Case Study Solution

This solution follows the steps outlined in the Intermediate Level Case study. Begin the case study by opening **StarterProgram.sas**.

Access Data

1. The tables were created in the **Sq** library after you ran 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
      1) What type is the **Year** column in each table?
         Year is character in the enplanement2017 table and numeric in the boarding2013_2016 table.
      2) What is the column name that holds the value of the number of passengers that boarded a plane in each table?
         In the enplanement2017 table, the column is named Enplanement, and in the boarding2013_2016 table, it is named Boarding.

```
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 for the ***entire table*** and for the following columns:
   a. **Airport_Code**
   b. **Claim_Site**
   c. **Disposition**
   d. **Claim_Type**
   e. **Date_Received**
   f. **Incident_Date**

```
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;
```

Results

Total Nonmissing Rows

TotalRow	TotalAirportCode	TotalClaimSite	TotalDisposition	TotalClaimType	TotalDateReceived	TotalIncidentDate
42,528	42,179	42,295	33,469	42,303	42,528	42,528

4. In one query, find the percentage of missing values in the following columns:
   a. **Airport_Code**
   b. **Claim_Site**
   c. **Disposition**
   d. **Claim_Type**
   e. **Date_Received**
   f. **Incident_Date**

```
/*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;

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;
```

Results

Percentage of Missing Rows					
PctAirportCode	PctClaimSite	PctDisposition	PctClaimType	PctDateReceived	PctIncidentDate
0.82%	0.55%	21.3%	0.53%	0.00%	0.00%

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**
      1) Replace the missing values with the value *Unknown*.
   b. **Disposition**
      1) Remove a leading space in front of *Closed: Canceled*.
      2) Add a *C* and remove the extra leading space in *losed: Contractor Claim*.
      3) Replace the missing values with the value *Unknown*.
   c. **Claim_Type**
      1) Replace *Passenger Property Loss/Personal Injur* with *Passenger Property Loss*.
      2) Replace *Passenger Property Loss/Personal Injury* with *Passenger Property Loss*.
      3) Replace *Property Damage/Personal Injury* with *Property Damage*.
      4) Replace the missing values with the value *Unknown*.
   d. The year from **Date_Received**
      (Hint: Use the PUT function.)
      1) Column values are correct.
   e. The year from **Incident_Date**
      (Hint: Use the PUT function.)
      1) Remove rows where the year of the incident is after 2017.

```
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 in which **Incident_Date** occurs **after Date_Received**

```
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;
```

Results

Number of Claims where Incident Date Occurred After the Date Received

Needs Review
65

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?
   It seems that there was a data entry error and that the Date_Received value is a year behind and should be 2018 instead of 2017.

```
proc sql;
select Claim_Number, Date_Received, Incident_Date
    from sq.claimsraw
    where Incident_Date > Date_Received;
quit;
```

Prepare Data

Using the information from the exploring stage, 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.

```
proc sql;
create table Claims_NoDup as
select distinct *
    from sq.claimsraw;
quit;
```

Log

```
NOTE: Table TSA.CLAIMS_NODUP created, with 42524 rows and 13 columns.
```

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 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 “Data Layout” section:
      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** column. 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. Add permanent labels to each column by replacing the underscore with a space.

```
proc sql;
create table sq.Claims_Cleaned as
select
/*a. Select the Claim_Number and 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")
       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 and apply the DOLLAR format.*/
       Close_Amount format=Dollar20.2 label="Close Amount",
/*g. Select the State column and uppercase 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;
```

Log

```
NOTE: Table TSA.CLAIMS_CLEANED created, with 42522 rows and 13 columns.
```

Partial Table

Claim_Number	Incident_Date	Date_Received	Airport_Name	Airport_Code	Claim_Type	Claim_Site	Disposition	Close_Amount	State	StateName	County	City
2013022602074	04FEB2013	19FEB2013	Lehigh Valley International Airport, Allentown	ABE	Property Damage	Checked Baggage	Deny	$0.00	PA	Pennsylvania	Lehigh	Allentown
2013031302547	05MAR2013	08MAR2013	Lehigh Valley International Airport, Allentown	ABE	Property Damage	Checked Baggage	Deny	$0.00	PA	Pennsylvania	Lehigh	Allentown
2013032002658	10MAR2013	13MAR2013	Lehigh Valley International Airport, Allentown	ABE	Passenger Property Loss	Checkpoint	Deny	$0.00	PA	Pennsylvania	Lehigh	Allentown
2013062304622	03MAY2013	23JUN2013	Lehigh Valley International Airport, Allentown	ABE	Property Damage	Checked Baggage	Unknown	.	PA	Pennsylvania	Lehigh	Allentown
2013060904074	06MAY2013	09JUN2013	Lehigh Valley International Airport, Allentown	ABE	Property Damage	Checked Baggage	Approve in Full	$97.96	PA	Pennsylvania	Lehigh	Allentown
2013080805751	08MAY2013	26JUN2013	Lehigh Valley International Airport, Allentown	ABE	Passenger Property Loss	Checked Baggage	Approve in Full	$100.00	PA	Pennsylvania	Lehigh	Allentown

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 want to 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 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;
```

Partial View

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

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 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;
```

Partial View

LocID	Enplanement	Year
0AK	3,123	2013
16A	3,652	2013
1G4	140,886	2013
2A3	2,336	2013
2A9	3,622	2013
4A2	2,500	2013
6R7	2,870	2013

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**.

```
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;
```

Log

```
NOTE: Table TSA.CLAIMSBYAIRPORT created, with 1438 rows and 8 columns.
```

Partial Table

Airport_Code	Airport_Name	City	State	Year	TotalClaims	Enplanement	PctClaims
ABE	Lehigh Valley International Airport, Allentown	Allentown	PA	2013	9	301,969	0.0030%
ABE	Lehigh Valley International Airport, Allentown	Allentown	PA	2014	3	298,306	0.0010%
ABE	Lehigh Valley International Airport, Allentown	Allentown	PA	2015	4	320,544	0.0012%
ABE	Lehigh Valley International Airport, Allentown	Allentown	PA	2016	5	324,511	0.0015%
ABE	Lehigh Valley International Airport, Allentown	Allentown	PA	2017	3	328,914	0.0009%
ABI	Abilene Regional	Abilene	TX	2013	4	82,758	0.0048%
ABI	Abilene Regional	Abilene	TX	2014	6	93,656	0.0064%

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

Analyze and Export Data

After you have prepared the data deliverables, open and run the **AnalysisProgram.sas** code at to create **FinalReport.html**. **Note**: You must have the final tables in the **Sq** library for the program to run correctly. **FinalReport.html** is created in your course code folder. Use this report to answer the quiz questions.