|  |
| --- |
| **Name: Ayush Patel**  **Roll: A232**  **Semester: VII**  **Class: Btech IT (4th year)**  **Branch: IT** |
| **Practical-3 Part A** |
| **Aim:** Program to Sort/Group Report data and Enhance Reports..  **a.** Write a SAS Program to performed following task   * Program to create SAS dataset Payroll with 10 observations. * EmployeeID * FirstName * LastName * Gender * Qualification * Salary * Job Title[Manager, SalesManager, Sales Rep. I, Sales Rep. II etc] * Country * Birthdate * Hire Date     Q2.  For above report generated Use appropriate Title and Footnote.  Q3. Implement Label and split option for given scenario |
| **Prerequisite:** Programming for Problem Solving |
| **Outcome:** To create dataset and performed data analysis and Sort dataset |
| Theory: |
| **Procedure:**   1. Open SAS Studio and write the SAS program |
| **Instructions:**   1. Write source code of all stored procedure 2. Copy code & paste in code section of Part B. |

|  |
| --- |
| **Part B** |
| **Code:**  **A)**  data payroll;  input empid Fname $ Lname $ Gender $ salary qualification $ jobtitle $ birthdate ddmmyy8. country $ hiredate ddmmyy8.;  datalines;  1 Ayush Patel M 99099 Graduate Manager 16092000 USA 09092024  2 Dhruvil Patel M 40000 UnderGrad HR 15052000 IND 02012022  3 Dhyanesh Patel M 80000 UnderGrad SalesManager 02082000 IND 03012022  4 Ajay Dabas M 60000 Graduate SalesRep 16062000 USA 03012024  5 Pavitra Maheswari M 45000 UnderGrad SalesRep 04042000 FRA 03012023  6 Tushar Agarwaal M 95000 UnderGrad SalesManager 03041999 JPN 03102025  7 Sakura Haruno F 49500 PhD SalesRep 06011989 JPN 03012004  8 Hinata Uzumaki F 90500 PhD Manager 08051995 USA 03122021  9 Snehil Raj M 65000 UnderGrad SalesRep 09012001 FRA 03012025  10 Alankar Uniyal M 99000 Graduate SalesManager 10012002 FRA 05062026  ;  proc sort data = payroll out=work.sorted\_payroll;  by country descending salary;  run;  proc print data = work.sorted\_payroll noobs;  format birthdate ddmmyy10. hiredate ddmmyy10.;  by country;  run;  **B)**  title 'Payroll Info'; footnote 'Employee Data'; proc sort data = payroll out=work.sorted\_payroll; by country descending salary; run;  proc print data = work.sorted\_payroll noobs; format birthdate ddmmyy10. hiredate ddmmyy10.; by country; run; title; footnote;  **C)**  title 'Payroll Info with column names';  footnote 'Confidential Employee Data';  proc print data=work.payroll label;  format birthdate ddmmyy10. hiredate ddmmyy10.;  label empid='Employee ID'  Fname = 'First Name'  Lname = 'Last Name'  Gender = 'Gender'  salary = 'Salary'  qualification = 'Qualification'  jobtitle = 'Designation'  birthdate = 'DOB'  country = 'Country'  hiredate = 'Hire Date';  run;  title;  footnote; |
| **Output:** |
| **Observation & Learning:**  Learnt about ‘label’ and other proc commands in SAS. |
| **Conclusion:**  Successfully implemented ‘label’ and other proc commands in SAS. |
| **Question:**      **Ans\_1— D & Ans\_2— D**  **Q3.**  **Explain following**   1. **Proc sort**   **Ans –**  The SORT procedure orders SAS data set observations by the values of one or more character or numeric variables. It either replaces the original data set or creates a new data set.   1. **Label Statement**   **Ans –**  It is a statement in base SAS which is used to rename a pre-existing column in SAS. Using a LABEL statement in a DATA step permanently associates labels with variables by affecting the descriptor information of the SAS data set that contains the variables. You can associate any number of variables with labels in a single LABEL statement   1. **Different Operators in Where statements**   **Ans –**  Operators Used in the WHERE Expression –  You can include both SAS operators and special WHERE-expression operators in the WHERE statement. For a complete list of the operators, see WHERE Statement Operators. For the rules that SAS follows when it evaluates WHERE expressions, see WHERE-Expression Processing in SAS Language Reference: Concepts.  This table lists the operators for the WHERE statement   |  |  |  | | --- | --- | --- | | **Operator Type** | **Symbol or Mnemonic** | **Description** | | Arithmetic |  |  | |  | \* | multiplication | |  | / | division | |  | + | addition | |  | − | subtraction | |  | \*\* | exponentiation | | Comparison |  |  | |  | = or EQ | equal to | |  | ^=, ¬=, ~=, or NE[1](https://go.documentation.sas.com/doc/en/pgmsascdc/9.4_3.5/lestmtsref/n1xbr9r0s9veq0n137iftzxq4g7e.htm#n1he5c28fww9xcn1kmqxe6x15d87) | not equal to | |  | > or GT | greater than | |  | < or LT | less than | |  | >= or GE | greater than or equal to | |  | <= or LE | less than or equal to | |  | IN | equal to one of a list | | Logical (Boolean) |  |  | |  | & or AND | logical and | |  | | or OR | logical or | |  | ~,^ , ¬, or NOT | logical not | | Other |  |  | |  | || | concatenation of character variables | |  | ( ) | indicate order of evaluation | |  | + prefix | positive number | |  | − prefix | negative number | | WHERE Expression Only |  |  | |  | BETWEEN–AND | an inclusive range | |  | ? or CONTAINS | a character string | |  | IS NULL or IS MISSING | missing values | |  | LIKE | match patterns | |  | =\* | sounds-like | |  | SAME-AND | add clauses to an existing WHERE statement without retyping original one | | **1** The caret (^), tilde (~), and the not sign (¬ ) all indicate a logical not. Use the character available on your keyboard, or use the mnemonic equivalent. | | | | **2** The OR symbol ( | ), broken vertical bar ( | ), and exclamation point (!) all indicate a logical or. Use the character available on your keyboard, or use the mnemonic equivalent. | | | | **3** Two OR symbols (| | ), two broken vertical bars ( | | ), or two exclamation points (!!) indicate concatenation. Use the character available on your keyboard. | | | | **4** You can use the colon modifier (:) with any of the comparison operators in order to compare only a specified prefix of a character string. | | | |