.com	137314.9	CA
normidameron9028@voidemail.com	127477.6	AZ

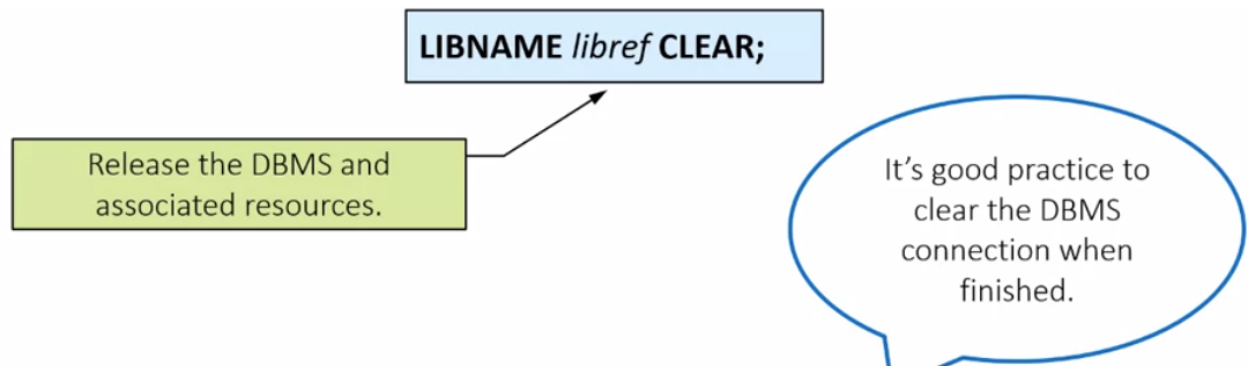
```
proc sql;
connect to pcfiles(path="&path/database/SQL_DB.accdb" dbpassword=sastest);
select UserID, Income format=dollar16., State
        from connection to pcfiles
        (select top 10, UserID, Income, State
        from customer
        order by Income desc);
disconnect from pcfiles;
quit;
```

UserID	Income	State
marremartinez6531@ismissing.com	$229,306	IL
kimlihuffman843@fakeemail.com	$228,165	NY
heacamoredock827@invalid.com	$220,092	NY
heljaboone8613@invalid.com	$188,954	NY
jambeerskin7521@fakeemail.com	$172,373	LA
aranihale6320@ismissing.com	$169,313	CA
terlowhite687@invalid.com	$165,776	FL
barmablanton521@n/a.com	$150,513	CT
wilhoflores5912@notreal.com	$137,315	CA
normidameron9028@voidemail.com	$127,478	AZ





LIBNAME *libref engine* *<SAS/ACCESS-engine-options>* **;**




```
************************************************************;
*  Using SAS/ACCESS LIBNAME Statement                      *;
************************************************************;
*  Syntax                                                  *;
*                                                          *;
*  LIBNAME libref engine <SAS/ACCESS-engine-options>;      *;
*                                                          *;
*  LIBNAME libref CLEAR;                                   *;
************************************************************;


************************************************************;
*  Demo                                                    *;
*  1) Open the s107d02.sas program in the demos folder     *;
*     and find the Demo section. If you have not already   *;
*     done so, run the libname.sas program to define the   *;
*     PATH macro variable.                                 *;
*  2) Begin by viewing all available libraries in your     *;
*     current session.                                     *;
*     Note: Your libraries might differ. Notice that       *;
*           there is no library named db.                  *;
*  3) Complete the LIBNAME statement to define a library   *;
*     named db that uses the PCFILES engine. Add the       *;
```

```
*    PATH= option to connect to the SQL_DB.accdb        *;
*    Microsoft Access database, and add the DBPASSWORD=  *;
*    option using the password sastest. Highlight the    *;
*    LIBNAME statement and run the selected code. Use    *;
*    the navigation pane to expand the db library.       *;
*    Note: In SAS Enterprise Guide, click Libraries and  *;
*         select Refresh to update the library list.    *;
*  4) Review the query below the LIBNAME statement. Add   *;
*    the library db to the beginning of the customer     *;
*    table name, and apply the DOLLAR16. format to the   *;
*    Income column.                                      *;
*  5) Add a statement to clear the db library. Highlight  *;
*    the entire demo program and run the selected code.  *;
************************************************************;

libname db pcfiles path="&path/database/SQL_DB.accdb" dbpassword=sastest;

proc sql outobs=10;
select UserID, Income, State
   from db.customer
   order by Income desc;
quit;

libname db clear;
```

UserID	Income	State
marremartinez6531@ismissing.com	229306	IL
kimlihuffman843@fakeemail.com	228165	NY
heacamoredock827@invalid.com	220092.2	NY
heljaboone8613@invalid.com	188953.8	NY
jambeerskin7521@fakeemail.com	172372.7	LA
aranihale6320@ismissing.com	169312.6	CA
terlowhite687@invalid.com	165776.3	FL
barmablanton521@n/a.com	150512.8	CT
wilhoflores5912@notreal.com	137314.9	CA
normidameron9028@voidemail.com	127477.6	AZ

```
PROC FEDSQL <options>;

SELECT col-name, col-name
      FROM input-table
      <WHERE clause>
      <GROUP BY clause>
      <HAVING clause>
      <ORDER BY clause>;

QUIT;
```

The foundation of the PROC FedSQL syntax is similar to PROC SQL.

```
LIBNAME libref engine <SAS/ACCESS-engine-options>;

PROC FEDSQL <options>;
    SELECT col-name, col-name
    FROM libref.table;
QUIT;
```

PROC FEDSQL needs the correct DBMS information through the SAS LIBNAME statement.

```
libname mkt oracle user=sas_user password=sastest
                   path=localhost;

proc fedsql;
select State,
       count(*) as TotalCustomer
    from mkt.customer
    where CreditScore > 700
    group by State
    order by TotalCustomer desc;
quit;

libname mkt clear;
```

Reference the table.

```
*********************************************************;
*  Activity 7.03                                        *;
*  1) Examine and run the query. Did it produce an error? *;
*  2) In the WHERE clause, replace the double quotation   *;
*     marks around NC with single quotation marks. Run    *;
*     the query. Did it run successfully?                 *;
*********************************************************;


proc fedsql;
select UserID, Income, State
        from sq.customer
        where State='NC'
        order by Income desc
        limit 10;
quit;
```

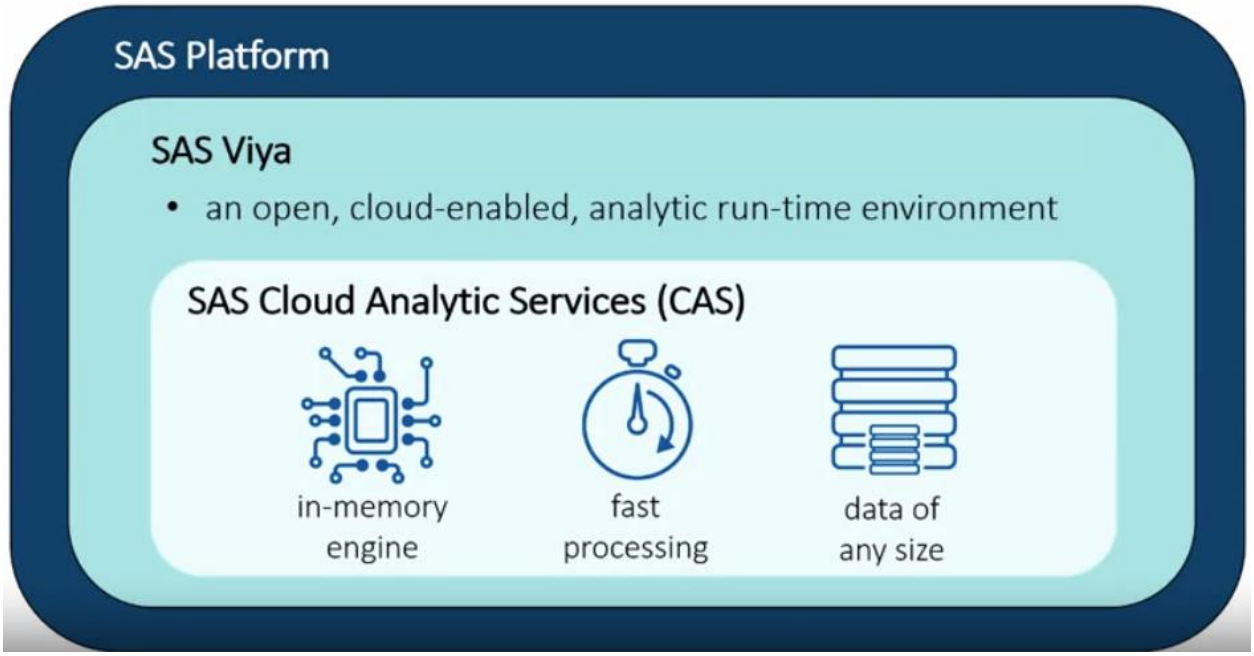
User ID	Income	State
monmagarcia7329@notreal.com	97905.92	NC
tonedgorman8119@notreal.com	94609.29	NC
johcakennon598@notreal.com	94249.01	NC
tyrjawolfe4919@isnull.com	93972.49	NC
donbeparker9115@notreal.com	91878.72	NC
alimiburchard6027@fakeemail.com	91875.87	NC
kenshlangill6215@invalid.com	91687.09	NC
rebbrgonyea9720@invalid.com	90823.13	NC
erirogilton8617@invalid.com	90533.26	NC
ellaaledy9626@voidemail.com	90188.78	NC

PROC FEDSQL IPTRACE;

```
libname mkt oracle user=s
                    path=

proc fedsql iptrace;
select State,
       count(*) as TotalCustomer
    from mkt.customer
    where CreditScore > 700
    group by State
    order by TotalCustomer desc;
quit;

libname mkt clear;
```

The IPTRACE option prints a text description of the PROC FEDSQL query plan.

```
IPTRACE: Query:
select State, count(*) as TotalCustomer from
IPTRACE: FULL pushdown to ORACLE SUCCESS!
IPTRACE: Retextualized child query:
select "SAS_USER"."CUSTOMER"."STATE", COUNT (
("SAS_USER"."CUSTOMER"."CREDITSCORE">700) )
IPTRACE: END
```



PROC SQL Versus PROC FEDSQL

The following table highlights some of the main differences between PROC SQL and PROC FEDSQL.

PROC SQL	**PROC FEDSQL**
SAS SQL implementation	Vendor neutral ANSI SQL
Follows ANSI Standard 2	ANSI Standard 3 compliant
Limited to SAS data types	Processes 17 ANSI data types
Is multi-threaded for sorting and indexing on the SAS Platform	Is fully multi-threaded on the SAS Platform
Includes many non-ANSI standard SAS enhancements	Includes very few non-ANSI SAS enhancements
Does not execute in CAS	Executes in CAS

```
*****************************************************************************;
*                    BEGIN THE SQL CASE STUDY                    *;
*****************************************************************************;
* NOTE: Be sure to first run casestudy_createdata.sas to create the data for *;
*      the case study. Data is created in the SQ library.              *;
* NOTE: The final deliverable tables Claims_Cleaned and ClaimsByAirport must *;
*      reside in the SQ library for the AnalysisProgram.sas to create the   *;
*      FinalReport.html output.                                  *;
*****************************************************************************;
*************************************************************;
* 1) If necessary, run your libname.sas program to define   *;
*    define the SQ library.                                                 *;
* 2) Run the program to generate the following tables      *;
*    to be used in the case study:                    *;
*        1. sq.claimsraw                        *;
*        2. sq.boarding2013_2016                    *;
*        3. sq.enplanement2017                    *;
* 3) After the data has been created, open              *;
*    StarterProgram.sas in the caseStudy folder and begin.  *;
*************************************************************;
/*Case Study: Guided Version
This suggested guide uses the steps of the SAS programming process to help you solve the
business problem. This is only one of many ways that you could solve this problem.
Begin the case study by opening and running the StarterProgram.sas.
Access Data
1. The tables are created in the Sq library after you run the casestudy_createdata.sas program.
Explore Data
2. Preview the first 10 rows and the descriptor portion of the following tables:
a. sq.claimsraw table
```

b. sq.enplanement2017 and sq.boarding2013_2016 tables

```
*/

proc sql outobs=10;

select * from sq.claimsraw;

select * from sq.enplanement2017;

select * from sq.boarding2013_2016;

quit;
```

Claim_Number	Date_Received	Incident_Date	Airport_Code	Airport_Name	Claim_Type	Claim_Site	Close_Amount	Disposition	StateName	State	County	City
2013042303294	23APR2013	04APR2013			Property Damage	Checked Baggage	.					
2013031302565	13MAR2013	13MAR2013					.					
2013031902651	11MAR2013	21FEB2013			Property Damage	Checked Baggage	.					
2013041503167	27MAR2013	08MAR2013			Property Damage	Checked Baggage	.					
2013040402988	22MAR2013	08FEB2013				Checked Baggage	.					
2013040202938	19MAR2013	05MAR2013			Passenger Property Loss	Checked Baggage	.					
2013061904529	09MAY2013	11MAR2013			Passenger Property Loss	Checked Baggage	.					
2013043003398	30APR2013	30APR2013			Property Damage	Checkpoint	.					
2013061804491	07MAY2013	18MAR2013			Property Damage	Checked Baggage	.					
2013040803043	08APR2013	08APR2013					.					

LocID	Enplanement	Year
ATL	50,251,964	2017
LAX	41,232,432	2017
ORD	38,593,028	2017
DFW	31,816,933	2017
DEN	29,809,097	2017
JFK	29,533,154	2017
SFO	26,900,048	2017
LAS	23,364,393	2017
SEA	22,639,124	2017
CLT	22,011,251	2017

LocID	Year	Boarding
ATL	2016	50501858
LAX	2016	39636042
ORD	2016	37589899
DFW	2016	31283579
JFK	2016	29239151
DEN	2016	28267394
SFO	2016	25707101
LAS	2016	22833267
SEA	2016	21887110
CLT	2016	21511880

```
proc sql;

describe table sq.claimsraw;

describe table sq.enplanement2017;

describe table sq.boarding2013_2016;

quit;
```

```
1          OPTIONS NONOTES NOSTIMER NOSOURCE NOSYNTAXCHECK;
 72
 73         proc sql;
 74         describe table sq.claimsraw;
 NOTE: SQL table SQ.CLAIMSRAW was created like:

 create table SQ.CLAIMSRAW( bufsize=131072 )
   (
    Claim_Number char(13) format=$13.,
    Date_Received num format=DATE9.,
    Incident_Date num format=DATE9.,
    Airport_Code char(3) format=$3.,
    Airport_Name char(48) format=$48.,
    Claim_Type char(39) format=$39.,
    Claim_Site char(15) format=$15.,
    Close_Amount num format=BEST12.,
    Disposition char(23) format=$23.,
    StateName char(17) format=$17.,
    State char(2) format=$2.,
    County char(20) format=$20.,
    City char(33) format=$33.
   );

 75         describe table sq.enplanement2017;
 NOTE: SQL table SQ.ENPLANEMENT2017 was created like:

 create table SQ.ENPLANEMENT2017( bufsize=131072 )
   (
    LocID char(14) format=$3.,
    Enplanement num format=COMMA15.,
    Year char(4)
   );

 76         describe table sq.boarding2013_2016;
 NOTE: SQL table SQ.BOARDING2013_2016 was created like:

 create table SQ.BOARDING2013_2016( bufsize=131072 )
   (
    LocID char(14) format=$3.,
    Year num,
    Boarding num
   );

 77         quit;
```

```
NOTE: PROCEDURE SQL used (Total process time):
      real time           0.00 seconds
      user cpu time       0.00 seconds
      system cpu time     0.01 seconds
      memory              1849.43k
      OS Memory           45704.00k
      Timestamp           05/27/2021 05:50:17 AM
      Step Count                        55  Switch Count  0
      Page Faults                       0
      Page Reclaims                     342
      Page Swaps                        0
      Voluntary Context Switches        15
      Involuntary Context Switches      0
      Block Input Operations            0
      Block Output Operations           16


78
79
80         OPTIONS NONOTES NOSTIMER NOSOURCE NOSYNTAXCHECK;
92
```

/*1) What type is the Year column in each table? Char4 in sq.enplanement2017 and num in sq.boarding2013_2016

2) What is the column name that holds the value of the number of passengers that boarded

a plane in each table? enplanement in sq.enplanement2017 and boarding in sq.boarding2013_2016

*/

proc freq data=sq.claimsraw;

tables airport_code claim_site disposition claim_type date_received incident_date;

run;

/*3. Count the number of nonmissing values in the entire table and in the following columns:

a. Airport_Code (missing: 349)

b. Claim_Site (missing: 233)

c. Disposition (missing: 9059)

d. Claim_Type (missing: 225)

e. Date_Received (no missing)

f. Incident_Date (no missing)

Results

*/

YUM	26	0.06	42029	99.64
ZZX	141	0.33	42170	99.98
ZZZ	9	0.02	42179	100.00
Frequency Missing = 349				

Claim_Site	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Bus Station	10	0.02	10	0.02
Checked Baggage	31156	73.66	31166	73.69
Checkpoint	10826	25.60	41992	99.28
Motor Vehicle	184	0.44	42176	99.72
Not Provided	1	0.00	42177	99.72
Other	110	0.26	42287	99.98
Pre-Check	8	0.02	42295	100.00
Frequency Missing = 233				

Disposition	Frequency	Percent	Cumulative Frequency	Cumulative Percent
*Insufficient	1733	5.18	1733	5.18
Approve in Full	6908	20.64	8641	25.82
Closed: Canceled	167	0.50	8808	26.32
Closed:Canceled	284	0.85	9092	27.17
Closed:Contractor Claim	115	0.34	9207	27.51
Deny	12213	36.49	21420	64.00
In Review	8938	26.71	30358	90.70
Pending Payment	1	0.00	30359	90.71
Received	14	0.04	30373	90.75
Settle	3023	9.03	33396	99.78
losed: Contractor Claim	73	0.22	33469	100.00
Frequency Missing = 9059				

Claim_Type	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Complaint	71	0.17	71	0.17
Compliment	3	0.01	74	0.17
Employee Loss (MPCECA)	23	0.05	97	0.23
Missed Flight	32	0.08	129	0.30
Motor Vehicle	140	0.33	269	0.64
Not Provided	2	0.00	271	0.64
Passenger Property Loss	23580	55.74	23851	56.38
Passenger Property Loss/Personal Injur	8	0.02	23859	56.40
Passenger Property Loss/Personal Injury	13	0.03	23872	56.43
Passenger Theft	14	0.03	23886	56.46
Personal Injury	413	0.98	24299	57.44
Property Damage	17973	42.49	42272	99.93
Property Damage/Personal Injury	14	0.03	42286	99.96
Property Loss	17	0.04	42303	100.00
Frequency Missing = 225				

```
proc sql;
select
    count(case when airport_code = '' then 1 end) as missing_airport_code,
    count(case when claim_site = '' then 1 end) as missing_claim_site,
    count(case when disposition = '' then 1 end) as missing_disposition,
    count(case when claim_type = '' then 1 end) as missing_claim_type,
    count(case when date_received = . then 1 end) as missing_date_received,
    count(case when incident_date = . then 1 end) as missing_incident_date,
    count(*) as Total_records,
    (calculated missing_airport_code / calculated total_records) as missing_airport_code_percent
format=percent7.2,
    (calculated missing_claim_site / calculated total_records) as missing_claim_site_percent
format=percent7.2,
    (calculated missing_disposition / calculated total_records) as missing_disposition_percent
format=percent7.2,
    (calculated missing_claim_type / calculated total_records) as missing_claim_type_percent
format=percent7.2,
```

```
        (calculated missing_date_received / calculated total_records) as
missing_date_received_percent format=percent7.2,

        (calculated missing_incident_date / calculated total_records) as missing_incident_date_percent
format=percent7.2

        from sq.claimsraw;

run;
```

missing_airport_code	missing_claim_site	missing_disposition	missing_claim_type	missing_date_received	missing_incident_date	Total_records
349	233	9059	225	0	0	42528

missing_airport_code_percent	missing_claim_site_percent	missing_disposition_percent	missing_claim_type_percent	missing_date_received_percent	missing_incident_date_percent
0.82%	0.55%	21.3%	0.53%	0.00%	0.00%

/*4. In one query, find the percentage of missing values in the following columns:

a. Airport_Code (0.82%)

b. Claim_Site (0.55%)

c. Disposition (21.3%)

d. Claim_Type (0.53%)

e. Date_Received (0%)

f. Incident_Date (0%)

Copyright © 2020, SAS Institute Inc., Cary, North Carolina, USA. ALL RIGHTS RESERVED.

Results

*/

```
proc freq data=sq.claimsraw;

        tables claim_site disposition claim_type date_received incident_date / nocum nopercent;

        format date_received incident_date year4.;

run;
```

The FREQ Procedure

Claim_Site	Frequency
Bus Station	10
Checked Baggage	31156
Checkpoint	10826
Motor Vehicle	184
Not Provided	1
Other	110
Pre-Check	8
Frequency Missing = 233	

Claim_Type	Frequency
Complaint	71
Compliment	3
Employee Loss (MPCECA)	23
Missed Flight	32
Motor Vehicle	140
Not Provided	2
Passenger Property Loss	23580
Passenger Property Loss/Personal Injur	8
Passenger Property Loss/Personal Injury	13
Passenger Theft	14
Personal Injury	413
Property Damage	17973
Property Damage/Personal Injury	14
Property Loss	17
Frequency Missing = 225	

Disposition	Frequency
*Insufficient	1733
Approve in Full	6908
Closed: Canceled	167
Closed:Canceled	284
Closed:Contractor Claim	115
Deny	12213
In Review	8938
Pending Payment	1
Received	14
Settle	3023
losed: Contractor Claim	73
Frequency Missing = 9059	

Date_Received	Frequency
2013	8476
2014	8798
2015	8663
2016	7973
2017	8618

Incident_Date	Frequency
2013	9536
2014	8680
2015	7721
2016	8186
2017	8403
2018	2

/*5. Explore the distinct values of the following columns to determine whether any adjustments are needed. Use the required column values in the Case Study Data Layout PDF.

a. Claim_Site

b. Disposition

c. Claim_Type

d. The year from Date_Received

(Hint: Use the PUT function.)

e. The year from Incident_Date

(Hint: Use the PUT function.)

*/

```
proc sql;
select count(*) as Quantity
        from sq.claimsraw
        where incident_date > date_received;
quit;
```

/*6. Count the number of rows in which Incident_Date occurs after Date_Received.

Results

*/

Quantity
65

```
proc sql;
select Claim_number, date_received, Incident_date
        from sq.claimsraw
        where incident_date > date_received;
quit;
```

/*7. Run a query to view the Claim_Number, Date_Received, and Incident_Date columns in the sq.claimsraw table in which Incident_Date occurs after Date_Received.

a. What assumption can you make about the Date_Received column values in your results?

The year on date_received should be the following year.

*/

Claim_Number	Date_Received	Incident_Date
2018011146749	02JAN2017	10DEC2017
2018011646813	05JAN2017	12DEC2017
2018011046737	02JAN2017	06DEC2017
2018011146748	02JAN2017	08DEC2017
2018010346412	03JAN2017	01NOV2017
2018011646797	05JAN2017	19DEC2017
2018010346402	03JAN2017	16NOV2017
2018010946655	09JAN2017	31DEC2017
2018010946640	02JAN2017	23DEC2017
2018010946664	04JAN2017	19NOV2017
2018011646808	05JAN2017	03DEC2017
2018011146746	02JAN2017	28DEC2017
2018011646818	05JAN2017	16DEC2017
2018010446435	04JAN2017	13DEC2017
2018011646825	16JAN2017	18DEC2017
2018010546553	08JAN2017	04SEP2017
2018011146758	02JAN2017	31DEC2017
2018010946636	02JAN2017	13NOV2017
2018011646804	05JAN2017	27DEC2017
2018011646815	08JAN2017	07DEC2017
2018011246771	05JAN2017	19DEC2017
2018011246792	05JAN2017	16DEC2017
2018010446479	04JAN2017	08DEC2017
2018010546528	05JAN2017	19DEC2017
2018011146752	02JAN2017	30NOV2017
2018010946652	02JAN2017	28DEC2017
2018011246767	12JAN2017	25DEC2017
2018010946619	09JAN2017	16NOV2017
2018010946583	09JAN2017	28NOV2017
2018010946679	08JAN2017	05DEC2017
2018010946647	02JAN2017	22NOV2017
2018011146762	11JAN2017	29OCT2017
2018010446437	04JAN2017	07DEC2017
2018011246766	05JAN2017	03SEP2017
2018011046725	10JAN2017	10DEC2017
2018011646806	16JAN2017	30NOV2017
2018011246765	05JAN2017	14DEC2017
2018011146755	02JAN2017	20NOV2017
2018010446519	04JAN2017	01DEC2017
2018010246388	02JAN2017	21NOV2017
2018011146745	02JAN2017	28NOV2017
2018010946642	02JAN2017	27DEC2017
2018011146760	02JAN2017	09DEC2017
2018011646810	16JAN2017	30DEC2017
2018011246773	05JAN2017	03JAN2018
2018011246794	12JAN2017	02DEC2017
2018011046729	10JAN2017	10JAN2018
2018011646801	16JAN2017	23DEC2017
2018010946645	02JAN2017	26DEC2017
2018011246768	05JAN2017	29MAY2017
2018010446445	04JAN2017	13DEC2017
2018011246770	12JAN2017	13OCT2017
2018010546552	05JAN2017	05OCT2017
2018010946631	02JAN2017	30DEC2017
2018011046722	10JAN2017	05DEC2017
2018011146764	05JAN2017	27DEC2017
2018011646811	16JAN2017	16DEC2017
2018011246790	12JAN2017	25NOV2017
2018010446423	04JAN2017	28NOV2017
2018010946686	09JAN2017	09DEC2017
2018010346403	03JAN2017	23NOV2017
2018010946628	09JAN2017	27NOV2017
2018010946632	09JAN2017	18NOV2017
2018011146754	02JAN2017	17OCT2017
2018010946586	09JAN2017	30NOV2017

/*Prepare Data

Use the information from the Explore stage to begin preparing the data for analysis.

8. Create a new table named Claims_NoDup that removes entirely duplicated rows. A duplicate claim exists if every value is duplicated.

Log

```
*/

proc sort data=sq.claimsraw

                 out=sq.Claims_NoDup nodup nodupkey;

                 by _all_;

run;
```

```
73          proc sort data=sq.claimsraw
74            out=sq.Claims_NoDup nodup nodupkey;
75            by _all_;
76          run;

NOTE: There were 42528 observations read from the data set SQ.CLAIMSRAW.
NOTE: 4 observations with duplicate key values were deleted.
NOTE: The data set SQ.CLAIMS_NODUP has 42524 observations and 13 variables.
```

/*9. Using the Claims_NoDup table, create a table named sq.Claims_Cleaned by doing the following:

a. Select the Claim_Number and Incident Date columns.

b. Fix the 65 date issues that you identified earlier by replacing the year 2017 with 2018 in the Date_Received column. (Hint: One method is by using the INTNX function.)

c. Select the Airport_Name column.

d. Replace missing values in the Airport_Code column with the value Unknown.

e. Clean the following columns by applying the requirements for the values in the Case Study Data Layout PDF:

1) Claim_Type

2) Claim_Site

3) Disposition

f. Select the Close_Amount column and format it with a dollar sign. Include two decimal places (for example, $130.28).

g. Select the State column and convert all values to uppercase.

h. Select the StateName, County, and City columns. Convert all values to proper case (for example, Raleigh).

i. Include only those rows where Incident_Date is between 2013 and 2017.

j. Order the results by Airport_Code and Incident_Date.

k. Assign permanent labels for columns by adding a space between words (for example, Close Amount).

```
*/
data sq.Claims_Cleaned;
	length Airport_Code $ 7;
	set sq.Claims_NoDup;
/* 9b. Fix 65 date issues */
	if (Incident_Date > Date_Received) then Date_Received = intnx('year', date_received, 1);
/* 9d. Clean the Airport_Code column */
	if Airport_Code in ('-','') then Airport_Code="Unknown";
/* 9e1. Clean the Claim_Type column */
	if Claim_Type in ('-','') then Claim_Type="Unknown";
	else if Claim_Type="Passenger Property Loss/Personal Injur" then Claim_Type="Passenger
Property Loss";
	else if Claim_Type="Passenger Property Loss/Personal Injury" then Claim_Type="Passenger
Property Loss";
	else if Claim_Type="Property Damage/Personal Injury" then Claim_Type="Property Damage";
/* 9e2. Clean the Claim_Site column */
	if Claim_Site in ('-','') then Claim_Site="Unknown";
/* 9e3. Clean the Disposition column */
	if Disposition in ('-','') then Disposition="Unknown";
	else if Disposition="losed: Contractor Claim" then Disposition="Closed:Contractor Claim";
	else if Disposition="Closed: Canceled" then Disposition="Closed:Canceled";
/* 9f. Format Close_Amount column with a dollar sign that includes two decimal places */
	format Incident_Date Date_Received Date9. Close_Amount dollar20.2;
/* 9gh. Convert All State values to Uppercase and all State_Name, County and City to proper case */
	State=upcase(state);
	StateName=propcase(StateName);
	County=propcase(County);
```

```
        City=propcase(City);
/* 9i. Include only those rows where Incident_Date is between 2013 and 2017. */
        where Year(Incident_date) between 2013 and 2017;
/* 9k. Add permanent labels */
        label Airport_Code="Airport Code"
                Airport_Name="Airport Name"
                Claim_Number="Claim Number"
                Claim_Site="Claim Site"
                Claim_Type="Claim Type"
                Close_Amount="Close Amount"
                Date_Issues="Date Issues"
                Date_Received="Date Received"
                Incident_Date="Incident Date"
                StateName="State Name";
/* 10. Drop County and City */
*       drop County City;
run;
```

```
NOTE: Variable Date_Issues is uninitialized.
NOTE: There were 42522 observations read from the data set SQ.CLAIMS_NODUP.
      WHERE (YEAR(Incident_date)>=2013 and YEAR(Incident_date)<=2017);
NOTE: The data set SQ.CLAIMS_CLEANED has 42522 observations and 13 variables.
```

```
proc sql;
select Claim_number, date_received, Incident_date
        from sq.Claims_Cleaned
        where incident_date > date_received;
quit;
```

```
73          proc sql;
74          select Claim_number, date_received, Incident_date
75          from sq.Claims_Cleaned
76          where incident_date > date_received;
NOTE: No rows were selected.
77          quit;
```

```
proc freq data=sq.Claims_Cleaned order=freq;
        tables Claim_Site
                Disposition
                Claim_Type
                Airport_Code / nocum nopercent;
run;
```

The FREQ Procedure

Claim Site	
Claim_Site	Frequency
Checked Baggage	31155
Checkpoint	10826
Unknown	232
Motor Vehicle	180
Other	110
Bus Station	10
Pre-Check	8
Not Provided	1

Disposition	Frequency
Deny	12213
Unknown	9055
In Review	8937
Approve in Full	6908
Settle	3023
*Insufficient	1732
Closed:Canceled	451
Closed:Contractor Claim	188
Received	14
Pending Payment	1

Claim Type	
Claim_Type	Frequency
Passenger Property Loss	23599
Property Damage	17987
Personal Injury	413
Unknown	225
Motor Vehicle	136
Complaint	71
Missed Flight	32
Employee Loss (MPCECA)	23
Property Loss	17
Passenger Theft	14
Compliment	3
Not Provided	2

/*Log

Partial Table

10. Use the sq.Claims_Cleaned table to create a view named TotalClaims to count the number of claims for each value of Airport_Code and Year.

a. Include Airport_Code, Airport_Name, City, State, and the year from Incident_Date. Name the new column Year.

b. Count the number of claims for each group using the COUNT function. Name the new column TotalClaims.

c. Group by the correct columns.

d. Order the table by Airport_Code and Year.

Note: Typically, you do not use an ORDER BY clause when creating a view. For the purpose of this case study, it is used to produce a similar result image for validation.

*/

```
proc sql;
create view sq.TotalClaims as
select Airport_Code, Airport_Name, City, State, Year(Incident_Date) as Year, count(*) as TotalClaims
        from sq.Claims_Cleaned
        group by 1,2,3,4,5
        order by Airport_Code, Year;
quit;
```

Table: SQ.TOTALCLAIMS | View: Column names | Filter: (none)

Columns

- Select all
- Airport_Code
- Airport_Name
- City
- State
- Year
- TotalClaims

Total rows: 1492 Total columns: 6 Rows 1-100

	Airport_Code	Airport_Name	City	State	Year	TotalClaims
1	ABE	Lehigh Valley International Airport, Allentown	Allentown	PA	2013	9
2	ABE	Lehigh Valley International Airport, Allentown	Allentown	PA	2014	3
3	ABE	Lehigh Valley International Airport, Allentown	Allentown	PA	2015	4
4	ABE	Lehigh Valley International Airport, Allentown	Allentown	PA	2016	5
5	ABE	Lehigh Valley International Airport, Allentown	Allentown	PA	2017	3
6	ABI	Abilene Regional	Abilene	TX	2013	4
7	ABI	Abilene Regional	Abilene	TX	2014	6

/*Partial View

11. Create a view named TotalEnplanements by using the OUTER UNION set operator to concatenate the enplanement2017 and boarding2013_2016 tables.

a. From the sq.enplanement2017 table, select the LocID and Enplanement columns. Create a new column named Year by converting the character Year column to numeric.

b. Use the OUTER UNION set operator with the CORR modifier.

c. From the sq.boarding2013_2016 table, select the LocID, Boarding, and Year columns. Change the name of the Boarding column to Enplanement.

d. Order the results by Year and LocID.

```
*/
proc sql;
create view sq.TotalEnplanements as
select LocId, Enplanement, input(Year,4.) as Year
        from sq.enplanement2017
        outer union corr
select LocId, Boarding as Enplanement, Year
        from sq.boarding2013_2016
        order by Year, LocID;
quit;
```

Table: SQ.TOTALENPLANEMENTS | View: Column names

Columns

- [x] Select all
- [x] LocID
- [x] Enplanement
- [x] Year

Total rows: 2543 Total columns: 3

	LocID	Enplanement	Year
1	0AK	3,123	2013
2	16A	3,652	2013
3	1G4	140,886	2013
4	2A3	2,336	2013
5	2A9	3,622	2013
6	4A2	2,500	2013

/*Partial View

12. Create a table named sq.ClaimsByAirport by joining the TotalClaims and TotalEnplanements views.

a. Select the Airport_Code, Airport_Name, City, State, Year, TotalClaims, and Enplanement columns.

b. Create a new column to calculate the percentage of claims by enplanements by dividing Enplanement by TotalClaims. Name the column PctClaims and format it using PERCENT10.4.

c. Perform an inner join using the criterion Airport_Code=LocID and the Year columns.

d. Order the results by Airport_Code and Year.

Log

Partial Table

Hint: You can solve steps 10 through 12 in one query using inline views.

*/

```
proc sql;

create table sq.ClaimsByAirport as

select c.Airport_Code, Airport_Name, City, State, c.Year, TotalClaims, Enplanement,

        (TotalClaims / Enplanement) as PctClaims format=percent10.4

      from sq.TotalClaims as c inner join

            sq.TotalEnplanements as e

            on c.Airport_Code = e.LocID

            and c.Year = e.Year

      order by c.Airport_Code, c.Year;

quit;
```

```
proc sql;
create table sq.ClaimsByAirport as
select c.Airport_Code, Airport_Name, City, State, c.Year, TotalClaims, Enplanement,
   (TotalClaims / Enplanement) as PctClaims format=percent10.4
from sq.TotalClaims as c inner join
 sq.TotalEnplanements as e
 on c.Airport_Code = e.LocID
 and c.Year = e.Year
order by c.Airport_Code, c.Year;
NOTE: Table SQ.CLAIMSBYAIRPORT created, with 1438 rows and 8 columns.
```

```
proc sql;

create table sq.ClaimsByAirport2 as

select c.Airport_Code, Airport_Name, City, State, c.Year, TotalClaims, Enplanement,

        (TotalClaims / Enplanement) as PctClaims format=percent10.4

      from (select Airport_Code, Airport_Name, City, State, Year(Incident_Date) as Year, count(*) as
TotalClaims

                  from sq.Claims_Cleaned

                  group by 1,2,3,4,5) as c inner join
```

```
                (select LocId, Enplanement, input(Year,4.) as Year
                        from sq.enplanement2017
                        outer union corr
                 select LocId, Boarding as Enplanement, Year
                        from sq.boarding2013_2016
                        ) as e
                on c.Airport_Code = e.LocID
                and c.Year = e.Year
        order by Airport_Code, Year;
quit;

*******************************************************************************
******************;
*                        ANALYZE AND EXPORT DATA                          *;
* NOTE: Do not edit the code below. Run the program after all data preparation is complete.        *;
* NOTE: The Claims_Cleaned and ClaimsByAirport tables must reside in the SQ library.              *;
* NOTE: The code below will create a report in the location of the PATH macro variable (your course   *;
*      code folder). The report will be named FinalReport.html.                                   *;
*******************************************************************************
******************;


/*****************************************BEGIN HTML
REPORT*************************************************/

/*Location of the input tables*/
%let inputLib=sq;

/*Close all default ODS output*/
ods _all_ close;
```

```
/*Output the HTML file to this location. Name the file 'FinalReport.html'*/
ods html5 file="FinalReport.html" path="&path";

/*Set up the HTML grid*/
ods layout gridded columns=3 rows=4 column_gutter=.25in row_gutter=.25in;

/*Set up the options*/
ods escapechar='^'; /*Used as an escape character for ODS TEXT*/
ods noproctitle; /*Remove all proctitles from the output*/
title;footnote; /*Clear any previously set titles or footnotes*/

/*Set colors and sizes for titles and text in the report*/
%let MainTitleColor=cx081d58;
%let MainTitleSize=28pt;
%let TitleColor=cx081d58;
%let TitleSize=16pt;
%let EmphasisNumbers=cx1d91c0;
%let EmphasisNumbersSize=26pt;
%let EmphasisText=14pt;


************************************************;
*Row 1                                          *;
************************************************;
ods region row=1 column=1 column_span=3;

/*Add the report's main title*/
```

```
ods text="^{style[textalign=c fontsize=&MainTitleSize color=&MainTitleColor]TSA Claims Case Study
Check}";



************************************************;
*Row 2                                *;
************************************************;
**************;
*COLUMN 1 & 2*;
**************;
ods region row=2 column=1 column_span=2;

/*Create a new table with the count of overall claims by year using the work.claims_cleaned table*/
proc sql;
create table TotalClaimsByYear as
select put(Incident_Date, year4.) as Year,
      count(*) as TotalClaims format=comma16.
   from &inputlib..Claims_Cleaned
   group by calculated Year
   order by Year;
quit;

/*Visualize the newly created table in a bar chart*/
ods graphics on /width=11in height=4.5in imagemap=on;
title h=&TitleSize color=&TitleColor "Total Claims by Year";
footnote "NOTE: Using the Claims Cleaned Table";
proc sgplot data=TotalClaimsByYear;
            vbar Year / response=TotalClaims colorresponse=TotalClaims
             dataskin=pressed
```

```
            barwidth=.8
            datalabel
            colormodel=(cxedf8b1 cx7fcdbb cx2c7fb8)
            tip=(Year TotalClaims);
           gradlegend / notitle;
           yaxis display=(noline noticks) grid label="Total Claims Filed";
quit;
title;
footnote;

**************;
*COLUMN 3    *;
**************;
ods region row=2 column=3;

/*Store the total enplanements value in a macro variable using the work.ClaimsByAirport table*/
proc sql noprint;
select sum(Enplanement) as TotalDateIssues format=comma16.
   into :TotalEnplanements trimmed
           from &inputlib..ClaimsByAirport;
quit;
ods text="^{style[just=c fontsize=&EmphasisText]Total Enplanements}";
ods text="^{style[just=r fontsize=&EmphasisNumbersSize
color=&EmphasisNumbers]&TotalEnplanements}";

/*Store the total claims filed value in a macro variable*/
proc sql noprint;
select count(*) as TotalClaims format=comma16.
   into :TotalClaims trimmed
```

```
        from &inputlib..Claims_Cleaned;
quit;
ods text="^{style[textalign=c fontsize=&EmphasisText color=&MainTitleColor]Total Claims Filed}";
ods text="^{style[textalign=c fontsize=&EmphasisNumbersSize
color=&EmphasisNumbers]&TotalClaims}";


/*Store the percentage of claims filed value in a macro variable*/
   /*Step 1. Remove commas from the macro variables*/
%let TotalEnplanementsNum=%sysfunc(compress("&TotalEnplanements",","));
%let TotalClaimsNum=%sysfunc(compress("&TotalClaims",","));
   /*Step 2. Calculate the percentage of total claims by enplanements and store in a macro variable*/
data _null_;
        Total=&TotalClaimsNum/&TotalEnplanementsNum;
        PercentValue=put(Total,percent8.4);
        call symputx('PctClaimsTotal',PercentValue);
run;
ods text="^{style[textalign=c fontsize=&EmphasisText color=&MainTitleColor]Percentage of Claims Filed
by Enplanements}";
ods text="^{style[textalign=c fontsize=&EmphasisNumbersSize
color=&EmphasisNumbers]&PctClaimsTotal}";


/*Store the average days of claim value in a macro variable*/
proc sql noprint;
select sum(Date_Received-Incident_Date)/count(*) as AvgDays format=5.1
   into :AvgDays
   from &inputlib..claims_cleaned;
quit;
ods text="^{style[textalign=c fontsize=&EmphasisText color=&MainTitleColor]Average Time (in days) to
File a Claim}";
ods text="^{style[textalign=c fontsize=&EmphasisNumbersSize color=&EmphasisNumbers]&AvgDays}";
```

```
/*Store average days of claim value in a macro variable*/
proc sql noprint;
select count(*) as TotalUnknownAirports format=comma5.
   into :TotalUnknownAirports
   from &inputlib..claims_cleaned
            where Airport_Code = "Unknown";
quit;
ods text="^{style[textalign=c fontsize=&EmphasisText color=&MainTitleColor]Total Unknown Airports}";
ods text="^{style[textalign=c fontsize=&EmphasisNumbersSize
color=&EmphasisNumbers]&TotalUnknownAirports}";



************************************************;
*Row 3                                *;
************************************************;
**************;
*COLUMN 1    *;
**************;
ods region row=3 column=1;

/*Visualize the frequency of Claim_Type*/
ods graphics on /width=5.5in height=4in;
title h=&TitleSize color=&TitleColor "Types of Claims Filed (Claim_Type)";
proc sgplot data=&inputlib..claims_Cleaned;
            hbar Claim_Type / categoryorder=respdesc datalabel fillattrs=(color= cx7fcdbb);
            yaxis display=(nolabel noticks noline)
        valueattrs=(color=gray33 size=9pt);
   xaxis grid labelattrs=(color=gray33 size=9pt)
```

```
        valueattrs=(color=gray33 size=9pt);
run;
title;

**************;
*COLUMN 2    *;
**************;
ods region row=3 column=2;

/*Visualize the frequency of Disposition*/
ods graphics on /width=5.5in height=4in ;
title h=&TitleSize color=&TitleColor "Final Result of a Claim (Disposition)";
proc sgplot data=&inputlib..claims_Cleaned;
            hbar Disposition / categoryorder=respdesc
                                                datalabel
                                                fillattrs=(color=cxedf8b1);
            yaxis display=(nolabel noticks noline)
        valueattrs=(color=gray33 size=9pt);
   xaxis grid labelattrs=(color=gray33 size=9pt)
        valueattrs=(color=gray33 size=9pt);
run;
title;

**************;
*COLUMN 3    *;
**************;
ods region row=3 column=3;

/*Visualize the frequency of Claim_Site*/
```

```
ods graphics on /width=5.5in height=4in;
title h=&TitleSize color=&TitleColor "Location Site of a Claim (Claim_Site)";
proc sgplot data=&inputlib..Claims_Cleaned;
        hbar Claim_Site / categoryorder=respdesc datalabel fillattrs=(color=cx2c7fb8);
        yaxis display=(nolabel noticks noline)
      valueattrs=(color=gray33 size=9pt);
  xaxis grid labelattrs=(color=gray33 size=9pt)
      valueattrs=(color=gray33 size=9pt);
run;


************************************************;
*Row 4                              *;
************************************************;
**************;
*COLUMN 1 & 2*;
**************;
ods region row=4 column=1 column_span=2;

/*Create a tablet view the top 20 airports by PctClaims with over 10,000,000 passengers*/
proc sql outobs=20;
create table top20airports as
select *
        from &inputlib..ClaimsByAirport
        where Enplanement > 10000000
        order by PctClaims desc;
quit;

/*Visualize the top20airports table*/
```

```
ods graphics on /width=12in height=4in;
title1 h=&TitleSize color=&TitleColor "Top 20 Airports by Highest Percetage of Claims Filed";
title2 h=&TitleSize color=&TitleColor "For Airports With More Than 10 Million Passengers";
proc sgplot data=top20airports;
        bubble x=Enplanement y=PctClaims size=TotalClaims / group=Airport_Code

        datalabel=Airport_Code

        transparency=.3

        tip=(Airport_Name PctClaims Enplanement TotalClaims State)

        datalabelattrs=(size=8 weight=bold);
        inset "Bubble size represents total number of claims" / position=topleft

                textattrs=(size=8);
        yaxis grid label="Percentage of Claims Per Passengers";
        xaxis grid label="Total Passengers";
run;
title;

**************;
*COLUMN 3    *;
**************;
ods region row=4 column=3;

/*Store values of the airport with the highest PctClaims for airports over 10,000,000 passengers.*/
proc sql noprint;
select Airport_Name, Year, PctClaims format=percent8.4
   into :Name trimmed, :Year trimmed, :PctClaims trimmed
```

```
   from top20airports(obs=1);
quit;

ods text=";
ods text=";
/*Airport Name*/
ods text="^{style[textalign=c fontsize=&EmphasisText color=&MainTitleColor]Airport with the Highest Percentage of Claims}";
ods text="^{style[textalign=c fontsize=&EmphasisNumbersSize color=&EmphasisNumbers]&Name}";
/*Year*/
ods text="^{style[textalign=c fontsize=&EmphasisText color=&MainTitleColor]Year}";
ods text="^{style[textalign=c fontsize=&EmphasisNumbersSize color=&EmphasisNumbers]&Year}";
/*PctClaims*/
ods text="^{style[textalign=c fontsize=&EmphasisText color=&MainTitleColor]Percentage of Claims Per Passengers}";
ods text="^{style[textalign=c fontsize=&EmphasisNumbersSize color=&EmphasisNumbers]&PctClaims}";

/*End gridded layout*/
ods layout end;

/*Close output to HTML5*/
ods html5 close;

/*********************************************END REPORT*************************************************/
```

Table: WORK.TOTALCLAIMSBYYEAR | View: Column names

Columns

- Select all
- Year
- TotalClaims

Total rows: 5 Total columns: 2

	Year	TotalClaims
1	2013	9,536
2	2014	8,680
3	2015	7,721
4	2016	8,182
5	2017	8,403

Table: WORK.TOP20AIRPORTS | View: Column names | Filter: (none)

Columns

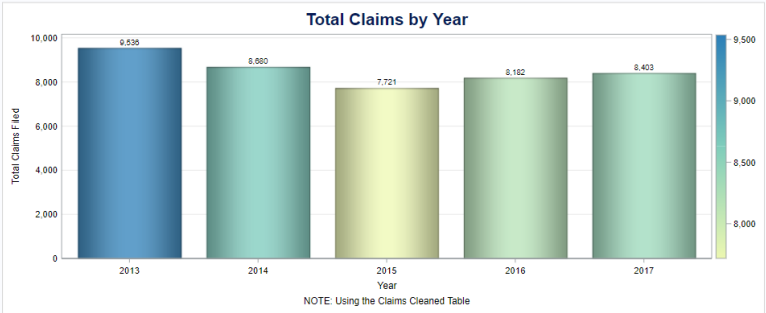
- Select all
- Airport_Code
- Airport_Name
- City
- State
- Year
- TotalClaims
- Enplanement
- PctClaims

Property	Value
Label	
Name	
Length	
Type	
Format	
Informat	

Total rows: 20 Total columns: 8 Rows 1-20

	Airport_Code	Airport_Name	City	State	Year	TotalClaims	Enplanement	PctClaims
1	JFK	John F. Kennedy International	New York	NY	2013	697	25,036,358	0.0028%
2	JFK	John F. Kennedy International	New York	NY	2014	616	26,244,928	0.0023%
3	MCO	Orlando International Airport	Orlando	FL	2013	362	16,884,524	0.0021%
4	MCO	Orlando International Airport	Orlando	FL	2014	351	17,278,608	0.0020%
5	EWR	Newark International Airport	Newark	NJ	2013	353	17,546,506	0.0020%
6	MIA	Miami International Airport	Miami	FL	2013	382	19,420,089	0.0020%
7	IAD	Washington Dulles International	Washington	DC	2014	203	10,415,948	0.0019%
8	IAD	Washington Dulles International	Washington	DC	2013	206	10,570,993	0.0019%
9	MCO	Orlando International Airport	Orlando	FL	2016	392	20,283,541	0.0019%
10	BOS	Boston (Logan) International Airport	Boston	MA	2013	273	14,810,153	0.0018%
11	EWR	Newark International Airport	Newark	NJ	2014	324	17,773,405	0.0018%
12	MIA	Miami International Airport	Miami	FL	2016	366	20,875,813	0.0018%
13	MCO	Orlando International Airport	Orlando	FL	2015	326	18,759,938	0.0017%
14	MCO	Orlando International Airport	Orlando	FL	2017	372	21,565,448	0.0017%
15	MIA	Miami International Airport	Miami	FL	2014	335	19,471,466	0.0017%
16	FLL	Ft. Lauderdale-Hollywood International	Fort Lauderdale	FL	2013	196	11,538,140	0.0017%
17	JFK	John F. Kennedy International	New York	NY	2015	468	27,782,369	0.0017%
18	MIA	Miami International Airport	Miami	FL	2017	343	20,709,225	0.0017%
19	JFK	John F. Kennedy International	New York	NY	2017	489	29,533,154	0.0017%
20	FLL	Ft. Lauderdale-Hollywood International	Fort Lauderdale	FL	2014	199	12,031,860	0.0017%

TSA Claims Case Study Check



Total Enplanements

3,950,117,888

Total Claims Filed

42,522

Percentage of Claims Filed by Enplanements

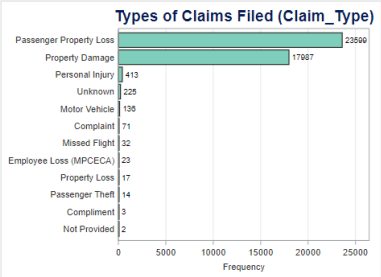
.0011%

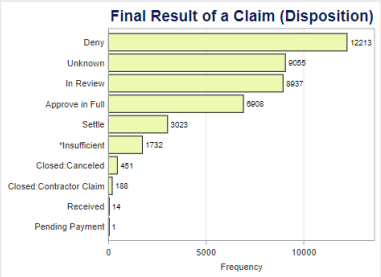
Average Time (in days) to File a Claim

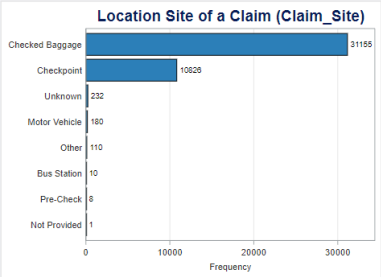
31.1

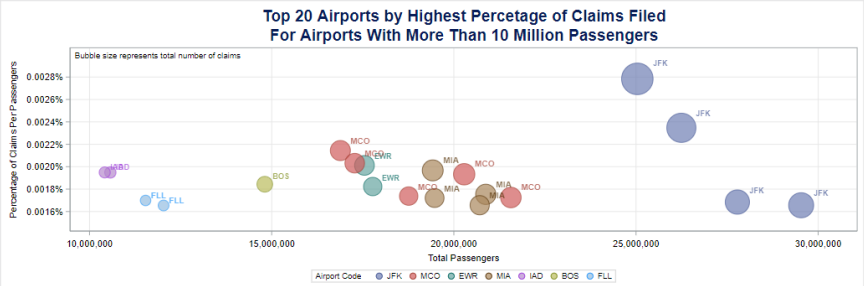
Total Unknown Airports

347









;ods text=

Airport with the Highest Percentage of Claims

John F. Kennedy International

Year

2013

Percentage of Claims Per Passengers

.0028%

```
*****************************************************************************
*********************
PROGRAM: SQL TSA Claims Case Study Solution
CREATED BY: Peter Styliadis
DATE CREATED: 5/2/2019
PROGRAM PURPOSE: Solution code to the SQL Essentials course case study. Goal
of the case study is for learners
                 to follow the directions and access/explore/prepare data
using SQL. Once complete, they
                 will run the code given to them from the analyze stage to
create a static dashboard. The code in the
                 analyze stage is to show them some visuals using SAS to
confirm the answers. They are out of scope
                 of the SQL course. Comments are throughout the program. The
code is broken up into the SAS Programming
                 Process:
                           - Access -> Explore -> Prepare -> Analyze ->
Export

*****************************************************************************
**********;
* THE FOLLOWING STEP BY STEP SOLUTION CORRESPONDS TO THE CASE STUDY - GUIDED
VERSION  *;
*****************************************************************************
**********;

*******************;
*ACCESS DATA      *;
*******************;
*****************************************************************************
***********;
* All tables are located in the SQ library for this case study. The code
below will set*;
* the path to the data and the library for you. Data must reside in the home
*;
* directory -> ESQ1M6. This will work for SAS OnDemand for Academics. Path
will        *;
* need to be changed if using another SAS interface.
*;
*****************************************************************************
***********;
%let path=~/ESQ1M6;
libname sq "&path/data";



*******************;
*EXPLORE DATA     *;
*******************;
/*2. Preview the first 10 rows and descriptor portion of the following
tables*/
*********;
* NOTES *;
*****************************************************************************
************************************;
* a. Year is character in the enplanement table, numeric in the boarding
table.                                  *;
```

```
* b. The number of passengers is called enplanement in the enplanement table,
and boarding in the boarding table *;
*************************************************************************
************************************;
proc sql outobs=10;
title "Table: CLAIMSRAW";
describe table sq.claimsraw;
select *
    from sq.claimsraw;
title "Table: ENPLANEMENT2017";
describe table sq.enplanement2017;
select *
    from sq.enplanement2017;
title "Table: BOARDING2013_2016";
describe table sq.boarding2013_2016;
select *
    from sq.boarding2013_2016;
title;
quit;


/*3. Count the number of nonmissing values in the following:*/
/* TotalRow  TotalAirportCode  TotalClaimSite TotalDisposition
TotalClaimType TotalDateReceived  TotalIncidentDate
   42,528    42,179            42,295         33,469            42,303
42,528             42,528             */
title "Total Nonmissing Rows";
proc sql;
select count(*) as TotalRow format=comma16.,
       count(Airport_Code) as TotalAirportCode format=comma16.,
       count(Claim_Site) as TotalClaimSite format=comma16.,
       count(Disposition) as TotalDisposition format=comma16.,
       count(Claim_Type) as TotalClaimType format=comma16.,
       count(Date_Received) as TotalDateReceived format=comma16.,
          count(Incident_Date) as TotalIncidentDate format=comma16.
    from sq.claimsraw;
quit;
title;


/*4. View percentage of missing values in the columns*/
/*Create a macro variable with the total number of rows - 42,528*/
proc sql noprint;
select count(*)
    into :TotalRows trimmed
    from sq.claimsraw;
quit;
%put &=TotalRows;

/*PctAirportCode PctClaimSite PctDisposition PctClaimType PctDateReceived
PctIncidentDate
  0.82%          0.55%        21.3%          0.53%        0.00%
0.00%*/
title "Percentage of Missing Rows";
proc sql;
select 1-(count(Airport_Code)/&TotalRows) as PctAirportCode
                                            format=percent7.2,
```

```
       1-(count(Claim_Site)/&TotalRows) as PctClaimSite
                                          format=percent7.2,
       1-(count(Disposition)/&TotalRows) as PctDisposition
                                         format=percent7.2,
       1-(count(Claim_Type)/&TotalRows) as PctClaimType
                                        format=percent7.2,
       1-(count(Date_Received)/&TotalRows) as PctDateReceived
                                           format=percent7.2,
       1-(count(Incident_Date)/&TotalRows) as PctIncidentDate
                                           format=percent7.2
    from sq.claimsraw;
quit;
title;


/*5. View the distinct values and frequencies*/
title "Column Distinct Values";
proc sql number;
/*Claim_Site*/
title2 "Column: Claim_Site";
select distinct Claim_Site
    from sq.claimsraw
    order by Claim_Site;
/*Disposition*/
title2 "Column: Disposition";
select distinct Disposition
    from sq.claimsraw
    order by Disposition;
/*Claim_Type*/
title2 "Column: Claim_Type";
select distinct Claim_Type
    from sq.claimsraw
    order by Claim_Type;
/*Date_Received*/
title2 "Column: Date_Received";
select distinct put(Date_Received, year4.) as Date_Received
    from sq.claimsraw
    order by Date_Received;
/*Incident_Date*/
title2 "Column: Incident_Date";
select distinct put(Incident_Date, year4.) as Incident_Date
    from sq.claimsraw
    order by Incident_Date;
quit;
title;


/*6. Count the number of rows where Incident_Date occurs AFTER Date_Recieved
- 65 rows*/;
title "Number of Claims where Incident Date Occurred After the Date
Received";
proc sql;
select count(*) label="Date Needs Review"
    from sq.claimsraw
    where Incident_Date > Date_Received;
quit;
title;
```

```
/*7. Run a query to view all rows and columns where Incident_Date occurs
AFTER Date_Received.
What assumption can you make about the dates in your results?*/
proc sql;
select Claim_Number, Date_Received, Incident_Date
    from sq.claimsraw
    where Incident_Date > Date_Received;
quit;


*******************;
*PREPARE DATA     *;
*******************;
/*8. Create a new table named Claims_NoDup that removes entirely duplicated
rows.
     A duplicate claim exists if every value is duplicated.*/
/*
NOTE: The data set work.CLAIMS_NODUP has 42524 observations and 13 variables.
*/
proc sql;
create table Claims_NoDup as
select distinct *
    from sq.claimsraw;
quit;



/*9. Prepare Data*/
proc sql;
create table sq.Claims_Cleaned as
select
/*a. Select the Claim_Number, Incident Date columns.*/
       Claim_Number label="Claim Number",
       Incident_Date format=date9. label="Incident Date",
/*b. Fix the 65 date issues you identified earlier by replacing the year 2017
with 2018 in the Date_Received column.*/
          case
                   when Incident_Date > Date_Received then
intnx("year",Date_Received,1,"sameday")
                       else Date_Received
          end as Date_Received label="Date Received" format=date9.,
/*c. Select the Airport_Name column*/
          Airport_Name label="Airport Name",
/*d. Replace missing values in the Airport_Code column with the value
Unknown.*/
       case
            when Airport_Code is null then "Unknown"
                else Airport_Code
          end as Airport_Code label="Airport Code",
/*e1. Clean the Claim_Type column.*/
       case
           when Claim_Type is null then "Unknown"
                  else scan(Claim_Type,1,"/","r") /*If I find a '/', scan and
retrieve the first word*/
       end as Claim_Type label="Claim Type",
```

```
/*e2. Clean the Claim_Site column.*/
       case
           when Claim_Site is null then "Unknown"
           else Claim_Site
       end as Claim_Site label="Claim Site",
/*e3. Clean the Disposition column.*/
       case
           when Disposition is null then "Unknown"
           when Disposition="Closed: Canceled" then "Closed:Canceled"
           when Disposition="losed: Contractor Claim" then "Closed:Contractor
Claim"
           else Disposition
       end as Disposition,
/*f. Select the Close_Amount column.*/
       Close_Amount format=Dollar20.2 label="Close Amount",
/*g. Select the State column and upper case all values.*/
       upcase(State) as State,
/*h. Select the StateName, County and City column. Proper case all values.*/
           propcase(StateName) as StateName label="State Name",
       propcase(County) as County,
       propcase(City) as City
        from Claims_NoDup
/*i. Remove all rows where year of Incident_Date occurs after 2017. */
    where year(Incident_Date) <= 2017
/*j. Order the results by Airport_Code, Incident_Date.*/
    order by Airport_Code, Incident_Date;
quit;


/***************Validate the Prepared Data***************/
proc sql;
select count(*) as TotalRows
    from sq.claims_cleaned;
quit;

title "SQL Distinct Values Validation";
proc sql;
/*Claim_Site*/
title2 "Column: Claim_Site";
select distinct Claim_Site
    from sq.claims_cleaned
    order by Claim_Site;
/*Disposition*/
title2 "Column: Disposition";
select distinct Disposition
    from sq.claims_cleaned
    order by Disposition;
/*Claim_Type*/
title2 "Column: Claim_Type";
select distinct Claim_Type
    from sq.claims_cleaned
    order by Claim_Type;
/*Date_Received*/
title2 "Column: Date_Received";
select distinct put(Date_Received, year4.) as Date_Received
    from sq.claims_cleaned
    order by Date_Received;
```

```
/*Incident_Date*/
title2 "Column: Incident_Date";
select distinct put(Incident_Date, year4.) as Incident_Date
    from sq.claims_cleaned
    order by Incident_Date;
quit;
title;
/***************End Validation****************************************/




/*10. Use the sq.Claims_Cleaned table to create a view named TotalClaims to
count the number of claims for each Airport_Code and Year.*/
/*NOTE: View work.TOTALCLAIMS created, with 1491 rows and 5 columns.*/
proc sql;
create view TotalClaims as
select Airport_Code, Airport_Name, City, State,
       year(Incident_date) as Year,
       count(*) as TotalClaims
    from sq.claims_cleaned
    group by Airport_Code, Airport_Name, City, State, calculated Year
    order by Airport_Code, Year;
quit;


/*11. Create a view name TotalEnplanements by using the OUTER UNION set
operator to concatenate the enplanement2017 and boarding2013_2016 tables.*/
proc sql;
create view TotalEnplanements as
select LocID, Enplanement, input(Year,4.) as Year
    from sq.enplanement2017
    outer union corr
select LocID, Boarding as Enplanement, Year
    from sq.boarding2013_2016
    order by Year, LocID;
quit;

/*12. Create a table named sq.ClaimsByAirport by joining the TotalClaims and
TotalEnplanements views.*/
proc sql;
create table sq.ClaimsByAirport as
select t.Airport_Code,t.Airport_Name, t.City, t.State,
       t.Year, t.TotalClaims, e.Enplanement,
       TotalClaims/Enplanement as PctClaims format=percent10.4
    from TotalClaims as t inner join
             TotalEnplanements as e
        on t.Airport_Code = e.LocID and
       t.Year = e.Year
        order by Airport_Code, Year;
quit;



*************************************************************************;
*  ALTERNATIVE: SOLVE STEPS 10-12 USING ONE QUERY WITH IN-LINE VIEWS     *;
*************************************************************************;
```

```
/*
proc sql;
create table sq.ClaimsByAirport as
select t.Airport_Code,t.Airport_Name, t.City, t.State,
       t.Year, t.TotalClaims, e.Enplanement,
       TotalClaims/Enplanement as PctClaims format=percent10.4
    from (select Airport_Code, Airport_Name, City, State,
                 year(Incident_date) as Year,
                 count(*) as TotalClaims
              from sq.claims_cleaned
              group by Airport_Code, Airport_Name, City, State, calculated
Year) as t inner join
             (select LocID, Enplanement, input(Year,4.) as Year
              from sq.enplanement2017
              outer union corr
          select LocID, Boarding as Enplanement, Year
              from sq.boarding2013_2016) as e
       on t.Airport_Code = e.LocID and
       t.Year = e.Year
       order by Airport_Code, Year;
quit;
*/
*************************************************************************;


*******************;
*EXPORT & ANALYSIS*;
*******************;
*************************************************************************
*********************;
* Run the following when complete. The statement runs the AnalysisProgram.sas
to create the      *;
* FinalResults.html in the location of the caseStudyFilesPath macro variable
set at the top of   *;
* this program.
*;
*************************************************************************
*********************;
*************************************************************************
***********;
* Specify the location of the AnalysisProgram.sas program. This is also the
location   *;
* for the FinalReport.html report output. The case study files must reside in
the home *;
* directory -> ESQ1M6 -> caseStudy. This will work for SAS OnDemand for
Academics. Path*;
* will need to be changed if using another SAS interface.
*;
*************************************************************************
***********;
%include "~/ESQ1M6/caseStudy/AnalysisProgram.sas";
```