San Diego PharmaSUG
June 1-4th 2025

A female megachile leafcutter bee collecting pollen from a blanket flower (*Gaillardia aristata*) to use in making a beeloaf. Image courtesy of Jim McCulloch.

Bee-yond the Basics: Harnessing SAS, SQL, and Python for Data Analytics in Pharma

PHARMASUG PRE-CONFERENCE SEMINAR

CHARU SHANKAR

TABLE OF CONTENTS

Introduction

What if the key to saving our ecosystems lies in the data? The sharp decline of native pollinators like bumblebees threatens more than just plant diversity—it puts entire food chains and agricultural systems at risk. Research[1] shows that plants pollinated by native bees are **2 to 5 times more likely** to survive and reproduce than those pollinated by non-native species. That's native precision: powerful in nature, and just as powerful in code.

In this session, we'll harness the strengths of Python and SAS by offering a side-by-side comparison in SAS Viya Workbench to uncover patterns in pollinator data, compare tools, and explore how the right data and platform can help protect the buzzing heroes of our ecosystems—native bees.

Tool Selection & Integration

In the world of data science, professionals often face a similar kind of buzz — especially when deciding between tools like **SAS and Python**. Studies[2] show that data scientists allocate approximately 45% of their time to data preparation activities, including loading and cleaning data. This not only slows down workflows but also adds unnecessary complexity to day-to-day tasks.

SAS Data Step has long been valued for its power, performance, and reliability, particularly with large datasets. Meanwhile, Python, with its flexible and expansive ecosystem — including libraries like Pandas, now with over 100,000 GitHub stars — is a go-to for many modern data scientists.

In this session, we'll harness the strengths of Python and SAS by offering **a** side-by-side comparison in SAS Viya Workbench to uncover patterns, compare tools, and explore how the right data, and the right platform can help protect what matters most.

Architecture

SAS is procedural, Python is object-oriented—two very different approaches that complement each other.

Procedural programming, like SAS, follows a step-by-step approach—code runs in a clear, linear order, which makes it great for data transformation and reporting. Object-oriented programming (OOP), like Python, organizes code into reusable objects that combine data and behavior, making it more flexible and modular. Both have their strengths, and understanding their differences helps you pick the right tool for the task.

Tool

In Workbench, you'll see software-as-a-service in its truest form: fully cloud-based, with data, storage, and memory ready on demand. It felt as groundbreaking as the first time I streamed Netflix.

You don't own the infrastructure, you rent just what you need. But the benefits are big: output persists, it's cloud-native, and accessible from anywhere. Designed for both data scientists and business users, you're no longer bound by on-prem limitations.

Workbench offers a unified environment for everything: data prep, visualization, advanced analytics, and machine learning—all in one place. The simple notebook UI(user interface), in-memory processing for faster compute, and elastic scalability mean it's ready for any workload. Need more power? Just add cores. And with built-in collaboration, multiple users can explore, code, and build together, seamlessly.

SAS Data Science Methodology

In this SAS Viya Workbench seminar, we'll walk through a practical, 5-step data journey—from access to insights.
* We'll start by loading the analysis tables (Access)
* then explore the data for quality issues (Explore)
* Next, we'll clean and enrich it (Prepare)
* uncover patterns (Analyze)
* and finally create interactive reports (Report).

It's a full workflow—streamlined, visual, and ready for action!

The Data

🐝 **Rusty Patched Bumble Bee**

Source: Rusty Patched Bumble Bee - USDA Forest Service

Pollinators like bees, butterflies, and birds play a vital role in keeping our ecosystems—and our food supply—thriving. They help fertilize plants, ensuring the growth of fruits, vegetables, and seeds. But pollinator populations are in decline due to habitat loss, pesticides, climate change, and disease. Their dwindling numbers are a warning buzz—we need to act to protect these tiny powerhouses before the ripple effects grow too large to ignore.

These are the 4 tables we will use
1. Pattern_decline_N_American_Bumblebees.csv[3]
2. Pattern_decline_Mexican_Bumblebees.csv[3]
3. Bumblebee_Others_Scientific_Common_Names - scientific and common name lookup[4]
4. Native_vs_nonnative_bumblebee_sighting_pollinators_of_farm_data_for_publication[5]

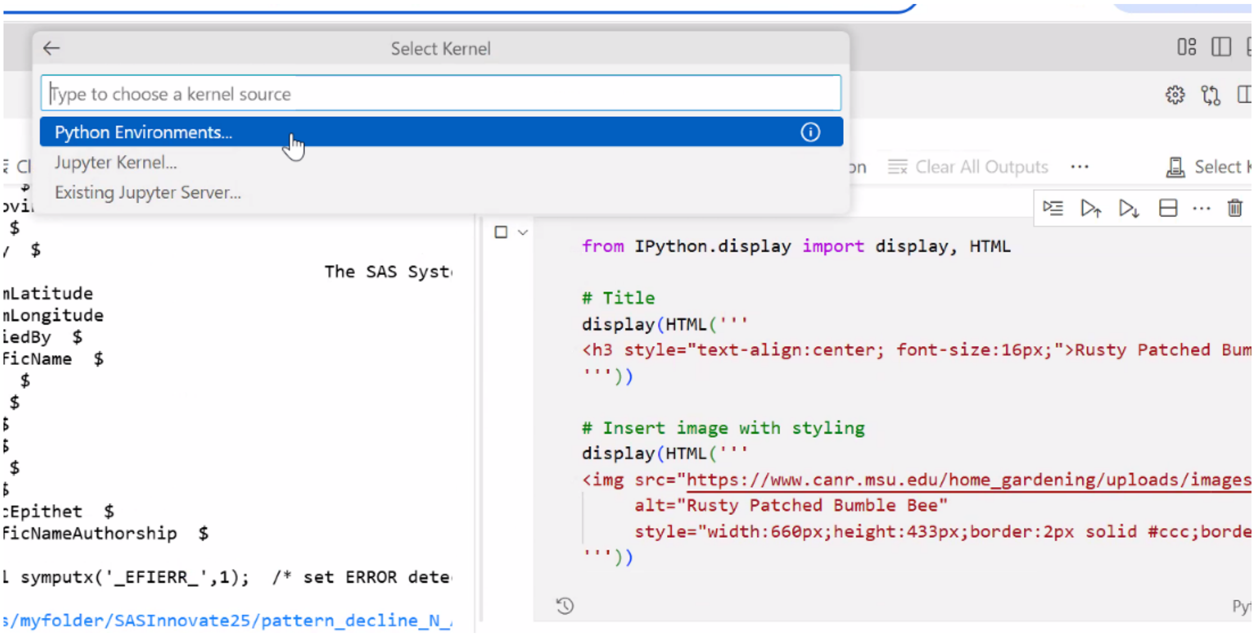
Terminology

Code Cells: These cells are where you write and run executable code.

Markdown Cells: These allow you to add formatted text—like headings, lists, and links—using lightweight Markdown syntax instead of code.

Kernel: The kernel is the computational engine that runs your code, tracks variables, and returns results within a notebook environment like Jupyter or SAS Viya Workbench.

Python Environment: In SAS Viya Workbench, this is a preconfigured, containerized workspace that includes a specific Python version (e.g., 3.10), essential libraries (like pandas, numpy, matplotlib), SAS integration via SASPy, and a Jupyter-compatible kernel—providing a consistent, isolated environment for analytics.



```
                                              The SAS Syst
mLatitude
mLongitude
iedBy  $
FicName  $
:Epithet  $
FicNameAuthorship  $

l symputx('_EFIERR_',1);  /* set ERROR dete

s/myfolder/SASInnovate25/pattern_decline_N_
```

```
from IPython.display import display, HTML

# Title
display(HTML('''
<h3 style="text-align:center; font-size:16px;">Rusty Patched Bum
'''))

# Insert image with styling
display(HTML('''
<img src="https://www.canr.msu.edu/home_gardening/uploads/images
     alt="Rusty Patched Bumble Bee"
     style="width:660px;height:433px;border:2px solid #ccc;borde
'''))
```

Data Access

SAS Data Access

In this project, we're using real-world data on North American bumblebees. We'll read the 4 CSV files into SAS datasets using the Viya Workbench, making it easy to manipulate and analyze as we explore trends and patterns in pollinator populations.

Importing an image

```
/* Begin HTML5 output to display content in a browser or output window */
ods html5;

/* Create a title for the output */
title "Rusty Patched Bumble Bee";
```

```
/* Use ODS HTML5 to insert an image into the output */
ods html5 text =
    '<img
src="https://images.ctfassets.net/cnu0m8re1exe/4nVzwubwk0QTMF15jLEoDI/05c89bb1ef18c12
20bf15b81f789c15f/Low-
Res_Rusty_patched_bumble_bee_cbolt_08082015_DSC9052_patched_72.jpg?fm=jpg&fl=progress
ive&w=660&h=433&fit=fill"
    alt="Rusty Patched Bumble Bee" /* Alternative text description for accessibility
*/
    style="width:660px;height:433px;border:2px solid #ccc;border-radius:10px;">' /*
Add styling to control image size and appearance */;


/* Close the ODS HTML5 output to stop writing to the output window */
ods html5 close;
```

🐝 Rusty Patched Bumble Bee

Source: Rusty Patched Bumble Bee - USDA Forest Service

Reading a CSV File into a SAS Dataset

Reading a CSV file into a SAS dataset is the first step in data analysis using SAS. This task involves using the PROC IMPORT procedure to load data from a CSV file into a SAS dataset, making the data accessible for various operations such as filtering, grouping, and aggregating. The dataset is stored in the WORK library for temporary use during the session.

SAS Code

```
/* Read the north american bumblebee CSV file into a SAS dataset for easy data
manipulation and analysis -workbench workspace*/
proc import
file="/workspaces/myfolder/Pharmasug25/pattern_decline_N_American_Bumblebees.csv"
out=dst1 dbms=csv replace;
run;
```

What information can you gather from the PROC IMPORT log about how SAS interpreted the columns in your CSV hive?
Hint: Check the log to review information about data types, lengths, headers, and delimiters. This will help understand how SAS is reading raw data before analysis begins.

When you run a PROC IMPORT in SAS, it often generates a DATA step in the background because SAS is essentially writing the code needed to read in your file—especially when importing from structured formats like CSV or Excel. Here's why:

What's Happening Behind the Scenes:

- PROC IMPORT acts as a helper procedure: it examines the structure of your input file (like variable names, types, and formats).
- Then, SAS auto-generates a DATA step with an INFILE statement (for CSVs) or LIBNAME reference (for Excel files).
- This code handles the actual reading and conversion of your file into a SAS dataset.

Why It's Helpful:

- You don't have to write the DATA step yourself—SAS does the heavy lifting.
- If you're curious, you can add the PROC IMPORT option OUT=your_dataset DBMS=CSV REPLACE; and check the LOG to see the generated DATA step code.

🧪 Bonus Tip:
You can even copy that auto-generated code from the log and tweak it for custom import behavior. Great for power users who want more control over data types, formats, or delimiters.

SAS Log

```
617  /** LOG_START_INDICATOR **/
618  title;footnote;ods _all_ close;
619  ods graphics on;
620  ods html5(id=vscode) style=HTMLEncore options(bitmap_mode='inline'
svg_mode='inline');
NOTE: Writing HTML5(VSCODE) Body file: sashtml7.htm
621  %let _SASPROGRAMFILE =
%nrquote(%nrstr(/workspaces/myfolder/Pharmasug25/1SASaccess.sasnb));
622  /* Read the north american bumblebee CSV file into a SAS dataset for easy data
manipulation and analysis */
623  proc import
file="/workspaces/myfolder/Pharmasug25/pattern_decline_N_American_Bumblebees.csv"
out=dst1 dbms=csv replace;
624  run;
625   /**********************************************************************
626   *   PRODUCT:   SAS
627   *   VERSION:   V.04.00
628   *   CREATOR:   External File Interface
629   *   DATE:      28APR25
630   *   DESC:      Generated SAS Datastep Code
631   *   TEMPLATE SOURCE:  (None Specified.)
632   ***********************************************************************/
633      data WORK.DST1    ;
634      %let _EFIERR_ = 0; /* set the ERROR detection macro variable */
635      infile
'/workspaces/myfolder/Pharmasug25/pattern_decline_N_American_Bumblebees.csv'
delimiter = ',' MISSOVER DSD
635! lrecl=32767 firstobs=2 ;
```

```
636          informat id best32. ;
637          informat institutionCode $8. ;
638          informat collectionCode $4. ;
639          informat basisOfRecord $17. ;
640          informat occurrenceID best32. ;
641          informat catalogNumber $12. ;
642          informat recordedBy $34. ;
643          informat year best32. ;
644          informat month best32. ;
645          informat day best32. ;
646          informat country $3. ;
647          informat stateProvince $7. ;
648          informat county $8. ;
649          informat locality $45. ;
650          informat verbatimLatitude best32. ;
651          informat verbatimLongitude best32. ;
652          informat identifiedBy $18. ;
653          informat scientificName $19. ;
654          informat kingdom $8. ;
655          informat phylum $10. ;
656          informat class $7. ;
657          informat order $11. ;
658          informat family $6. ;
659          informat genus $6. ;
660          informat specificEpithet $12. ;
661          informat scientificNameAuthorship $12. ;
662          format id best12. ;
663          format institutionCode $8. ;
664          format collectionCode $4. ;
17                                                          The SAS System
Monday, April 28, 2025 08:03:00 PM
665          format basisOfRecord $17. ;
666          format occurrenceID best12. ;
667          format catalogNumber $12. ;
668          format recordedBy $34. ;
669          format year best12. ;
670          format month best12. ;
671          format day best12. ;
672          format country $3. ;
673          format stateProvince $7. ;
674          format county $8. ;
675          format locality $45. ;
676          format verbatimLatitude best12. ;
677          format verbatimLongitude best12. ;
678          format identifiedBy $18. ;
679          format scientificName $19. ;
680          format kingdom $8. ;
681          format phylum $10. ;
682          format class $7. ;
683          format order $11. ;
684          format family $6. ;
685          format genus $6. ;
686          format specificEpithet $12. ;
687          format scientificNameAuthorship $12. ;
688       input
```

```
689                    id
690                    institutionCode  $
691                    collectionCode  $
692                    basisOfRecord  $
693                    occurrenceID
694                    catalogNumber  $
695                    recordedBy  $
696                    year
697                    month
698                    day
699                    country  $
700                    stateProvince  $
701                    county  $
702                    locality  $
703                    verbatimLatitude
704                    verbatimLongitude
705                    identifiedBy  $
706                    scientificName  $
707                    kingdom  $
708                    phylum  $
709                    class  $
710                    order  $
711                    family  $
712                    genus  $
713                    specificEpithet  $
714                    scientificNameAuthorship  $
715         ;
716         if _ERROR_ then call symputx('_EFIERR_',1);  /* set ERROR detection macro 
variable */
717         run;
NOTE: The infile 
'/workspaces/myfolder/Pharmasug25/pattern_decline_N_American_Bumblebees.csv' is:

Filename=/workspaces/myfolder/Pharmasug25/pattern_decline_N_American_Bumblebees.csv,
      Owner Name=sas,Group Name=sas,
      Access Permission=-rw-rw----,
18                                                          The SAS System
Monday, April 28, 2025 08:03:00 PM
      Last Modified=01Apr2025:18:57:36,
      File Size (bytes)=15836577
NOTE: 66907 records were read from the infile 
'/workspaces/myfolder/Pharmasug25/pattern_decline_N_American_Bumblebees.csv'.
      The minimum record length was 173.
      The maximum record length was 333.
NOTE: The data set WORK.DST1 has 66907 observations and 26 variables.
NOTE: DATA statement used (Total process time):
      real time           0.13 seconds
      cpu time            0.15 seconds

66907 rows created in WORK.DST1 from 
/workspaces/myfolder/Pharmasug25/pattern_decline_N_American_Bumblebees.csv.



NOTE: WORK.DST1 data set was successfully created.
```

```
NOTE: The data set WORK.DST1 has 66907 observations and 26 variables.
NOTE: PROCEDURE IMPORT used (Total process time):
      real time           0.22 seconds
      cpu time            0.23 seconds

718  ;*';*";*/;run;quit;ods html5(id=vscode) close;
```

SAS Code

```
/* Read the mexican bumblebee CSV file into a SAS dataset*/
proc import
file="/workspaces/myfolder/Pharmasug25/pattern_decline_Mexican_Bumblebees.csv"
out=dst2 dbms=csv replace;
run;
```

SAS Log

```
719  /** LOG_START_INDICATOR **/
720  title;footnote;ods _all_ close;
721  ods graphics on;
722  ods html5(id=vscode) style=HTMLEncore options(bitmap_mode='inline'
svg_mode='inline');
NOTE: Writing HTML5(VSCODE) Body file: sashtml8.htm
723  %let _SASPROGRAMFILE =
%nrquote(%nrstr(/workspaces/myfolder/Pharmasug25/1SASaccess.sasnb));
724  /* Read the mexican bumblebee CSV file into a SAS dataset for easy data
manipulation and analysis*/
725  proc import
file="/workspaces/myfolder/Pharmasug25/pattern_decline_Mexican_Bumblebees.csv"
out=dst2 dbms=csv replace;
726  run;
727   /**********************************************************************
728   *   PRODUCT:   SAS
729   *   VERSION:   V.04.00
730   *   CREATOR:   External File Interface
731   *   DATE:      28APR25
732   *   DESC:      Generated SAS Datastep Code
733   *   TEMPLATE SOURCE:  (None Specified.)
734   ***********************************************************************/
735     data WORK.DST2    ;
736     %let _EFIERR_ = 0; /* set the ERROR detection macro variable */
737     infile
'/workspaces/myfolder/Pharmasug25/pattern_decline_Mexican_Bumblebees.csv' delimiter =
',' MISSOVER DSD lrecl=32767
737!  firstobs=2 ;
738        informat id best32. ;
739        informat institutionCode $8. ;
740        informat collectionCode $4. ;
741        informat basisOfRecord $17. ;
742        informat occurrenceID $1. ;
743        informat catalogNumber $10. ;
744        informat recordedBy $1. ;
745        informat year best32. ;
746        informat month best32. ;
747        informat day best32. ;
```

```
748          informat country $6. ;
749          informat stateProvince $7. ;
19                                                          The SAS System
Monday, April 28, 2025 08:03:00 PM
750          informat county $1. ;
751          informat locality $35. ;
752          informat verbatimLatitude best32. ;
753          informat verbatimLongitude best32. ;
754          informat identifiedBy $1. ;
755          informat scientificName $20. ;
756          informat kingdom $8. ;
757          informat phylum $10. ;
758          informat class $7. ;
759          informat order $11. ;
760          informat family $6. ;
761          informat genus $6. ;
762          informat specificEpithet $13. ;
763          informat scientificNameAuthorship $13. ;
764          format id best12. ;
765          format institutionCode $8. ;
766          format collectionCode $4. ;
767          format basisOfRecord $17. ;
768          format occurrenceID $1. ;
769          format catalogNumber $10. ;
770          format recordedBy $1. ;
771          format year best12. ;
772          format month best12. ;
773          format day best12. ;
774          format country $6. ;
775          format stateProvince $7. ;
776          format county $1. ;
777          format locality $35. ;
778          format verbatimLatitude best12. ;
779          format verbatimLongitude best12. ;
780          format identifiedBy $1. ;
781          format scientificName $20. ;
782          format kingdom $8. ;
783          format phylum $10. ;
784          format class $7. ;
785          format order $11. ;
786          format family $6. ;
787          format genus $6. ;
788          format specificEpithet $13. ;
789          format scientificNameAuthorship $13. ;
790       input
791                  id
792                  institutionCode  $
793                  collectionCode  $
794                  basisOfRecord  $
795                  occurrenceID  $
796                  catalogNumber  $
797                  recordedBy  $
798                  year
799                  month
800                  day
```

```
801                    country  $
802                    stateProvince  $
803                    county  $
804                    locality  $
805                    verbatimLatitude
806                    verbatimLongitude
807                    identifiedBy  $
20                                                         The SAS System
Monday, April 28, 2025 08:03:00 PM
808                    scientificName  $
809                    kingdom  $
810                    phylum  $
811                    class  $
812                    order  $
813                    family  $
814                    genus  $
815                    specificEpithet  $
816                    scientificNameAuthorship  $
817          ;
818          if _ERROR_ then call symputx('_EFIERR_',1);  /* set ERROR detection macro 
variable */
819          run;
NOTE: The infile 
'/workspaces/myfolder/Pharmasug25/pattern_decline_Mexican_Bumblebees.csv' is:

Filename=/workspaces/myfolder/Pharmasug25/pattern_decline_Mexican_Bumblebees.csv,
      Owner Name=sas,Group Name=sas,
      Access Permission=-rw-rw----,
      Last Modified=01Apr2025:18:57:36,
      File Size (bytes)=5490
NOTE: 24 records were read from the infile 
'/workspaces/myfolder/Pharmasug25/pattern_decline_Mexican_Bumblebees.csv'.
      The minimum record length was 197.
      The maximum record length was 234.
NOTE: The data set WORK.DST2 has 24 observations and 26 variables.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.01 seconds

24 rows created in WORK.DST2 from 
/workspaces/myfolder/Pharmasug25/pattern_decline_Mexican_Bumblebees.csv.



NOTE: WORK.DST2 data set was successfully created.
NOTE: The data set WORK.DST2 has 24 observations and 26 variables.
NOTE: PROCEDURE IMPORT used (Total process time):
      real time           0.05 seconds
      cpu time            0.06 seconds

820  ;*';*";*/;run;quit;ods html5(id=vscode) close;
```

SAS Code

```
/* Read the scientific and common name lookup csv file into a SAS dataset */
```

```
proc import
file="/workspaces/myfolder/Pharmasug25/Bumblebee_Others_Scientific_Common_Names.csv"
out=dst3 dbms=csv replace;
run;
```

SAS Log

```
821  /** LOG_START_INDICATOR **/
822  title;footnote;ods _all_ close;
823  ods graphics on;
824  ods html5(id=vscode) style=HTMLEncore options(bitmap_mode='inline'
svg_mode='inline');
NOTE: Writing HTML5(VSCODE) Body file: sashtml9.htm
825  %let _SASPROGRAMFILE =
%nrquote(%nrstr(/workspaces/myfolder/Pharmasug25/1SASaccess.sasnb));
826  /* Read the scientific and common name lookup csv file into a SAS dataset */
827  proc import
file="/workspaces/myfolder/Pharmasug25/Bumblebee_Others_Scientific_Common_Names.csv"
out=dst3 dbms=csv replace;
828  run;
829   /**********************************************************************
830   *   PRODUCT:   SAS
831   *   VERSION:   V.04.00
832   *   CREATOR:   External File Interface
833   *   DATE:      28APR25
834   *   DESC:      Generated SAS Datastep Code
835   *   TEMPLATE SOURCE:  (None Specified.)
21                                                        The SAS System
Monday, April 28, 2025 08:03:00 PM
836   **********************************************************************/
837      data WORK.DST3    ;
838      %let _EFIERR_ = 0; /* set the ERROR detection macro variable */
839      infile
'/workspaces/myfolder/Pharmasug25/Bumblebee_Others_Scientific_Common_Names.csv'
delimiter = ',' MISSOVER DSD
839! lrecl=32767 firstobs=2 ;
840         informat ScientificName $22. ;
841         informat Species $11. ;
842         informat specificEpithet $14. ;
843         informat CommonName $27. ;
844         informat Description $115. ;
845         informat Source $13. ;
846         format ScientificName $22. ;
847         format Species $11. ;
848         format specificEpithet $14. ;
849         format CommonName $27. ;
850         format Description $115. ;
851         format Source $13. ;
852      input
853                  ScientificName  $
854                  Species  $
855                  specificEpithet  $
856                  CommonName  $
857                  Description  $
858                  Source  $
859      ;
```

```
860      if _ERROR_ then call symputx('_EFIERR_',1);  /* set ERROR detection macro 
variable */
861      run;
NOTE: The infile 
'/workspaces/myfolder/Pharmasug25/Bumblebee_Others_Scientific_Common_Names.csv' is:
      
Filename=/workspaces/myfolder/Pharmasug25/Bumblebee_Others_Scientific_Common_Names.cs
v,
      Owner Name=sas,Group Name=sas,
      Access Permission=-rw-rw----,
      Last Modified=01Apr2025:18:57:36,
      File Size (bytes)=28357
NOTE: 162 records were read from the infile 
'/workspaces/myfolder/Pharmasug25/Bumblebee_Others_Scientific_Common_Names.csv'.
      The minimum record length was 38.
      The maximum record length was 234.
NOTE: The data set WORK.DST3 has 162 observations and 6 variables.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds
      

162 rows created in WORK.DST3 from 
/workspaces/myfolder/Pharmasug25/Bumblebee_Others_Scientific_Common_Names.csv.
  
  
  
NOTE: WORK.DST3 data set was successfully created.
NOTE: The data set WORK.DST3 has 162 observations and 6 variables.
NOTE: PROCEDURE IMPORT used (Total process time):
      real time           0.04 seconds
      cpu time            0.05 seconds
      

862  ;*';*";*/;run;quit;ods html5(id=vscode) close;
```

SAS Code

```
/* Read the native vs non-native bee csv file into a SAS dataset */
proc import 
file="/workspaces/myfolder/Pharmasug25/native_vs_nonnative_bumblebee_sighting_pollina
tors_of_farm_data_for_publication.csv" out=work.dst4 dbms=csv replace;
run;
```

SAS Log

```
863  /** LOG_START_INDICATOR **/
864  title;footnote;ods _all_ close;
22                                                         The SAS System                  
Monday, April 28, 2025 08:03:00 PM
865  ods graphics on;
866  ods html5(id=vscode) style=HTMLEncore options(bitmap_mode='inline' 
svg_mode='inline');
NOTE: Writing HTML5(VSCODE) Body file: sashtml10.htm
```

```
867  %let _SASPROGRAMFILE =
%nrquote(%nrstr(/workspaces/myfolder/Pharmasug25/1SASaccess.sasnb));
868  /* Read the native vs nonnative bee csv file into a SAS dataset */
869  proc import
869!
file="/workspaces/myfolder/Pharmasug25/native_vs_nonnative_bumblebee_sighting_pollina
tors_of_farm_data_for_publication.csv"
869! out=work.dst4 dbms=csv replace;
870  run;
871   /**********************************************************************
872   *   PRODUCT:   SAS
873   *   VERSION:   V.04.00
874   *   CREATOR:   External File Interface
875   *   DATE:      28APR25
876   *   DESC:      Generated SAS Datastep Code
877   *   TEMPLATE SOURCE:  (None Specified.)
878   ***********************************************************************/
879      data WORK.DST4    ;
880      %let _EFIERR_ = 0; /* set the ERROR detection macro variable */
881      infile
881!
'/workspaces/myfolder/Pharmasug25/native_vs_nonnative_bumblebee_sighting_pollinators_
of_farm_data_for_publication.csv'
881! delimiter = ',' MISSOVER DSD lrecl=32767 firstobs=2 ;
882         informat "no of specimens in sample"N $1. ;
883         informat date mmddyy10. ;
884         informat year best32. ;
885         informat season $11. ;
886         informat site $1. ;
887         informat plot $10. ;
888         informat site_plot $4. ;
889         informat site_plot_year $9. ;
890         informat sampling $12. ;
891         informat "plant species"N $17. ;
892         informat "start time"N time20.3 ;
893         informat "end time"N $1. ;
894         informat "vegetation cover"N $1. ;
895         informat "floral cover"N $1. ;
896         informat "height of vegetation (cm)"N $1. ;
897         informat "Achillea millefolium"N $1. ;
898         informat "Agastache foeniculum"N $1. ;
899         informat "Calendula officinalis"N $1. ;
900         informat "Leucanthemum vulgare"N $1. ;
901         informat "Leucanthemum maximum"N $1. ;
902         informat "Cichorium intybus"N $1. ;
903         informat "Coronilla varia"N $1. ;
904         informat "Cosmos bipinnatus"N $1. ;
905         informat "Daucus carota"N $1. ;
906         informat "Linum perenne"N $1. ;
907         informat "Lobularia maritima"N $1. ;
908         informat "Lotus corniculatus"N $1. ;
909         informat "Melilotus officinalis"N $1. ;
910         informat "Origanum vulgare"N $1. ;
911         informat "Papaver rhoeas"N $1. ;
912         informat "Salvia officinalis"N $1. ;
```

```
913         informat "Trifolium incarnatum"N $1. ;
914         informat "Trifolium pratense"N $1. ;
915         informat "Trifolium repens"N $1. ;
916         informat "Viola cornuta"N $1. ;
23                                                         The SAS System
Monday, April 28, 2025 08:03:00 PM
917         informat "Asclepias tuberosa"N $1. ;
918         informat "Baptisia australis"N $1. ;
919         informat "Bidens aristosa"N $1. ;
920         informat "Chamaecrista fasciculata"N $1. ;
921         informat "Chamaecrista nictitans"N $1. ;
922         informat "Eupatorium perfoliatum"N $1. ;
923         informat "Helenium flexuosum"N $1. ;
924         informat "Lespedeza virginica"N $1. ;
925         informat "Liatris pilosa"N $1. ;
926         informat "Lupinus perennis"N $1. ;
927         informat "Monarda punctata"N $1. ;
928         informat "Penstemon laevigatus"N $1. ;
929         informat "Pycnanthemum tenuifolium"N $1. ;
930         informat "Rudbeckia hirta"N $1. ;
931         informat "Rudbeckia triloba"N $1. ;
932         informat "Sisyrinchium angustifolium"N $1. ;
933         informat "Solidago odora"N $1. ;
934         informat "Solidago nemoralis"N $1. ;
935         informat "Symphyotrichum laeve"N $1. ;
936         informat "Tradescantia virginiana"N $1. ;
937         informat "Verbena hastata"N $1. ;
938         informat "no bee"N $1. ;
939         informat "small green metallic bee"N $1. ;
940         informat "lrg green bee"N $1. ;
941         informat "bumble bee"N $1. ;
942         informat "lrg carpenter bee"N $1. ;
943         informat "small dark bee"N $1. ;
944         informat "honey bee"N $1. ;
945         informat Megachile $1. ;
946         informat Anthidium $1. ;
947         informat Species $18. ;
948         informat Sex $2. ;
949         informat "only genus level"N best32. ;
950         informat "bee specialist"N $2. ;
951         informat "specialized on"N $1. ;
952         informat parasitic $2. ;
953         informat nesting $6. ;
954         informat status $1. ;
955         informat "non-native bee"N $2. ;
956         informat "other species characteristics"N $1. ;
957         format "no of specimens in sample"N $1. ;
958         format date mmddyy10. ;
959         format year best12. ;
960         format season $11. ;
961         format site $1. ;
962         format plot $10. ;
963         format site_plot $4. ;
964         format site_plot_year $9. ;
965         format sampling $12. ;
```

```
966         format "plant species"N $17. ;
967         format "start time"N time20.3 ;
968         format "end time"N $1. ;
969         format "vegetation cover"N $1. ;
970         format "floral cover"N $1. ;
971         format "height of vegetation (cm)"N $1. ;
972         format "Achillea millefolium"N $1. ;
973         format "Agastache foeniculum"N $1. ;
974         format "Calendula officinalis"N $1. ;
24                                                         The SAS System
Monday, April 28, 2025 08:03:00 PM
975         format "Leucanthemum vulgare"N $1. ;
976         format "Leucanthemum maximum"N $1. ;
977         format "Cichorium intybus"N $1. ;
978         format "Coronilla varia"N $1. ;
979         format "Cosmos bipinnatus"N $1. ;
980         format "Daucus carota"N $1. ;
981         format "Linum perenne"N $1. ;
982         format "Lobularia maritima"N $1. ;
983         format "Lotus corniculatus"N $1. ;
984         format "Melilotus officinalis"N $1. ;
985         format "Origanum vulgare"N $1. ;
986         format "Papaver rhoeas"N $1. ;
987         format "Salvia officinalis"N $1. ;
988         format "Trifolium incarnatum"N $1. ;
989         format "Trifolium pratense"N $1. ;
990         format "Trifolium repens"N $1. ;
991         format "Viola cornuta"N $1. ;
992         format "Asclepias tuberosa"N $1. ;
993         format "Baptisia australis"N $1. ;
994         format "Bidens aristosa"N $1. ;
995         format "Chamaecrista fasciculata"N $1. ;
996         format "Chamaecrista nictitans"N $1. ;
997         format "Eupatorium perfoliatum"N $1. ;
998         format "Helenium flexuosum"N $1. ;
999         format "Lespedeza virginica"N $1. ;
1000         format "Liatris pilosa"N $1. ;
1001         format "Lupinus perennis"N $1. ;
1002         format "Monarda punctata"N $1. ;
1003         format "Penstemon laevigatus"N $1. ;
1004         format "Pycnanthemum tenuifolium"N $1. ;
1005         format "Rudbeckia hirta"N $1. ;
1006         format "Rudbeckia triloba"N $1. ;
1007         format "Sisyrinchium angustifolium"N $1. ;
1008         format "Solidago odora"N $1. ;
1009         format "Solidago nemoralis"N $1. ;
1010         format "Symphyotrichum laeve"N $1. ;
1011         format "Tradescantia virginiana"N $1. ;
1012         format "Verbena hastata"N $1. ;
1013         format "no bee"N $1. ;
1014         format "small green metallic bee"N $1. ;
1015         format "lrg green bee"N $1. ;
1016         format "bumble bee"N $1. ;
1017         format "lrg carpenter bee"N $1. ;
1018         format "small dark bee"N $1. ;
```

```
1019         format "honey bee"N $1. ;
1020         format Megachile $1. ;
1021         format Anthidium $1. ;
1022         format Species $18. ;
1023         format Sex $2. ;
1024         format "only genus level"N best12. ;
1025         format "bee specialist"N $2. ;
1026         format "specialized on"N $1. ;
1027         format parasitic $2. ;
1028         format nesting $6. ;
1029         format status $1. ;
1030         format "non-native bee"N $2. ;
1031         format "other species characteristics"N $1. ;
1032      input
25                                                              The SAS System
Monday, April 28, 2025 08:03:00 PM
1033                  "no of specimens in sample"N  $
1034                  date
1035                  year
1036                  season  $
1037                  site  $
1038                  plot  $
1039                  site_plot  $
1040                  site_plot_year  $
1041                  sampling  $
1042                  "plant species"N  $
1043                  "start time"N
1044                  "end time"N  $
1045                  "vegetation cover"N  $
1046                  "floral cover"N  $
1047                  "height of vegetation (cm)"N  $
1048                  "Achillea millefolium"N  $
1049                  "Agastache foeniculum"N  $
1050                  "Calendula officinalis"N  $
1051                  "Leucanthemum vulgare"N  $
1052                  "Leucanthemum maximum"N  $
1053                  "Cichorium intybus"N  $
1054                  "Coronilla varia"N  $
1055                  "Cosmos bipinnatus"N  $
1056                  "Daucus carota"N  $
1057                  "Linum perenne"N  $
1058                  "Lobularia maritima"N  $
1059                  "Lotus corniculatus"N  $
1060                  "Melilotus officinalis"N  $
1061                  "Origanum vulgare"N  $
1062                  "Papaver rhoeas"N  $
1063                  "Salvia officinalis"N  $
1064                  "Trifolium incarnatum"N  $
1065                  "Trifolium pratense"N  $
1066                  "Trifolium repens"N  $
1067                  "Viola cornuta"N  $
1068                  "Asclepias tuberosa"N  $
1069                  "Baptisia australis"N  $
1070                  "Bidens aristosa"N  $
1071                  "Chamaecrista fasciculata"N  $
```

```
1072                      "Chamaecrista nictitans"N  $
1073                      "Eupatorium perfoliatum"N  $
1074                      "Helenium flexuosum"N  $
1075                      "Lespedeza virginica"N  $
1076                      "Liatris pilosa"N  $
1077                      "Lupinus perennis"N  $
1078                      "Monarda punctata"N  $
1079                      "Penstemon laevigatus"N  $
1080                      "Pycnanthemum tenuifolium"N  $
1081                      "Rudbeckia hirta"N  $
1082                      "Rudbeckia triloba"N  $
1083                      "Sisyrinchium angustifolium"N  $
1084                      "Solidago odora"N  $
1085                      "Solidago nemoralis"N  $
1086                      "Symphyotrichum laeve"N  $
1087                      "Tradescantia virginiana"N  $
1088                      "Verbena hastata"N  $
1089                      "no bee"N  $
1090                      "small green metallic bee"N  $
26                                                         The SAS System
Monday, April 28, 2025 08:03:00 PM
1091                      "lrg green bee"N  $
1092                      "bumble bee"N  $
1093                      "lrg carpenter bee"N  $
1094                      "small dark bee"N  $
1095                      "honey bee"N  $
1096                      Megachile  $
1097                      Anthidium  $
1098                      Species  $
1099                      Sex  $
1100                      "only genus level"N
1101                      "bee specialist"N  $
1102                      "specialized on"N  $
1103                      parasitic  $
1104                      nesting  $
1105                      status  $
1106                      "non-native bee"N  $
1107                      "other species characteristics"N  $
1108          ;
1109          if _ERROR_ then call symputx('_EFIERR_',1);  /* set ERROR detection macro
variable */
1110          run;
NOTE: The infile

'/workspaces/myfolder/Pharmasug25/native_vs_nonnative_bumblebee_sighting_pollinators_
of_farm_data_for_publication.csv' is:


Filename=/workspaces/myfolder/Pharmasug25/native_vs_nonnative_bumblebee_sighting_poll
inators_of_farm_data_for_publication.cs
      v,
      Owner Name=sas,Group Name=sas,
      Access Permission=-rw-rw----,
      Last Modified=01Apr2025:18:57:36,
      File Size (bytes)=833192
```

```
NOTE: 3744 records were read from the infile

'/workspaces/myfolder/Pharmasug25/native_vs_nonnative_bumblebee_sighting_pollinators_
of_farm_data_for_publication.csv'.
      The minimum record length was 134.
      The maximum record length was 392.
NOTE: The data set WORK.DST4 has 3744 observations and 75 variables.
NOTE: DATA statement used (Total process time):
      real time           0.02 seconds
      cpu time            0.02 seconds

3744 rows created in WORK.DST4 from
/workspaces/myfolder/Pharmasug25/native_vs_nonnative_bumblebee_sighting_pollinators_o
f_farm_data_for_publication.csv.



NOTE: WORK.DST4 data set was successfully created.
NOTE: The data set WORK.DST4 has 3744 observations and 75 variables.
NOTE: PROCEDURE IMPORT used (Total process time):
      real time           0.14 seconds
      cpu time            0.14 seconds

1111  ;*';*";*/;run;quit;ods html5(id=vscode) close;
```

SQL Data Access

The PROC SQL equivalent of a PROC IMPORT step doesn't exist in Base SAS, because PROC SQL cannot directly read external CSV files.

Python Data Access

On the Python side we'll work with the same real-world data on North American bumblebees. We'll use Python in SAS Viya Workbench to read four CSV files into dataframes, making it easy to manipulate, explore, and analyze trends and patterns in pollinator populations.

Importing an image in Python

Python Code

```
from IPython.display import display, HTML

# Title
display(HTML('''
<h3 style="text-align:center; font-size:16px;">Rusty Patched Bumble Bee</h3>
'''))

# Insert image with styling
display(HTML('''
<img src="https://www.canr.msu.edu/home_gardening/uploads/images/Photo1-
Rusty.jpg?language_id=1"
     alt="Rusty Patched Bumble Bee"
     style="width:660px;height:433px;border:2px solid #ccc;border-radius:10px;">
'''))
```

Output

Rusty Patched Bumble Bee

Bumble bees are vital pollinators for wildflowers and crops, thriving in cooler temperatures and low light. Their unique "buzz pollination" technique—vibrating flowers to release pollen—benefits plants like tomatoes, peppers, and cranberries.

Unfortunately, bumble bee populations are in sharp decline. Recent research by the Xerces Society and the IUCN Bumble Bee Specialist Group shows that over 28% of North American species face extinction risks. While some species have gained conservation support, others, like the Suckley and variable cuckoo bumble bees, remain overlooked.

Learn more about efforts to protect the rusty patched bumble bee here and explore this story map.

Reading a CSV File into a DataFrame

Reading a CSV file into a DataFrame is the first step in data analysis using pandas. This task involves loading data from a CSV file into a pandas DataFrame, which provides a powerful and flexible data structure for data manipulation and analysis. The read_csv function is used to read the CSV file, making the data easily accessible for various operations such as filtering, grouping, and aggregating.

Pandas is a powerful Python library used for data manipulation, cleaning, and analysis, especially with structured data like tables and spreadsheets.

Python Code

```
# Import the pandas library for data manipulation and analysis
import pandas as pd
```

The Concept of the Log-Python Console

Warning messages are output to the **console** where the Python code is being run, and is often seen in **Jupyter Notebooks**, **IPython** shells, or terminal-based Python sessions. It helps users debug potential issues with their code or data. In SAS, you have the log, In python you have a Python Console.

Python Code

```
# Read the North American bumblebee CSV file into a DataFrame for easy data
manipulation and analysis.
df1=pd.read_csv('/workspaces/myfolder/SASPythonDataScientists/pattern_decline_N_Ameri
can_Bumblebees.csv', encoding='latin-1')
```

Python Console

```
/tmp/ipykernel_785/3399206490.py:3: DtypeWarning: Columns (6,16) have mixed types.
Specify dtype option on import or set low_memory=False.

df1=pd.read_csv('/workspaces/myfolder/SASPythonDataScientists/pattern_decline_N_Ameri
can_Bumblebees.csv', encoding='latin-1')
```

This warning means columns 6 and 16 have mixed data types (e.g., numbers and text). You can resolve it by specifying the correct dtype or using low_memory=False to process the file in chunks. Specifying dtype is more precise, while low_memory=False is a quick but less reliable fix.

Python Code

```
# Read the North American bumblebee CSV file into a DataFrame for easy data
manipulation and analysis, forcing column 6 and 16 to be strings
```

```
df1=pd.read_csv('/workspaces/myfolder/Pharmasug25/pattern_decline_N_American_Bumblebe
es.csv', dtype={6: str, 16: str}, encoding='latin-1')
```

Python Code

```
# Read the Mexican bumblebee CSV file into a DataFrame for easy data manipulation and
analysis.
df2=pd.read_csv('/workspaces/myfolder/Pharmasug25/pattern_decline_Mexican_Bumblebees.
csv' , encoding='latin-1')
```

Python Code

```
# Read the scientific and common name lookup csv file into a DataFrame
df3=pd.read_csv('/workspaces/myfolder/Pharmasug25/Bumblebee_Others_Scientific_Common_
Names.csv' , encoding='latin-1')
```

Python Code

```
# Read the native vs non native bee data into a DataFrame for easy data manipulation
and analysis.
df4=pd.read_csv('/workspaces/myfolder/Pharmasug25/native_vs_nonnative_bumblebee_sight
ing_pollinators_of_farm_data_for_publication.csv' , encoding='latin-1')
```

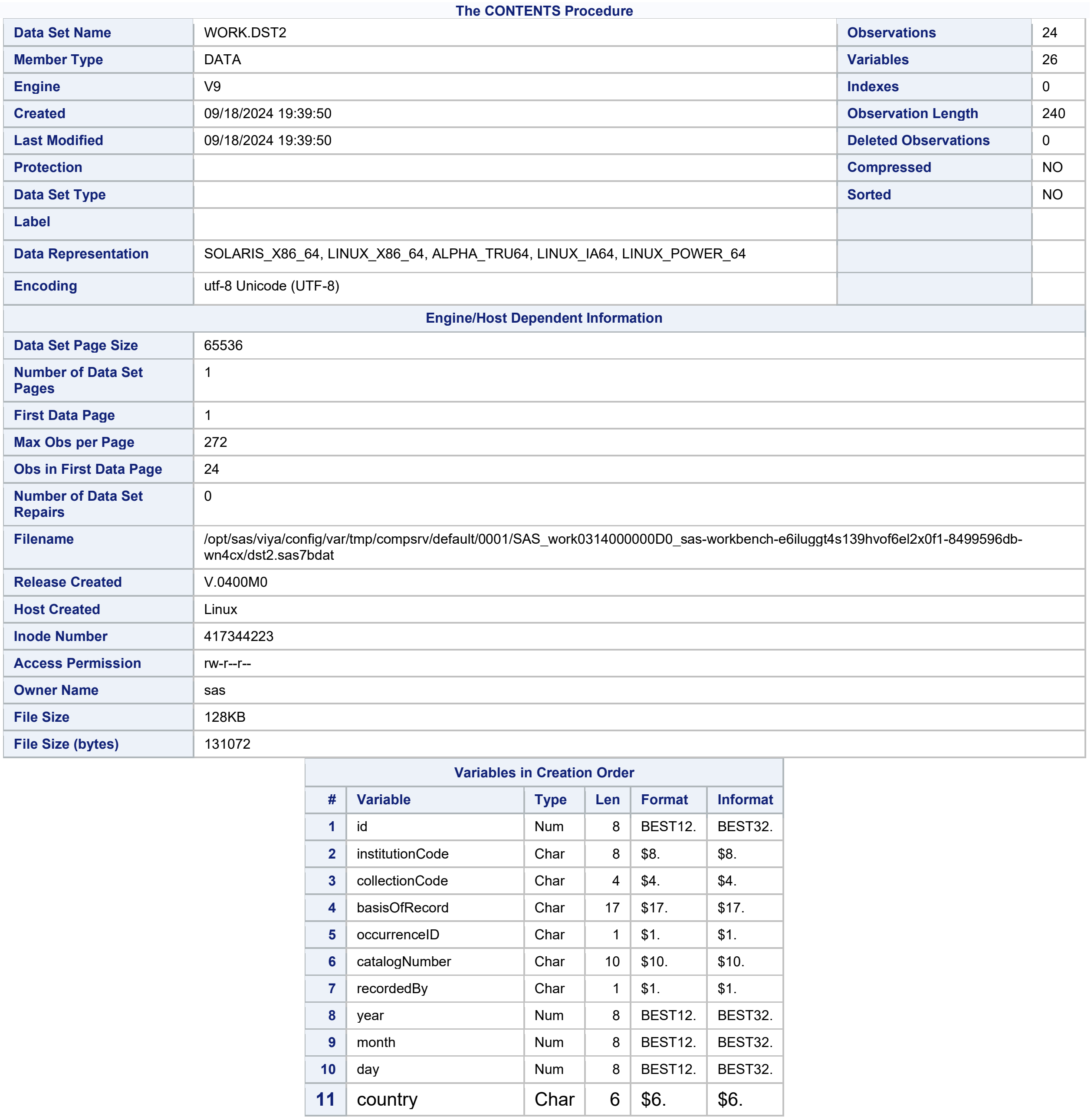
Data Exploration

SAS Data Exploration

Use PROC CONTENTS to display metadata about a SAS dataset, including variable names, types, and labels—like a bee collecting all the essential details from flowers!

```
proc contents data=dst2 varnum;
run;
```

SAS Results



The CONTENTS Procedure

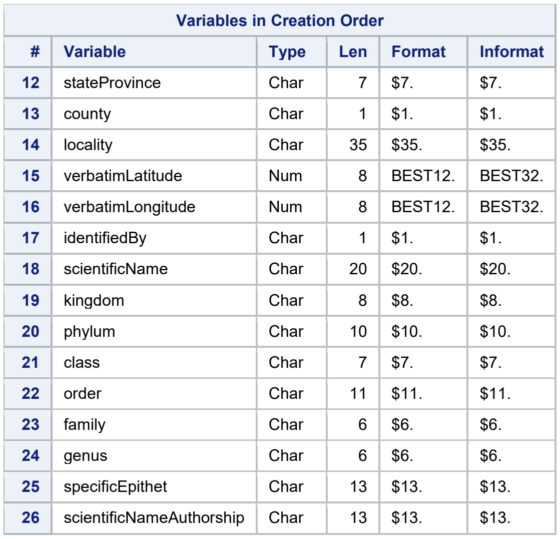
Data Set Name	WORK.DST2	Observations	24
Member Type	DATA	Variables	26
Engine	V9	Indexes	0
Created	09/18/2024 19:39:50	Observation Length	240
Last Modified	09/18/2024 19:39:50	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	SOLARIS_X86_64, LINUX_X86_64, ALPHA_TRU64, LINUX_IA64, LINUX_POWER_64		
Encoding	utf-8 Unicode (UTF-8)		

Engine/Host Dependent Information

Data Set Page Size	65536
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	272
Obs in First Data Page	24
Number of Data Set Repairs	0
Filename	/opt/sas/viya/config/var/tmp/compsrv/default/0001/SAS_work0314000000D0_sas-workbench-e6iluggt4s139hvof6el2x0f1-8499596db-wn4cx/dst2.sas7bdat
Release Created	V.0400M0
Host Created	Linux
Inode Number	417344223
Access Permission	rw-r--r--
Owner Name	sas
File Size	128KB
File Size (bytes)	131072

Variables in Creation Order

#	Variable	Type	Len	Format	Informat
1	id	Num	8	BEST12.	BEST32.
2	institutionCode	Char	8	$8.	$8.
3	collectionCode	Char	4	$4.	$4.
4	basisOfRecord	Char	17	$17.	$17.
5	occurrenceID	Char	1	$1.	$1.
6	catalogNumber	Char	10	$10.	$10.
7	recordedBy	Char	1	$1.	$1.
8	year	Num	8	BEST12.	BEST32.
9	month	Num	8	BEST12.	BEST32.
10	day	Num	8	BEST12.	BEST32.
11	country	Char	6	$6.	$6.



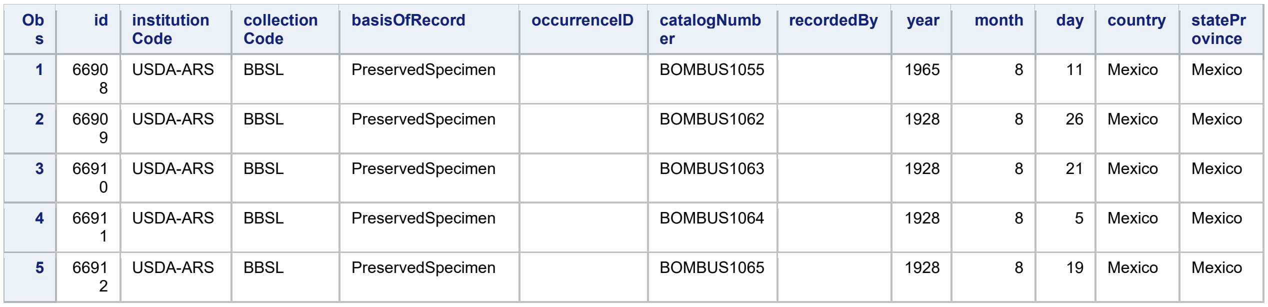
Variables in Creation Order					
#	Variable	Type	Len	Format	Informat
12	stateProvince	Char	7	$7.	$7.
13	county	Char	1	$1.	$1.
14	locality	Char	35	$35.	$35.
15	verbatimLatitude	Num	8	BEST12.	BEST32.
16	verbatimLongitude	Num	8	BEST12.	BEST32.
17	identifiedBy	Char	1	$1.	$1.
18	scientificName	Char	20	$20.	$20.
19	kingdom	Char	8	$8.	$8.
20	phylum	Char	10	$10.	$10.
21	class	Char	7	$7.	$7.
22	order	Char	11	$11.	$11.
23	family	Char	6	$6.	$6.
24	genus	Char	6	$6.	$6.
25	specificEpithet	Char	13	$13.	$13.
26	scientificNameAuthorship	Char	13	$13.	$13.

Dive into the data hive by previewing the first 5 rows of the dataset with PROC PRINT.

SAS Code

```
proc print data=dst2(obs=5);
run;
```

Partial SAS Results



Obs	id	institutionCode	collectionCode	basisOfRecord	occurrenceID	catalogNumber	recordedBy	year	month	day	country	stateProvince
1	66908	USDA-ARS	BBSL	PreservedSpecimen		BOMBUS1055		1965	8	11	Mexico	Mexico
2	66909	USDA-ARS	BBSL	PreservedSpecimen		BOMBUS1062		1928	8	26	Mexico	Mexico
3	66910	USDA-ARS	BBSL	PreservedSpecimen		BOMBUS1063		1928	8	21	Mexico	Mexico
4	66911	USDA-ARS	BBSL	PreservedSpecimen		BOMBUS1064		1928	8	5	Mexico	Mexico
5	66912	USDA-ARS	BBSL	PreservedSpecimen		BOMBUS1065		1928	8	19	Mexico	Mexico

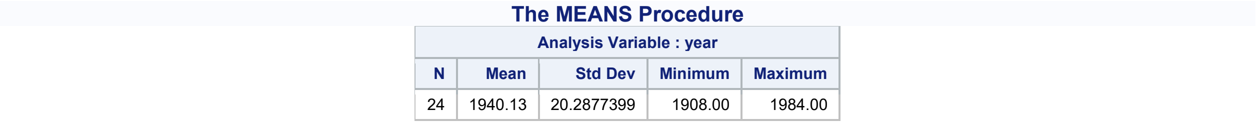
Summary Statistics

Buzz through the data with PROC MEANS to calculate summary statistics—mean, median, and standard deviation—just like a pollinator gathering the best data from every flower!

SAS Code

```
proc means data=dst2;
   var year;
run;
```

SAS Results



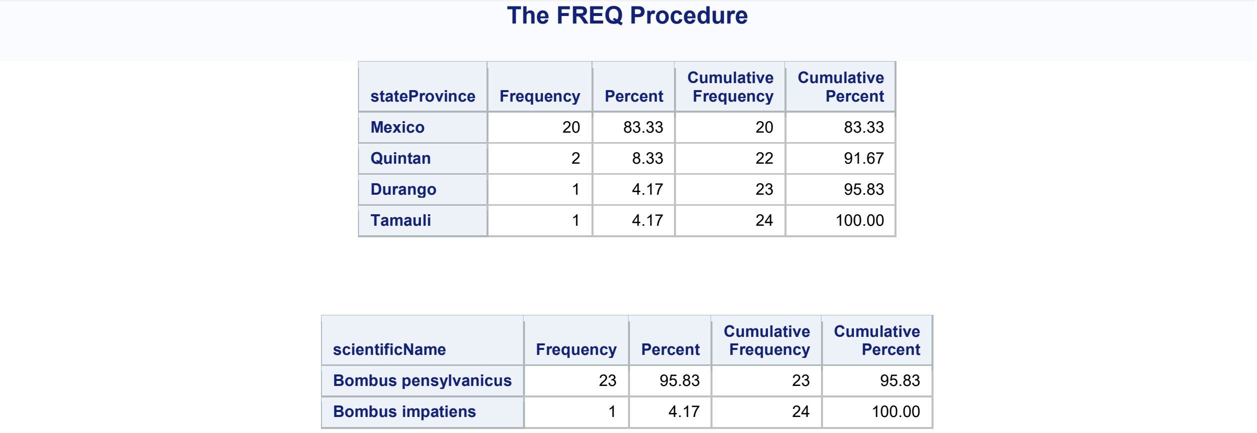
The MEANS Procedure

Analysis Variable : year				
N	Mean	Std Dev	Minimum	Maximum
24	1940.13	20.2877399	1908.00	1984.00

Tally up the hive activity by generating frequency counts for the dataset dst2, ordered by frequency with PROC FREQ!

SAS Code

```
proc freq data=dst2 order=freq;
run;
```



The FREQ Procedure

stateProvince	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Mexico	20	83.33	20	83.33
Quintan	2	8.33	22	91.67
Durango	1	4.17	23	95.83
Tamauli	1	4.17	24	100.00

scientificName	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Bombus pensylvanicus	23	95.83	23	95.83
Bombus impatiens	1	4.17	24	100.00

SQL Data Exploration

Programming in SAS is largely procedural with a step-by-step data flow, PROC SQL offers a declarative, SQL-based approach for working with structured data, while Python is fully object-oriented, enabling greater flexibility, modularity, and integration with diverse libraries for data manipulation and analysis.

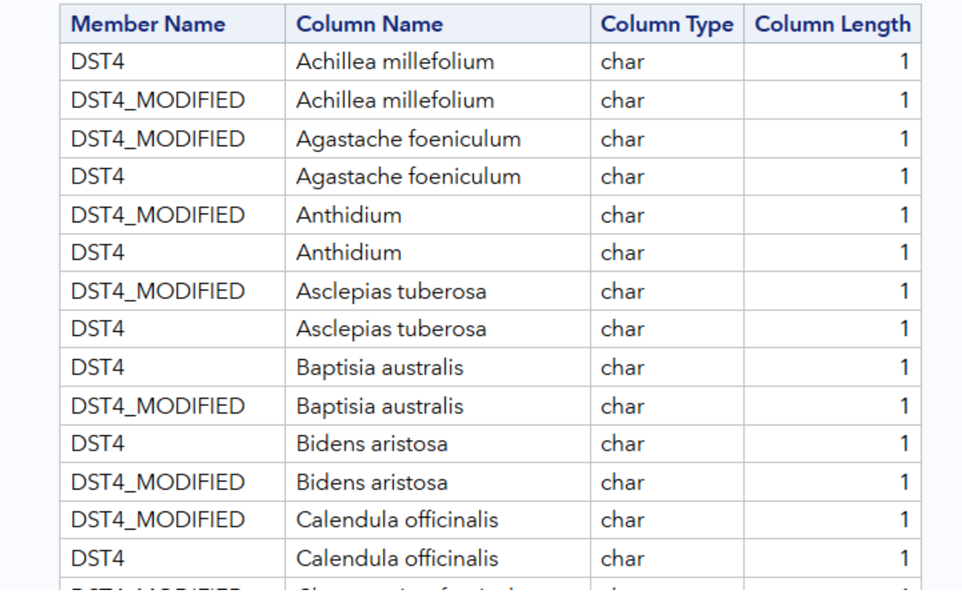
Obtain metadata info by mimicking PROC CONTENTS using PROC SQL

```
proc sql;
    describe table dictionary.columns;
```

```
proc sql;
    select memname, name, type, length from dictionary.columns
    where libname="WORK" and upcase(memname)="DST2"
```

The powerful ability of SQL to explore Metadata is examined in the code below. Locate common columns in all tables by sweeping the WORK library

```
proc sql;
    select memname, name, type, length
    from dictionary.columns
    where libname="WORK"
    group by name
    having count(name) > 1
    order by 2;
quit;
```



Member Name	Column Name	Column Type	Column Length
DST4	Achillea millefolium	char	1
DST4_MODIFIED	Achillea millefolium	char	1
DST4_MODIFIED	Agastache foeniculum	char	1
DST4	Agastache foeniculum	char	1
DST4_MODIFIED	Anthidium	char	1
DST4	Anthidium	char	1
DST4_MODIFIED	Asclepias tuberosa	char	1
DST4	Asclepias tuberosa	char	1
DST4	Baptisia australis	char	1
DST4_MODIFIED	Baptisia australis	char	1
DST4	Bidens aristosa	char	1
DST4_MODIFIED	Bidens aristosa	char	1
DST4_MODIFIED	Calendula officinalis	char	1
DST4	Calendula officinalis	char	1

PROC SQL to print

```
proc sql outobs=5;
    select *
    from dst2;
quit;
```

Breakdown : The OUTOBS= option restricts the number of rows that PROC SQL displays or writes to a table. For example, if you specify OUTOBS=10 and insert values into a table by using a query, then PROC SQL inserts a maximum of 10 rows into the resulting table. OUTOBS= is similar to the SAS data set option OBS=.

In PROC SQL, mean, min, max, std, and count replicate the default statistics from PROC MEANS.

```
title "Analysis Variable : year";
proc sql;
    select
        count(year) as N,
        mean(year) as Mean,
         std(year) as StdDev,
         min(year) as Minimum,
        max(year) as Maximum
    from dst2;
quit;
```

Getting PROC SQL To do PROC FREQ work

```
/*Step 1: Get frequency and percent using PROC SQL*/
proc sql;
    create table freq_state as
    select
        stateProvince,
        count(*) as Frequency,
        calculated Frequency / total_count * 100 as Percent format=6.2
    from
        (select * from dst2),
        (select count(*) as total_count from dst2)
    group by stateProvince
    order by Frequency desc;
quit;
```

Breakdown: This PROC SQL code creates a summary table showing the frequency and percent of each stateProvince value in dst2, by grouping the data and dividing each count by the total number of rows to mimic PROC FREQ output.

```
/*Step 2: Add cumulative frequency and percent using a DATA step*/
data freq_state_final;
    set freq_state;
    retain CumFreq CumPercent 0;
    CumFreq + Frequency;
    CumPercent + Percent;
run;
```

Breakdown : The above DATA step adds cumulative frequency and cumulative percent to each row in freq_state by retaining running totals across observations.

```
/*Step 3: Print the resulting dataset*/
proc sql;
    select * from freq_state_final;
quit;
```

Python Data Exploration

Python Code

```
import pandas as pd
```

Metadata time to see what’s buzzing under the surface!" 🐝

Python Code

```
#metadata
df2.info()
```

Python Console

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[21], line 2
      1 #metadata
----> 2 df2.info()


NameError: name 'df2' is not defined
```

Q. Why do I have to import the same data for each notebook in Workbench? Why can't I just import it in one notebook and use it across others?

A. In SAS Viya Workbench (and most notebook environments like Jupyter), each notebook runs in its own separate memory space (called a kernel).

- When you import data into one notebook, it lives only inside that notebook’s memory.
- Other notebooks can’t see or share that memory unless you explicitly save the data somewhere they can both access — like saving it to a file (CSV, SAS7BDAT, etc.) or a shared database.

Think of it like this:
🐝 Each notebook is like its own private hive — it doesn't know what's buzzing in the next hive unless you share the honey (data) in a common place.

Python Code

```
import pandas as pd

df1=pd.read_csv('/workspaces/myfolder/Pharmasug25/pattern_decline_N_American_Bumblebees.csv', dtype={6: str, 16: str}, encoding='latin-1')
df2=pd.read_csv('/workspaces/myfolder/Pharmasug25/pattern_decline_Mexican_Bumblebees.csv' , encoding='latin-1')
df3=pd.read_csv('/workspaces/myfolder/Pharmasug25/Bumblebee_Others_Scientific_Common_Names.csv' , encoding='latin-1')
df4=pd.read_csv('/workspaces/myfolder/Pharmasug25/native_vs_nonnative_bumblebee_sighting_pollinators_of_farm_data_for_publication.csv' , encoding='latin-1')
```

Q. 🐝 Why doesn’t Python leave as many 'honey trails' of progress like SAS does?

A. Python stays quiet unless you ask it to speak (with print(), logging, or verbose settings), while SAS automatically logs every step to meet strict audit needs in industries like healthcare and finance. If you want more buzz in Python, you can add manual print()s, use the logging library, or turn on verbose options!

🐝 ✨ Let’s create a tiny SAS-style log in Python to show you how it can feel “chattier” during program execution.

```
import pandas as pd
import logging

# Set up a basic logger
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s -
%(message)s')

logging.info('Starting program execution...')

# Step 1: Read data
try:
    df =
pd.read_csv('/workspaces/myfolder/Pharmasug25/pattern_decline_N_American_Bumblebees.c
sv', encoding='latin-1')
    logging.info('CSV file successfully read into a DataFrame.')
except Exception as e:
    logging.error(f'Error reading CSV file: {e}')

# Step 2: Check basic information
logging.info('Displaying dataset structure:')
print(df.info())

# Step 3: Calculate basic statistics
summary = df.describe()
logging.info('Calculated summary statistics.')

# Step 4: Preview data
logging.info('Here are the first 5 rows of the dataset:')
print(df.head())

logging.info('Program execution completed successfully.')
```

Python Console

```
2025-04-29 01:53:49,659 - INFO - Starting program execution...
/tmp/ipykernel_2434/540410752.py:11: DtypeWarning: Columns (6,16) have mixed types. Specify
dtype option on import or set low_memory=False.
  df =
pd.read_csv('/workspaces/myfolder/Pharmasug25/pattern_decline_N_American_Bumblebees.csv',
encoding='latin-1')
2025-04-29 01:53:49,913 - INFO - CSV file successfully read into a DataFrame.
2025-04-29 01:53:49,914 - INFO - Displaying dataset structure:
2025-04-29 01:53:50,011 - INFO - Calculated summary statistics.
2025-04-29 01:53:50,011 - INFO - Here are the first 5 rows of the dataset:
2025-04-29 01:53:50,026 - INFO - Program execution completed successfully.
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 66907 entries, 0 to 66906
Data columns (total 26 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   id                        66907 non-null  int64 
 1   institutionCode           66907 non-null  object
 2   collectionCode             66907 non-null  object
 3   basisOfRecord              66907 non-null  object
 4   occurrenceID               66907 non-null  int64
 5   catalogNumber              66907 non-null  object
 6   recordedBy                 25350 non-null  object
 7   year                       65778 non-null  float64
 8   month                      66368 non-null  float64
 9   day                        63897 non-null  float64
 10  country                    66818 non-null  object
 11  stateProvince              66818 non-null  object
 12  county                     59648 non-null  object
 13  locality                   62342 non-null  object
 14  verbatimLatitude           65980 non-null  float64
 15  verbatimLongitude          65980 non-null  float64
 16  identifiedBy               25309 non-null  object
 17  scientificName             66907 non-null  object
 18  kingdom                    66907 non-null  object
 19  phylum                     66907 non-null  object
...
3  Hymenoptera  Apidae  Bombus    occidentalis              Greene 1858
4  Hymenoptera  Apidae  Bombus        bifarius             Cresson 1878

[5 rows x 26 columns]
```

What this does:

logging.info() shows friendly notes as you move through each step (just like SAS NOTES).

If something goes wrong, logging.error() prints an error (just like SAS ERRORS).

It timestamps each message automatically!

Python Code

```
# Now that we have re-read the DF2 dataframe, we can look at the metadata
df2.info()
```

Python Console

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 24 entries, 0 to 23
Data columns (total 26 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   id                        24 non-null     int64
 1   institutionCode           24 non-null     object
 2   collectionCode            24 non-null     object
 3   basisOfRecord             24 non-null     object
 4   occurrenceID              0 non-null      float64
 5   catalogNumber             24 non-null     object
 6   recordedBy                0 non-null      float64
 7   year                      24 non-null     int64
 8   month                     24 non-null     int64
 9   day                       24 non-null     int64
 10  country                   24 non-null     object
 11  stateProvince             24 non-null     object
 12  county                    0 non-null      float64
 13  locality                  23 non-null     object
 14  verbatimLatitude          23 non-null     float64
 15  verbatimLongitude         23 non-null     float64
 16  identifiedBy              0 non-null      float64
 17  scientificName            24 non-null     object
 18  kingdom                   24 non-null     object
 19  phylum                    24 non-null     object
...
 24  specificEpithet           24 non-null     object
 25  scientificNameAuthorship  24 non-null     object
dtypes: float64(6), int64(4), object(16)

memory usage: 5.0+ KB
```

The method info() provides technical information about a DataFrame, so let’s view the output in more detail:

df2 is a DataFrame. There are 24 entries, i.e. 24 rows. Each row has a row label (aka the index) with values ranging from 0 to 0 to 23.
The table has 25 columns. Most columns have a value for each of the rows (all values are non-null).
There are some columns with textual data (strings, aka object). The other columns are numerical data with some of them whole numbers (aka integer) and others are real numbers (aka float).
The kind of data (characters, integers,…) in the different columns are summarized by listing the dtypes.
The approximate amount of RAM used to hold the DataFrame is provided as well.

Python Code

```
#read the first 5 rows of df1 using the head method just like a PROC PRINT
df2.head()
```

Partial Python Console

	id	institutionCode	collectionCode	basisOfRecord	catalogNumber	year	month	scientificName
0	66908	USDA-ARS	BBSL	PreservedSpecimen	BOMBUS1055	1965	8	Bombus pensylvanicus
1	66909	USDA-ARS	BBSL	PreservedSpecimen	BOMBUS1062	1928	8	Bombus pensylvanicus
2	66910	USDA-ARS	BBSL	PreservedSpecimen	BOMBUS1063	1928	8	Bombus pensylvanicus
3	66911	USDA-ARS	BBSL	PreservedSpecimen	BOMBUS1064	1928	8	Bombus pensylvanicus
4	66912	USDA-ARS	BBSL	PreservedSpecimen	BOMBUS1065	1928	8	Bombus pensylvanicus

Understanding hive activity by generating some descriptive statistics for one column, year-just like PROC MEANS

Python Code

```
df2['year'].describe()
```

Python Console

```
count      24.00000
mean     1940.12500
std        20.28774
min      1908.00000
25%      1928.00000
```

```
50%      1928.00000
75%      1962.00000
max      1984.00000
Name: year, dtype: float64
```

Learn where the bumblebees like to buzz around the most by getting frequency counts, similar to PROC FREQ

Python Code

```
stateProvince_freq = df2['stateProvince'].value_counts()
print(stateProvince_freq)
scientificName_freq = df2['scientificName'].value_counts()
print(scientificName_freq)
```

Python Console

```
stateProvince
Mexico          20
Quintana Roo     2
Durango          1
Tamaulipas       1
Name: count, dtype: int64
scientificName
Bombus pensylvanicus    23
Bombus impatiens         1

Name: count, dtype: int64
```

Data Preparation

SAS Data Prepare

Concatenate North American(exclude Alaska) and Mexican Bumblebee data

Know thy data using PROC CONTENTS. Before concatenating tables, first ensure that both datasets have the same structure (i.e., same variable names and types).

SAS Code

```
proc contents data=dst1;
run;
proc contents data=dst2;
run;
```

Merge your buzzing data colonies! 🐝
Use the SET statement to bring together the North American and Mexican bumblebee datasets into one vibrant hive of pollinator insights. 🌎 🍯

SAS Code

```
data dsconc;
set dst1 dst2;
run;
```

SAS Log

```
516  /** LOG_START_INDICATOR **/
517  title;footnote;ods _all_ close;
518  ods graphics on;
519  ods html5(id=vscode) style=HTMLEncore options(bitmap_mode='inline'
svg_mode='inline');
NOTE: Writing HTML5(VSCODE) Body file: sashtml6.htm
520  %let _SASPROGRAMFILE =
%nrquote(%nrstr(/workspaces/myfolder/Pharmasug25/3SASPrepare.sasnb));
521  data dsconc;
522  set dst1 dst2;
ERROR: Variable occurrenceID has been defined as both character and numeric.
523  run;
NOTE: The SAS System stopped processing this step because of errors.
NOTE: Due to ERROR(s) above, SAS set option OBS=0, enabling syntax check mode.
      This prevents execution of subsequent data modification statements.
WARNING: The data set WORK.DSCONC may be incomplete.  When this step was stopped
there were 0 observations and 26 variables.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds

524  ;*';*";*/;run;quit;ods html5(id=vscode) close;
```

Why did the concatenation fail?

During the Access phase, we used PROC IMPORT to convert CSVs into SAS datasets. By default, PROC IMPORT "guesses" the data structure by examining the first 20 rows to determine variable types and lengths. It assigns the most prevalent data type (numeric or character) to each column. If most of the first 20 rows are missing, SAS defaults to the character data type, and any subsequent numeric values are set to missing. This is why the column occurenceID defined as Character in the Mexican bumblebee data conflicts with the numeric OccurenceId in the North American Bumblebee dataset.   The DATA step, however, offers more control, granularity, and precision for importing data.

SAS Code

```
data dst1;
infile
"/workspaces/myfolder/SASPythonDataScientists/pattern_decline_N_American_Bumblebees.c
sv"  dsd firstobs=2;
input
id
institutionCode : $8.
collectionCode : $4.
basisOfRecord : $17.
occurrenceID :$9.
catalognumber: $12.
recordedBy $
year
month
day
country :$6.
stateProvince  : $16.
county : $17.
locality  : $37.
verbatimLatitude
verbatimLongitude
identifiedBy : $18.
scientificName  : $20.
kingdom  : $8.
phylum  : $10.
class : $7.
order  : $11.
family  : $6.
genus :  $6.
specificEpithet : $13.
scientificNameAuthorship : $13.
;

run;
```

```
data dst2;
infile
"/workspaces/myfolder/SASPythonDataScientists/pattern_decline_Mexican_Bumblebees.csv"
 dsd firstobs=2;
input
id
institutionCode : $8.
collectionCode : $4.
basisOfRecord : $17.
occurrenceID :$9.
catalognumber: $12.
recordedBy $
year
month
day
country :$6.
stateProvince  : $16.
county : $17.
locality  : $37.
verbatimLatitude
verbatimLongitude
identifiedBy : $18.
scientificName  : $20.
kingdom  : $8.
phylum  : $10.
class : $7.
order  : $11.
family  : $6.
genus :  $6.
specificEpithet : $13.
scientificNameAuthorship : $13.
;
run;
```

🐝 Before diving into the nectar of analysis, let’s peek inside each hive's blueprint! Here's where we inspect the metadata of our two bee tables — and keep a close eye on the column OccurrenceId, which might be causing a buzz due to mismatched types. 🍯 🔍

SAS Code

```
proc contents data=dst1 varnum;
run;
proc contents data=dst2 varnum;
run;
```

🐝 *Time to cross-pollinate our data!* Scientific names like *Bombus pensylvanicus* may wow the entomologists, but for the rest of us, a friendly common name makes the buzz more relatable. Let's merge our grand North American + Mexican bumblebee dataset(dsconc) with a lookup table of common names(ds3) — so every bee gets a name we can all appreciate. 🌼 🔗

SAS Code

```
data dsconc;
set dst1(where=(country <> 'Alaska')) dst2;
run;
```

```
1049  /** LOG_START_INDICATOR **/
1050  title;footnote;ods _all_ close;
1051  ods graphics on;
1052  ods html5(id=vscode) style=HTMLEncore options(bitmap_mode='inline'
svg_mode='inline');
NOTE: Writing HTML5(VSCODE) Body file: sashtml23.htm
1053  %let _SASPROGRAMFILE =
%nrquote(%nrstr(/workspaces/myfolder/Pharmasug25/3SASPrepare.sasnb));
1054  data dsconc;
1055  set dst1(where=(country <> 'Alaska')) dst2;
NOTE: The "<>" operator is interpreted as "not equals".
1056  run;
NOTE: There were 66907 observations read from the data set WORK.DST1.
      WHERE country not = 'Alaska';
NOTE: There were 24 observations read from the data set WORK.DST2.
NOTE: The data set WORK.DSCONC has 66931 observations and 26 variables.
NOTE: DATA statement used (Total process time):
      real time           0.02 seconds
      cpu time            0.03 seconds

1057  ;*';*";*/;run;quit;ods html5(id=vscode) close;
```

SAS Code

```
proc contents data=dsconc;
run;

proc contents data=dst3;
run;
```

PROC Contents reveals that the common variable is ScientificName

SAS Code

```
data dsmerge;
    merge dsconc dst3;
    by scientificname;
run;
```

SAS Log

```
1058  /** LOG_START_INDICATOR **/
1059  title;footnote;ods _all_ close;
1060  ods graphics on;
1061  ods html5(id=vscode) style=HTMLEncore options(bitmap_mode='inline'
svg_mode='inline');
NOTE: Writing HTML5(VSCODE) Body file: sashtml24.htm
1062  %let _SASPROGRAMFILE =
%nrquote(%nrstr(/workspaces/myfolder/Pharmasug25/3SASPrepare.sasnb));
1063  data dsmerge;
1064      merge dsconc dst3;
1065      by scientificname;
1066  run;
29                                                          The SAS System
Tuesday, April 29, 2025 01:55:00 PM
WARNING: Multiple lengths were specified for the BY variable scientificName by input
data sets. This might cause unexpected results.
WARNING: Multiple lengths were specified for the variable specificEpithet by input
data set(s). This can cause truncation of data.
ERROR: BY variables are not properly sorted on data set WORK.DSCONC.
id=2 institutionCode=USDA-ARS collectionCode=BBSL basisOfRecord=PreservedSpecimen
occurrenceID=699384988 catalognumber=BBSL241571
recordedBy=W. Apper year=1970 month=7 day=27 country=USA stateProvince=Arizona
county=Apache locality=Paradise Creek
verbatimLatitude=34.0328 verbatimLongitude=-109.7142 identifiedBy=
scientificName=Bombus occidentalis kingdom=Animalia
phylum=Arthropoda class=Insecta order=Hymenoptera family=Apidae genus=Bombus
specificEpithet=occidentalis
scientificNameAuthorship=Greene 1858 Species=Bombus CommonName=Western Bumblebee
Description=Found in western U.S.; enjoys wildflowers and garden plants; active
during the day; family Apidae. Source=IUCN Red List
FIRST.scientificName=1 LAST.scientificName=0 _ERROR_=1 _N_=65
NOTE: The SAS System stopped processing this step because of errors.
NOTE: There were 3 observations read from the data set WORK.DSCONC.
NOTE: There were 65 observations read from the data set WORK.DST3.
WARNING: The data set WORK.DSMERGE may be incomplete.  When this step was stopped
there were 64 observations and 30 variables.
WARNING: Data set WORK.DSMERGE was not replaced because this step was stopped.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds

1067  ;*';*";*/;run;quit;ods html5(id=vscode) close;
```

Prep the datasets for merge by running a PROC SORT to ensure both datasets are ordered properly by scientificName.

SAS Code

```
proc sort data=dsconc;
    by scientificname;
run;
proc sort data=dst3;
```

```
by scientificname;
run;
```

SAS Log

```
1068  /** LOG_START_INDICATOR **/
1069  title;footnote;ods _all_ close;
1070  ods graphics on;
1071  ods html5(id=vscode) style=HTMLEncore options(bitmap_mode='inline'
svg_mode='inline');
NOTE: Writing HTML5(VSCODE) Body file: sashtml25.htm
1072  %let _SASPROGRAMFILE =
%nrquote(%nrstr(/workspaces/myfolder/Pharmasug25/3SASPrepare.sasnb));
1073  proc sort data=dsconc;
1074      by scientificname;
1075  run;
NOTE: There were 66931 observations read from the data set WORK.DSCONC.
NOTE: The data set WORK.DSCONC has 66931 observations and 26 variables.
NOTE: PROCEDURE SORT used (Total process time):
      real time           0.04 seconds
      cpu time            0.04 seconds

1076  proc sort data=dst3;
1077  by scientificname;
1078  run;
NOTE: Input data set is already sorted, no sorting done.
NOTE: PROCEDURE SORT used (Total process time):
      real time           0.00 seconds
      cpu time            0.01 seconds

1079  ;*';*";*/;run;quit;ods html5(id=vscode) close;
```

🐝 Bringing bees together! This code merges two hives—dsconc and dst3—by scientificname, keeping only the bees (rows) found in both colonies. The in= flags help check who's buzzing in from where! 🐝 🔗

SAS Code

```
data dsmerge;
    merge dsconc(in=inc) dst3(in=ind);
    by scientificname;
    if inc and ind;
run;
```

SQL Data Preparation

Concatenation

```
proc sql;
    create table dsconc as
    select * from dst1 where country <> 'Alaska'
    union corr
    select * from dst2;
quit;
```

What is the difference between Data step & SQL during concatenation?

Joining

```
proc sql;
    create table dsmerge as
    select *
    from dsconc as a
    inner join dst3 as b
    on a.scientificName = b.scientificName;
quit;
```

The merge ... by ... if inc and ind; logic keeps only matching records from both datasets, which is exactly what an INNER JOIN does.

Aliases a and b allow you to reference columns uniquely if needed.

Python Data Preparation

🐝 Time to merge the buzz! We're joining bee data with scientific names to build one vibrant hive of insights—revealing where the buzz is and who's doing the pollinating. ✿ 📊

First read the csv into pandas dataframes- Review-A Pandas DataFrame is like a spreadsheet in Python—it's a two-dimensional table where you can store and work with data using rows and columns, just like you would in Excel or a SAS dataset.

Python Code

```
import pandas as pd

df1=pd.read_csv('/workspaces/myfolder/Pharmasug25/pattern_decline_N_American_Bumblebees.csv', dtype={6: str, 16: str}, encoding='latin-1')
df2=pd.read_csv('/workspaces/myfolder/Pharmasug25/pattern_decline_Mexican_Bumblebees.csv' , encoding='latin-1')
```

```
df3=pd.read_csv('/workspaces/myfolder/Pharmasug25/Bumblebee_Others_Scientific_Common_Names.csv' , encoding='latin-1')
df4=pd.read_csv('/workspaces/myfolder/Pharmasug25/native_vs_nonnative_bumblebee_sighting_pollinators_of_farm_data_for_publication.csv' , encoding='latin-1')
```

Before concatenating 2 data frames to combine North American(excluding Alaska) and Mexican Bumblebees, Take a quick look at the dimensions of the 2 dataframes we are about to concatenate.

Python Code

```
# North American bumblebee decline dataframe
df1.shape
```

Python Console

```
(66907, 26)
```

Python Code

```
# Mexican bumblebee decline dataframe
df2.shape
```

Python Console

```
(24, 26)
```

Concatenation is a way to stitch dataframes along an axis, either row axis or column axis. use concat() and pass it a list of DataFrames that you want to concatenate.

Python Code

```
dfconc=pd.concat([df1,df2])
dfconc.shape
```

Python Console

```
(66931, 26)
```

Merging data frames

Male bees sleeping. Credit: J. Beckham

We're buzzing into bumblebee data with Python by merging common names and nesting habits—giving each bee its name tag at the hive party! 🐝 This makes connecting Latin and everyday names a breeze for sweet, streamlined analysis. 🍯

The command list(df3) in Python, when using pandas, will return a list of the column names in the DataFrame df3. It’s a quick way to view the structure of the DataFrame and understand what variables (columns) it contains.

Python Code

```
list(df3)
```

Python Console

```
['ScientificName',
 'Species',
 'specificEpithet',
 'CommonName',
 'Description',
 'Source']
```

The command print(df3) in Python will display the entire contents of the DataFrame df3 in the console or output window. This allows you to see all the rows and columns of data contained in df3, providing a full view of the dataset.

Python Code

```
print(df3)
```

Python Console

```
       ScientificName      Species specificEpithet  \
0              Agapostemon  Agapostemon             NaN
1     Agapostemon sericeus  Agapostemon        sericeus
2    Agapostemon splendens  Agapostemon       splendens
3      Agapostemon texanus  Agapostemon         texanus
```

```
4    Agapostemon virescens  Agapostemon       virescens
..                     ...          ...             ...
157        Osmia bucephala        Osmia       bucephala
158       Osmia collinsiae        Osmia      collinsiae
159        Osmia distincta        Osmia       distincta
160         Osmia georgica        Osmia        georgica
161           Osmia pumila        Osmia          pumila

                      CommonName  \
0             Metallic Green Bee
1              Silky Agapostemon
2           Splendid Agapostemon
3              Texas Agapostemon
4    Bicolored Striped Sweat Bee
..                           ...
157       Large-headed Mason Bee
158           Collins' Mason Bee
159           Distinct Mason Bee
160            Georgia Mason Bee
161             Little Mason Bee
...
160  Found in southeastern U.S.; enjoys wildflowers...  Discover Life
161  Found in gardens and woodlands; enjoys small f...  Discover Life

[162 rows x 6 columns]
```

The command df3.describe() in Python is used to generate summary statistics for the numerical columns in the DataFrame df3

Python Code

```
df3.describe()
```

Python Console

	ScientificName	Species	specificEpithet	CommonName	Description	Source
count	162	162	156	162	161	161
unique	162	23	151	161	122	7
top	Osmia pumila	Bombus	texana	Modest Masked Bee	Found in western U.S.; enjoys wildflowers and ...	Discover Life
freq	1	55	2	2	6	65

Take a quick look at the dimensions of the tables we are about to merge

Python Code

```
dfconc.shape
```

Python Console

```
(66931, 26
```

Python Code

```
df3.shape
```

Python Console

```
(162, 6)
```

In the world of pandas, DataFrames have a merge() method, with similar functionality to SAS merges. No need to sort ahead of time—perform all kinds of different joins by simply using the how keyword. It's like a hive of possibilities for your data!

Python Code

```
inner_join = dfconc.merge(df3, on=["SCIENTIFICNAME"], how="inner")
```

Python Console

```
KeyError                                  Traceback (most recent call last)
/tmp/ipykernel_15474/3012165160.py in ?()
----> 1 inner_join = dfconc.merge(df3, on=["SCIENTIFICNAME"], how="inner")

/usr/local/lib64/python3.11/site-packages/pandas/core/frame.py in ?(self, right, how,
on, left_on, right_on, left_index, right_index, sort, suffixes, copy, indicator,
validate)
  10828         validate: MergeValidate | None = None,
  10829     ) -> DataFrame:
  10830         from pandas.core.reshape.merge import merge
  10831
> 10832         return merge(
  10833             self,
  10834             right,
  10835             how=how,

/usr/local/lib64/python3.11/site-packages/pandas/core/reshape/merge.py in ?(left,
right, how, on, left_on, right_on, left_index, right_index, sort, suffixes, copy,
indicator, validate)
    166             validate=validate,
    167             copy=copy,
    168         )
    169     else:
--> 170         op = _MergeOperation(
    171             left_df,
    172             right_df,
    173             how=how,
...
   1912
   1913         # Check for duplicates
   1914         if values.ndim > 1:
```

```
KeyError: 'SCIENTIFICNAME'
```

Dataframe column names are essentially string values, which are case sensitive in Python. Because of this, you will need to be careful whenever you utilize column names, such as when renaming a column, accessing columns or performing functions on them.

Python Code

```
dfconc.columns = dfconc.columns.str.lower()
```

Python Code

```
list(dfconc)
```

Python Console

```
['id',
 'institutioncode',
 'collectioncode',
 'basisofrecord',
 'occurrenceid',
 'catalognumber',
 'recordedby',
 'year',
 'month',
 'day',
 'country',
 'stateprovince',
 'county',
 'locality',
 'verbatimlatitude',
 'verbatimlongitude',
 'identifiedby',
 'scientificname',
 'kingdom',
 'phylum',
 'class',
 'order',
 'family',
 'genus',
 'specificepithet',
 'scientificnameauthorship']
```

Python Code

```
df3.columns = df3.columns.str.lower()
```

Python Code

```
list(df3)
```

Python Console

```
['scientificname',
 'species',
 'specificepithet',
 'commonname',
 'description',
 'source']
```

Use an inner join to merge dfconc and df3 on the scientificname column, keeping only the rows where there's a match in both tables—like inviting only the bees who appear on both guest lists! 🐝

Python Code

```
df_inner = dfconc.merge(df3, on=["scientificname"], how="inner")
```

The command df_inner.head() in Python (using pandas) shows the first 5 rows of the df_inner DataFrame by default.

🐝 Think of it as peeking at the top of the hive—just a quick glance to see what kind of data is buzzing inside! If you want to see more or fewer rows, you can pass a number like df_inner.head(10).

Python Code

```
df_inner.head
```

```
<bound method NDFrame.head of           id institutioncode collectioncode
basisofrecord  occurrenceid  \
0          1        USDA-ARS          BBSL  PreservedSpecimen   699384987.0
1          2        USDA-ARS          BBSL  PreservedSpecimen   699384988.0
2          3        USDA-ARS          BBSL  PreservedSpecimen   699384989.0
3          4        USDA-ARS          BBSL  PreservedSpecimen   699384990.0
4          5        USDA-ARS          BBSL  PreservedSpecimen   699384991.0
...      ...             ...           ...                ...           ...
66926  66927        USDA-ARS          BBSL  PreservedSpecimen           NaN
66927  66928        USDA-ARS          BBSL  PreservedSpecimen           NaN
66928  66929        USDA-ARS          BBSL  PreservedSpecimen           NaN
66929  66930        USDA-ARS          BBSL  PreservedSpecimen           NaN
66930  66931        USDA-ARS          BBSL  PreservedSpecimen           NaN

      catalognumber       recordedby    year  month   day  ...         order  \
0         BBSL221088      W. Apperson  1970.0    7.0  27.0  ...  Hymenoptera
1         BBSL241571      W. Apperson  1970.0    7.0  27.0  ...  Hymenoptera
2              76122        B. Hevron  1989.0    6.0  16.0  ...  Hymenoptera
3           JPS30053  P.S. Bartholomew  1970.0    9.0  15.0  ...  Hymenoptera
4         BBSL226571      W.J. Hanson  1961.0    8.0  15.0  ...  Hymenoptera
...              ...              ...     ...    ...   ...  ...          ...
66926     BOMBUS1219              NaN  1928.0    8.0  19.0  ...  Hymenoptera
66927     BOMBUS1348              NaN  1928.0    7.0  13.0  ...  Hymenoptera
66928    BOMBUS33485              NaN  1930.0    9.0  13.0  ...  Hymenoptera
66929    BOMBUS33755              NaN  1944.0    1.0   8.0  ...  Hymenoptera
66930    BOMBUS37213              NaN  1933.0    6.0  19.0  ...  Hymenoptera
...
66928  Bumblebees of North America
66929  Bumblebees of North America
66930  Bumblebees of North America

[66931 rows x 31 columns]>
```

Data Analysis

SAS Data Analysis

We need to find bumblebees with names ending in "ern," "ed," or a charming hyphen. Our Queen Bee, a Southern belle with a flair for magnolias, believes these names hint at the finest nectar. Can you be on the lookout for maybe a Buzz-ern, Dappled-ed, or Polka-dotted bee to keep her hive the envy of the meadows!

Perl regex (regular expressions) is a powerful tool for pattern matching and text manipulation, allowing complex searches, substitutions, and transformations within strings.

Perl in SAS is used for advanced string manipulation and regular expression tasks, often through the PRX functions, allowing for more complex text processing than traditional SAS methods.

SAS Code

```
/*locate certain bee populations by name pattern*/
/*regex -specificity, precision & density*/
/*match "ed" or "ern" followed by a space (or boundary) or a dash*/

proc print data=dst3(obs=100);
/*ed OR ern FOLLOWED BY a SPACE or -*/
/*OR*/
/* any value with a dash*/
where prxmatch('/((ed|ern)\b)|\-/i', commonname);
run;
```

Breakdown:

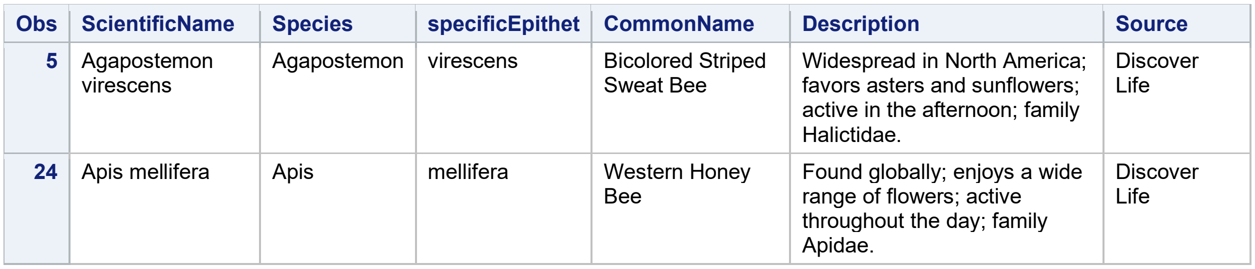
(ed|ern) matches either "ed" or "ern".

(\s|-) ensures that the "ed" or "ern" is followed by either a space (\s) or a hyphen (-).

|\- allows for matching any word that contains a hyphen, even if it doesn't end in "ed" or "ern".

With the /i at the end of the regex, it's case-insensitive, ensuring you capture both "ed" and "ED", "ern" and "ERN", etc.

SAS Results



Obs	ScientificName	Species	specificEpithet	CommonName	Description	Source
5	Agapostemon virescens	Agapostemon	virescens	Bicolored Striped Sweat Bee	Widespread in North America; favors asters and sunflowers; active in the afternoon; family Halictidae.	Discover Life
24	Apis mellifera	Apis	mellifera	Western Honey Bee	Found globally; enjoys a wide range of flowers; active throughout the day; family Apidae.	Discover Life

Obs	ScientificName	Species	specificEpithet	CommonName	Description	Source
29	Bombus affinis	Bombus	affinis	Rusty Patched Bumblebee	Found in the eastern U.S.; favors wildflowers and garden plants; active during the day; family Apidae.	IUCN Red List
30	Bombus appositus	Bombus	appositus	White-shouldered Bumblebee	Found in woodlands and gardens; enjoys clover and thistles; active during the day; family Apidae.	Bumblebees of
34	Bombus balteatus	Bombus	balteatus	Golden-belted Bumblebee	Found in western U.S.; prefers wildflowers and garden plants; active throughout the day; family Apidae.	Bumblebees of
35	Bombus bifarius	Bombus	bifarius	Two-form Bumblebee	Found in various North American habitats; enjoys flowering plants; active during the day; family Apidae.	Bumblebees of
36	Bombus bimaculatus	Bombus	bimaculatus	Two-spotted Bumblebee	Found in northeastern U.S.; enjoys clover and meadow flowers; active throughout the day; family Apidae.	Bumblebees of
38	Bombus borealis	Bombus	borealis	Northern Amber Bumblebee	Found in northern U.S. and Canada; enjoys wildflowers and garden plants; active during the day; family Apidae.	Bumblebees of
48	Bombus flavifrons	Bombus	flavifrons	Yellow-fronted Bumblebee	Found in U.S.; enjoys wildflowers and garden plants; active during the day; family Apidae.	Bumblebees of
50	Bombus fraternus	Bombus	fraternus	Southern Plains Bumblebee	Found in southern U.S.; enjoys a variety of wildflowers; active during the day; family Apidae.	Bumblebees of

SAS Code

```
/* Can the contains operator perform better? */
proc print data=dst3;
where commonname contains 'ed' or commonname contains 'ern' or commonname contains '-
';
run;
```

Breakdown:

This approach doesn't work because the CONTAINS operator matches substrings anywhere, without checking word boundaries or ensuring that "ed" or "ern" appear at the end of the word, leading to broad and imprecise matches (e.g., "Red-backed").

SAS Code

```
/* Certainly, the Like operator must perform better */
proc print data=dst3(obs=10);
where commonname like '%ed%'
   or commonname like '%ern%'
   or commonname contains '-';
run;
```

Breakdown:

The code doesn't work as expected because the LIKE and CONTAINS operators in SAS behave differently. LIKE matches substrings anywhere in the string (e.g., %ed% matches "ed" anywhere), while CONTAINS does the same but without checking specific positions. Combining them with OR leads to overly broad matches, such as any string containing a hyphen, which may not align with your pattern requirements.

Grouping Aggregating Data

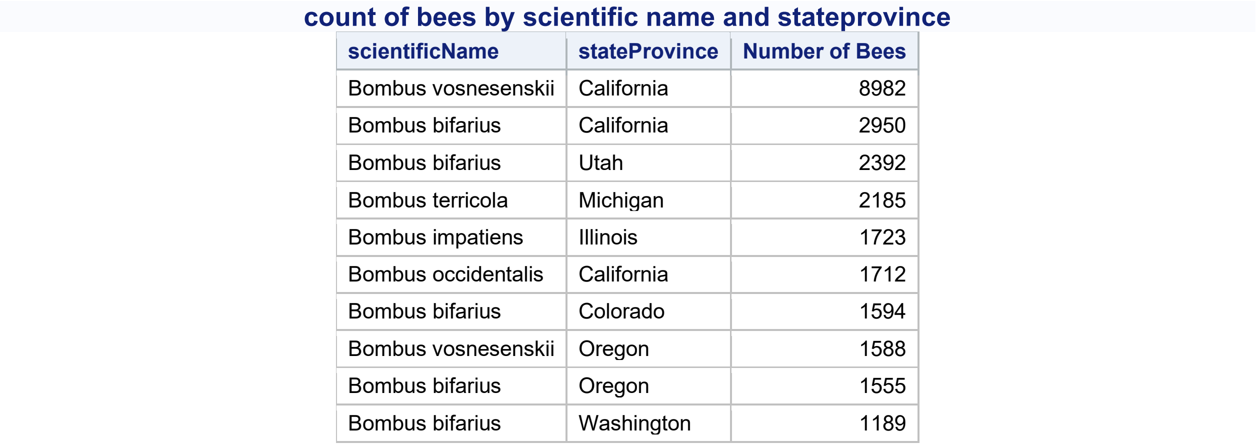
[4] *Georgia Mason Bee-a solitary bee. Credit: NPS*

Buzzing Around: Mapping Bumblebee Hotspots!
Let's track down where these fuzzy friends are hanging out the most. From hot & arid Arizona to the cool climes of Ontario, grab your data nets and let’s discover the ultimate bee hangouts

```
title "Count of Bees by Scientific Name and StateProvince";

/*Sort data by stateProvince and scientificName to prepare for grouped analysis*/
proc sort data=dst1 out=sorted;
    by stateProvince scientificName;
run;

/*Count # of observations for each unique stateProvince and scientificName pair*/
proc means data=sorted noprint nway;
    class stateProvince scientificName;
    output out=bee_counts (drop=_type_ _freq_)
        n=Count;
run;

/*Sort results by count (highest first), then by state and scientific name*/
proc sort data=bee_counts;
    by descending Count stateProvince scientificName;
```

```
run;

/*Print the final table with a custom label for the count column*/
proc print data=bee_counts label;
    label Count = "Number of Bees";
run;
```

SAS Partial Results



count of bees by scientific name and stateprovince

scientificName	stateProvince	Number of Bees
Bombus vosnesenskii	California	8982
Bombus bifarius	California	2950
Bombus bifarius	Utah	2392
Bombus terricola	Michigan	2185
Bombus impatiens	Illinois	1723
Bombus occidentalis	California	1712
Bombus bifarius	Colorado	1594
Bombus vosnesenskii	Oregon	1588
Bombus bifarius	Oregon	1555
Bombus bifarius	Washington	1189

SQL Data Analysis

```
proc sql outobs=10;
    select *
    from dst3
    where prxmatch('/(ed|ern)(\s|-)|\-/i', commonname);
quit;
```

Breakdown
prxmatch('/(ed|ern)(\s|-)|\-/i', commonname) applies the same regex filter.

Grouping Aggregate Data

```
title "Count of Bees by Scientific Name and StateProvince";
proc sql;
select  scientificname, stateprovince,  count(scientificname) as count 'Number of 
Bees'
from dst1
group by 2,  1
order by 3 desc,2, 1
;
```

Python Data Analysis

Queen Bee’s Pattern Parade:
👑🐝 The Queen Bee is on a mission to find the sweetest bee names with the best patterns! while SAS flexed its data-handling muscles to reveal hidden patterns, watch as Python weaves its web of regex wizardry. Which tool will uncover the juiciest insights or are they both equal? Let the name hunt begin!

To obey the queen, we will find all instances of bee names ending in 'ern' 'ed' or with a '-' using perl regular expression.

Python Code

```
import pandas as pd
import re

# Read the scientific and common name lookup csv file into a DataFrame
df3=pd.read_csv('/workspaces/myfolder/SASPythonDataScientists/Bumblebee_Others_Scientific_Common_Names.csv' , encoding='latin-1')
```

re is a standard library module (or "package") in Python that provides support for regular expressions—a powerful way to search, match, and manipulate strings based on patterns.

Since it's built into Python, you don't need to install it separately—just import re, and you're ready to start buzzing through text with regex! 🐝

Python Code

```
df3.info()
```

Python Console

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 162 entries, 0 to 161
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   ScientificName   162 non-null    object
 1   Species          162 non-null    object
 2   specificEpithet  156 non-null    object
 3   CommonName       162 non-null    object
 4   Description      161 non-null    object
 5   Source           161 non-null    object
dtypes: object(6)
memory usage: 7.7+ KB
```

DataFrames can be filtered in multiple ways; the most intuitive of which boolean indexing creating a series of True/False values

Python Code

```
#  regex pattern (case-insensitive and end-of-word)
```

```
pattern = r'(?i)\b(?:ed|ern)\b|-'  # (?i) = ignore case, \b = word boundary

# Apply the filter
df_regex = df3[df3['CommonName'].str.contains(pattern, regex=True)]

# Display the filtered DataFrame
print(df_regex)
```

Python Console

```
ScientificName       Species specificEpithet  \
29           Bombus appositus        Bombus       appositus
33           Bombus balteatus        Bombus       balteatus
34            Bombus bifarius        Bombus        bifarius
35         Bombus bimaculatus        Bombus     bimaculatus
47          Bombus flavifrons        Bombus      flavifrons
51        Bombus griseocollis        Bombus    griseocollis
57             Bombus lucorum        Bombus         lucorum
58         Bombus melanopygus        Bombus     melanopygus
59              Bombus mixtus        Bombus          mixtus
67         Bombus rotundiceps        Bombus     rotundiceps
68         Bombus rufocinctus        Bombus     rufocinctus
74           Bombus ternarius        Bombus       ternarius
75          Bombus terrestris        Bombus      terrestris
76           Bombus terricola        Bombus       terricola
77              Bombus vagans        Bombus          vagans
80        Bombus vosnesenskii        Bombus    vosnesenskii
88      Coelioxys octodentata     Coelioxys     octodentata
94        Halictus rubicundus      Halictus      rubicundus
96       Hoplitis pilosifrons      Hoplitis     pilosifrons
112   Lasioglossum fuscipenne  Lasioglossum      fuscipenne
116  Lasioglossum leucozonium  Lasioglossum     leucozonium
127          Megachile brevis     Megachile          brevis
128          Megachile exilis     Megachile          exilis
133       Megachile montivaga     Megachile       montivaga
...
142                                        BugGuide
147                                        BugGuide
156                                   Discover Life
157                                        BugGuide
```

Breakdown
this regex looks for:

Words ending in "ed" or "ern" followed by a non-word character (like a space, hyphen, or punctuation).

Or simply a hyphen (-) anywhere in the string.

(?i) makes the pattern case insensitive.

\b ensures "ed" or "ern" appear at word boundaries, i.e., the end of a word.

Python Code

```
df_regex.shape
```

```
Explanation:
df_regex.shape is a Pandas DataFrame attribute that returns a tuple representing the dimensions of the DataFrame – specifically, the number of rows and columns.

since df_regex.shape returns (33, 6), that means the filtered DataFrame df_regex has:

33 rows (bee records that matched the regex pattern), and

6 columns (like ScientificName, Species etc.).
```

[5] Green Metallic Sweat Bee - Unlike other sweat bees, they are not attracted to human sweat.

Grouping Aggregating Data

Python Code

```
import pandas as pd

# Read the North American bumblebee CSV file into a DataFrame for easy data manipulation and analysis, forcing column 6 and 16 to be strings
df1=pd.read_csv('/workspaces/myfolder/Pharmasug25/pattern_decline_N_American_Bumblebees.csv', dtype={6: str, 16: str}, encoding='latin-1')
```

Python Code

```
df1.groupby(['scientificName','stateProvince']).size().reset_index(name='count').sort_values(by='count', ascending=False).head(20)
```

This line of code performs a grouped count summary in Pandas and returns the top 20 combinations of scientific name and state/province by frequency, sorted in descending order.

🐝 In bee-speak: this is like tallying up how many times each bee species shows up in each state, ranking them from the most spotted to the least — the top 20 buzziest combos!

```
Explanation
df1.groupby(['scientificName','stateProvince'])
# Group the DataFrame by both scientific name and state/province

.size()
# Count # of rows (i.e., bee observations) in each group

.reset_index(name='count')
# Convert result to a DF, name the count column 'count'

.sort_values(by='count', ascending=False)
# Sort the counts from highest to lowest

.head(20)
# Show only the top 20 results
```

Partial Python Console

	scientificName	stateProvince	count
629	Bombus vosnesenskii	California	8982
85	Bombus bifarius	California	2950
94	Bombus bifarius	Utah	2392
557	Bombus terricola	Michigan	2185
306	Bombus impatiens	Illinois	1723
410	Bombus occidentalis	California	1712
86	Bombus bifarius	Colorado	1594
634	Bombus vosnesenskii	Oregon	1588
92	Bombus bifarius	Oregon	1555
95	Bombus bifarius	Washington	1189
436	Bombus pensylvanicus	Illinois	1097
466	Bombus pensylvanicus	Texas	1071

Data Reporting

SAS Data Reporting

🐝 To compare flowering periods of native vs non native plants, Let's first get the data ready. clean up the garden log by tossing empty flower labels, tag plants by year, and jot down what month blooms happen to help the bees!

SAS Code

```
data dst4_modified;
  set dst4;
  where 'plant species'n ne ' ';
  PlantSpecies_Year=catx('-','plant species'n,year);
  Month=month(date);
run;
```

SAS Log

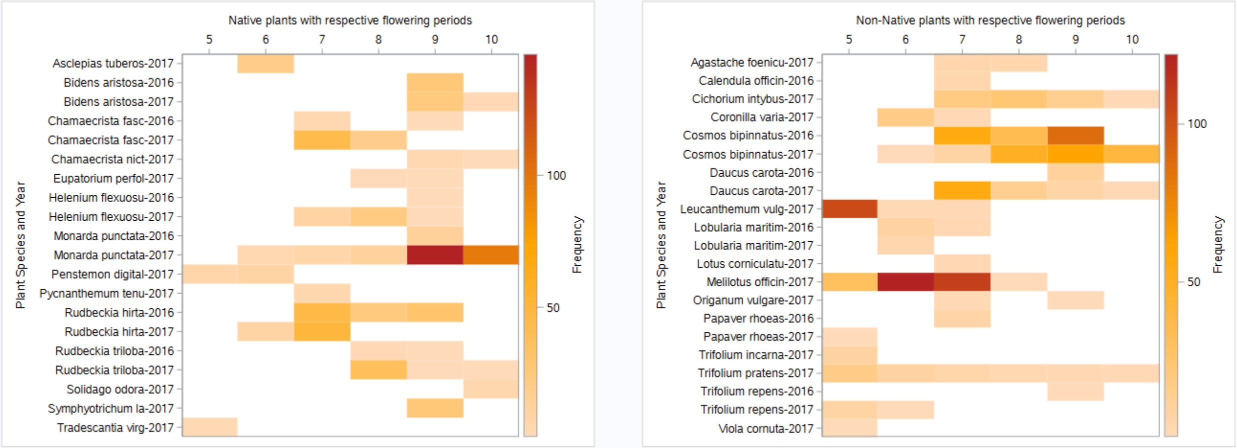
```
508  /** LOG_START_INDICATOR **/
509  title;footnote;ods _all_ close;
510  ods graphics on;
511  ods html5(id=vscode) style=HTMLEncore options(bitmap_mode='inline'
svg_mode='inline');
NOTE: Writing HTML5(VSCODE) Body file: sashtml5.htm
512  %let _SASPROGRAMFILE =
%nrquote(%nrstr(/workspaces/myfolder/Pharmasug25/5SASReport.sasnb));
513  data dst4_modified;
514    set dst4;
515    where 'plant species'n ne ' ';
516    PlantSpecies_Year=catx('-','plant species'n,year);
517    Month=month(date);
518  run;
14                                                       The SAS System
Wednesday, April 30, 2025 02:32:00 AM
NOTE: There were 1708 observations read from the data set WORK.DST4.
      WHERE 'plant species'n not = ' ';
NOTE: The data set WORK.DST4_MODIFIED has 1708 observations and 77 variables.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.01 seconds

519  ;*';*";*/;run;quit;ods html5(id=vscode) close;
```

SAS Code

```
title;
ods layout gridded columns=2 column_gutter=1cm;
ods region;
proc sgplot data=dst4_modified;
```

```
  where plot='native';
  heatmap x=Month y=PlantSpecies_Year
          / discretex x2axis colormodel=(peachpuff orange firebrick);
  x2axis values=('5' '6' '7' '8' '9' '10') label='Native plants with respective
flowering periods';
  yaxis discreteorder=formatted reverse label='Plant Species and Year';
run;

ods region;
proc sgplot data=dst4_modified;
  where plot='non-native';
  heatmap x=Month y=PlantSpecies_Year
          / discretex x2axis colormodel=(peachpuff orange firebrick);
  x2axis values=('5' '6' '7' '8' '9' '10') label='Non-Native plants with respective
flowering periods';
  yaxis discreteorder=formatted reverse label='Plant Species and Year';
run;
ods layout end;
```

SAS Results



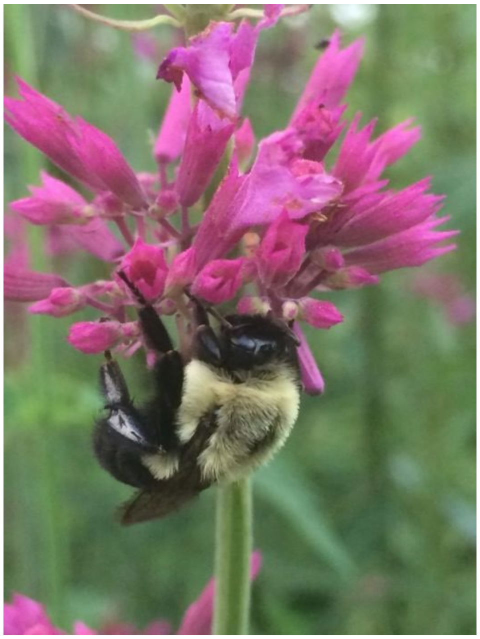
Explanation:
This SAS code creates side-by-side heatmaps that compare the flowering periods of native and non-native plants using PROC SGPLOT with HEATMAP. It uses ODS LAYOUT GRIDDED to organize the visual output into two columns, each showing a heatmap of plant species (y-axis) across months (x-axis). The dataset dst4_modified is filtered into native and non-native plots, and for each, it plots how flowering activity (via PlantSpecies_Year) is distributed over time (May–October). Color intensities range from peachpuff to firebrick, visually highlighting blooming trends. This allows for a clear, compact comparison of seasonal flowering behavior between plant types. ✿

SQL Data Reporting

The PROC SQL step is for querying and manipulating data — not for creating plots or layouts, so the SGPLOT and ODS LAYOUT steps cannot be converted into PROC SQL.

Python Data Reporting

Truly a buzz-worthy battle between Python and SAS as we compare their skills in visualizing native versus non-native flowering plants! Watch as Python whips up colorful heat maps and SAS turns data into dazzling visuals. Who will create the most vibrant bloom? Let’s dive in and see which tool gets the hive buzzing!



7 *‘Pink Panther’ Anise Hyssop is a bee magnet*

Python Code

```
#Import Libraries
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib.backends.backend_pdf import PdfPages
#Load sas dataset into a pandas dataframe using the pandas read_sas method
df4=pd.read_csv('/workspaces/myfolder/Pharmasug25/native_vs_nonnative_bumblebee_sighting_pollinators_of_farm_data_for_publication.csv' , encoding='latin-1')
```

Python Code

```
#We want to plot a heatmap of months in which the plants flower, assuming we have all flowering plants in the data
```

```
# Confirm date column is datetime
df4['date'] = pd.to_datetime(df4['date'], errors='coerce')

# Extract only the needed columns (make sure these column names exist)
df4 = df4[['year','plot','date','plant species']]

#Create new columns from date
df4['month'] = df4['date'].dt.month
df4['year'] = df4['date'].dt.year
df4.head(10)
```

SAS Console

	year	plot	date	plant species	month
0	2016	non-native	2016-09-21	Trifolium repens	9
1	2016	non-native	2016-09-21	Cosmos bipinnatus	9
2	2016	non-native	2016-09-21	Cosmos bipinnatus	9
3	2016	native	2016-09-21	Monarda punctata	9
4	2016	native	2016-09-21	Monarda punctata	9
5	2016	native	2016-09-21	Monarda punctata	9
6	2016	native	2016-09-21	Bidens aristosa	9
7	2016	native	2016-09-21	Bidens aristosa	9
8	2016	native	2016-09-21	Bidens aristosa	9
9	2016	native	2016-09-21	Bidens aristosa	9

Python Code

```
#Turn the months into dummy coded columns that we can sum over using get_dummies

#Pandas library function that converts categorical variable(s) into dummy/indicator variables (one-hot encoded format).
df4 = pd.get_dummies(df4, columns=['month'], dtype=float)

#Rename months for clarity and plotting
df4.rename(columns=dict(month_4='4',month_5='5',month_6='6',month_7='7',month_8='8',month_9='9',month_10='10'), inplace=True)

df4.head(10)
```

Python Console

	year	plot	date	plant species	4	5	6	7	8	9	10
0	2016	non-native	2016-09-21	Trifolium repens	0.0	0.0	0.0	0.0	0.0	1.0	0.0
1	2016	non-native	2016-09-21	Cosmos bipinnatus	0.0	0.0	0.0	0.0	0.0	1.0	0.0
2	2016	non-native	2016-09-21	Cosmos bipinnatus	0.0	0.0	0.0	0.0	0.0	1.0	0.0
3	2016	native	2016-09-21	Monarda punctata	0.0	0.0	0.0	0.0	0.0	1.0	0.0
4	2016	native	2016-09-21	Monarda punctata	0.0	0.0	0.0	0.0	0.0	1.0	0.0
5	2016	native	2016-09-21	Monarda punctata	0.0	0.0	0.0	0.0	0.0	1.0	0.0
6	2016	native	2016-09-21	Bidens aristosa	0.0	0.0	0.0	0.0	0.0	1.0	0.0
7	2016	native	2016-09-21	Bidens aristosa	0.0	0.0	0.0	0.0	0.0	1.0	0.0
8	2016	native	2016-09-21	Bidens aristosa	0.0	0.0	0.0	0.0	0.0	1.0	0.0
9	2016	native	2016-09-21	Bidens aristosa	0.0	0.0	0.0	0.0	0.0	1.0	0.0

Python Code

```
#Separate data into native and non-native plants for plotting a heatmap
native_df = df4[df4['plot']=='native']
non_native_df = df4[df4['plot']=='non-native']

#Roll up data to get number of records for each plant for each month
native_plot = native_df.groupby(['plant
species','year'])[['4','5','6','7','8','9','10']].agg('sum')
non_native_plot = non_native_df.groupby(['plant
species','year'])[['4','5','6','7','8','9','10']].agg('sum')

native_plot.head()
```

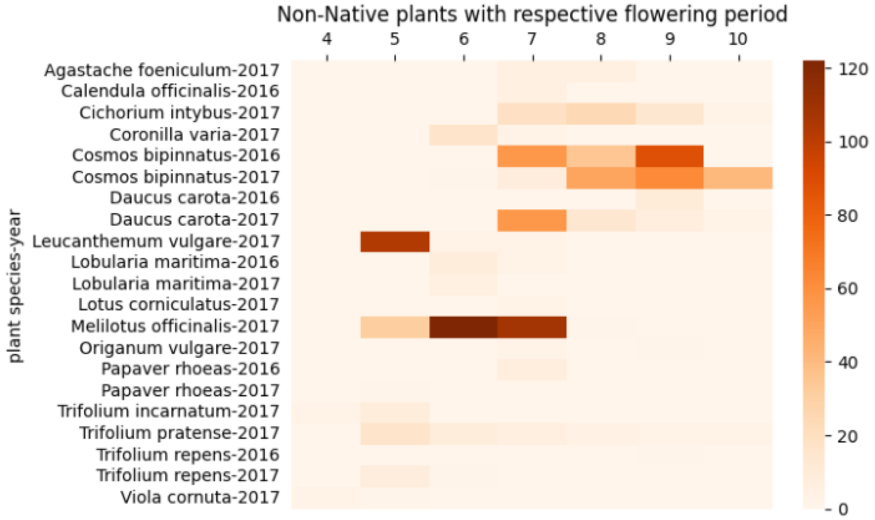
Python Console

		4	5	6	7	8	9	10
plant species	**year**							
Asclepias tuberosa	2017	0.0	0.0	20.0	0.0	0.0	0.0	0.0
Bidens aristosa	2016	0.0	0.0	0.0	0.0	0.0	26.0	0.0
	2017	0.0	0.0	0.0	0.0	0.0	24.0	2.0
Chamaecrista fasciculata	2016	0.0	0.0	0.0	4.0	0.0	0.0	0.0
	2017	0.0	0.0	0.0	44.0	19.0	0.0	0.0

Python Code

```
#create a heatmap for flowering period of non-native plants
ax = sns.heatmap(non_native_plot, cmap='Oranges')
ax.set_title('Non-Native plants with respective flowering period')
ax.xaxis.tick_top()
ax.tick_params(left=False)
plt.savefig('seaborn_plot.pdf', format='pdf')
```

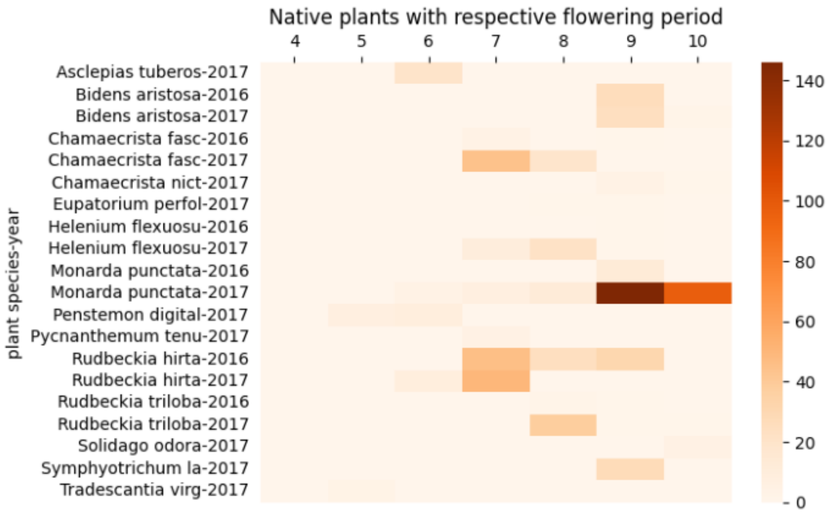
Python Console



Python Code

```
#create a heatmap for flowering period of native plants
ax = sns.heatmap(native_plot, cmap='Oranges') #Create the heatmap
ax.set_title('Native plants with respective flowering period');
ax.xaxis.tick_top()
ax.tick_params(left=False)
```

Python Console



Export results-heatmap to Pdf

Python Code

```
#Open PDF File for Multiple Pages
with PdfPages('/workspaces/myfolder/SASPythonDataScientists/multiple_plots.pdf') as
pdf:

    plt.figure #Create and Save the First Plot
    ax = sns.heatmap(native_plot, cmap='Oranges') #Create the heatmap
    ax.set_title('Native plants with respective flowering period');
    ax.xaxis.tick_top()
    ax.tick_params(left=False)
    pdf.savefig(bbox_inches='tight')  # Saves current figure to the PDF with a
    #tight bounding box, which adjusts layout to fit plot content without extra
whitespace.
    plt.close()    # Close the figure

    plt.figure
    ax = sns.heatmap(non_native_plot, cmap='Oranges')
    ax.set_title('Non-Native plants with respective flowering period')
    ax.xaxis.tick_top()
    ax.tick_params(left=False)
    pdf.savefig(bbox_inches='tight')  # Save the current figure into the PDF
    plt.close()    # Close the figure

#https://matplotlib.org/stable/gallery/color/colormap_reference.html
#above is a reference to the colormap if you want to change how the plot looks just
change the cmap option

#https://seaborn.pydata.org/generated/seaborn.heatmap.html
#above is the doc page for the heatmap plot we are using
```

Credits

Special thanks to Nicole Ball for her beautifully crafted SAS heat map code, and to Ari Zitin for his sharp Python heat map translation. A heartfelt thanks as well to my manager, James Waite, whose constant support for creative endeavors allows me to learn by doing what I love most - teaching and empowering users. This project is a true testament to what happens when teamwork, curiosity, and a passion for learning come together.

The Author can be contacted at Charu.shankar@sas.com

How can you help?

1. **Plant native** blooming trees, shrubs, and wildflowers to provide pollinators with nectar and pollen to eat. There are plenty of helpful resources on native plants for your area. One of the most comprehensive ones is the Lady Bird Johnson Wildlflower database.
2. **Be careful about what plants you buy.** Even though evidence is building that neonics are bad for bees, many commercial plants are still sprayed with this systemic herbicide before they are shipped to the big box stores and garden centers. Check the label of each plant for a warning to see if it was sprayed for aphids and other insects. If it is, then set it back down for pollinators' sake.

3. **Plant for variety in color, sizes and seasons.** Having a buffet of flowering options is best to help pollinators, especially bees. While many bees are generalists and don't care about the flower species, there are some that are specialists (i.e. they only visit specific native nectar plant species). Some can prefer a certain size of flower so providing many different types of flowers is helpful.
4. **Provide nesting habitats for solitary bees.** As mentioned above, solitary bees have different nesting needs than hive bees. Keeps areas of bare soil where ground-nesting bees can burrow. Provide pithy plant stalks like sunflowers where the bees can hollow out the inside for their nest. If you choose to use a bee hotel, they will need to be disinfected after every season to prevent the spread of bee diseases.
5. **Participate in citizen science activities!** There are thousands of native bee species in the U.S., and there is still much we need to learn about individual species. Professor Beckhams credits observations from citizen scientists on iNaturalist to help track bees for her studies in Texas.

Citations

1 Plants pollinated by non-native honeybees are less likely to survive -*Proceedings of the Royal Society B*. Researchers from the University of California San Diego

2 Data Prep Still Dominates Data Scientists' Time, Survey Finds

3 Patterns of widespread decline in North American bumble bees

4 Georgia Mason Solitary Bee

5 Are native and non-native pollinator friendly plants equally valuable for native wild bee communities

6 Sweat Bees

7 Liatris for bees