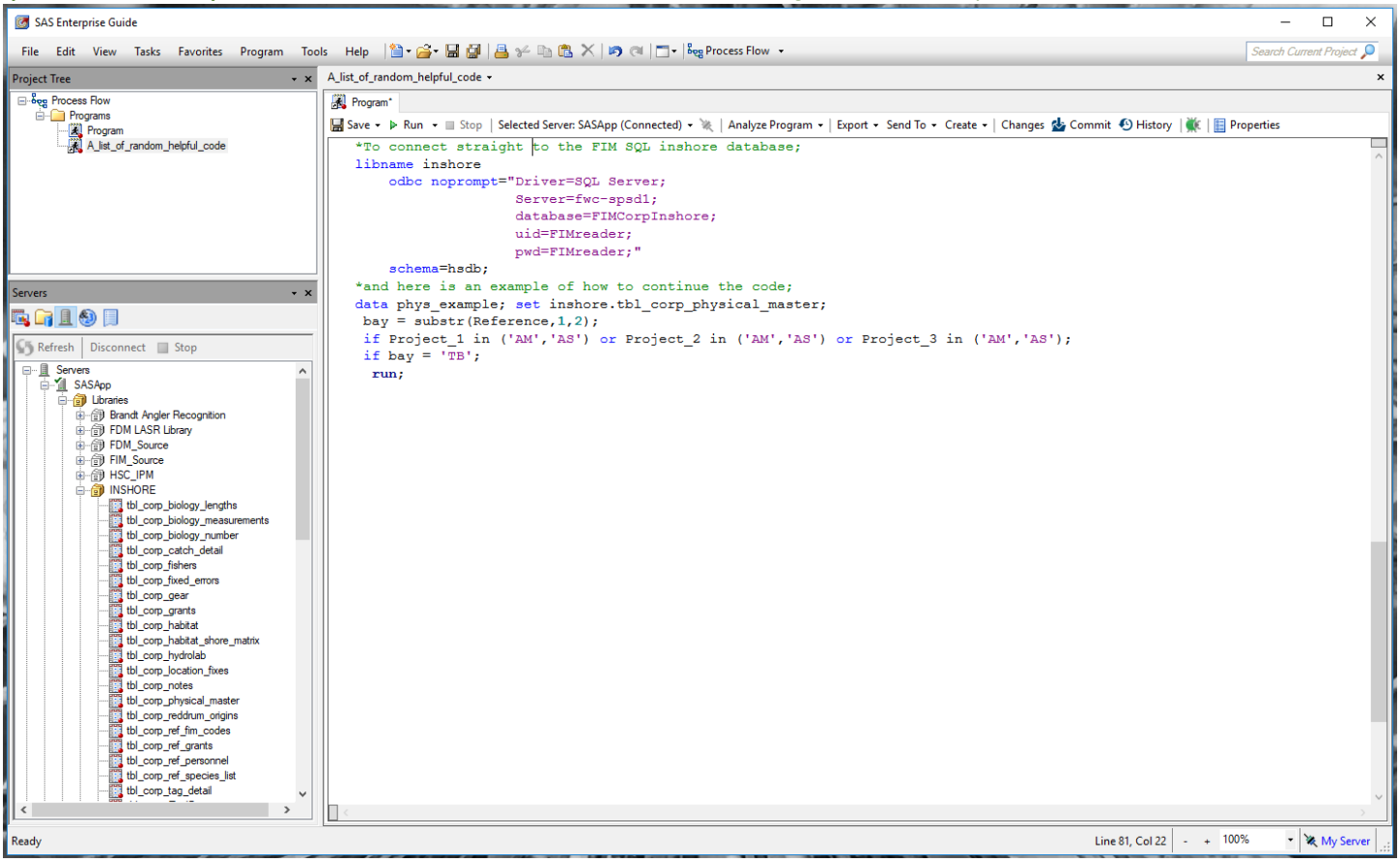
Connecting to the FIM SQL database through SAS and R

1. If you use SAS: use a simple libname statement

```
*To connect straight to the FIM SQL inshore database;
libname inshore
      odbc noprompt="Driver=SQL Server;
                     Server=fwc-spsd2;
                     database=FIMCorpInshore;
                   uid=FIMreader;
                   pwd=FIMreader;"
      schema=hsdb;

*Here is an example of how to continue the code;
data phys_example; set inshore.tbl_corp_physical_master;
 bay = substr(Reference,1,2);
 if Project_1 in ('AM','AS') or Project_2 in ('AM','AS') or Project_3 in
('AM','AS');
 if bay = 'TB';
  run;

*Here is what it looks like in SAS, where you can see that the “INSHORE” library
(on the left) contains all the tables of the SQL database;
```



Connecting to the FIM SQL database through SAS and R

2. If you use R: here you have multiple options, but these are the two I've found most useful so far

You might want to make sure R, RStudio, and Java are all the same (32-bit or 64-bit) if you are going to be using outputs/reporting features of R (i.e. R Markdown).

Option 1: Connect to SQL and **use SQL code** (copy & paste) in R with the "dbGetQuery" function

```
#Install & load packages
library(odbc)
library(DBI)

#Connect to FIM SQL database
connex <- dbConnect(odbc::odbc(),
                    driver = "SQL Server",
                    server = "fwc-spsd2",
                    uid = "FIMreader",
                    pwd = "FIMreader")

#Test the connection with a code to find a species name or NODCCODE from the species list table
Find.Species <- dbGetQuery(connex, "SELECT [NODCCODE]
                           ,[TSN]
                           ,[spp_code]
                           ,[Scientificname]
                           ,[Commonname]
                           ,[Manufactured_nodccode]
                           ,[Manufactured_tsn]
                           ,[spp_code_old]
                           FROM [FIMCorpInshore].[hsdb].[tbl_corp_ref_species_list]
                                where Commonname like '%pinfish%'
                      ")

#View the Find.Species results
Find.Species
```

```
> Find.Species
    NODCCODE    TSN        spp_code      Scientificname       Commonname
Manufactured_nodccode Manufactured_tsn  spp_code_old
1 8835430201 169187 Lag. rhomboides  Lagodon rhomboides          Pinfish
FALSE            FALSE L. rhomboides
2 8835430401 169192 Dip. holbrookii Diplodus holbrookii Spottail Pinfish
FALSE            FALSE  D. holbrooki
```

Connecting to the FIM SQL database through SAS and R

Option 2: Connect to SQL but then use R to query and analyze

This allows you to set up multiple connections, for each database

If you have datasets in same database but different schema, the "in_schema" function becomes very helpful when pulling data

This option allows you to filter/process data remotely before "collecting" it to your computer – if you can do your large query steps on the server, I recommend it before “collecting” data to your local machine. Just keep in mind that in order to query data on the server, the R code has to be translated (in the background) to SQL language (this is what the dbplyr package is for), so not all functions are able to be translated.

Therefore, sometimes it is beneficial to "collect" more to R and then process using R language

```
#Install & load packages
library(odbc)
library(dplyr)  #for manipulating data; supports tidyverse code
library(dbplyr)  #this package will transfer dplyr-like commands into SQL commands in the background

#Connect to FIMCorpInshore SQL database
Con.corp <- dbConnect(odbc::odbc(),
                      driver = "SQL Server",
                      server = "fwc-spsd2",
                      database = "FIMCorpInshore",
                      uid = "FIMreader",
                      pwd = "FIMreader")

#Test the connection by pulling time data from the table of physical data
timedat <- tbl(Con.corp,in_schema("hsdb", "tbl_corp_physical_master")) %>%
  select(Reference, TripID, Gear, StartTime) %>%
  collect()  #use collect to actually pull the data down from SQL to your local environment

# View top few rows of the timedat dataset
head(timedat)
```

```
> head(timedat)
# A tibble: 6 x 4
  Reference     TripID       Gear StartTime
  <chr>         <chr>       <int> <chr>
1 ANM2004100101 ANM20041001   300 09:31:00
2 ANM2004100102 ANM20041001   300 10:11:00
3 ANM2004100103 ANM20041001   301 12:11:00
4 ANM2004100104 ANM20041001    23 12:34:00
5 ANM2004100105 ANM20041001    23 12:49:00
6 ANM2004100106 ANM20041001    23 13:09:00
```