|  |  |  |
| --- | --- | --- |
| **SQL** | | |
| **proc** **sql;**  **create** **table** sql\_base **as**  **select** name **label**="name标签", sex,  (weight\*0.4536)/((height\*0.0254)\*\*2)as bmi  **label**="bmi标签" **format**=5.2,  **case**  **when** calculated bmi<16 **then** "低"  when 16<=calculated bmi<=18 then "中"  when calculated bmi>18 then "高"  **else** "NA"  **end as** bmilevel  from sashelp.class  /\*calculated bmi\*/ /\*where语句以及case语句会优先于select语句,新计算出来的变量用calculated来进行筛选\*/  **order by** bmi desc;  **quit**; | | **proc sql;**  **select** Subsidiary, count(product) as n **label**="产品种类数" from sashelp.shoes  **group by** Subsidiary  having n>6 /\*having 筛选的是分组数据，受group by影响；where用于筛选行，不受group by影响\*/  order by n desc;  quit;  **proc sql;**  **select** Region,Product,sum(Sales) as total\_sale **label**="产品销售额"  **from** sashelp.shoes  **group by** Region,Product  **order by** Region,total\_sale desc; /\*在region下哪种产品销售额最大，如果只order total\_sale，则汇总的是全部的sale\*/  quit; |
| /\* SQL横向合并数据 \*/  libname scores xlsx "/home/u64080144/SQL/Score.xlsx";  proc sql number;  /\* number在最开始添加列，表示行索引 \*/  create table scores.test123 as  select coalesce(a.name,b.name) as name,sex,age,exam,Chinese,Math,English  /\* coalesce 相当于 a.name，因为两个表都有name \*/  from scores.classinfo as a  left join scores.test1 as b  on a.name = b.name;  quit;  proc sql;  create table deltest as  select \* from scores.test123  where name ^= "Alfred";  /\* 某一行去除去除 \*/  quit; | | proc sql outobs=5 number;  create table xx as  select col1 **label**=" " **format**=6.2, col2, col1/col2 as col3,**compress(upcase**(col4))as col4  case  when calculated col3<3 then “val1"  when calculated col3>8 then "val2"  else “val3"  end as col4  from table(keep=col1,col2)  group by col1,col2  having col1>6  order by col1,col2 desc; /\*在col1排序的情况下对col2排序\*/  quit; |
| /\* SQL横向合并数据 \*/  proc sql;  create table table1 as  select coalesce(a.col1,b.col1) as col1,col2,col3  from table1 as a  where a.col1 ^= "str"  left join table2 as b  on a.col1=b.col1;  quit; | | /\* SQL纵向合并数据 \*/  proc sql number;  select Name, Exam,.,Math,English from scores.test1(drop=Chinese)  outer union corr  select Name, Exam,Chinese,.,English from scores.test2(drop=Math)  outer union corr  select Name, Exam,Chinese,Math,. from scores.test3(drop=English);  quit; |
| Format | |  |
| proc format;  value bmifmt  low-18.4 = "Underweight"  18.5-24.9 = "Normal"  25-29.9 = "Overweight"  30-high = "Obese";  run; | | proc format;  value $bmifmt  "U" = "Underweight"  "N" = "Normal"  "O" = "Overweight"  "B" = "Obese";  run; |
| **Summary** | | |
| proc means proc univariate 定量资料的描述统计  proc means data=sortclass maxdec=2;  by sex;  var age weight height;  run;    proc means data=sortclass maxdec=2;  class sex;  var age weight height;  run; | | ods output Summary=sum1;  ods trace on;  \*缺失值会自动删除掉;  proc means data=bmi maxdec=2 missing n mean std min max median nmiss;  \*missing将缺失的变量单独统计出来  by sex;  class bmilevel;  var age weight height;  output out=sum2;  run;  ods trace off;  ods output close; |
| by与class区别：  by要sort  by之后根据分类变量生成n个表  class之后只有一个表 | |  |
| proc freq 定性资料的描述统计    proc freq data=bmi;  tables sex; \*一维的  run;    proc freq data=bmi;  tables sex\*bmilevel /nmissing;\*二维的  run; | | proc freq data=tb1 **nlevels** **order**=freq; \*按照freq进行排序;  tables col1 / **out**=tb2 **outpct**;  \*outpct百分比;  run;  proc freq data=tb1;  tables col1\*col2/ **out**=tb2 **outpct**;  run;  proc freq data=tb1;  tables col1\*col2/ missing; \*把缺失值也统计出来了  run; |
| **Proc sort** | |  |
| libname scores xlsx "/home/u64080144/Score.xlsx";    data grades;  set scores.test1 scores.test123;  run;    **proc sort data**=grades;  by **descending** name **descending** exam;  run;    /\* sort去重 nodup\*/  proc sort data=grades out=nodouble nodup dupout=nodup1;  by \_all\_; \*nodup去除相邻的重复行，因此by要all;  run;    /\* sort去部分重复的行nodupkey \*/  proc sort data=grades out=nodouble **nodupkey** dupout=nodup2;  by name; \*nodup去除namek相同的行;  run;    /\* sort去部分重复的行nodupkey \*/  proc sort data=grades out=nodouble nodupkey dupout=nodup2;  by name; \*nodup去除namek相同的行;  run; | | |
| **Base** | | |
| data sales;  input storeid $4. Q1 Q2 Q3 Q4;  datalines;  a001 3200 4530 2345 3226  a002 3203 4535 2347 3227  a003 3204 4536 2344 3222  ;  run;    data total\_means;  set sales;  total\_sales1=Q1+Q2+Q3+Q4;  total\_sales2=sum(Q1, Q2, Q3, Q4);  total\_sales2=sum(of Q1-Q4);  mean\_sales1=(Q1+Q2+Q3+Q4)/4;  mean\_sales2=mean(Q1, Q2, Q3, Q4);  mean\_sales3=mean(of Q1-Q4);  run; | /\* data数据读取 \*/  data ;  infile "xxxx" dlm="," firstobs=2 pad;  \*dlm指定文件的分隔符 firstobs指定从第二行开始读 pad使每一行长度相同，用空格填充;  input col1 $9. col2-col4;  str=bird||"\*\*\*"; \*字符串拼接;  len = length(str);  run;  /\* prco 获取导入数据 -作用：将变量名自动导入，并数据规范（防止里面有的数据有引号有的没引号）\*/  proc import datafile="xxx"  dbms=dlm out="outputfile"; \*数据库管理系统 什么样的文件：分隔符文件;  \*dbms=xlsx;  \*sheet="xlsx sheet 名字";  delimiter=",";  getnames=yes; \*第一行读取变量名;  guessingrows=max; \*指定行数以判断变量类型与长度 默认20行,仅对分隔符文件有效;  run; | |
| **merge-**横向合并数据 |  | |
| libname scores xlsx "/home/u64080144/Score.xlsx";    /\*一对一合并\*/  data work.test1;  merge scores.classinfo scores.test1;  run;    /\*一对一匹配合并-每个表的by变量只有一个\*/  \*不能直接对libname的逻辑库进行修改，因此sort后要加out，逻辑库再换一个;  proc sort data=scores.classinfo out=classinfo;  by name;  run;    proc sort data=scores.test2 out=test2;  by name;  run;    data test2;  merge scores.classinfo scores.test2;  by name;  \*by前要对他排序;  run;  **set-**纵向合并数据 | /\* 一对多匹配合并-有个表的by变量有多个 \*/  data scores.natest;  set scores.test123;  if name="Alfred" then delete; \*删除的是某一条语句;  run;    proc sort data=scores.natest out=natest;  by name;  run;    data test123;  merge scores.classinfo(in=ds1) natest(in=ds2);  \*数据集选项in=ds1 dss1=ds1 相当于立一个flag临时变量，用于标志某个表是否有值;  by name;  dss1=ds1;  dss2=ds2;  if dss1=1 and dss2=0 then delete;  run; | |
| ibname scores xlsx "/home/u64080144/Score.xlsx";    data exam123;  set scores.test1 - scores.test3;  \*不用by的话，则默认无脑的将所有表向下拓展，用by的话将对by变量group;  by name;  run;    /\* 逐步set的话，每个set会覆盖前一个set  则需要手动控制输出 在每个set后加一个output\*/  data exam123;  set scores.test1;output;  set scores.test2;output;  set scores.test3;output;  run;    proc print;  run;    /\* proc append将data内的数据追加到base的末尾，当两个数据集变量完全一致时候使用 \*/  proc appent base=test1;data=test2;run; | proc sort data=tb1 out=tb2;  by col1;  run;    data tb1;  set tb1 tb2 tb3-tb5;  by col1;  run; | |
| **数据集选项（where/drop/keep）** | | |
| data male;  set sashelp.class;  if sex="M";  bmi=(weight\*0.4)/(height\*0.3);  run;    /\* where选项 \*/  data male;  set sashelp.class(**where**=(sex="M"));  bmi=(weight\*0.4)/(height\*0.3);  run;  where可以有语句可以有选项，选项在set后  set tab1(**where**=(name="alive"))  drop可以有语句可以有选项，选项在set后/data后  drop选项只作用于前面所跟的数据  data male(**drop**=sex weight);  set sashelp.class(**drop**=sex weight);  drop语句位置没关系，只作用于输出的表 | /\* drop选项 drop语句-drop语句的位置没影响，只会对最终生成的表有影响 \*/  data male;  set sashelp.class(**drop**=sex weight);  bmi=(weight\*0.4)/(height\*0.3);  run;    data male(drop=sex weight);  set sashelp.class;  bmi=(weight\*0.4)/(height\*0.3);  run;    data male;  set sashelp.class;  **drop** sex weight;  bmi=(weight\*0.4)/(height\*0.3);  if sex="F" then output male;  run; | |
| **if-then** | | |
| data bmi;  set sashelp.class;  bmi=weight/height;  if bmi<1 then  bmilevel="normal";  else if 0<bmi<1 then  bmilevel="fat";  else  bmilevel="fat";  run;    data normal fat fated;  length bmilevel $21;  set sashelp.class;  bmi=weight/height;  \*do end 语句把then后面的>=2条操作整成一个组，使得后面的else能够和前面的if配套;  if bmi<1 then  do;  bmilevel="normal";  output normal;  end;  else if 0<bmi<1 then  do;  bmilevel="fat";  output fat;  end;  else  do;  bmilevel="fated";  output fated;  end;  run;    data class;  set sashelp.class;  gender=ifc(sex="M","男","女");  gender=ifn(sex="M",1,2);  gender=ifn(sex,1,2,99);  run; | data table1;  set table;  if col1="m" then  do;  col2=1;  end;  else if col1="f" then col2=2;  else col2=3;  run;  **循环语句**  数值型  do i=1 to 20 by -1;  put i;  end;    字符型  do i='a','b','c';  put i=;  end; | |
| **GLM** | | |
| 是 SAS 中用于执行\*\*通用线性模型（General Linear Model）\*\*分析的过程，可以处理广泛的统计建模问题，包括回归、方差分析（ANOVA）、协方差分析（ANCOVA）等 | | |
| proc glm data=dataset;  class categorical-variables;  model dependent-variable = independent-variables;  means class-variable / options;  output out=output-dataset predicted=predicted residual=residuals;  run;  quit; | | |
| MODEL 语句   * 定义因变量（dependent-variable）和自变量（independent-variables）。 * 形式为 dependent-variable = independent-variables。   MEANS 语句   * 用于计算分类变量组的均值。 * 可以附加选项进行多重比较检验。   OUTPUT 语句   * 将预测值、残差等结果保存到一个新数据集。 | | |
| **简单线性回归**  proc glm data=example;  model y = x;  run;  quit; | | |
| 多因素方差分析（ANOVA）  proc glm data=example;  class factor1 factor2;  model response = factor1 factor2 factor1\*factor2;  means factor1 factor2 / tukey;  run;  quit; class：指定分类变量 factor1 和 factor2。  model：指定因变量 response 和效应模型，包括主效应和交互效应。  means：生成组均值并进行 Tukey 多重比较检验。 | | |
| 协方差分析（ANCOVA）  proc glm data=example;  class group;  model response = group covariate;  run;  quit;  group 是分类变量（如处理组）。  covariate 是协变量，控制其对因变量的影响。 | | |
| proc glm data=example;  model y = x1 x2;  output out=results predicted=pred residual=res;  run;  quit;  将预测值保存到 results 数据集中，变量名为 pred。  将残差保存到 results 数据集中，变量名为 res。 | | |
| **MODEL 语句中的选项**   | **选项** | **功能** | | --- | --- | | solution | 输出模型参数估计值（如回归系数）。 | | nouni | 禁止输出单变量分析结果。 | | ss1/ss2/ss3 | 指定计算平方和的类型（用于不平衡设计）。 | | | |
| **MEANS 语句中的选项**   | **选项** | **功能** | | --- | --- | | Tukey | 执行 Tukey 的多重比较检验。 | | Bonferroni | 执行 Bonferroni 的多重比较检验。 | | CLM | 生成均值的置信区间。 | | | |
| 删除某条语句  data步内  if name="Alfred" then delete; \*删除的是某一条语句;  if dss1=1 and dss2=0 then delete;  删除某列数据  data步内  set scores.test1(drop=Chinese) | | |