Paper PO-03

LAG Function Combined with Conditional Functions – Useful in Identifying Differences in Like Data

Andy Hummel, Delta Air Lines, Atlanta, GA

ABSTRACT

The LAG function is useful in identifying subtle differences in rows with similar data. This is especially valuable when the data set contains a large number of rows. Additionally, when conditional functions are used in conjunction with the LAG function specific limits can be used to flag only particular differences between rows.

INTRODUCTION

The LAG function can return a value from a previous row of data, or compare the current row value to a previous row. LAG can be used to look back 1 row or more than 1,000 rows depending on the programmer's needs. When LAG is used in combination with conditional functions such as IF, AND, OR, =, or NOT = it becomes a powerful evaluator of duplicate data. This paper will demonstrate an applied use of LAG in combination with conditional functions to flag duplicate rows of data.

Data that is manually entered into a database can often contain duplicate and inconsistent data. This is especially true when the data is entered by multiple users in a dynamic environment. Duplicate and conflicting records can lead to redundant expenses, such as a hotel room booked in two different cities for the same night and employee.

The data used in the following examples was manually entered into the database by multiple coordinators who set up hotel stays for employees. Due to a number of factors including multiple hotel requests from the same employee, irregular operational issues and user error the database may contain discrepancies. The following SAS® examples will show how to flag the discrepancies. The variables used in the data set are: employee number, city name, hotel name and night of hotel stay.

STEP 1: SORT THE DATA

The first and most critical step is to sort the data based on the variables that the LAG function will evaluate. The data must be sorted correctly in order for the lag function to properly evaluate the data. In this example we want to evaluate Empl_Nbr, Airport_City and Hotel_Name. Empl_Nbr will be sorted first since this is the primary variable, followed by Airport_City and Hotel_Name since they are the secondary variables.

```
/* SORTING THE DATA */
PROC SORT DATA=raw_hotel_data OUT=hotel_check_1;
       BY empl_nbr airport_city hotel_name;
RUN;
```

Table 1. Data Sorted by Empl_Nbr, Airport_City, Hotel_Name

Empl_Nbr	Airport_City	Hotel_Name	5/4/2010
1741	ATL	Bobs Best Hotel	X
1741	ATL	Crabby Inn	X
2292	BWI	Crabby Inn	X
2786	BWI	Crabby Inn	X
3413	BWI	Crabby Inn	X
3792	ATL	Bobs Best Hotel	X
3876	ATL	Crabby Inn	X
3876	BWI	Crabby Inn	X
4379	ATL	Bobs Best Hotel	X
4379	ATL	Crabby Inn	X
5083	BWI	Crabby Inn	X
5298	ATL	Bobs Best Hotel	X
5712	BWI	Crabby Inn	X
6359	ATL	Bobs Best Hotel	X
6807	BWI	Crabby Inn	X
6920	ATL	Crabby Inn	X
6920	BWI	Crabby Inn	X
7335	ATL	Bobs Best Hotel	X
7335	BWI	Crabby Inn	X
8241	ATL	Bobs Best Hotel	X
9218	BWI	Crabby Inn	X
9240	BWI	Crabby Inn	X
9827	ATL	Bobs Best Hotel	X
9959	ATL	Bobs Best Hotel	X

STEP 2: EVALUATE FOR SAME EMPLOYEE, SAME CITY, DIFFERENT HOTEL

Here the lag function is used to evaluate the data to determine if an employee is booked in the same city but in different hotels. In this example the Empl_nbr and Airport_Name in the current row must match the previous row, while the Hotel_Name in the current row must be different from the Hotel_Name in the previous row in order for the row to be flagged. The flagged rows are highlighted in the below table.

```
DATA hotel_check_2;
      SET hotel_check_1;

      /* STEP 2 */
      IF empl_nbr = LAG(empl_nbr)
            AND airport_city = LAG(airport_city)
            AND hotel_name  NE LAG(hotel_name)
      THEN same_city_diff_hotel='Yes';
      ELSE same_city_diff_hotel='No';
RUN;
```

Table 2. Same Employee, Same City, Different Hotel

Empl_Nbr	Airport_City	Hotel_Name	5/4/2010	same_city_diff_hotel
1741	ATL	Bobs Best Hotel	X	No
1741	ATL	Crabby Inn	X	Yes
2292	BWI	Crabby Inn	X	No
2786	BWI	Crabby Inn	X	No
3413	BWI	Crabby Inn	X	No
3792	ATL	Bobs Best Hotel	X	No
3876	ATL	Crabby Inn	X	No
3876	BWI	Crabby Inn	X	No
4379	ATL	Bobs Best Hotel	X	No
4379	ATL	Crabby Inn	X	Yes
5083	BWI	Crabby Inn	X	No
5298	ATL	Bobs Best Hotel	X	No
5712	BWI	Crabby Inn	X	No
6359	ATL	Bobs Best Hotel	X	No
6807	BWI	Crabby Inn	X	No
6920	ATL	Crabby Inn	X	No
6920	BWI	Crabby Inn	X	No
7335	ATL	Bobs Best Hotel	X	No
7335	BWI	Crabby Inn	X	No
8241	ATL	Bobs Best Hotel	X	No
9218	BWI	Crabby Inn	X	No
9240	BWI	Crabby Inn	X	No
9827	ATL	Bobs Best Hotel	X	No
9959	ATL	Bobs Best Hotel	X	No

STEP 3: EVALUATE FOR SAME EMPLOYEE, SAME HOTEL, DIFFERENT CITIES

Here the lag function evaluates the data to see if an employee is booked in the same hotel but in different cities. In this example the Empl_nbr and Hotel_Name in the current row must match the previous row, while the Airport_City in the current row must be different from the Airport_City in the previous row in order for the row to be flagged. The flagged rows are highlighted in the below table.

```
DATA hotel_check_3;
      SET hotel_check_1;

      /* STEP 3 */
      IF empl_nbr = LAG(empl_nbr)
            AND airport_city NE LAG(airport_city)
            AND hotel_name    = LAG(hotel_name)
      THEN same_hotel_diff_city='Yes';
      ELSE same_hotel_diff_city='No';

RUN;
```

Table 3. Same Employee, Same Hotel, Different City

Empl_Nbr	Airport_City	Hotel_Name	5/4/2010	same_city_diff_hotel	same_hotel_diff_city
1741	ATL	Bobs Best Hotel	X	No	No
1741	ATL	Crabby Inn	X	Yes	No
2292	BWI	Crabby Inn	X	No	No
2786	BWI	Crabby Inn	X	No	No
3413	BWI	Crabby Inn	X	No	No
3792	ATL	Bobs Best Hotel	X	No	No
3876	ATL	Crabby Inn	X	No	No
3876	BWI	Crabby Inn	X	No	Yes
4379	ATL	Bobs Best Hotel	X	No	No
4379	ATL	Crabby Inn	X	Yes	No
5083	BWI	Crabby Inn	X	No	No
5298	ATL	Bobs Best Hotel	X	No	No
5712	BWI	Crabby Inn	X	No	No
6359	ATL	Bobs Best Hotel	X	No	No
6807	BWI	Crabby Inn	X	No	No
6920	ATL	Crabby Inn	X	No	No
6920	BWI	Crabby Inn	X	No	Yes
7335	ATL	Bobs Best Hotel	X	No	No
7335	BWI	Crabby Inn	X	No	No
8241	ATL	Bobs Best Hotel	X	No	No
9218	BWI	Crabby Inn	X	No	No
9240	BWI	Crabby Inn	X	No	No
9827	ATL	Bobs Best Hotel	X	No	No
9959	ATL	Bobs Best Hotel	X	No	No

STEP 4: EVALUATE FOR SAME EMPLOYEE, DIFFERENT CITIES, DIFFERENT HOTELS

Here we evaluate the data to see if an employee is booked in different hotels in different cities. In this example the Empl_nbr in the current row must match the Empl_Nbr in the previous row, while the Airport_City and Hotel_Name in the current row must be different from the Airport_City and Hotel_Name in the previous row. The flagged rows are highlighted in the below table.

```
DATA hotel_check_3;
       SET hotel_check_1;

       /* STEP 4 */
       IF empl_nbr = LAG(empl_nbr)
             AND airport_city NE LAG(airport_city)
             AND hotel_name   NE LAG(hotel_name)
       THEN diff_hotel_diff_city='Yes';
       ELSE diff_hotel_diff_city='No';

RUN;
```

Table 4. Same Employee, Different City, Different Hotel

Empl_Nbr	Airport_City	Hotel_Name	5/4/2010	same_city_diff_hotel	same_hotel_diff_city	diff_hotel_diff_city
1741	ATL	Bobs Best Hotel	X	No	No	No
1741	ATL	Crabby Inn	X	Yes	No	No
2292	BWI	Crabby Inn	X	No	No	No
2786	BWI	Crabby Inn	X	No	No	No
3413	BWI	Crabby Inn	X	No	No	No
3792	ATL	Bobs Best Hotel	X	No	No	No
3876	ATL	Crabby Inn	X	No	No	No
3876	BWI	Crabby Inn	X	No	Yes	No
4379	ATL	Bobs Best Hotel	X	No	No	No
4379	ATL	Crabby Inn	X	Yes	No	No
5083	BWI	Crabby Inn	X	No	No	No
5298	ATL	Bobs Best Hotel	X	No	No	No
5712	BWI	Crabby Inn	X	No	No	No
6359	ATL	Bobs Best Hotel	X	No	No	No
6807	BWI	Crabby Inn	X	No	No	No
6920	ATL	Crabby Inn	X	No	No	No
6920	BWI	Crabby Inn	X	No	Yes	No
7335	ATL	Bobs Best Hotel	X	No	No	No
7335	BWI	Crabby Inn	X	No	No	Yes
8241	ATL	Bobs Best Hotel	X	No	No	No
9218	BWI	Crabby Inn	X	No	No	No
9240	BWI	Crabby Inn	X	No	No	No
9827	ATL	Bobs Best Hotel	X	No	No	No
9959	ATL	Bobs Best Hotel	X	No	No	No

ADDITIONAL USES

The above examples used the LAG function to evaluate the immediate preceding row of data. LAG also has the ability to evaluate or return the value from further back than one previous row. To specify how far back for LAG to look, include the number of rows to look back after "LAG". In the below example LAG is evaluating the current row to the 4th pervious row since the number 4 is placed after LAG.

```
/* EVALUATING THE CURRENT ROW TO THE 4TH PREVIOUS ROW OF DATA */
IF empl_nbr = LAG4(empl_nbr)
THEN same_empl_nbr='Yes';
ELSE same_empl_nbr='No';
```

By adding the conditional function of "OR", LAG can compare the current row to multiple rows of previous data. The following example will flag the current row if the employee number in the current row matches the employee number in any of 1st through 5th previous rows. This was accomplished by placing the numbers 1,2,3,4 and 5 after LAG.

```
/* EVALUATING BACK 4 ROWS OF DATA */
IF (empl_nbr = LAG1(empl_nbr)) OR (empl_nbr = LAG2(empl_nbr))
      OR (empl_nbr = LAG3(empl_nbr)) OR (empl_nbr = LAG4(empl_nbr))
      OR (empl_nbr = LAG5(empl_nbr))
THEN same_empl_nbr='Yes';
ELSE same_empl_nbr='No';
```

CONCLUSION

The above examples illustrated how to use the LAG function in conjunction with conditional functions as an evaluator of like data. In these examples once the duplicate records are flagged the user will need to evaluate the data to determine which record is the legitimate record to keep. The use of LAG with conditional functions makes this a particularly powerful tool for setting detailed and specific flags.

CONTACT INFORMATION

Your comments and questions are encouraged. Contact the author for the program and data presented in this paper at:

Andy Hummel
Delta Air Lines, Inc.
Department 028
P.O. Box 20706
Atlanta, GA 30320-6001
Work Phone: 404-715-1270
Email: Andrew.Hummel@delta.com