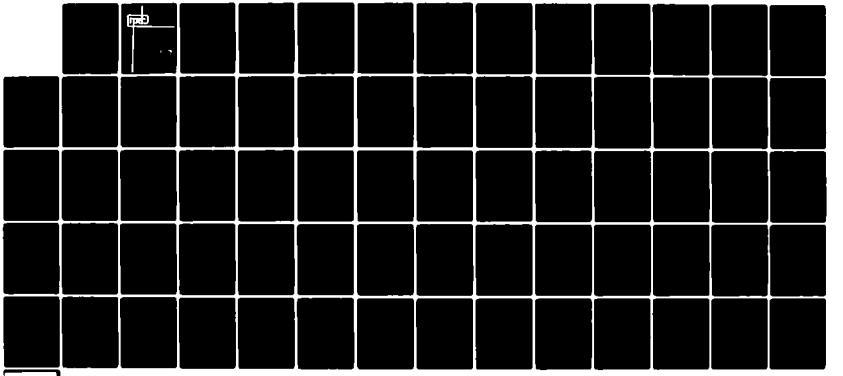
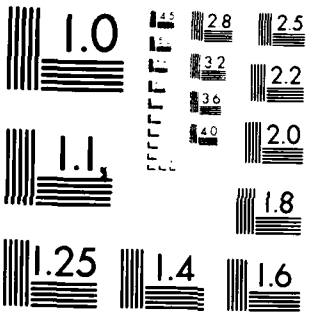
AD-A131 901 GURU: A COMPUTER PROGRAM FOR ANALYZING CATEGORIZED DATA 1/1
(U) NAVY PERSONNEL RESEARCH AND DEVELOPMENT CENTER SAN
DIEGO CA J A RIEDEL ET AL. DEC 76 NPRDC-TN-77-4
UNCLASSIFIED SBI-AD-F630 058 F/G 9/2 NL



END
DATE FILMED
9-83
DTIC



NAVY PERSONNEL RESEARCH AND DEVELOPMENT CENTER SAN DIEGO CALIFORNIA 92152

NPRDC TN 77-4

DECEMBER 1976

GURU: A COMPUTER PROGRAM FOR ANALYZING CATEGORIZED DATA

Technical Note 77-4 December 1976

GURU: A COMPUTER PROGRAM FOR ANALYZING CATEGORIZED DATA

James A. Riedel
Janet D. Dodson

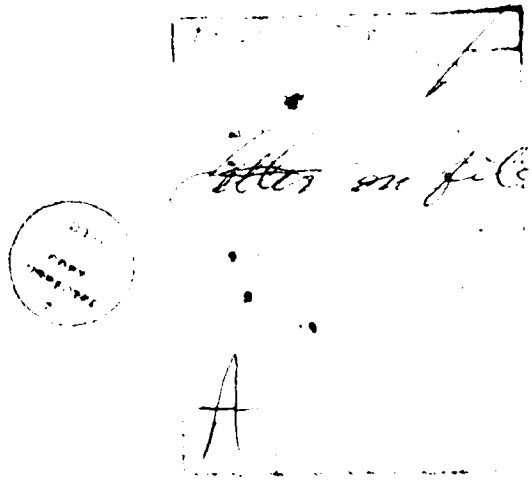
Reviewed by
Robert Penn

Navy Personnel Research and Development Center
San Diego, California 92152

FOREWORD

The development of the computer program described in this report was in support of Work Unit PF55.521.021.03.01 (Attitude Assessment Techniques). Appreciation is expressed to Drs. William H. Githens and L. A. Broedling, of the Navy Personnel Research and Development Center, for their guidance and support, and to Dr. Richard P. Barthol of the University of California at Los Angeles for his assistance in the effort.

J. J. CLARKIN
Commanding Officer



SUMMARY

Problem

A research effort was conducted at the Navy Personnel Research and Development Center (NAVPERSRANDCEN) using the ECHO technique, a technique for attitude assessment in which responses are elicited to repeated open-ended questions. During the course of this research, it was found that the existing computer programs available for analyzing ECHO and other categorized data were inadequate for carrying out the proposed analysis plan.

Objective

The objective of the present effort was to develop an efficient computer program to analyze the data generated by use of semi-structured data collection techniques such as ECHO.

Approach

A review of software packages, both those developed at NAVPERSRANDCEN and those available commercially, revealed that no program existed which satisfied the project requirements. Therefore, the development of a comprehensive computer program was undertaken. This program was designed to satisfy the data analysis requirements of the ongoing ECHO project and to be useful to other investigators employing open-ended question techniques such as ECHO.

Results

A computer program called GURU was developed. The program provides extensive descriptive statistics of categorized data and allows great flexibility in comparing various groups of respondents as well as different classifications of the same data.

CONTENTS

INTRODUCTION

Problem

A research effort was conducted at the Navy Personnel Research and Development Center (NAVPERSRANDCEN) using the ECHO technique, a technique for attitude assessment in which responses are elicited to repeated open-ended questions. During the course of this research, it was found that the existing computer programs available for analyzing ECHO and other categorized data were inadequate for carrying out the proposed analysis plan.

Background

The ECHO technique is a semistructured data collection technique used to observe, quantify, and describe ". . . the patterns of value and influence that are felt, verbally expressed, and often acted on in human society" (Barthol & de Mille, 1969).

Basically the ECHO technique involves giving each respondent 20 computer cards, each containing two questions. One of the questions is designed to reveal how the respondent feels about some aspect of the topic being studied; that is, it requests him to list some event that indicates his value judgment dealing with some area being studied. The second question is designed to reveal who would be concerned with that aspect or judgment. Each card is prepunched to distinguish the respondent, the group, the specific card, and the specific questions asked. However, the personal identity of the respondent is not provided.

Ten of the cards are called "goods," because the questions they contain are of a positive nature. For example, a "good" might ask: "What is a good thing to do on the job?" and "Who would approve?" The other 10 are called "bads," because of their negative nature. For example, a "bad" corresponding to the "good" above would ask: "What is a bad thing to do on the job?" and "Who would disapprove?" Subjects are asked to answer each question on the card and to supply demographic information.[1]

The responses obtained are inductively categorized by a team of three people. The responses to "goods" and "bads" are analyzed separately under specified sets of procedures that sometimes require a number of classification iterations before reliability is achieved. Finally, the classification codes are inserted into the computer cards and the data are processed through such computer programs as UNIKOUNT, PERZPROB, and ROLLEM. These programs provide (1) lists of response category hierarchies, (2) tables comparing groups, and (3) a comparison of the classification schemes (Barthol & de Mille, 1969).

---

[1]See Barthol and de Mille (1969) and Barthol and Bridge (1968) for a more complete description of the ECHO technique.

Objective

The present effort was undertaken to develop an efficient computer program to analyze data generated by use of semistructured data collection techniques such as ECHO. It was felt that a computer program should be designed and developed that would (1) reduce the data card handling and operator time required for analysis, (2) allow for user flexibility with regard to both input format and desired output, and (3) accommodate categorized data generated by unstructured or semistructured methods other than the ECHO technique.

APPROACH

To fulfill the objective, a search was conducted to determine if an adequate computer program was available. Both NAVPERSRANDCEN resources and those programs available commercially were examined. Since an adequate program was not found, the development of a comprehensive computer program was undertaken. This program was designed to satisfy the data analysis requirements of the ongoing ECHO project and to be useful to other investigators employing open-ended question techniques such as ECHO.

RESULTS

A computer program called GURU was developed for use in research projects where responses are generated by unstructured or semistructured methods and then organized through content analysis or categorization. It is designed to run on the PL/I OPTIMIZING COMPILER using the IBM 360/65 monitor.

Potential Applications

The use of a structured data collection technique, such as a forced-choice survey or a checklist, is appropriate when an investigator has knowledge about the relevant variables in the population being studied. With such knowledge, he can ask the right questions to assess the saliency of these variables with respect to a particular topic area.

However, for exploratory research, unstructured, open-ended or semi-structured techniques such as ECHO are more appropriate because they presume that the investigator has limited knowledge about the relevant variables of a particular population. Such techniques can be used to assess the values or basic attitudes held by members of the group by asking the following kinds of questions:

1. What benefits have you obtained from the training program?

2. Why did you decide to leave (or join) the Navy?

3. If you have objections to women serving aboard ship, what are they?

4. What aspects of your job motivate you to produce to full capacity?

5. What turns you off or demotivates you from accomplishing anything on the job?

In research projects where responses are generated by unstructured or semistructured methods and then organized through content analysis or categorization, the GURU computer program described herein is useful in gaining quantitative insights into the data obtained by:

1. Counting responses in each category of classified data, computing percentages and ranks, and printing tables of rank-ordered categories with their titles.

2. For any subgroup in the population, computing and presenting the frequency, percentage, and rank of each category and then computing the difference between groups with the corresponding probabilities.

3. Printing and/or punching the written responses occurring in the individual categories.

4. Comparing different categorizations of the same data.

This means that, from any data base, the user could (1) print tables of rank-ordered categories with their frequencies, percentages, and titles, (2) print the actual responses listed by category and (if desired) provide this listing on subsamples within the total sample, (3) compute the possibility of observing different percentages between the groups in each of the categories, and (4) if the responses were classified two or more times, see how the responses in one category scheme are distributed over a second set of categories.

If, for example, the work motivation of employees was being investigated by asking open-ended questions, the responses to the questions would be categorized and the classification codes entered on input data cards. The GURU program would provide descriptive statistics (i.e., the frequency and percentage of responses in each category, lists of response category hierarchies, and lists of classified responses). In this example, the program might show that (1) 40 percent of the respondents find supervision is demotivating them from contributing to their full capacity, (2) 20 percent object to the plant facilities (inadequate lighting, cold buildings, etc.), or (3) 25 percent are satisfied with the pay they receive. The investigator is observing values and attitudes emitted from a population in response to open-ended questions. The descriptive information supplied by the program provides an index of the saliency of these feelings. The program would also allow the user flexibility in comparing various groups of respondents. For example, if the sample included first-level supervisors, as well as workers, the investigator could analyze data from the two groups separately, look at the similarities and differences in their respective response patterns, and print the actual responses occurring in each category under the appropriate category title. This permits inspection of the raw data in its categorized form. This option can be especially effective in providing individuals (e.g., management), a clear picture of the data making up each of the categories. Additionally, if more than one data categorization is conducted, either by using the same category scheme or generating new categories, the user can compare the two.

This program also has potential utility in the area of training research. In addition to obtaining objective performance criteria, an investigator may desire to incorporate a trainee's subjective reactions into the evaluation of a new training program. Thus, after training is complete, a trainee could be asked to respond to an open-ended technique of gathering information such as writing a paragraph, listing remarks, responding in an interview, etc., to tap their attitudes/reactions to the training. Once this information has been collected, it can be categorized as previously described in the worker attitude example. At this point the computer program can provide the descriptive statistics, as well as the options of (1) making group comparisons, (2) listing the actual responses, and (3) providing classification comparisons.

Program Description

The GURU program is comprised of four major components: (1) UNIKOUNT, (2) PERZPROB, (3) PRINT/PUNCH, and (4) CATCOMP. These components are described in detail in the following paragraphs.

UNIKOUNT

As described in Barthol and de Mille (1969), "program UNIKOUNT counts responses in each category and respondents having at least one response in each category." A sample output is provided in Appendix A.

UNIKOUNT aggregates all occurrences of each category number punched in a specified reference field in the data cards to arrive at the response frequency for each category (one card per response). The program computes the percentage that each response frequency represents of the total number of responses in the analysis. The categories are ordered by card frequency and rank numbers are computed with rank number 1 (or lowest tie number) being assigned to the category having the most cards. A rank-ordered table is printed, with each row showing the category number, card frequency, percentage, rank number, and title for that category (see page A-4 and A-5). The output also includes the following: (1) main title describing the input data (e.g., "WAREHOUSEMEN"), (2) title of the ref ice field, (3) total number of data records input, (4) number of error car . any (i.e., cards having no category number punched in the specified re nce field), and (5) cards remaining in the analysis after deletion of the error cards.

The second UNIKOUNT table shows, for each category, the number of respondents (subjects) having at least one response in that category (see pages A-6 and A-7). To arrive at this information, UNIKOUNT excludes any redundant responses for each subject in the category; that is, one response is excluded every time the unique identification number of the respondent is repeated in one category. The program computes the percentage that the resulting frequency represents of the total number of respondents found in the sample, and displays these percentages in a table similar to the first UNIKOUNT table. The number of redundant responses excluded is also noted.

PERZPROB

The PERZPROB option (see Barthol & de Mille, 1969) computes the probability of observing different percentages of two groups of respondents or response cards represented in a category. A sample output is provided in Appendix B.

The program prints a percentage table (pages B-1 and B-2) showing categories as rows and groups of respondents or response cards as columns. Each cell of the table shows the percentage of one group of respondents (subjects) or response cards that were represented in one category.

It also prints a probability table (pages B-3 and B-4) showing the computed probabilities that a percentage difference as large as the observed difference between each pair of groups, in each category, might have arisen by chance. Since the different groups do not include the same respondents, the percentages or proportions are independent, and the sampling distribution of the difference between two proportions is approximately normal.

The formula for computing Z, the normal deviate (Wallis & Roberts, 1956) is:

$$Z = \frac{P_1 - P_2}{\sqrt{(PQ)\ (1/n_1 + 1/n_2)}}$$

where

P_1 = the proportion of one group represented in the category,

P_2 = the proportion of the other group represented in the category,

n_1 = the number of respondents in the first group,

n_2 = the number of respondents in the second group,

$$P = \frac{n_1 p_1 + n_2 p_2}{n_1 + n_2}$$

and

$$Q = 1 - P.$$

Since the possible values of $p_1 - p_2$ are not continuously distributed, a correction for discreteness is applied before the probability is computed, reducing by 0.5 the frequency corresponding to the larger proportion ($n_1 p_1$) and increasing by 0.5 the frequency corresponding to the smaller proportion ($n_2 p_2$).

In certain instances, the user may have an extreme p or q or small N. Thus, the program follows this rule: p or q, whichever is smaller, multiplied by N_1 or N_2, whichever is smaller, must result in a product greater than 5 (Downey & Heath, 1974). If the product is less than or equal to 5, the program prints zeros followed by an asterisk where the probability would normally be printed. This is done because computing Z on comparisons where this product is 5 or less is inappropriate. The Fisher's exact test is the correct test for probability with such comparisons (Hays, 1963). Package computer programs are available for this test. However, if the product is greater than 5, the program computes and prints the probability associated with that comparison. Each value in the probability table is two tailed, including the probability of the difference $p_1 > p_2$, as well as the difference $p_2 > p_1$.

Additional information associated with the percentage table includes: (1) main title (e.g., "WAREHOUSEMEN"), (2) number and title of each category, (3) number and title of each group, and (4) the number of respondents or response cards in each group.

Each column in the probability table compares two groups in the percentage table. The difference associated with the comparison is followed by the probability. After selecting a suitable probability level (for example, 0.50, 0.010, 0.001), the user can circle the cells in the probability table that meet that criterion. The circled cells then constitute an index of significant differences in the percentage table.[2]

The input data for PERZPROB are the percentages printed in the UNIKOUNT table. Thus, UNIKOUNT must be run to obtain PERZPROB output. However, there is an option in GURU to suppress the printing of the UNIKOUNT tables.

PRINT/PUNCH

GURU is capable of printing or punching individual verbal responses. For example, if responses were categorized and the user wanted to see the actual responses represented by the frequencies and percentages displayed in UNIKOUNT, the verbal responses could be listed by category (see Appendix C). This option is particularly useful when the user is interested in seeing the raw data displayed in classified form. Punching may be useful if the user wants to create a new data set for reclassification.

CATCOMP

CATCOMP allows the user to compare two or more classifications of the same data. The program provides a listing of how the responses in each category of one classification are distributed among the categories of one or more other classifications and prints the associated category titles, frequencies, and percentages for each (see sample output in Appendix D).

---

[2]It has been stated that the group percentages are independent because the different groups do not contain the same respondents. This means that any entry in the probability table refers to a comparison of two independent percentages. A word of caution is necessary, however, when the user circles two or more entries in the same column of the probability table. The degree of dependence of entries in the same column is indeterminate but should not be assumed to be zero. Conditions tending to introduce dependence are: (1) each respondent may be represented in several cells of the column, (2) each respondent is allowed a limited number of responses, which may be distributed across a larger number of cells, and (3) the cells themselves may be assumed not to exist prior to, or independent of, classification of the responses into categories. Because of this intracolumn dependence, the user should not assume a simple correspondence between the number of cells circled in a column in the probability table and the degree of dissimilarity between two groups of respondents.

It is also capable of comparing classification schemes within subsamples. For example, an investigator may have a set of 1000 responses that have been independently categorized by two classifiers producing separate classifications C_1 and C_2. If C_1 and C_2 were comprised of 20 and 26 categories respectively, the user might be interested in empirically comparing the classifications to get an index of the reliability between the two classifications or to study category formation/classification strategies of the different classifiers. The CATCOMP program will take a particular classification C_1 and show by category how the responses within each of the categories are distributed over another set of categories C_2. It could show that the responses in category 3 for C_1 all occurred in category 8 for C_2 or that the 50 responses in category 5 for C_1 were distributed over several categories in C_2. A comparison of the corresponding category titles of the two classification schemes could provide the investigator useful information about what the classifier in C_1 called the same responses classified by the classifier in C_2. Information concerning classifiers' perceptions and level of abstraction during categorization may be derived. Additionally, the investigator could make similar comparisons between classifications on specified subsamples of the population of responses. Such a comparison is demonstrated in Appendix D.

Input Data Format

The GURU program reads fixed length data records of any size. The columns for each classification must be the same on each record. Each subject can have any number of classification records. This would be used if a subject was asked more than one question or a single question more than one time.

If the option to PRINT/PUNCH the actual verbal responses is chosen, those responses should be on fixed length data records of any size (not necessarily the same record size as the classification records), in a separate file. An identification field of consecutive columns must appear on both the classification cards and the verbal response cards. However, these fields need not be in the same columns on both types of records.

Capabilities and Limitations

The capabilities and limitations of the GURU program are listed below:

1. GURU allows 999 runs to be made in a single submittal.
2. Run title cards allow for an 80-character description.
3. There may be up to 10 groups per run.
4. Group title cards allow for a 40-character description.

5. There may be 20 numeric values for each group.

6. The program can group nonconsecutive values.

7. The reference field title may be up to 40 characters.

8. If there is more than one classificiation record per subject, those records must be located together on the input file, not necessarily in any sorted order. This requirement applies when the percent subjects option is chosen.

Usage

The necessary job control language to execute GURU is provided in Appendix E. A sample of a source listing is provided in Appendix F.

REFERENCES

Barthol, R. P., & Bridge, R. G. The ECHO multi-response method for surveying value and influence patterns in groups. Psychological Reports, 1968, 22, 1345-1354.

Barthol, R. P., & de Mille, R. Project ECHO: Final Report (DAHC04-C-0018-2). Santa Barbara, CA: General Research Corporation, 1969.

Downey, N. M., & Heath, R. W. Basic statistical methods (4th ed.). New York: Harper and Row, 1974.

Hays, W. L. Statistics for psychologists. New York: Holt, Rinehart, and Winston, Inc., 1963.

Wallis, W. A., & Roberts, H. B. Statistics, a new approach, Glenroe, ILL: The Free Press, 1956.

APPENDIX A

GURU INPUT DATA AND UNIKOUNT SAMPLE OUTPUT

```
NUMBER OF RUNS TO BE MADE=001
LOGICAL RECORD LENGTH FOR CONTROL CARD=0100
CONTROL CARD ID COLUMN=81
CONTROL CARD ID WIDTH=19
SUBJECT NUMBER COLUMN=003  SUBJECT NUMBER WIDTH=03
RESPONSE CARD=1
RESPONSE CARD LOGICAL RECORD LENGTH=0200
RESPONSE CARD ID COLUMN=181
RESPONSE CARD ID WIDTH=19
NUMBER OF RESPONSE SECTIONS=2
1 80 101 80
SAMPLE RUN OF GURU PROGRAM
NUMBER OF GROUPS FOR THIS RUN=02
NUMBER OF CATEGORIES=32
MAXIMUM NUMBER EXPECTED FOR ANY ONE GROUP=00500
NUMBER OF GROUP VALUES TO FOLLOW
6 4
GROUP VALUES TO FOLLOW
30111 30321 30112 03033 30113 03013
30211 03052 30212 30322
GROUP TITLES   (40 CHARACTERS)
BROADWAY WAREHOUSEMEN
ANNEX WAREHOUSEMEN
GROUP COLUMN=047    GROUP WIDTH=005
CATEGORY VALUES AND TITLES TO FOLLOW
     01          MISCELLANEOUS
     02          SELF-IMPROVEMENT
     03          JOB SECURITY
     04          TAKING ON NEW RESPONSIBILITY
     05          HAVING A JOB / WORK TO DO EVERYDAY
     06          DOESN'T ANSWER QUESTION
     07          TIME
     08          PAY / FRINGE BENEFITS
     09          WORKING CONDITIONS
     10          THE OPPORTUNITY FOR ADVANCEMENT
     11          KNOWING JOB IS IMPORTANT TO OVERALL NSC MISSION
     12          ROTATION / TRAINING / LEARNING NEW THINGS
     13          MY OWN ATTITUDE
     14          NEGATIVE RESPONSES / GRIPES
     15          NOTHING / NONE
     16          ACCOMPLISHING A GOOD DAYS WORK
     17          MEETING TIME REQUIREMENTS
     18          SUPERVISOR STAYING OFF MY BACK
     19          WEEKENDS OFF
     20          BEING ABLE TO DEPEND ON CO-WORKERS-COOPERATIVE/RELIABLE  CO-WO 5
     21          HELPING/BE HELPED -CO-WORKERS
     22          LIKING PARTICULAR PARTS OF THE JOB
     23          THE CHALLENGE OF GETTING THE WORK OUT
     24          LIKING THE JOB
     25          RECOGNITION / FEEDBACK
     26          WORKING WITH OTHERS
     27          BEING MY OWN BOSS
     28          TAKING PRIDE / SELF-SATISFACTION IN MY WORK
     29          HAVING RESPECT OF OTHERS
     30          SUPERVISION
     31          SAME
     32          KNOWING THAT THERE IS A JOB TO BE DONE
PRINT=1   PUNCH=0   LIST=1
COMPUTE=1  % CARDS=1  % SUBJECTS=1 UNIKOUNT ONLY=0 PRINT UNIKOUNT=1
CATEGORY COMPARISON=1
```

```
REFERENCE FIELD TITLE= INDIGENOUS CLASSIFICATION
REFERENCE FIELD COLUMN=0014
REFERENCE FIELD WIDTH=002
NUMBER OF OTHER CLASSIFICATIONS FOR CATCOMP=1
MAXIMUM NUMBER OF CATEGORIES IN THE OTHER CLASSIFICATIONS=030
PROFESSIONAL CLASSIFICATION              00110002030
      01          MISCELLANEOUS
      02          THE THREAT OF A REPRIMAND  NEGATIVE MOTIVATOR
      03          MY POSITIVE ATTITUDE
      04          WORKING WITH OTHER PEOPLE
      05          THE RELIABILITY OF OTHERS SO I CAN GET MY JOB DONE
      06          BEING HELPFUL AND COOPERATIVE ON THE JOB
      07          MEETING DEADLINES
      08          OTHERS COOPERATING WITH ME
      09          THE CONDITIONS OF WORK
      10          PRIDE IN DOING GOOD WORK
      11          KNOWING THE IMPORTANCE OF MY JOB
      12          NOTHING ABOUT THIS JOB
      13          NEGATIVE RESPONSES
      14          TIME
      15          SAME
      16          BEING ABLE TO SUPERVISE/LEAD OTHERS
      17          THE JOB ITSELF
      18          FREEDOM ON THE JOB
      19          BEING PROMOTED/IMPROVING MY WORK STATUS
      20          DOING THE RIGHT AMOUNT OF WORK
      21          LEARNING MORE ABOUT MY WORK/SELF IMPROVEMENT
      22          KNOWING THERE IS WORK TO BE DONE
      23          GOOD SUPERVISION
      24          DOING MY JOB WELL
      25          RECEIVE MONEY/FRINGE BENEFITS
      26          GETTING AWAY FROM WORK
      27          THINGS OTHER THAN THE WORK ITSELF
      28          RECEIVING RECOGNITION
      29          PERFORMING SPECIFIC ASPECTS OF THE JOB
      30          A FEELING OF JOB SECURITY
```

```
GROUP    1013 COULD NOT BE FOUND
GROUP       0 COULD NOT BE FOUND
GROUP       0 COULD NOT BE FOUND
GROUP    1013 COULD NOT BE FOUND
GROUP       0 COULD NOT BE FOUND
GROUP       0 COULD NOT BE FOUND
GROUP       0 COULD NOT BE FOUND
GROUP       0 COULD NOT BE FOUND
GROUP       0 COULD NOT BE FOUND
GROUP       0 COULD NOT BE FOUND
GROUP       0 COULD NOT BE FOUND
GROUP       0 COULD NOT BE FOUND
GROUP   32012 COULD NOT BE FOUND
GROUP   32012 COULD NOT BE FOUND
GROUP   30213 COULD NOT BE FOUND
GROUP    1013 COULD NOT BE FOUND
GROUP    5311 COULD NOT BE FOUND
GROUP    1013 COULD NOT BE FOUND
GROUP   30213 COULD NOT BE FOUND
GROUP   30213 COULD NOT BE FOUND
GROUP   30213 COULD NOT BE FOUND
GROUP   30213 COULD NOT BE FOUND
GROUP   30213 COULD NOT BE FOUND
GROUP   30213 COULD NOT BE FOUND
GROUP    3042 COULD NOT BE FOUND
GROUP    3021 COULD NOT BE FOUND
GROUP    3021 COULD NOT BE FOUND
GROUP       0 COULD NOT BE FOUND
GROUP    3021 COULD NOT BE FOUND
GROUP   30213 COULD NOT BE FOUND
GROUP       0 COULD NOT BE FOUND
GROUP       0 COULD NOT BE FOUND
GROUP       0 COULD NOT BE FOUND
GROUP       0 COULD NOT BE FOUND
GROUP       0 COULD NOT BE FOUND
GROUP       0 COULD NOT BE FOUND
GROUP       0 COULD NOT BE FOUND
GROUP       0 COULD NOT BE FOUND
GROUP    1013 COULD NOT BE FOUND
GROUP    1013 COULD NOT BE FOUND
GROUP   30213 COULD NOT BE FOUND
GROUP   30213 COULD NOT BE FOUND
GROUP   30213 COULD NOT BE FOUND
GROUP    3042 COULD NOT BE FOUND
GROUP       0 COULD NOT BE FOUND
GROUP    3042 COULD NOT BE FOUND
GROUP    3051 COULD NOT BE FOUND
GROUP       0 COULD NOT BE FOUND
GROUP       0 COULD NOT BE FOUND
GROUP       0 COULD NOT BE FOUND
GROUP    3021 COULD NOT BE FOUND
GROUP    3021 COULD NOT BE FOUND
GROUP    3021 COULD NOT BE FOUND
GROUP    3021 COULD NOT BE FOUND
GROUP    3021 COULD NOT BE FOUND
GROUP       0 COULD NOT BE FOUND
GROUP   30213 COULD NOT BE FOUND

NUMBER IN GROUP  1 IS   470
NUMBER IN GROUP  2 IS   350
```

Note: This page is a listing of missing cases.

SAMPLE RUN OF GURU PROGRAM

REFERENCE FIELD - INDIGENOUS CLASSIFICATION

GROUP NAME = ANNEX WAREHOUSEMEN

TOTAL CARDS	350
NUMBER OF CARDS WITH ERRORS	216
NUMBER OF CARDS REMAINING IN ANALYSIS	134

CATEGORY	RAW FREQUENCY	PERCENT CARDS	RANK	CATEGORY TITLE
26	12	9.0	1.0	WORKING WITH OTHERS
15	10	7.5	2.0	NOTHING / NONE
28	10	7.5	2.0	TAKING PRIDE / SELF-SATISFACTION IN MY WORK
8	8	6.0	5.0	PAY / FRINGE BENEFITS
23	8	6.0	5.0	THE CHALLENGE OF GETTING THE WORK OUT
24	8	6.0	5.0	LIKING THE JOB
30	7	5.2	7.0	SUPERVISION
25	6	4.5	8.0	RECOGNITION / FEEDBACK
20	6	4.5	8.0	BEING ABLE TO DEPEND ON CO-WORKERS-COOPERATIVE/RELIABLE CO-WO 5
16	5	3.7	11.0	ACCOMPLISHING A GOOD DAYS WORK
10	5	3.7	11.0	THE OPPORTUNITY FOR ADVANCEMENT
17	5	3.7	11.0	MEETING TIME REQUIREMENTS
1	4	3.0	15.0	MISCELLANEOUS
18	4	3.0	15.0	SUPERVISOR STAYING OFF MY BACK
2	4	3.0	15.0	SELF-IMPROVEMENT
9	4	3.0	15.0	WORKING CONDITIONS
14	4	3.0	15.0	NEGATIVE RESPONSES / GRIPES
32	4	3.0	15.0	KNOWING THAT THERE IS A JOB TO BE DONE
12	3	2.2	20.0	ROTATION / TRAINING / LEARNING NEW THINGS
21	3	2.2	20.0	HELPING/BE HELPED -CO-WORKERS
4	3	2.2	20.0	TAKING ON NEW RESPONSIBILITY

22	2	1.5	23.0	LIKING PARTICULAR PARTS OF THE JOB
19	2	1.5	23.0	WEEKENDS OFF
27	2	1.5	23.0	BEING MY OWN BOSS
5	1	0.7	27.0	HAVING A JOB / WORK TO DO EVERYDAY
7	1	0.7	27.0	TIME
11	1	0.7	27.0	KNOWING JOB IS IMPORTANT TO OVERALL NSC MISSION
6	1	0.7	27.0	DOESN'T ANSWER QUESTION
29	1	0.7	27.0	HAVING RESPECT OF OTHERS

UNIKOUNT

SAMPLE RUN OF GURU PROGRAM

REFERENCE FIELD - INDIGENOUS CLASSIFICATION

GROUP NAME = ANNEX WAREHOUSEMEN

CATEGORY	RAW FREQUENCY	PERCENT SUBJECTS	RANK	CATEGORY TITLE
26	11	37.9	1.0	WORKING WITH OTHERS
28	10	34.5	2.0	TAKING PRIDE / SELF-SATISFACTION IN MY WORK
8	7	24.1	3.0	PAY / FRINGE BENEFITS
24	7	24.1	3.0	LIKING THE JOB
25	6	20.7	6.0	RECOGNITION / FEEDBACK
23	6	20.7	6.0	THE CHALLENGE OF GETTING THE WORK OUT
30	6	20.7	6.0	SUPERVISION
16	5	17.2	8.0	ACCOMPLISHING A GOOD DAYS WORK
20	5	17.2	8.0	BEING ABLE TO DEPEND ON CO-WORKERS-COOPERATIVE/RELIABLE CO-WO 5
10	4	13.8	12.0	THE OPPORTUNITY FOR ADVANCEMENT
1	4	13.8	12.0	MISCELLANEOUS
2	4	13.8	12.0	SELF-IMPROVEMENT
14	4	13.8	12.0	NEGATIVE RESPONSES / GRIPES
32	4	13.8	12.0	KNOWING THAT THERE IS A JOB TO BE DONE
17	4	13.8	12.0	MEETING TIME REQUIREMENTS
12	3	10.3	18.0	ROTATION / TRAINING / LEARNING NEW THINGS
21	3	10.3	18.0	HELPING/BE HELPED -CO-WORKERS
18	3	10.3	18.0	SUPERVISOR STAYING OFF MY BACK
9	3	10.3	18.0	WORKING CONDITIONS
4	3	10.3	18.0	TAKING ON NEW RESPONSIBILITY
22	2	6.9	21.0	LIKING PARTICULAR PARTS OF THE JOB
19	2	6.9	21.0	WEEKENDS OFF
15	1	3.4	26.0	NOTHING / NONE
11	1	3.4	26.0	KNOWING JOB IS IMPORTANT TO OVERALL NSC MISSION

5	1	3.4	26.0	HAVING A JOB / WORK TO DO EVERYDAY
6	1	3.4	26.0	DOESN'T ANSWER QUESTION
7	1	3.4	26.0	TIME
27	1	3.4	26.0	BEING MY OWN BOSS
29	1	3.4	26.0	HAVING RESPECT OF OTHERS

NUMBER OF SUBJECTS 29

NUMBER EXPELLED 21

APPENDIX B

PERZPROB SAMPLE OUTPUT

PERZPROB

SAMPLE RUN OF GURU PROGRAM

RUN FOR PERCENT CARDS

NUMBER OF CATEGORIES 32

NUMBER OF GROUPS 2

BROADWAY WAREHOUSEMEN
ANNEX WAREHOUSEMEN

		GROUP	1	2
		NUMBER IN GROUP	219	134
	CATEGORY INDEX AND TITLE		PERCENTAGES	
1	MISCELLANEOUS		2.7	3.0
2	SELF-IMPROVEMENT		1.4	3.0
3	JOB SECURITY		1.4	0.0
4	TAKING ON NEW RESPONSIBILITY		0.9	2.2
5	*HAVING A JOB / WORK TO DO EVERYDAY*		4.6	0.7
6	DOESN'T ANSWER QUESTION		0.9	0.7
7	TIME		3.7	0.7
8	PAY / FRINGE BENEFITS		6.4	6.0
9	WORKING CONDITIONS		1.4	3.0
10	THE OPPORTUNITY FOR ADVANCEMENT		3.2	3.7
11	KNOWING JOB IS IMPORTANT TO OVERALL NSC MISSION		1.8	0.7
12	ROTATION / TRAINING / LEARNING NEW THINGS		4.6	2.2
13	MY OWN ATTITUDE		0.5	0.0
14	NEGATIVE RESPONSES / GRIPES		1.4	3.0
15	NOTHING / NONE		8.7	7.5
16	ACCOMPLISHING A GOOD DAYS WORK		5.5	3.7
17	MEETING TIME REQUIREMENTS		0.0	3.7
18	SUPERVISOR STAYING OFF MY BACK		1.8	3.0
19	WEEKENDS OFF		0.9	1.5

20	BEING ABLE TO DEPEND ON CO-WORKERS-COOPERATIVE/RELIABLE	CO-WO 5	2.7	4.5
21	HELPING/BE HELPED -CO-WORKERS		2.7	2.2
22	LIKING PARTICULAR PARTS OF THE JOB		3.2	1.5
23	THE CHALLENGE OF GETTING THE WORK OUT		5.9	6.0
24	LIKING THE JOB		5.3	6.0
25	RECOGNITION / FEEDBACK		6.4	4.5
26	WORKING WITH OTHERS		5.5	9.0
27	BEING MY OWN BOSS		0.0	1.5
28	TAKING PRIDE / SELF-SATISFACTION IN MY WORK		7.8	7.5
29	HAVING RESPECT OF OTHERS		0.0	0.7
30	SUPERVISION		1.8	5.2
31	SAME		5.5	0.0
32	KNOWING THAT THERE IS A JOB TO BE DONE		1.4	3.0

TWO-TAILED PROBABILITY OF A DIFFERENCE OF THIS MAGNITUDE ARISING BY CHANCE

(* INDICATES THAT THE PROBABILITY TEST FOR THIS COMPARISON IS INAPPROPRIATE (SEE DOCUMENTATION))

1/ 2

1	0.002	0.000	*
2	0.016	0.000	*
3	0.014	0.000	*
4	0.013	0.000	*
5	0.038	0.000	*
6	0.002	0.000	*
7	0.029	0.000	*
8	0.004	0.946	
9	0.016	0.000	*
10	0.005	0.000	*
11	0.011	0.000	*
12	0.023	0.000	*
13	0.005	0.000	*
14	0.016	0.000	*
15	0.012	0.839	
16	0.017	0.625	
17	0.037	0.000	*
18	0.012	0.000	*
19	0.006	0.000	*
20	0.017	0.000	*
21	0.005	0.000	*
22	0.017	0.000	*
23	0.000	0.827	
24	0.005	0.965	
25	0.019	0.604	
26	0.035	0.298	

APPENDIX C

PRINT/PUNCH SAMPLE OUTPUT

RESPONSES TO CATEGORIES WITHIN GROUPS

BROADWAY-WAREHOUSEMEN

1 MISCELLANEOUS

NO COMMENT
MORE WORK
A HEAD AND DO IT INSTEAD OF LETTING SOMEONE ELSE DO IT
WAREHOUSEMAN TO WORK WITH
COMPETITION
FOR GETTING OLD
I HAD EXPERIENCE IN MY JOB WHEN I WAS IN THE NAVY
20 YEARS IN THE NAVY
OK BUT WHEN IAM PICKING UP THE PLT AND MOVE THING TO THE FLOOR
NO COMMENT
RECESSION
YOUNG PEOPLE

2 SELF-IMPROVEMENT

A STRONG DESIRE TO GET SOMEPLACE
MYSELF
BETTER FUTURE IF LOOKING FOR THAT
USE PAY FOR OUTSIDE INVESTMENTS
WHEN THERE IS TALK OF UPWARD MOBILITY AND I THINK SOMETHING IS GOING TO HAPPEN
NSC

3 JOB SECURITY

THE FEELING THAT I HAVE A SECURE JOB
MY JOB
A PLACE TO GO
ME
HOPES OF BECOMING A FULL TIME/PERMANENT EMPLOYEE THROUGH DOING WELL IN WORK
MY OWN AMBITION

4 TAKING ON NEW RESPONSIBILITY

32 KNOWING THAT THERE IS A JOB TO BE DONE

THE NEED TO GET THE JOB DONE
WHEN MATERIAL BACKS UP
KNOWING THAT THE JOB HAS TO BE DONE
ME
TO ACCOMPLISH THE JOB
NO COMMENT
THE WORK HAS TO BE DONE IF NOT DONE BY ME IT PILES UP AND STILL HAS TO BE DONE
NO COMMENT

Note: Output for categories 5-31 have been omitted to save space.

APPENDIX D

CATCOMP SAMPLE OUTPUT

CATEGORY COMPARISONS FOR - SAMPLE RUN OF GURU PROGRAM
REFERENCE FIELD - INDIGENOUS CLASSIFICATION

CATEGORY 1 - MISCELLANEOUS
CATEGORY FREQUENCY 10

GROUP 1 BROADWAY WAREHOUSEMEN
GROUP 1 FREQUENCY 6

PROFESSIONAL CLASSIFICATION CATEGORY NUMBER AND TITLE		FREQUENCY	% CATEGORY 1	% CATEGORY 1 IN GROUP 1
1	MISCELLANEOUS	4	40.00	66.67
5	THE RELIABILITY OF OTHERS SO I CAN GET MY JOB DONE	1	10.00	16.67
17	THE JOB ITSELF	1	10.00	16.67

GROUP 2 ANNEX WAREHOUSEMEN
GROUP 2 FREQUENCY 4

PROFESSIONAL CLASSIFICATION CATEGORY NUMBER AND TITLE		FREQUENCY	% CATEGORY 1	% CATEGORY 1 IN GROUP 2
1	MISCELLANEOUS	2	20.00	50.00
2	THE THREAT OF A REPRIMAND NEGATIVE MOTIVATOR	1	10.00	25.00
3	MY POSITIVE ATTITUDE	1	10.00	25.00

CATEGORY COMPARISONS FOR - SAMPLE RUN OF GURU PROGRAM
REFERENCE FIELD - INDIGENOUS CLASSIFICATION

CATEGORY 2 - SELF-IMPROVEMENT
CATEGORY FREQUENCY 7

GROUP 1 BROADWAY WAREHOUSEMEN
GROUP 1 FREQUENCY 3

PROFESSIONAL CLASSIFICATION CATEGORY NUMBER AND TITLE		FREQUENCY	% CATEGORY 2	% CATEGORY 2 IN GROUP 1
19	BEING PROMOTED/IMPROVING MY WORK STATUS	2	28.57	66.67
21	LEARNING MORE ABOUT MY WORK/SELF IMPROVEMENT	1	14.29	33.33

GROUP 2 ANNEX WAREHOUSEMEN
GROUP 2 FREQUENCY 4

PROFESSIONAL CLASSIFICATION CATEGORY NUMBER AND TITLE		FREQUENCY	% CATEGORY 2	% CATEGORY 2 IN GROUP 2
3	MY POSITIVE ATTITUDE	1	14.29	25.00
21	LEARNING MORE ABOUT MY WORK/SELF IMPROVEMENT	3	42.86	75.00

CATEGORY COMPARISONS FOR - SAMPLE RUN OF GURU PROGRAM
REFERENCE FIELD - INDIGENOUS CLASSIFICATION

CATEGORY 3 - JOB SECURITY
CATEGORY FREQUENCY 3

GROUP 1 BROADWAY WAREHOUSEMEN
GROUP 1 FREQUENCY 3

PROFESSIONAL CLASSIFICATION CATEGORY NUMBER AND TITLE		FREQUENCY	% CATEGORY 3	% CATEGORY 3 IN GROUP 1
19	BEING PROMOTED/IMPROVING MY WORK STATUS	1	33.33	33.33
22	KNOWING THERE IS WORK TO BE DONE	1	33.33	33.33
30	A FEELING OF JOB SECURITY	1	33.33	33.33

GROUP 2 ANNEX WAREHOUSEMEN
GROUP 2 FREQUENCY 0

CATEGORY 4 - TAKING ON NEW RESPONSIBILITY
CATEGORY FREQUENCY 5

GROUP 1 BROADWAY WAREHOUSEMEN
GROUP 1 FREQUENCY 2

PROFESSIONAL CLASSIFICATION CATEGORY NUMBER AND TITLE		FREQUENCY	% CATEGORY 4	% CATEGORY 4 IN GROUP 1
18	FREEDOM ON THE JOB	2	40.00	100.00

GROUP 2 ANNEX WAREHOUSEMEN
GROUP 2 FREQUENCY 3

PROFESSIONAL CLASSIFICATION CATEGORY NUMBER AND TITLE		FREQUENCY	% CATEGORY 4	% CATEGORY 4 IN GROUP 2
16	BEING ABLE TO SUPERVISE/LEAD OTHERS	2	40.00	66.67
23	GOOD SUPERVISION	1	20.00	33.33

Note: Output for categories 5-30 have been omitted to save space.

CATEGORY COMPARISONS FOR - SAMPLE RUN OF GURU PROGRAM
REFERENCE FIELD - INDIGENOUS CLASSIFICATION

CATEGORY 31 - SAME
CATEGORY FREQUENCY 12

GROUP 1 BROADWAY WAREHOUSEMEN
GROUP 1 FREQUENCY 12

PROFESSIONAL CLASSIFICATION CATEGORY NUMBER AND TITLE		FREQUENCY	% CATEGORY 31	% CATEGORY 31 IN GROUP 1
1	MISCELLANEOUS	1	8.33	8.33
15	SAME	11	91.67	91.67

GROUP 2 ANNEX WAREHOUSEMEN
GROUP 2 FREQUENCY 0

CATEGORY 32 - KNOWING THAT THERE IS A JOB TO BE DONE
CATEGORY FREQUENCY 7

GROUP 1 BROADWAY WAREHOUSEMEN
GROUP 1 FREQUENCY 3

PROFESSIONAL CLASSIFICATION CATEGORY NUMBER AND TITLE		FREQUENCY	% CATEGORY 32	% CATEGORY 32 IN GROUP 1
17	THE JOB ITSELF	1	14.29	33.33
22	KNOWING THERE IS WORK TO BE DONE	2	28.57	66.67

GROUP 2 ANNEX WAREHOUSEMEN
GROUP 2 FREQUENCY 4

PROFESSIONAL CLASSIFICATION CATEGORY NUMBER AND TITLE		FREQUENCY	% CATEGORY 32	% CATEGORY 32 IN GROUP 2
22	KNOWING THERE IS WORK TO BE DONE	2	28.57	50.00
24	DOING MY JOB WELL	2	28.57	50.00

APPENDIX E

GURU JOB CONTROL LANGUAGE

GURU JOB CONTROL LANGUAGE

The necessary job control language to execute GURU follows:

a. //GO EXEC PGM=GURU,REGION=248K,
   // PARM='ISA(100K)'

b. //GO.STEPLIB DD DSN=ECHO,DISP=SHR

c. //GO.SYSIN DD *
   [SETUP CARDS]

d. //GO.SYSPRINT DD SYSOUT=A

e. //GO.PNCH DD SYSOUT=B,DCB=BLKSIZE=80

f.1. //GO.TEMP1 DD DSN=&&TEMP1,
   // UNIT=SYSDA,DISP=NEW,
   // SPACE=(6000,(15,10)),
   // DCB=(RECFM=FB,LRECL=100,BLKSIZE=6000)

   Supply as many of card f.1's as the largest number of groups in any of the runs. E.G. for 3 groups, TEMP1, TEMP2 & TEMP3 need be supplied.

f.2. //GO.TEMP3 DD DUMMY

   Supply f.2's through TEMP10 following the f.1's. E.G. if the maximum number of groups is 3, there should be 3 f.1 cards and 7 f.2 cards.

g. //GO.WORDS DD file containing verbal response records

h. //GO.CARD1 DD file containing classification records

i. //GO.PLIDUMP DD SYSOUT=A

j. /*

SETUP

The following is the general setup for GURU.

All numbers are right-justified.

Card A

Col 27-29 Number of runs to be made.

Card B

Col 40-43 Logical record length for classification records.

Card C

Col 24-25 Classification record identification column.

Card D

Col 23-24 Classification record identification width.

Card E

Col 23-25 Subject number column.

Col 49-50 Subject number width.

Card F

Col 15 Flag for existence of verbal response records.

0 - No

1 - Yes

[CARDS G-K REQUIRED ONLY IF CARD F IS 1]

Card G

Col 37-40 Logical record length for verbal response records.

Card H

Col 25-27 Verbal response record identification column.

Card I

Col 24-25 Verbal response record identification width.

Card J

Col 29 — Number of verbal response sections (e.g., if responses are on two cards with an ID in col 72-80, the number of sections would be 2, card 1, cols 1-72; card 2, cols 1-72).

Card K

Free Format — Verbal response sections followed by their lengths (e.g., in the example for card J, card G would have 160, card J would have 2, and card K would have 1 72 81 72).

Card L

Col 1-80 — Title card for the run.

Card M

Col 31-32 — Number of groups for this run.

Card N

Col 22-23 — Number of categories to be found in the reference field.

Card O

Col 43-47 — Maximum number of records expected for any one group.

Card P

Col 1-80 — "Number of group values to follow"

Card Q

Free format — Number of values per group (e.g., if there are 4 groups in a run, 4 numbers would appear on this card, separated by one or more blanks).

Card R

Col 1-80 — "Group values to follow"

Card(s) S

Free format	Actual group values. One card S must appear for each group. Values appear on the cards separated by a blank (e.g., if there are four groups for this run four S cards, containing the values in each group, must be supplied).

Card T

Col 1-80	"Group titles (40 characters)".

Card(s) U

Col 1-40	Group title. One card U must appear for each group.

Card V

Col 14-16	Starting column of the variable used to split groups.
Col 33-35	Width of that variable.

Card W

Col 1-80	"Category values and titles to follow"

Card(s) X

Col 6-7	Category number.
Col 17-80	Category title. These categories are determined by the classifiers. There should be the same number of X cards as specified on card N.

Card Y

Col 7	Print switch. If any printing or punching of the actual responses is desired, this switch should be set to one.

APPENDIX F

GURU SOURCE LISTING

Col 17 — Punch switch. If the responses are to be punched on cards, this switch should be set to one.

Col 26 — List switch. If the actual responses are to be printed, this switch should be set to one.

Card Z

Col 9 — Compute switch. If UNIKOUNT or PERZPROB is to be executed, this switch should be set to one.

Col 20 — If the UNIKOUNT output for the percent responses is desired, this column should be set to one.

Col 34 — If the UNIKOUNT output for the percent subjects is desired, this column should be set to one.

Col 50 — If only UNIKOUNT is to be performed, this column should be set to one.

Col 67 — If PERZPROB is being performed, in order to get the UNIKOUNT output as well, this column should be set to one.

Card AA

Col 21 — Flag for category comparisons.

0 - No

1 - Yes

Card BB

Col 24-63 — Reference field title.

Card CC

Col 24-27 — Reference field column.

Card DD

Col 23-25 Reference field width.

[CARDS EE-HH REQUIRED ONLY IF CARD AA = 1]

Card EE

Col 45 Number of other classifications to be compared with main reference field.

Card FF

Col 59-61 Maximum number of categories in other classifications.

Card GG

Col 1-40 Classification title

Col 41-44 Classification starting column

Col 45-48 Classification width

Col 49-51 Number of categories in this classification.

Card HH

Col 6-7 Category number.

Col 17-80 Category title. There should be the same number of HH cards as specified in cols 49-51 of the GG card.

NOTE:

1. There should be the same number of FF cards as specified in col 45 in card EE.

2. There should be the same number of cards L-HH as specified in card A.

SOURCE LISTING

```
STMT LEV NT

   1      0  (SUBSCRIPTRANGE):
             GURU: PROC OPTIONS(MAIN);

                /*  THIS PROCEDURE IS THE DRIVER ROUTINE FOR THE GURU PROGRAM.
                    PROGRAMMER:              JANET D. DODSON
                    DATE:                    DECEMBER 23, 1976                    */

   2   1  0       DCL (NRUNS,NGROUPS,PRIN ,PUNCH,LIST,COMPUTE,
                    PERCENT_CARDS,PERCENT_SUBJECTS,UNIKOUNT_ONLY,PRINT_UNIKOUNT)
                    BIN FIXED;
   3   1  0       DCL TITLE CHAR(80);
   4   1  0       DCL (GROUP_COLUMN,GROUP_WIDTH) BIN FIXED;
   5   1  0       DCL (COL1,WID1,COL2,WID2) BIN FIXED;
   6   1  0       DCL GROUP BIN FIXED;
   7   1  0       DCL FILES(10) FILE;
   8   1  0       DCL (TEMP1,TEMP2,TEMP3,TEMP4,TEMP5,
                    TEMP6,TEMP7,TEMP8,TEMP9,TEMP10) FILE;
   9   1  0       DCL NMAX BIN FIXED(31);
  10   1  0       DCL (CARDS,GOODS,BADS) BIN FIXED;
  11   1  0       DCL CAT_COL BIN FIXED;
  12   1  0       DCL CAT_VALUE BIN FIXED;
  13   1  0       DCL Z DEC FLOAT(16);
  14   1  0       DCL CARD3 CHAR(80);
  15   1  0       DCL (SUBCOL,SUBWID) BIN FIXED(15);

  16   1  0         OPEN FILE(SYSIN) OUTPUT;
  17   1  0         OPEN FILE(SYSPRINT) LINESIZE(130);

                /*  ASSIGN TEMPORARY FILES  */

  18   1  0         FILES(1) = TEMP1;
  19   1  0         FILES(2) = TEMP2;
  20   1  0         FILES(3) = TEMP3;
  21   1  0         FILES(4) = TEMP4;
  22   1  0         FILES(5) = TEMP5;
  23   1  0         FILES(6) = TEMP6;
  24   1  0         FILES(7) = TEMP7;
  25   1  0         FILES(8) = TEMP8;
  26   1  0         FILES(9) = TEMP9;
  27   1  0         FILES(10) = TEMP10;

                /*  CONVERSION ERROR BLOCK  */

  28   1  0       DCL (ONCHAR,ONSOURCE) BUILTIN;
  29   1  0       DCL CHAR CHAR(1);
  30   1  0       DCL SOURCE CHAR(100) VARYING;
  31   1  0         ON CONVERSION BEGIN;
  32   2  0           CHAR = ONCHAR;
```

```
    STMT LEV NT

      33   2  0        SOURCE = ONSOURCE;
      34   2  0        PUT SKIP DATA(CHAR,SOURCE);
      35   2  0      END;

                   /*  LIST DECK SETUP  */

      36   1  0      ON ENDFILE(INPUT) GO TO GO_AHEAD;
      37   1  0      DO WHILE('1'B);
      38   1  1        GET FILE(INPUT) SKIP EDIT(CARD3) (A(80)) COPY;
      39   1  1        PUT FILE(SYSIN) SKIP EDIT(CARD3) (A);
      40   1  1      END;

                   /*  READ IN NUMBER OF RUNS TO BE MADE  */

      41   1  0  GO_AHEAD:

                     CLOSE FILE(SYSIN);
      42   1  0      OPEN FILE(SYSIN) INPUT;

      43   1  0      GET EDIT(NRUNS) (X(26),F(3));
      44   1  0      COL2,WID2=1;
      45   1  0      NSECTIONS=1; LRECL2=1;

                   /*  LRECL FOR CONTROL CARD  */

      47   1  0      GET SKIP EDIT(LRECL1) (X(39),F(4));

                   /*  CONTROL ID COLUMN & WIDTH  */

      48   1  0      GET SKIP EDIT(COL1,WID1) (X(23),F(4),SKIP,X(22),F(2));

                   /*  READ IN SUBJECT NUMBER COLUMN & WIDTH  */

      49   1  0      GET SKIP EDIT(SUBCOL,SUBWID) (X(22),F(3),X(23),F(2));

                   /*  IS THERE A RESPONSE CARD ?  */

      50   1  0      GET SKIP EDIT(ICARD2) (X(14),F(1));
      51   1  0      IF ICARD2 ^= 0 THEN DO;

                   /*  RESPONSE LRECL  */

      52   1  1        GET SKIP EDIT(LRECL2) (X(36),F(4));

                   /*  RESPONSE ID COLUMN & WIDTH  */
```

```
PL/I OPTIMIZING COMPILER          (SUBSCRIPTRANGE):

    STMT LEV NT

      53   1  1        GET SKIP EDIT(COL2,WID2) (X(24),F(4),SKIP,X(23),F(2));

                    /*  NUMBER OF SECTIONS TO RESPONSE CARD  */

      54   1  1        GET SKIP EDIT(NSECTIONS) (X(28),F(1));
      55   1  1      END;

      56   1  0    BEGIN;
      57   2  0      DCL CARD CHAR(LRECL1);
      58   2  0      DCL CARD2 CHAR(LRECL2);
      59   2  0      DCL ID1 CHAR(WID1) DEF CARD POS(COL1);
      60   2  0      DCL ID2 CHAR(WID2) DEF CARD2 POS(COL2);
      61   2  0      DCL COLS(NSECTIONS) BIN FIXED(15);
      62   2  0      DCL WIDS(NSECTIONS) BIN FIXED(15);

                    /*  READ IN RESPONSE COLS & WIDTHS  */

      63   2  0      IF ICARD2 ^= 0 THEN
                      GET SKIP LIST((COLS(I),WIDS(I) DO I=1 TO NSECTIONS));
      64   2  0      DO IRUNS = 1 TO NRUNS;
      65   2  1      PUT PAGE;
      66   2  1        CLOSE FILE(FILES(1)),FILE(FILES(2)),FILE(FILES(3)),FILE(FILES(4))
                       ,FILE(FILES(5)),FILE(FILES(6)),FILE(FILES(7)),FILE(FILES(8)),
                        FILE(FILES(9)),FILE(FILES(10));

                    /*  READ IN RUN TITLE  */

      67   2  1        GET SKIP EDIT(TITLE) (A(80));

                    /*  READ IN NUMBER OF GROUPS FOR THIS RUN  */

      68   2  1        GET SKIP EDIT(NGROUPS) (X(30),F(2));

                    /*  READ IN NUMBER OF CATEGORIES FOR THIS RUN  */

      69   2  1      GET SKIP EDIT(ICAT) (X(21),F(2));

                    /*  READ IN MAXIMUM NUMBER EXPECTED FOR ANY ONE GROUP  */

      70   2  1      GET SKIP EDIT(NMAX) (X(42),F(5));

      71   2  1    BEGIN;
      72   3  1      DCL CAT_HOLD(ICAT) BIN FIXED;
      73   3  1      DCL CAT_TITLE_HOLD(ICAT) CHAR(64);
      74   3  1      DCL PER2_CARDS(NGROUPS,ICAT) DEC FLOAT;
      75   3  1      DCL PER2_SUBJS(NGROUPS,ICAT) DEC FLOAT;
      76   3  1      DCL NO(NGROUPS,NMAX) BIN FIXED;
```

```
PL/I OPTIMIZING COMPILER          (SUBSCRIPTRANGE):

     STMT LEVEL NT

       77    3  1    DCL NSUBJECTS(NGROUPS);
       78    3  1    DCL NUMBER_GROUP_VALUES(NGROUPS) BIN FIXED;
       79    3  1    DCL GROUP_VALUES(NGROUPS,20) BIN FIXED(31);
       80    3  1    DCL GROUP_TITLE(NGROUPS) CHAR(40);
       81    3  1    DCL NSUBS(NGROUPS);
       82    3  1    DCL CATEGORY(ICAT) BIN FIXED;
       83    3  1    DCL CAT_TITLE(ICAT) CHAR(64);
       84    3  1    DCL RANK(ICAT);
       85    3  1    DCL CAT(NGROUPS,NMAX) BIN FIXED;
       86    3  1    DCL FREQ(ICAT) BIN FIXED;
       87    3  1    DCL (JONE(200),JTWO(200));
       88    3  1    DCL NOUT(NGROUPS);
       89    3  1    DCL ALL_CAT(ICAT);
       90    3  1    DCL GROUP_CAT(ICAT,NGROUPS);
       91    3  1      NUMBER_GROUP_VALUES = 0;
       92    3  1      GROUP_VALUES = 0;
       93    3  1      GROUP_TITLE = (40) ' ';
       94    3  1      CATEGORY = 0;
       95    3  1      CAT_TITLE = (64) ' ';
       96    3  1      PERZ_CARDS,PERZ_SUBJS=0;
       97    3  1      NSUBJECTS,NSUBS = 0;
       98    3  1      NOUT = 0;
       99    3  1      NO = 0;
      100    3  1      CAT = 0;
      101    3  1      ALL_CAT = 0;
      102    3  1      GROUP_CAT = 0;

                  /*  READ IN NUMBER OF VALUES PER GROUP  */

      103    3  1       GET SKIP;
      104    3  1      GET SKIP LIST((NUMBER_GROUP_VALUES(J) DO J=1 TO NGROUPS));

                  /*  READ IN GROUP VALUES  */

      105    3  1       GET SKIP;
      106    3  1       DO J = 1 TO NGROUPS;
      107    3  2         GET SKIP LIST((GROUP_VALUES(J,K)
                             DO K=1 TO NUMBER_GROUP_VALUES(J)));
      108    3  2       END;

                  /*  READ IN GROUP TITLES  */

      109    3  1       GET SKIP;
      110    3  1       DO J = 1 TO NGROUPS;
      111    3  2         GET SKIP EDIT(GROUP_TITLE(J)) (A(40));
      112    3  2       END;

                  /*  READ IN COLUMN & WIDTH FOR GROUP VARIABLE  */

      113    3  1         GET SKIP EDIT(GROUP_COLUMN,GROUP_WIDTH)
```

```
    STMT LEV NT

                        (X(13),F(3),X(16),F(3));

                   /*  READ IN CATEGORY NUMBERS AND TITLES  */

     114    3  1        GET SKIP;
     115    3  1       DO I=1 TO ICAT;
     116    3  2           GET SKIP EDIT(CATEGORY(I),CAT_TITLE(I))
                          (X(5),F(2),X(9),A(64));
     117    3  2         END;
     118    3  1       CAT_HOLD = CATEGORY;
     119    3  1       CAT_TITLE_HOLD = CAT_TITLE;

                   /*  READ IN PRINT OPTIONS  */

     120    3  1         GET SKIP EDIT(PRIN ,PUNCH,LIST) (X(6),F(1),X(9),F(1),X(8),F(1));

                   /*  READ IN COMPUTE OPTIONS  */

     121    3  1         GET SKIP EDIT(COMPUTE,PERCENT_CARDS,PERCENT_SUBJECTS,
                          UNIKOUNT_ONLY,PRINT_UNIKOUNT)
                          (X(8),F(1),X(10),F(1),X(13),F(1),
                          X(15),F(1),X(16),F(1));

                   /*  READ IN CATEGORY COMPARISON OPTIONS  */

     122    3  1       GET SKIP EDIT(ICOMP) (X(20),F(1));

     123    3  1     DCL RTITLE CHAR(40);
     124    3  1     DCL RWID BIN FIXED(15);

                   /*  READ IN REFERENCE FIELD TITLE  */

     125    3  1       GET SKIP EDIT(RTITLE) (X(22),A(40));

                   /*  READ IN REFERENCE FIELD COLUMN & WIDTH  */

     126    3  1       GET SKIP EDIT(CAT_COL,RWID) (X(23),F(4),SKIP,X(22),F(3));
     127    3  1         CLOSE FILE(CARD1);
     128    3  1         ON ENDFILE(CARD1) GO TO WRITTEN;

                   /*  WRITE DATA INTO TEMPORARY GROUP FILES  */

     129    3  1   MORE:

                          GET FILE(CARD1) SKIP EDIT(CARD) (A(LRECL1));
     130    3  1         GET STRING(CARD) EDIT(GROUP) (X(GROUP_COLUMN-1),F(GROUP_WIDTH));
     131    3  1         DO I = 1 TO NGROUPS;
```

```
    STMT LEV NT

     132    3  2          DO J = 1 TO NUMBER_GROUP_VALUES(I);
     133    3  3            IF GROUP = GROUP_VALUES(I,J) THEN DO;
     134    3  4              PUT FILE(FILES(I)) SKIP EDIT(CARD) (A);
     135    3  4              NOUT(I) = NOUT (I) + 1;
     136    3  4              GO TO MORE; END;
     138    3  3          END;
     139    3  2        END;
     140    3  1        PUT SKIP EDIT('GROUP ',GROUP,' COULD NOT BE FOUND') (A,F(6),A);
     141    3  1      GO TO MORE;

                  /*  PRINT NUMBER IN EACH GROUP  */

     142    3  1  WRITTEN:

                      PUT SKIP(2);
     143    3  1      DO JK=1 TO NGROUPS;
     144    3  2        PUT SKIP EDIT('NUMBER IN GROUP ',JK,' IS ',NOUT(JK))
                          (A,F(2),A,F(5));
     145    3  2      END;
     146    3  1        CLOSE FILE(FILES(1)),FILE(FILES(2)),FILE(FILES(3)),FILE(FILES(4))
                        ,FILE(FILES(5)),FILE(FILES(6)),FILE(FILES(7)),FILE(FILES(8)),
                         FILE(FILES(9)),FILE(FILES(10));
     147    3  1        IF PRIN =1 THEN CALL PRINT;

                  /*  GO ON TO DO THE COMPUTING  */

     148    3  1      IF COMPUTE ^= 1 THEN GO TO NEXT_RUN;
     149    3  1      IF NGROUPS ^= 1 THEN GO TO MULTIPLE_GROUPS;

                  /*  SINGLE GROUP  (NO GROUP BREAKOUTS)  */

     150    3  1      IGROUP=1;
     151    3  1      RANK = 0; FREQ = 0;
     153    3  1      CALL UNIK1;
     154    3  1      IF PERCENT_SUBJECTS ^= 1 THEN GO TO CHECK_CATCOMP;
     155    3  1      RANK = 0; FREQ = 0;
     157    3  1      NSUBS = NSUBJECTS;
     158    3  1      CALL UNIK2;
     159    3  1      GO TO CHECK_CATCOMP;

                  /*  SAMPLE BROKEN INTO GROUPS  */

     160    3  1  MULTIPLE_GROUPS:

                      DO IGROUP = 1 TO NGROUPS;
     161    3  2      RANK = 0; FREQ = 0;
     163    3  2      CALL UNIK1;

     164    3  2  NEXT_GROUP:
```

```
    STMT LEV NT

                      END;
     165    3  1       IF PERCENT_CARDS ^= 1 THEN GO TO SUBJECTS2;
     166    3  1       IF UNIKOUNT_ONLY ^= 1 THEN CALL PERZPRB;

     167    3  1  SUBJECTS2:

                      IF PERCENT_SUBJECTS ^= 1 THEN GO TO CHECK_CATCOMP;
     168    3  1       DO IGROUP=1 TO NGROUPS;
     169    3  2       RANK = 0; FREQ = 0;
     171    3  2       NSUBS = NSUBJECTS;
     172    3  2         CALL UNIK2;
     173    3  2       IF UNIKOUNT_ONLY = 1 THEN  GO TO NEXT_GROUP2;

     174    3  2  NEXT_GROUP2:

                        END;
     175    3  1       IF UNIKOUNT_ONLY ^= 1 THEN CALL PERZPRB;

     176    3  1  CHECK_CATCOMP:

                      IF ICOMP=1 THEN CALL CATCOMP;
```

```
      STMT LEV NT

       177    3  1  PRINT: PROC;

                     /*  THIS PROCEDURE OPERATES ON THE FOLLOWING FLAGS:
                           PUNCH=1, PUNCH RESPONSES TO QUESTIONS WITHIN GROUPS
                           LIST = 1, LIST RESPONSES TO QUESTIONS WITHIN
                             CATEGORIES WITHIN GROUPS  */

       178    4  1       IF ICARD2 ^= 1 THEN
                            DO;
       179    4  2           PUT PAGE EDIT('THERE ARE NO RESPONSE CARDS TO BE PRINTED OR PUN
                          CHED') (A);
       180    4  2           RETURN;
       181    4  2         END;

       182    4  1       IF PUNCH = 1 THEN
                            DO;
       183    4  2           DO I = 1 TO NGROUPS;
       184    4  3             ON ENDFILE(FILES(I)) BEGIN;
       185    5  3               CLOSE FILE(FILES(I)); GO TO NEXT_GROUP; END;
       188    4  3             PUT FILE(PNCH    ) SKIP EDIT(GROUP_TITLE(I)) (A);
       189    4  3             ON ENDFILE(WORDS) BEGIN;
       190    5  3               CLOSE FILE(WORDS); GO TO READ_WORDS; END;

       193    4  3  READ_FILES:

                                 GET FILE(FILES(I)) SKIP EDIT(CARD) (A(LRECL1));

       194    4  3  READ_WORDS:

                                 GET FILE(WORDS) SKIP EDIT(CARD2) (A(LRECL2));
       195    4  3               IF ID1 = ID2 THEN DO;
       196    4  4         DO KK=1 TO NSECTIONS;
       197    4  5           PUT FILE(PNCH) SKIP EDIT(SUBSTR(CARD2,COLS(KK),WIDS(KK))) (A);
       198    4  5         END;
       199    4  4               GO TO READ_FILES;
       200    4  4             END;
       201    4  3             GO TO READ_WORDS;

       202    4  3  NEXT_GROUP:

                              END;
       203    4  2       DO KK=1 TO NSECTIONS;
       204    4  3         PUT FILE(PNCH) SKIP EDIT(SUBSTR(CARD2,COLS(KK),WIDS(KK))) (A);
       205    4  3       END;
       206    4  2         END;
       207    4  1       IF LIST=1 THEN
                            DO;
       208    4  2           DO I = 1 TO NGROUPS;
       209    4  3             CLOSE FILE(FILES(I)),FILE(WORDS);
       210    4  3             ON ENDFILE(FILES(I)) BEGIN;
       211    5  3               CLOSE FILE(FILES(I)); GO TO NEXT_CATEGORY; END;
       214    4  3           PUT PAGE EDIT('RESPONSES TO CATEGORIES WITHIN GROUPS')
                                (COL(45),A);
```

```
    STMT LEV NT

     215   4  3            PUT SKIP(2) EDIT(GROUP_TITLE(I)) (A);
     216   4  3       PUT SKIP(0) EDIT('----------------------------------------') (A);
     217   4  3            ON ENDFILE(WORDS) BEGIN;
     218   5  3              CLOSE FILE(WORDS); GO TO READ_WORDS2;
     220   5  3              END;
     221   4  3          DO J=1 TO ICAT;
     222   4  4            PUT SKIP(2) EDIT(CATEGORY(J),CAT_TITLE(J))
                              (X(5),F(2),X(3),A);
     223   4  4            PUT SKIP(2);

     224   4  4  READ_FILES2:

                            GET FILE(FILES(I)) SKIP EDIT(CARD) (A(LRECL1));
     225   4  4            GET STRING(CARD) EDIT(CAT_VALUE) (X(CAT_COL-1),F(RWID));
     226   4  4             IF CAT_VALUE ^= CATEGORY(J) THEN GO TO READ_FILES2;

     227   4  4  READ_WORDS2:

                            GET FILE(WORDS) SKIP EDIT(CARD2) (A(LRECL2));
     228   4  4            IF ID1 = ID2 THEN DO;
     229   4  5              PUT SKIP EDIT(SUBSTR(CARD2,COLS(1),WIDS(1))) (COL(10),A);
     230   4  5              IF NSECTIONS > 1 THEN DO;
     231   4  6              DO III=2 TO NSECTIONS;
     232   4  7                PUT SKIP EDIT(SUBSTR(CARD2,COLS(III),WIDS(III)))
                                (COL(15),A);
     233   4  7              END;
     234   4  6              END;
     235   4  5              GO TO READ_FILES2;
     236   4  5            END;
     237   4  4            GO TO READ_WORDS2;

     238   4  4  NEXT_CATEGORY:
                       END;  /* END DO J  */
     239   4  3          END;  /*  END DO I   */
     240   4  2        END;  /*  END DO  */
     241   4  1  END PRINT;
```

STATEMENT

```
242   *  1  UNIK1: PROC;

             /*  THIS PROCEDURE IS THE UNIKOUNT PROCEDURE FOR PERCENT CARDS  */

243   4  1    DCL SUBJ BIN FIXED;
244   4  1    DCL CARD4 CHAR(LRECL1);
245   4  1      IN_UNIK1 = 1;
246   4  1      IF PERCENT_CARDS = 0 ! PRINT_UNIKOUNT=0 THEN GO TO PASS1;
247   4  1      PUT PAGE EDIT('UNIKOUNT',TITLE,'REFERENCE FIELD  -  ',RTITLE)
                 (COL(62),A,SKIP(2),COL(20),A,SKIP(2),A,A);
248   4  1      PUT SKIP(2) EDIT('GROUP NAME = ',GROUP_TITLE(IGROUP)) (A);
249   4  1      PUT SKIP(2);
250   4  1      IF NOUT(IGROUP) = 0 THEN DO;
251   4  2        PUT SKIP(2) EDIT('NO CARDS IN THIS GROUP') (A);
252   4  2        RETURN; END;

254   4  1  PASS1:

             ON ENDFILE(FILES(IGROUP)) GO TO BUBBLE;
255   4  1      CLOSE FILE(FILES(IGROUP));
256   4  1      CARDS,GOODS,BADS = 0;
257   4  1      NSUBJECTS(IGROUP) = 0;

             /*  SORT CARDS INTO GOODS AND BADS  */

258   4  1  MORE:

             GET FILE(FILES(IGROUP)) SKIP EDIT(CARD4) (A(LRECL1));
259   4  1      GET STRING(CARD4) EDIT(SUBJ) (X(SUBCOL-1),F(SUBWID));
260   4  1      GET STRING(CARD4) EDIT(CAT_VALUE) (X(CAT_COL-1),F(RWID));
261   4  1      CARDS = CARDS + 1;
262   4  1      IF SUBJ <= 0 ! CAT_VALUE <= 0 THEN
                 DO;
263   4  2          BADS = BADS + 1;
264   4  2          GO TO MORE;
265   4  2        END;
266   4  1      GOODS = GOODS + 1;
267   4  1      NO(IGROUP,GOODS) = SUBJ;
268   4  1      CAT(IGROUP,GOODS) = CAT_VALUE;
269   4  1      JM = FIND(CAT_VALUE);
270   4  1      IF JM = 0 THEN GO TO MORE;
271   4  1      ALL_CAT(JM) = ALL_CAT(JM) + 1;
272   4  1      GROUP_CAT(JM,IGROUP) = GROUP_CAT(JM,IGROUP) + 1;
273   4  1      GO TO MORE;

274   4  1  BUBBLE:

             IF PERCENT_CARDS = 0 ! PRINT_UNIKOUNT=0 THEN GO TO PASS2;
275   4  1      PUT SKIP(2) EDIT('TOTAL CARDS',CARDS,
                 'NUMBER OF CARDS WITH ERRORS',BADS,
                 'NUMBER OF CARDS REMAINING IN ANALYSIS',GOODS)
                 (A,COL(45),F(5),SKIP(2));
```

```
PL/I OPTIMIZING COMPILER          (SUBSCRIPTRANGE):

     STMT LEV NT

      276   4  1  PASS2:

                       NSUBJECTS(IGROUP) = GOODS;
      277   4  1       IF PERCENT_CARDS ^= 1 THEN RETURN;
      278   4  1       FREQ = 0;

                     /*  COUNT CATEGORY FREQUENCIES  */

      279   4  1       DO I = 1 TO GOODS;
      280   4  2         DO J = 1 TO ICAT;
      281   4  3           IF CAT(IGROUP,I) = CATEGORY(J) THEN
                            DO;
      282   4  4             FREQ(J) = FREQ(J) + 1;
      283   4  4             GO TO NEXT_GOOD;
      284   4  4            END;
      285   4  3           END;

      286   4  2  NEXT_GOOD:

                     END;
      287   4  1       CALL DOWN2(FREQ,CATEGORY,CAT_TITLE,ICAT);
      288   4  1       DO I = 1 TO ICAT;
      289   4  2         IF FREQ(I) = 0 THEN GO TO FOUND2;
      290   4  2       END;
      291   4  1       L = I + 1;

      292   4  1  FOUND2:

                       L = I - 1;
      293   4  1       CALL RANKS(FREQ,RANK,L);
      294   4  1       IF PRINT_UNIKOUNT = 1 THEN
                       PUT SKIP(2) EDIT('CATEGORY','RAW FREQUENCY',
                         'PERCENT CARDS','RANK','CATEGORY TITLE')
                         (A,COL(14),A,COL(30),A,COL(48),A,COL(55),A);
      295   4  1       CALL PRNT;
      296   4  1  END UNIK1;
```

```
    STMT LEV NT

     297   3  1  SWAP: PROC(J,K);

                   /*  THIS PROCEDURE EXCHANGES VALUES IN CONSECUTIVE
                       ELEMENTS IN AN ARRAY  */

     298   4  1    DCL I;
     299   4  1      I = J ;
     300   4  1      J = K;
     301   4  1      K = I;
     302   4  1  END SWAP;




     303   3  1  DOWN2: PROC(J,K,M,N);

                   /*  THIS PROCEDURE SORTS ARRAYS J,K & M IN DESCENDING ORDER  */

     304   4  1    DCL (J(*),K(*)) BIN FIXED;
     305   4  1    DCL M(*) CHAR(64);
     306   4  1      LIMIT = N - 1;

     307   4  1  ONE:

                       INDEX = 1;
     308   4  1      DO I = 1 TO LIMIT;
     309   4  2      I1 = I + 1;
     310   4  2      IF J(I) >= J(I1) THEN GO TO TWO;
     311   4  2      CALL SWAP(J(I),J(I1));
     312   4  2      CALL SWAP(K(I),K(I1));
     313   4  2      CALL SWAP2(M(I),M(I1));
     314   4  2      INDEX = I;

     315   4  2  TWO:

                       END;
     316   4  1      IF INDEX <= 1 THEN RETURN;
     317   4  1      LIMIT = INDEX - 1;
     318   4  1      GO TO ONE;
     319   4  1  END DOWN2;
```

```
PL/I OPTIMIZING COMPILER          (SUBSCRIPTRANGE):

    STMT LEV NT

     320   3  1  PRNT: PROC;

                   /*  THIS PROCEDURE CALCULATES THE CATEGORY PERCENTAGES FOR UNIKOUNT  */

     321   4  1       TOT = GOODS;
     322   4  1       DO I = 1 TO L;
     323   4  2         C = CATEGORY(I);
     324   4  2         F = FREQ(I);
     325   4  2         PRCNT = F/TOT * 100;
     326   4  2       DO LL=1 TO ICAT;
     327   4  3         IF CAT_HOLD(LL) = CATEGORY(I) THEN GOTO FOUND_CAT;
     328   4  3       END;

     329   4  2  FOUND_CAT:

                       IF IN_UNIK1 = 1 THEN PERZ_CARDS(IGROUP,LL) = PRCNT;
     330   4  2       ELSE PERZ_SURJS(IGROUP,LL) = PRCNT;
     331   4  2       IF PRINT_UNIKOUNT = 1 THEN
                         PUT SKIP(2) EDIT(C,F,PRCNT,RANK(I),
                           CAT_TITLE(I))
                           (F(5),COL(15),F(5),COL(34),F(5,1),
                           COL(48),F(4,1),COL(55),A);
     332   4  2       END;
     333   4  1  END PRNT;
```

```
    STMT LEV NT

     334    *  1  RANKS: PROC (FREQ,RANK,N):

                   /*  THIS PROCEDURE RANKS PERCENTAGES IN DESCENDING ORDER  */

     335    4  1    DCL FREQ(*) BIN FIXED:
     336    4  1    DCL RANK(*):
     337    4  1    DCL L:
     338    4  1      L = 1:

     339    4  1  ONE:

                       LP1 = L + 1:
     340    4  1      IF FREQ(L) > FREQ(LP1) THEN GO TO FIVE:
     341    4  1      SUM = L + LP1:
     342    4  1      LP2 = L + 2:
     343    4  1      IF LP2 > N THEN GO TO THREE:

     344    4  1  TWO:

                       IF FREQ(L) > FREQ(LP2) THEN GO TO THREE:
     345    4  1      SUM = SUM + LP2:
     346    4  1      LP2 = LP2 + 1:
     347    4  1      IF LP2 <=N THEN GO TO TWO:

     348    4  1  THREE:

                       LP2 = LP2 - 1:
     349    4  1      FN = LP2 - L + 1:
     350    4  1      R = SUM / FN:
     351    4  1      DO K = L TO LP2:
     352    4  2        RANK(K) = R:
     353    4  2      END:
     354    4  1      L = LP2 + 1:
     355    4  1      GO TO SIX:

     356    4  1  FIVE:

                       RANK(L) = L:
     357    4  1      L = L + 1:

     358    4  1  SIX:

                       IF L <  N THEN GO TO ONE:
     359    4  1  END RANKS:
```

```
PL/I OPTIMIZING COMPILER          (SUBSCRIPTRANGE):

    STMT LEV NT

     360   3  1  UNIK2: PROC;

                   /*  THIS PROCEDURE IS THE UNIKOUNT PROCEDURE FOR PERCENT SUBJECTS  */

     361   4  1    DCL SUM BUILTIN;
     362   4  1      IN_UNIK1 = 0;

                   /*  ARRANGE NO BY SUBJECT NUMBER  */

     363   4  1      IF NSUBJECTS(IGROUP) = 0 THEN DO;
     364   4  2      PUT PAGE EDIT('UNIKOUNT',TITLE,'REFERENCE FIELD  -  ',RTITLE)
                      (COL(62),A,SKIP(2),COL(20),A,SKIP(2),A,A);
     365   4  2      PUT SKIP(2) EDIT('GROUP NAME = ',GROUP_TITLE(IGROUP)) (A);
     366   4  2        PUT SKIP(2) EDIT('NO CARDS IN THIS GROUP') (A); RETURN; END;

     369   4  1      CALL UP2(CAT,NO,NSUBJECTS(IGROUP));
     370   4  1      CALL UP2(NO,CAT,NSUBJECTS(IGROUP));
     371   4  1      NSUBJECTS(IGROUP) = 0;
     372   4  1      NSS = 0; NEXPEL = 0;
     374   4  1      LAST_SUB = 0;
     375   4  1      DO ISUBS = 1 TO NSUBS(IGROUP);
     376   4  2        IF LAST_SUB ^= NO(IGROUP,ISUBS) THEN LAST_CAT = 0;
     377   4  2      LAST_SUB = NO(IGROUP,ISUBS);
     378   4  2      IF CAT(IGROUP,ISUBS) = LAST_CAT THEN GO TO OUT1;

                   /*  COUNT CATEGORY FREQUENCIES  */

     379   4  2        DO IFIND = 1 TO ICAT;
     380   4  3          IF CAT(IGROUP,ISUBS) = CATEGORY(IFIND) THEN  DO;
     381   4  4            FREQ(IFIND) = FREQ(IFIND) + 1;
     382   4  4        IF LAST_CAT=0 THEN NSS=NSS + 1;
     383   4  4        LAST_CAT = CATEGORY(IFIND);
     384   4  4        GO TO OUT2;
     385   4  4      END;
     386   4  5          END;

     387   4  2  OUT1:

                       NEXPEL = NEXPEL + 1;

     388   4  2  OUT2:

                     END;
     389   4  1      CALL DOWN2(FREQ,CATEGORY,CAT_TITLE,ICAT);
     390   4  1      DO I = 1 TO ICAT;
     391   4  2        IF FREQ(I) = 0 THEN GO TO FOUND3;
     392   4  2      END;
     393   4  1      I = I + 1;

     394   4  1  FOUND3:
```

STMT LEV NT

```
                    L = I - 1;
395     4  1        CALL RANKS(FREQ,RANK,L);
396     4  1        GOODS = NSS;
397     4  1        IF PRINT_UNIKOUNT = 1 THEN DO;
398     4  2        PUT PAGE EDIT('UNIKOUNT',TITLE,'REFERENCE FIELD  -  ',RTITLE)
                      (COL(62),A,SKIP(2),COL(20),A,SKIP(2),A,A);
399     4  2        PUT SKIP(2) EDIT('GROUP NAME = ',GROUP_TITLE(IGROUP)) (A);
400     4  2        PUT SKIP(2);
401     4  2        PUT SKIP(2) EDIT('CATEGORY','RAW FREQUENCY',
                      'PERCENT SUBJECTS','RANK','CATEGORY TITLE')
                      (A,COL(14),A,COL(29),A,COL(48),A,COL(55),A);
402     4  2        END;
403     4  1        CALL PRNT;
404     4  1        IF PRINT_UNIKOUNT=1 THEN DO;
405     4  2        PUT SKIP(2) EDIT('NUMBER OF SUBJECTS',NSS) (A,F(5));
406     4  2        PUT SKIP(2) EDIT('NUMBER EXPELLED',NEXPEL) (A,F(5));
407     4  2        END;
408     4  1        NSUBJECTS(IGROUP) = NSS;
409     4  1  END UNIK2;
```

```
PL/I OPTIMIZING COMPILER          (SUBSCRIPTRANGE):

    STMT LEV NT

     410    3  1  PERZPRB: PROC;

                  /*  THIS PROCEDURE CALCULATES PERCENTAGES WITHIN CATEGORIES ACROSS
                      GROUPS.  IT CALCULATES THE TWO-TAILED PROBABILITY OF DIFFERENCES
                      BETWEEN THOSE PERCENTAGES  */

     411    4  1    DCL SIG(ICAT,200) CHAR(1);
     412    4  1    DCL PROB(ICAT,200) ;
     413    4  1    DCL DIFF(ICAT,200);

                  /*  OUTPUT PAGE HEADING  */

     414    4  1      PUT PAGE EDIT('PERZPROB') (COL(62),A);
     415    4  1      PUT SKIP(2) EDIT(TITLE) (A);
     416    4  1      IF IN_UNIK1 = 1 THEN
                        PUT SKIP(2) EDIT('RUN FOR PERCENT CARDS') (A);
     417    4  1        ELSE PUT SKIP(2) EDIT('RUN FOR PERCENT SUBJECTS') (A);
     418    4  1      PUT SKIP(2) EDIT('NUMBER OF CATEGORIES',ICAT,
                        'NUMBER OF GROUPS',NGROUPS)
                        (A,F(3),SKIP(2),A,F(3));
     419    4  1      PUT SKIP(2) EDIT((GROUP_TITLE(JJ) DO JJ=1 TO NGROUPS)) (A,SKIP);
     420    4  1      PUT SKIP(3) EDIT('GROUP',( I DO I=1 TO NGROUPS))
                        (COL(64),A,10 F(5));
     421    4  1      PUT SKIP(2) EDIT('NUMBER IN GROUP',
                        (NSUBJECTS(I) DO I=1 TO NGROUPS))
                        (COL(54),A,10 F(5));
     422    4  1      PUT SKIP(2) EDIT('CATEGORY INDEX AND TITLE',
                        'PERCENTAGES') (COL(10),A,COL(70),A);
     423    4  1      PUT SKIP(2);
     424    4  1      IF IN_UNIK1 = 1 THEN
                        DO;
     425    4  2          DO I = 1 TO ICAT;
     426    4  3            PUT SKIP(2) EDIT(CAT_HOLD(I),CAT_TITLE_HOLD(I),
                            (PERZ_CARDS(K,I) DO K=1 TO NGROUPS))
                            (F(2),X(2),A,COL(70),10 F(5,1));
     427    4  3          END;
     428    4  2        END;
     429    4  1      ELSE
                        DO;
     430    4  2          DO I = 1 TO ICAT;
     431    4  3            PUT SKIP(2) EDIT(CAT_HOLD(I),CAT_TITLE_HOLD(I),
                            (PERZ_SUBJS(K,I) DO K=1 TO NGROUPS))
                            (F(2),X(2),A,COL(70),10 F(5,1));
     432    4  3          END;
     433    4  2        END;

                  /*  COMPUTE PROBABILITIES  */

     434    4  1      JJ1 = NGROUPS - 1;
     435    4  1      K = 0;
     436    4  1      PROB=1; DIFF = 0;
```

```
PL/I OPTIMIZING COMPILER         (SUBSCRIPTRANGE):

     STMT LEV NT

      438    4  1       SIG=' ';
      439    4  1       DO J1=1 TO NG1;
      440    4  2         JP1 = J1 + 1;
      441    4  2         DO J2=JP1 TO NGROUPS;
      442    4  3           K = K + 1;
      443    4  3           JONE(K) = J1;
      444    4  3           JTWO(K) = J2;
      445    4  3           N1 = NSUBJECTS(J1);
      446    4  3           N2 = NSUBJECTS(J2);
      447    4  3           DO I=1 TO ICAT;
      448    4  4       IF IN_UNIK1=1 THEN DO; IF PERZ_CARDS(J1,I)=0 THEN P1=0;
      450    4  5         ELSE P1=PERZ_CARDS(J1,I); END;
      452    4  4         ELSE DO; IF PERZ_SUBJS(J1,I) = 0 THEN P1=0;
      454    4  5           ELSE P1=PERZ_SUBJS(J1,I); END;
      456    4  4       IF IN_UNIK1=1 THEN DO; IF PERZ_CARDS(J2,I) = 0 THEN P2=0;
      458    4  5         ELSE P2=PERZ_CARDS(J2,I); END;
      460    4  4         ELSE DO; IF PERZ_SUBJS(J2,I) = 0 THEN P2=0; ELSE
      462    4  5         P2=PERZ_SUBJS(J2,I); END;
      464    4  4       P1 = P1 / 100;
      465    4  4       P2 = P2 / 100;
      466    4  4           IF ABS(P1-P2) <= .0001 THEN GOTO L42;
      467    4  4         DIFF(I,K) = ABS(P1-P2);
      468    4  4           X1 = P1 * N1;
      469    4  4           X2 = P2 * N2;
      470    4  4           IF P1 < P2 THEN GO TO L40;
      471    4  4           X1 = X1 - .5;
      472    4  4           X2 = X2 + .5;
      473    4  4           GO TO L41;

      474    4  4  L40:

                           X1 = X1 + .5;
      475    4  4           X2= X2 - .5;

      476    4  4  L41:

      477    4  4         IF N1=0 THEN P1=0; ELSE
                           P1 = X1 / N1;
      478    4  4         IF N2=0 THEN P2=0; ELSE
      479    4  4           P2 = X2 / N2;
      480    4  4         IF N1+N2 = 0 THEN P=0; ELSE
      481    4  4           P = (X1+X2) / (N1 + N2);
      482    4  4           Q = 1 - P;
      483    4  4             RESULT = 0;
      484    4  4           IF P<Q THEN DO;
      485    4  5             IF N1 < N2 THEN RESULT=P*N1; ELSE RESULT=P*N2;
      487    4  5             END;
      488    4  4           ELSE DO;
      489    4  5             IF N1<N2 THEN RESULT=Q*N1; ELSE RESULT=Q*N2;
      491    4  5             END;
      492    4  4           IF RESULT < 5 THEN DO; PROB(I,K) = 0;
      494    4  5             SIG(I,K) = '*'; GO TO L42; END;
      497    4  4         IF N1=0 ! N2=0 THEN Z=0; ELSE
      498    4  4           Z = (P1-P2)/SQRT(P*Q*(1/N1 + 1/N2));
```

```
STMT LEV NT

 499   4  4          PR=GAUCOF(Z);
 500   4  4          IF PR > .5 THEN PR=1 - PR;
 501   4  4          PROB(I,K) = 2 * PR;
 502   4  4      L42: END;
 503   4  3        END;
 504   4  2      END;

                /*  COMPUTE NUMBER OF ROWS NEEDED TO OUTPUT PROBABILITIES  */

 505   4  1          LIM = K;
 506   4  1      ROWS = LIM / 8;
 507   4  1      NROWS = ROWS;
 508   4  1      REMAIN = ROWS - NROWS;
 509   4  1      IF ROWS <= 1 THEN NROWS = 1;
 510   4  1        ELSE IF REMAIN > 0 THEN DO;
 511   4  2          ROWS = ROWS + 1;
 512   4  2          NROWS = ROWS;
 513   4  2        END;
 514   4  1        ELSE NROWS = ROWS;
 515   4  1      DO ITIMES = 1 TO NROWS;
 516   4  2      IF LIM <=8 THEN
 517   4  3        DO; KSTART = 1; KEND = LIM; END;
 520   4  2        ELSE
 521   4  3          DO; IF ITIMES < NROWS THEN
                    DO;
 522   4  4            IF ITIMES = 1 THEN DO; KSTART=1; KEND=8; END;
 526   4  4            ELSE DO; KSTART=KEND + 1; KEND=KSTART+7; END;
 530   4  4          END;
 531   4  3        ELSE DO; KSTART = KEND+ 1; KEND = LIM; END;
 535   4  3      END;

                /*  OUTPUT PROBABILITIES  */

 536   4  2      PUT PAGE EDIT(
                'TWO-TAILED PROBABILITY OF A DIFFERENCE OF THIS MAGNITUDE ARISING BY CH
                ANCE') (COL(29),A);
 537   4  2      PUT SKIP(2) EDIT('(* INDICATES THAT THE PROBABILITY TEST FOR THIS C
                OMPARISON IS INAPPROPRIATE (SEE DOCUMENTATION))') (X(12),A);
 538   4  2      PUT SKIP(2) EDIT((JONE(M),'/',JTWO(M)
                  DO M=KSTART TO KEND))
                  (X(3),8(F(5),A,F(2),X(7)));
 539   4  2      PUT SKIP(3);
 540   4  2      DO JJ=1 TO ICAT;
 541   4  3        PUT SKIP(2) EDIT(CAT_HOLD(JJ),
                    (DIFF(JJ,KK), PROB(JJ,KK),
                    SIG(JJ,KK) DO KK=
                    KSTART TO KEND))
                    (F(2),X(1),8(F(6,3),F(F+3),X(1),A(1),X(1)));
 542   4  3      END;
 543   4  2      END;
 544   4  1  END PERZOR;
```

```
    STMT LEV NT

     545    3  1  GAUCOF: PROC(Z);

                   /*  THIS FUNCTION RETURNS A GAUCE VALUF GIVEN AN INITIAL Z VALUE  */

     546    4  1    DCL I;
     547    4  1    DCL Z DEC FLOAT(16);
     548    4  1    DCL Q;
     549    4  1    DCL A(7) DEC FLOAT(16) INIT
                     (.000043063,.00027656,.00015201,.00927052,
                     .042282,.070523,1);
     550    4  1      IF Z  = 0 THEN
                      RETURN(.5);
     551    4  1      Q = .70710675 * ABS(Z);
     552    4  1      DENOM = A(1);
     553    4  1      DO I = 2 TO 7;
     554    4  2        DENOM = DENOM * Q + A(I);
     555    4  2      END;
     556    4  1      DENOM = DENOM ** 16;
     557    4  1  RETURN(.5+SIGN(Z)*(ABS(.5-.5/DENOM)));
     558    4  1  END GAUCOF;
```

```
    STMT LEV NT

     559   3  1  UP2: PROC (J,K,N);

                   /*  THIS PROCEDURE SORTS ARRAYS J & K IN ASCENDING ORDER  */

     560   4  1    DCL J(*,*);
     561   4  1    DCL K(*,*);
     562   4  1    DCL I;
     563   4  1      LIMIT = N-1;

     564   4  1  ONE:

                       INDEX = 1;
     565   4  1      DO I=1 TO LIMIT;
     566   4  2        I1 = I + 1;
     567   4  2      IF J(IGROUP,I) <= J(IGROUP,I1) THEN GO TO TWO;
     568   4  2      CALL SWAP(J(IGROUP,I),J(IGROUP,I1));
     569   4  2      CALL SWAP( K(IGROUP,I),K(IGROUP,I1));
     570   4  2        INDEX = I;

     571   4  2  TWO:

                       END;
     572   4  1      IF INDEX <= 1 THEN GO TO THREE;
     573   4  1      LIMIT = INDEX - 1;
     574   4  1      GO TO ONE;

     575   4  1  THREE:

                   END UP2;
```

```
    STMT LEV NT

     576   3  1  SWAP2: PROC(J,K);

                   /*  THIS PROCEDURE EXCHANGES VALUES IN CONSECUTIVE ELEMENTS
                       IN AN ARRAY  */

     577   4  1    DCL (J,K,M) CHAR(64);
     578   4  1      M = J;
     579   4  1      J = K;
     580   4  1      K = M;
     581   4  1  END SWAP2;




     582   3  1  FIND: PROC (I);

                   /*  THIS PROCEDURE FINDS THE INDEX OF THE CATEGORY
                       NUMBER STORED IN CAT_HOLD  */

     583   4  1    DCL (I,LL);
     584   4  1      DO LL=1 TO ICAT;
     585   4  2        IF CAT_HOLD(LL) = I THEN GO TO FOUND_I;
     586   4  2      END;
     587   4  1      RETURN(0);

     588   4  1  FOUND_I:

                    RETURN(LL);
     589   4  1  END FIND;
```

```
PL/I OPTIMIZING COMPILER          (SUBSCRIPTRANGE):

    STMT LEV NT

     590    *  1  CATCOMP:PROC;

                  /*  THIS PROCEDURE COMPUTES THE BREAK OUTS OF THE CURRENT
                      REFERENCE FIELD WITHIN OTHER CLASSIFICATIONS  */


                  /*  READ IN THE NUMBER OF OTHER CLASSIFICATIONS TO BE COMPARED  */

     591    4  1      GET SKIP EDIT(NCLASS) (X(44),F(1));


                  /*  READ IN MAXIMUM NUMBER OF CATEGORIES IN THE OTHER
                      CLASSIFICATIONS  */

     592    4  1      GET SKIP EDIT(NCATMAX) (X(58),F(3));

     593    4  1    BEGIN;

     594    5  1      DCL CLASS_COL(NCLASS) BIN FIXED;
     595    5  1      DCL CLASS_WID(NCLASS) BIN FIXED;
     596    5  1      DCL CLASS_TITLE(NCLASS) CHAR(40);
     597    5  1      DCL CLASS_CAT(NCLASS) BIN FIXED;
     598    5  1      DCL COUNTS(NCLASS,NCATMAX) BIN FIXED(31);
     599    5  1      DCL CAT_LABELS(NCLASS,NCATMAX) CHAR(64);

                    /*  READ IN TITLES, COLUMNS, WIDTHS, AND NUMBER OF CATEGORIES
                        FOR EACH CLASSIFICATION */

     600    5  1      DO KM = 1 TO NCLASS;
     601    5  2        GET SKIP EDIT(CLASS_TITLE(KM),CLASS_COL(KM),
                          CLASS_WID(KM),CLASS_CAT(KM))
                          (A(40),F(4),F(4),F(3));
     602    5  2        DO KMM=1 TO CLASS_CAT(KM);
     603    5  3          GET SKIP EDIT(CAT_LABELS(KM,KMM)) (X(16),A(64));
     604    5  3        END;
     605    5  2      END;

                    /*  OUTPUT PAGE HEADING  */

     606    5  1      PUT PAGE EDIT('CATEGORY COMPARISONS FOR  -  ',TITLE) (A);
     607    5  1      PUT SKIP EDIT('REFERENCE FIELD',' -  ',RTITLE) (X(9),A,A,A);
     608    5  1      ILINE = 2;
     609    5  1      DO II=1 TO ICAT;
     610    5  2        IF ILINE > 25 THEN
                        DO;
     611    5  3          PUT PAGE EDIT('CATEGORY COMPARISONS FOR  -  ',TITLE) (A);
     612    5  3          PUT SKIP EDIT('REFERENCE FIELD',' -  ',RTITLE) (X(9),A,A,A);
     613    5  3          ILINE = 2;
     614    5  3        END;
     615    5  2        PUT SKIP EDIT((15)' ','-') ();
     616    5  2        PUT SKIP(2) EDIT('CATEGORY ',CAT_VAL(II),' -  ',
```

```
    STMT LEV IT

                        CAT_TITLE_HOLD(II))
                       (A,F(3),A,A);
  617   5  2          PUT SKIP EDIT('CATEGORY FREQUENCY',ALL_CAT(II)) (COL(15),A,F(6));
  618   5  2          ILINE = ILINE + 4;
  619   5  2          DO JJ=1 TO NGROUPS;
  620   5  3            PUT SKIP(3) EDIT('GROUP ',JJ,GROUP_TITLE(JJ))
                         (COL(5),A,F(2),COL(20),A);
  621   5  3            PUT SKIP EDIT('GROUP ',JJ,' FREQUENCY',GROUP_CAT(II,JJ))
                         (COL(20),A,F(2),A,F(6));
  622   5  3            ILINE = ILINE + 4;
  623   5  3            IF GROUP_CAT(II,JJ) = 0 THEN GO TO NEXT_GRP;
  624   5  3                COUNTS = 0;
  625   5  3            DO LL=1 TO NCLASS;
  626   5  4            PUT SKIP(3) EDIT(CLASS_TITLE(LL),'FREQUENCY','% CATEGORY',II,
                        '% CATEGORY',II,'CATEGORY NUMBER AND TITLE',' IN GROUP',JJ)
                         (COL(10),A,COL(90),A,COL(102),A,F(3),COL(118),A,F(3),
                          SKIP,COL(16),A,COL(119),A,F(3));
  627   5  4            ILINE = ILINE + 4;
  628   5  4              ON ENDFILE(FILES(JJ)) GO TO PRINT_IT;
  629   5  4              CLOSE FILE(FILES(JJ));

  630   5  4  ANOTHER:

                          GET FILE(FILES(JJ)) SKIP EDIT(CARD) (A(LRECL1));
  631   5  4                GET STRING(CARD) EDIT(CAT_VALUE) (X(CAT_COL-1),F(RWID));
  632   5  4                IF CAT_VALUE ^= CAT_HOLD(II) THEN GO TO ANOTHER;
  633   5  4                  GET STRING(CARD) EDIT(IVAL) (X(CLASS_COL(LL)-1),
                              F(CLASS_WID(LL)));
  634   5  4                  IF IVAL^=0 & IVAL<=CLASS_CAT(LL) THEN
                              COUNTS(LL,IVAL) = COUNTS(LL,IVAL) + 1;
  635   5  4                GO TO ANOTHER;

  636   5  4  PRINT_IT:

                          PUT SKIP;
  637   5  4                ILINE = ILINE + 1;
  638   5  4                DO NN = 1 TO CLASS_CAT(LL);
  639   5  5                  IF COUNTS(LL,NN) > 0 THEN DO;
  640   5  5                  PUT SKIP EDIT( NN,CAT_LABELS(LL,NN),COUNTS(LL,NN),
                                (COUNTS(LL,NN) / ALL_CAT(II))*100,
                                (COUNTS(LL,NN) / GROUP_CAT(II,JJ))*100)
                             (COL(11),F(3),COL(16),A,COL(92),F(5),COL(105),F(6,2),
                              COL(121),F(6,2));
  641   5  6                    ILINE = ILINE + 1;
  642   5  6                  END;
  643   5  5                  END;  /* END WRITING COUNTS */
  644   5  4                END;  /* END WRITING CLASSIFICATIONS */

  645   5  3  NEXT_GRP:

                        END;    /* END WRITING GROUPS */
  646   5  2               END;     /* END WRITING CATEGORIES */
  647   5  1               END;  /* END BEGIN */
  648   4  1  END CATCOMP;
```

```
    STMT LEV NT

     649    *  1  NEXT_RUN:

                  END;  /*  END DO IRUNS  */
     650    2  1      END;  /* BEGIN  */
     651    2  0      END;    /*  BEGIN  */

     652    1  0  END GURU;
```


ATE
LME