Lesson 1

Interface neutral

SAS

Base SAS

SAS Framework

SAS Programming process

Main file types:

Raw Data files

SAS Data Set

SAS Program File

Lesson 2

Programs – steps.

DATA steps – creates a SAS dataset. Variables are columns in data.

```
data work.news;
run;
```

PROC steps – or procedure step. Creates reports.

```
Proc print data=work.news;
Run;
```

Step boundaries – Each step has a begging in and ending.

Free formatted style

Comments - /* */ block and *comment; comment statement.

Unbalanced quotation marks.

Lesson 3: Accessing Data

SAS Libraries – collection of data files recognized by SAS. Temporary library ex work. Permanent Library is saved after session ex sasuser.

Library

SAS data set

Member of the Library

Library reference (libref) – references a particular reference location that a particular environment recognizes. Work, sasuser, and sashelp are references to physical locations BUT each library references the SAME location.

Libref.data-set-name; ex: work.newdataSet (2 level name)

A one level name is assumed to be a temp library.

LIBNAME libref 'SAS-library' <options>; *This is an example of making a connection to a lib. It is a global statement.

LIBNAME – is the key word

Libref – is a valid lib (1-8 chars, begin with letter or underscore and NOT numbers.

SAS-library – file path or your location

An example using a macro variable:

```
%let path=/folders/myfolders/ecprg193;

libname orion "&path";

PROC CONTENTS DATA=libref._ALL_ NODS;

RUN; *List all the contents of the library. Shoes Directory info, Member Types (data or index), and the name. NODS suppress the descriptor dat for each individual dile in the library.

PROC PRINT DATA=libref.SAS-data-set;

RUN;

LIBNAME libref CLEAR; *This disconnects the library or dissociates the tem libr name.
```

Descriptor – gives the meta data and Variable properties.

Data values.

Missing data values – missing values are recorded in the table. (blank for chars and . for numbers) Or do MISSING='null' or MISSING='char'

** Numeric values are stored in floating point notation in 8 bytes of storage, allowing a maximum of 16-17 digits.

** Any library that you create is a permanent library and is available in subsequent SAS sessions. When a data set is in the temporary **work** library, you can optionally use a one-level name. A one-level name consists of only the data set name, such as **sales** or **newsalesemps**. When you specify a one-level name, SAS assumes that the data set is stored in the **work** library because **work** is the default libref.

** the default length of numeric variables is 8 bytes.

Lesson 4: Producing Detail Reports

*By default a proc print step displays all observations and variables in a dataset.

Var last_name first_name; *displays the variables you want. Kind of like a select in SQL.

Sum Salary Prices; *Recorded in the last row of the report.

Where Statement. Used to subset. **Where where-expression**; where Sal<2444; 'M' **Character constant**.

Comparison operators. Examples: eq, ne, lt, <=, in etc. **where country in ('AU', 'US')**

Arithmetic operators. Examples: *, +, -, **(this is an exponent). **Where Sal+Bonous<=10000**;

Logical Operators. Examples: &, |, AND, NOT, OR, ~ **where sal>3 and me='M'**

noobs. Proc print data=orion.sales noobs; (gets rid of the observation numbers in the report.

Contains operator. Where Job_Title contains 'Rep' (searches a substring to match)

Special Where operators.

(not)Between-and

Where same and

Is null

Is missing (could also use sal=.; or sal='';)

Like (% any number of chars, _ at lest one char occupies) Example: where name like '%N'; where name like 'T_m%'; **where Name like '%, M%';**

ID statements. Id Customer; (will make the customer ID replace the obs numbers in report)

Proc Sort.

Proc sort data=orion.sales;

Out=work.salesSorted;

By first;

Run;

Proc sort data=input-SAS-data-set

<out=output-SAS-data-set>;

By <descending> by-variable(s);

Run;

* PROC SORT replaces the original data set unless you specify an output data set in the OUT= option.

Grouping. Proc print data=work.sales2;

By country;

Run;

Titles and footnotes. **TITLEn 'text'; FOOTNOTEn 'text';** where n is the line number. Ex: title3 'Wellness Clinic Insurance'; title1; (cancels a title)

LABEL variable='label'; Change variables names for a report.

Example:

```
title1 'Australian Sales Employees';
title2 'Senior Sales Representatives';
footnote1 'Job_Title: Sales Rep. IV';

proc print data=orion.sales noobs;
  where Country='AU' and
      Job_Title contains 'Rep. IV';
  var Employee_ID First_Name Last_Name
     Gender Salary;
run;
title;
footnote;
```

Lesson 5: Formatting Data Values

FORMAT variable(s) format;

Ex: format Salary dollar8.;

Dollar12.2 (.2 is the decimal places 12 is the width)

Dates – MMDDYY10 = 01/01/1960

PROC FORMAT;

VALUE format-name value-or-range1='formatted-value1'

value-or-range2='formatted-value2'

...;

RUN;

**Character formats begin with a dollar sign and must be followed by a letter or underscore. user-defined formats cannot be the name of a SAS format.

**In a User Defined format statement you have to include the period at the end when calling the format in a proc print.

Sample Programs

Lesson 6: Reading SAS data sets

Below is a data step:

```
DATA output-SAS-data-set; /*data statement provides the name of the data set we create */
    SET input-SAS-data-set; /*set statement. From the existing data set. SET is the key word.*/
    WHERE where-expression;
RUN; /*run statement */
```

EX:

```
data work.subset1;
  set orion.sales;
  where Country='AU' and
      Job_Title contains 'Rep';
Discount=.25; /*This will add a column with the value of .25 and Discount as the column name.
run;
```

SAS date constant – 'ddmmm<yy>yy'D SAS converts this to a SAS **data value.**

**If an operand in an arithmetic expression has a missing value, the result is a missing value.

DROP variable-list; /*vars to drop in the dataset*/

KEEP variable-list;/*vars to keep. Make sure you add any vars you added*/

**Which one depends on how many you want to write.

SAS Possessing of the DATA Step:

Compilations Phase – scans code for errors. Creates a **Program data vector** that has the descriptor for the data.

Execution Phase – Reads and processes observations and creates the data protion.

IF expression;/*ONLY use when you are adding new vars not in the set data set you are reading from*/

**When the expression is false, SAS excludes the observation from the output data set and continues processing.

Choosing a statement for Subletting Observations -

Example Code:

Sample Programs

**To subset observations in a PROC step, you must use a WHERE statement. You cannot use a subsetting IF statement.

**Labels and formats that you specify in PROC steps override the permanent labels and formats in the current step. However, the permanent labels and formats are not changed.

Using Libraries to Read in (and Write to) Excel (.xlsx) Files

proc contents data=np.parks;

run;

Lesson 8: Reading Raw Data Files

Raw Data file can be: text file, CSV, or ASCII.

Delimited File – use commas or spaces to separate data.

Fixed column file -

Record Layout.

** A delimited raw data file is an external text file in which the values are separated by spaces or other special characters. The file is not software-specific.

Need to know: Specification, type of Data, Arrangement.

Standard Data – data that SAS can read without any special instructions.

Nonstandard Data – Hard to read data like (23) or $67.23. Needs extra instructions to read this data.

DATA output-SAS-data-set-name;

    INFILE 'raw-data-file-name' DLM='delimiter';

INPUT variable1 <$> variable2 <$> ... variableN <$>;

RUN;

INPUT – use $ after any var you want to be a character variable. Else it would be a numeric var. the input describes the data in the set from left to right. YOU CANNOT SKIP FIELDS. You can read up to a nonstandard field and leave it out. The default length is 8 bytes regardless of type.

** A libref is used to access SAS data sets in a SAS data library. The INFILE statement references the raw data file, so you do not need to use a libref to point to it.

Compilation for input – checks for errors, creates a **input buffer** (only for reading raw data), then a PDV is created (the _N_ and _ERROR_ are automatic data vars not written to the data set),then it creates a descriptor portion for the dataset.

**Missing numeric values have a (.) and characters are just blank.

** SAS uses an input buffer only if the input data is a raw data file. At compile time, the PDV holds the variable name, type, and length, but not the initial value. The descriptor portion is the last item created during compilation.

** When reading a raw data file, SAS reads the data from the file into the input buffer, and the input buffer is an area of memory whose default length depends on the operating system. The only true statement here is that at the bottom of the DATA step, SAS writes the contents of the PDV to the output data set.

Length – 8 bytes can hold up to 15-16 digits. Characters can be long.

LENGTH variable(s) <$> length;

Length needs to precede the input statement.

Modified List Input –

Formats – instructions on how to write

Informats – instructions on how to read

INPUT variable <$> variable <:informat>;

First_Name :$12. (the : makes it read up to a delimitor and the 12 is the length)

**A character variable can hold ANY data.

Dates: Birth_Data :date. Hire_Date :mmddyy.; The Default length is 7. Ignore the widths and just us a : to tell it to read up to the next delimiter.

DATALINES;

<data line 1>

<data line 2>

...

; /*This is a null statement*/

Datalines - Reads in instream data.

Validating Data

**When SAS encounters a data error, it prints messages and a ruler in the log and assigns a missing value to the affected variable. Then SAS continues processing.

DSD – Delimiter sensitive data

Sets the default delimiter to a comma

Treats ***consecutive*** delimiters as missing values

Enables SAS to read values with embedded delimiters.

INFILE 'raw-data-file-name' <DLM=> <DSD> <MISSOVER>;

Missover – skips over missing values at the end. Assigns then to missing values.

Lesson 9

Pitfall – var=sum(1,2,missing,3) vs var=1+missing

Date functions. TODAY(), DATE(),

```
**data orion.newloan;
  set orion.records;
  TotalPaid=sum(TotLoan+Interest);
  if Code='1' then Type='Fixed';
  else Type='Variable';
  length Type $ 10;
run;
```

In the DATA step below, what is the length of the new variable, Type? Ans = 5

The length of a new variable is determined by the first reference in the DATA step. In this case, the length of Type, the new variable, is determined by the value Fixed, which is five characters long. Although a LENGTH statement is included, it is in the wrong place. It should appear before

any other reference to the variable in the DATA step. The LENGTH statement cannot change the length of an existing variable.