SAS® SQL 1: Essentials Appendix A

Case Study

SAS® SQL 1: Essentials Appendix A Case Study

Book code E714091, prepared date 25Sep2019. Appendix A, LWSQ1M6_001

Appendix A Case Study

1.1 Case Study Introduction

In this case study, you solve a real-world business problem by applying concepts that you learned in the SAS SQL 1: Essentials course. Be aware that there are numerous solutions to this problem, and some can include concepts that are outside the scope of the course.

Creating Case Study Files (Required)

Download the **casestudy_cre8data.sas** program and open it in your SAS session. Modify the %LET statement to provide a writeable location in your SAS environment for the **Path** macro variable, and then run the program. The program creates the following case study files:

- **StarterProgram.sas** SAS program
- **AnalysisProgram.sas** SAS program
- **work.claimsraw** SAS table
- **work.enplanement2017** SAS table
- **work.boarding2013_2016** SAS table

How to Attempt the Case Study

There are three ways to complete this case study. Please follow the method that fits your skill set.

Advanced Level

- If you feel comfortable with the topics in the SQL 1: Essentials course and want to treat this as a real-world problem, read Section 1.3, “Advanced Level,” of the PDF and begin. During the process, feel free to use your notes, Google, or SAS documentation.

Intermediate Level

- If you think you might need a bit of assistance in the case study, read Section 1.4, “Intermediate Level,” of the PDF for a guide. This section does not give you the solutions, but instead it provides a roadmap on how to solve the problem. If you are stuck on a specific task, you can find the solution in Section 1.5, “Beginner Level.”

Beginner Level

- If you are not familiar with SQL and want to use the case study as a walk-through demo, feel free to do so. You can read Section 1.5, “Beginner Level,” and follow a roadmap to solve the problem using a suggested solution for each task. After you run through the case study as a demo, we recommend that you go back and attempt it on your own.

If you have any questions regarding the case study, if you came up with different solutions and want to show them off, or if you want to share additional visualizations, post in the SAS Training Forum. We would love to hear from you!

Business Problem

The Transportation Security Administration (TSA) is an agency of the United States Department of Homeland Security that has authority over the security of the traveling public. A claim is filed if you are injured or your property is lost or damaged during the screening process at an airport.

Your first project is to prepare TSA Airport Claims and Enplanement data from 2013 through 2017 for analysis. You are trying to determine how many claims occur at each airport every year, how many passengers fly at each airport every year, and what is the percentage of claims per airport.

The analysis program has already been completed, but it is your job to follow the requirements and prepare the tables correctly. After the tables are prepared, you can run the analysis program provided to answer the business questions.

Deliverables

There are three deliverables that need to be completed by the end of the case study. The final deliverable tables **must be placed in the Work library**:

- **work.Claims_Cleaned** - a new table that cleans and prepares the **work.claimsraw** table
- **work.ClaimsByAirport** - a new table that is created by summarizing claims for each airport and year from the **work.Claims_Cleaned** table and then joining the summarized data with the enplanement information (**work.enplanement2017** and **work.boarding2013_2016** tables)
- **FinalReport.html** - the final HTML report produced by running the **AnalysisProgram.sas** on the prepared tables.

When creating these three deliverables, you might need to create other temporary tables or views.

Data Information

These tables were created from the following:

- TSA Airport Claims data from https://www.dhs.gov/tsa-claims-data
- FAA Airport Facilities data from https://www.faa.gov/airports/airport_safety/airportdata_5010/
- FAA Enplanements data from https://www.faa.gov/airports/planning_capacity/passenger_allcargo_stats/passenger/

Here are a few notes regarding the data:

- All data is public data, and accuracy is not guaranteed.
- The **claimsraw** table was created by concatenating each individual TSA Airport Claims table.
- The concatenated results of the TSA Airport Claims tables were joined with the FAA Airport Facilities data. The column **Airport_Codes** from the TSA Airport Claims data has been joined with **Location_ID** from the FAA Airports Facilities data to obtain airport information. Some **Airport_Codes** values do not correspond to **Location_ID** values.
- Columns in the TSA Airport Claims data have changed over the years. Because of this, some of the original columns were removed from the data for this case study.
- Some column names and data types have been changed in the FAA Enplanement data for this case study.

1.2 Data Layout

Below is the data layout for each table in the case study. The layout lists the column name, a description, and any value requirements for that column.

claimsraw table

Column	Description
Claim_Number	Number for each claim. Some claims can have duplicate claim numbers but different information for each claim. Those claims are considered valid for this case study. Any ***entirely*** duplicate row should be removed from the data.
Date_Received	Date on which the claim was received. **Requirements:** From 2013 through 2018 **Incident_Date** should always occur before **Date_Received**.
Incident_Date	Date on which the incident occurred. **Requirements**: From 2013 through 2017 **Incident_Date** should always occur before **Date_Received**.
Airport_Code	The abbreviated airport codes. **Requirements:** All missing values are considered unknown.
Airport_Name	Full name of the airport.
Claim_Type	Category of the claim. If the claim is separated into two types by a slash, **Claim_Type** is the first type. For example: *Personal Property Loss/Injury* is considered *Personal Property Loss*. Possible values (14): • *Bus Terminal* • *Complaint* • *Compliment* • *Employee Loss (MPCECA)* • *Missed Flight* • *Motor Vehicle* • *Not Provided* • *Passenger Property Loss* • *Passenger Theft* • *Personal Injury*

Column	Description
	• *Property Damage* • *Property Loss* • *Unknown* • *Wrongful Death* **Requirements:** All missing values are considered unknown.
Claim_Site	Airport location of the claim. Possible values (8): • *Bus Station* • *Checked Baggage* • *Checkpoint* • *Motor Vehicle* • *Not Provided* • *Other* • *Pre-Check* • *Unknown* **Requirements:** All missing values are considered unknown.
Close_Amount	The dollar amount that a claim was closed for.
Disposition	The final settlement of the claim. Possible values (10): • **Insufficient* • *Approve in Full* • *Closed:Canceled* • *Closed:Contractor Claim* • *Deny* • *In Review* • *Pending Payment* • *Received* • *Settle* • *Unknown* **Requirements:** All missing values are considered unknown.
StateName	Associated airport state name (for example, *NEW YORK*). **Requirements:** Values should be in proper case (for example, *New York*).

Column	Description
State	Associated airport state code. This is the standard two-letter abbreviation used by the post office for US states and territories. **Requirements:** Values should be in all uppercase (for example, *NY*).
County	Airport associated county (or parish) name (for example, *COOK*). **Requirements:** Values should be in proper case (for example, *Cook*).
City	Associated airport city name (for example, *CHICAGO*). **Requirements:** Values should be in proper case (for example, *Chicago*).

enplanement2017 table

Column	Description
LocID	Airport code abbreviation.
Year	Character year of the enplanement information.
Enplanement	Total passengers boarding (enplanements).

boarding2013_2016 table

Column	Description
LocID	Airport code abbreviation.
Year	Numeric year of the boarding information.
Boarding	Total passengers boarding (enplanements).

1.3 Advanced Level

In this version of the case study, you receive the high-level requirements to solve the business problem. There are multiple solutions to the problem, and how you solve it is your decision.

To solve the business problem, follow the requirements below given to you by your supervisor. Be aware that these requirements are only assumed for this case study. They are not an accurate representation of TSA or FAA requirements.

Begin the case study by opening the **StarterProgram.sas** and accessing the tables.

Deliverables and Requirements

Your job is to prepare two tables for analysis. After the tables are prepared, you can run the provided code in **AnalysisProgram.sas** to analyze the results. For the analysis program to run correctly, follow the requirements for each deliverable listed below.

To create the following deliverables, you can use a variety of different methods that can include temporary or permanent tables, views, and in-line views. Be sure to explore the tables and columns and compare with the "Data Layout" section for all column requirements.

1. **work.Claims_Cleaned** – Create a new table named **work.Claims_Cleaned** that cleans and prepares the **work.claimsraw** table. Here is what the **work.Claims_Clenaed** table must do:
   a. Include all columns from the **work.claimsraw** table and remove duplicated rows.
   b. Change all missing values to *Unknown* for the following columns: **Airport_Code**, **Claim_Type**, **Claim_Site**, and **Disposition**. Follow the requirements in the "Data Layout" section for the column values.
   c. Fix all rows where **Incident_Date** occurs ***after*** **Date_Received** by adding one year to the **Date_Received** value.
   d. **StateName**, **City**, and **County** values should be in proper case (for example, *Raleigh*).
   e. **State** values should be in uppercase.
   f. Include only those rows where **Incident_Date** is between 2013 and 2017.
   g. Currency columns should be permanently formatted with a dollar sign and include two decimal places (for example, $130.28).
   h. All dates should be permanently formatted in the style 01JAN2000.
   i. Assign permanent labels for columns by adding a space between words (for example, **Close Amount**).
   j. Order the final table by **Airport_Code** and **Incident_Date**.

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	Close_Amount	Disposition	StateName	State	County	City
2013022602074	04FEB2013	19FEB2013	Lehigh Valley International Airport, Allentown	ABE	Property Damage	Checked Baggage	$0.00	Deny	Pennsylvania	PA	Lehigh	Allentown
2013031302547	05MAR2013	08MAR2013	Lehigh Valley International Airport, Allentown	ABE	Property Damage	Checked Baggage	$0.00	Deny	Pennsylvania	PA	Lehigh	Allentown
2013032002658	10MAR2013	13MAR2013	Lehigh Valley International Airport, Allentown	ABE	Passenger Property Loss	Checkpoint	$0.00	Deny	Pennsylvania	PA	Lehigh	Allentown
2013062304622	03MAY2013	23JUN2013	Lehigh Valley International Airport, Allentown	ABE	Property Damage	Checked Baggage	.	Unknown	Pennsylvania	PA	Lehigh	Allentown
2013060904074	06MAY2013	09JUN2013	Lehigh Valley International Airport, Allentown	ABE	Property Damage	Checked Baggage	$97.96	Approve in Full	Pennsylvania	PA	Lehigh	Allentown
2013080805751	09MAY2013	26JUN2013	Lehigh Valley International Airport, Allentown	ABE	Passenger Property Loss	Checked Baggage	$100.00	Approve in Full	Pennsylvania	PA	Lehigh	Allentown
2013061304330	09MAY2013	13MAY2013	Lehigh Valley International Airport, Allentown	ABE	Passenger Property Loss	Checked Baggage	$99.95	Approve in Full	Pennsylvania	PA	Lehigh	Allentown

2. **work.ClaimsByAirport** – Create a new table named **work.ClaimsByAirport** by summarizing claims for each airport and year from the **work.Claims_Cleaned** table. Then perform an inner join on the summarized data with the **work.enplanement2017** and **work.boarding2013_2016** tables.

   a. Include the following columns from **work.Claims_Cleaned** table: **Airport_Code**, **Airport_Name**, **City**, **State**, and the year of the **Incident_Date**. Name the new column **Year**.

   b. Three new columns need to be added.

      1) Create the column **TotalClaims** as the number of claims for each group.

      2) Retrieve the total passengers boarding for each **Year** and **Airport_Code** and name the column **Enplanement**. The information can be found in the **work.enplanement2017** and **work.boarding2013_2016** tables.

      3) Calculate the percentage of claims for each group by dividing **TotalClaims** by **Enplanement**. Name the new column **PctClaims** and format accordingly.

      4) Order the results by **Airport_Code** and **Year**.

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%

3. **FinalReport.html –** Now the data is fully prepared. Run the code at the bottom of the **StarterProgram.sas** and answer the business questions below. For the program to run correctly, the final tables must be in the **Work** library. After you run the code, a new file named **FinalReport.html** is created in the location specified by the **Path** macro variable. You can compare your results with the solution results in Section 1.6, "Validate Your Results."

Answer the Following Business Questions:

Using the **FinalReport.html** file, answer the following business questions. You can compare your results to the results in Section 1.6, "Validate Your Results."

1. How many total enplanements are in final report?
2. How many total claims have been filed?

3. What is the percentage of claims filed by enplanements?
4. What is the average time in days to file a claim?
5. How many unknown airport codes are in the results?
6. What type of claim is typically filed? How many?
7. How many claims resulted as *Closed:Canceled*?
8. What is the most frequent location identified for claims? How many claims were filed for this location?
9. Which airport with more than 10 million passengers has the highest percentage of claims?

1.4 Intermediate Level

Below is a suggested guide to help you solve the business problem. Be aware that there are multiple solutions to this problem and that you do not need to follow the steps below.

The steps below follow the SAS programming process. How you solve each task is your choice, but if you are stuck, you can refer to the "Beginner Level" section in this document for solutions to the specific task, or post a question in the SAS Training Community.

Begin the case study by opening the **StarterProgram.sas**.

Access Data

1. The tables were created in the **Work** library after you ran the **casestudy_cre8data.sas** program.

Explore Data

2. Preview the first ***10 rows*** and the ***descriptor portion*** of the following tables:
   a. **work.claimsraw** table
   b. **work.enplanement2017** and **work.boarding2013_2016** tables
      1) What type is the **Year** column in each table?
      2) What is the column name that holds the value of how many passengers boarded a plane in each table?
3. Count the number of nonmissing values in the ***entire table*** and in the following columns:
   a. **Airport_Code**
   b. **Claim_Site**
   c. **Disposition**
   d. **Claim_Type**
   e. **Date_Received**
   f. **Incident_Date**

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**

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 using the required column values in the "Data Layout" section:
   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.)
6. Count the number of rows in which **Incident_Date** occurs ***after*** **Date_Received**.

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 **work.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?

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.

Log

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

9. Using the **Claims_NoDup** table, create a table named **work.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** 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).

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	09MAY2013	26JUN2013	Lehigh Valley International Airport, Allentown	ABE	Passenger Property Loss	Checked Baggage	Approve in Full	$100.00	PA	Pennsylvania	Lehigh	Allentown

10. Use the **work.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.

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

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

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 **work.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

```
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

Now the data is fully prepared. Run the code at the bottom of the **StarterProgram.sas** and answer the business questions below. For the program to run correctly, the final tables must be in the **Work** library. After you run the code, a new file named **FinalReport.html** is created in the location specified by the **Path** macro variable. You can compare your results with the solution results in Section 1.6, "Validate Your Results."

Answer the Following Business Questions:

Using the **FinalReport.html** file, answer the following business questions. You can compare your answers to the answers in Section 1.6, "Validate Your Results."

1. How many total enplanements are in final report?
2. How many total claims have been filed?
3. What is the percentage of claims filed by enplanements?
4. What is the average time in days to file a claim?
5. How many unknown airport codes are in the results?
6. What type of claim is typically filed? How many?
7. How many claims resulted as *Closed:Canceled*?
8. What is the most frequent location identified for claims? How many claims were filed for this location?
9. Which airport with more than 10 million passengers has the highest percentage of claims?

1.5 Beginner Level

This section is a step-by-step guide to solve the case study with solutions for each task. The steps are the same steps from Section 1.4. "Intermediate Level," but here the solution to the step is available.

For more information, you can visit the SAS Documentation or post a question in the SAS Training Community.

Begin the case study by opening **StarterProgram.sas**.

Access Data

1. The tables were created in the **Work** library after you ran the **casestudy_cre8data.sas** program.

Explore Data

2. Preview the first ***10 rows*** and the ***descriptor portion*** of the following tables:
   a. **work.claimsraw** table
   b. **work.enplanement2017** and **work.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 how many passengers 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 work.claimsraw;
select *
    from work.claimsraw;
title "Table: ENPLANEMENT2017";
describe table work.enplanement2017;
select *
    from work.enplanement2017;
title "Table: BOARDING2013_2016";
describe table work.boarding2013_2016;
select *
    from work.boarding2013_2016;
title;
quit;
```

3. Count the number of nonmissing values in the ***entire table*** and in 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 work.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 work.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 work.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 using the required column values in the "Data Layout" section:

   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 work.claimsraw
    order by Claim_Site;
/*Disposition*/
title2 "Column: Disposition";
select distinct Disposition
    from work.claimsraw
    order by Disposition;
```

```
/*Claim_Type*/
title2 "Column: Claim_Type";
select distinct Claim_Type
    from work.claimsraw
    order by Claim_Type;
/*Date_Received*/
title2 "Column: Date_Received";
select distinct put(Date_Received, year4.) as Date_Received
    from work.claimsraw
    order by Date_Received;
/*Incident_Date*/
title2 "Column: Incident_Date";
select distinct put(Incident_Date, year4.) as Incident_Date
    from work.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 work.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 **work.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 work.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 work.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 **work.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 work.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	09MAY2013	26JUN2013	Lehigh Valley International Airport, Allentown	ABE	Passenger Property Loss	Checked Baggage	Approve in Full	$100.00	PA	Pennsylvania	Lehigh	Allentown

10. Use the **work.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 work.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 **work.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 **work.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 work.enplanement2017
    outer union corr
select LocID, Boarding as Enplanement, Year
    from work.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 **work.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 work.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

Now the data is fully prepared. Run the code at the bottom of the **StarterProgram.sas** and answer the business questions below. For the program to run correctly, the final tables must be in the **Work** library. After you run the code, a new file named **FinalReport.html** is created in the location specified by the **Path** macro variable. You can compare your results with the solution results in Section 1.6, "Validate Your Results."

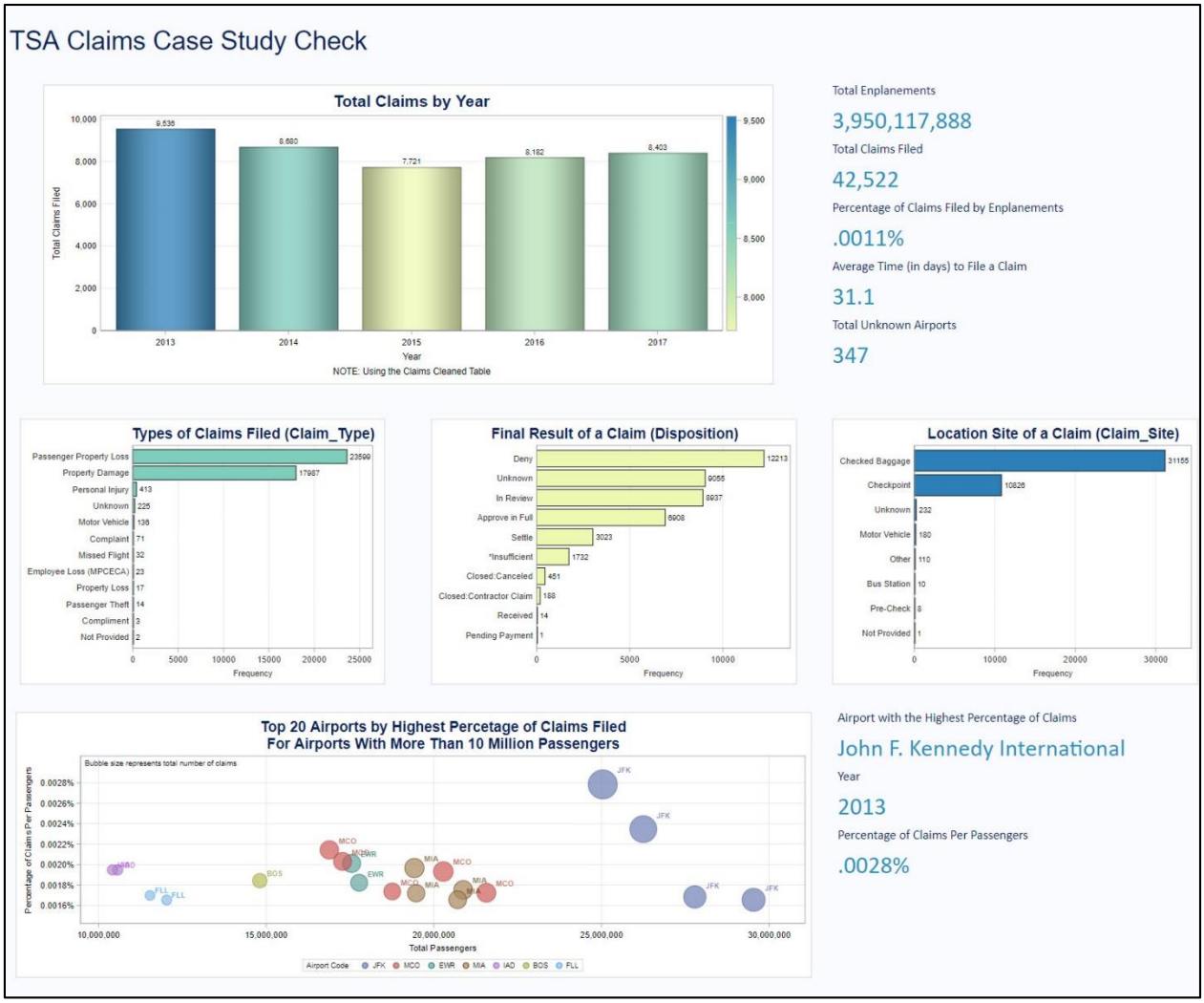
Answer the Following Business Questions:

Using the **FinalReport.html** file, answer the following business questions. You can compare your answers to the answers in Section 1.6, "Validate Your Results."

1. How many total enplanements are in final report?
2. How many total claims have been filed?
3. What is the percentage of claims filed by enplanements?
4. What is the average time in days to file a claim?
5. How many unknown airport codes are in the results?
6. What type of claim is typically filed? How many?
7. How many claims resulted as *Closed:Canceled*?
8. What is the most frequent location identified for claims? How many claims were filed for this location?
9. Which airport with more than 10 million passengers has the highest percentage of claims?

1.6 Validate Your Results

Compare your results to the **FinalReport.html** solution and business question answers below.



Business Question Solutions:

1. How many total enplanements are in final report? **3,950,117,888**
2. How many total claims have been filed? **42,522**
3. What is the percentage of claims filed by enplanements? **.0011%**
4. What is the average time in days to file a claim? **31.1**
5. How many unknown airport codes are in the results? **347**
6. What type of claim is typically filed? How many? **Passenger Property Loss, 23,599**
7. How many claims resulted as *Closed:Canceled*? **451**
8. What is the most frequent location identified for claims? How many claims were filed for this location? **Checked Baggage, 31,155**
9. Which airport with over 10 million passengers had the highest percentage of claims file? **John F. Kennedy International**