Canada Revenue Agency | Agence du revenu du Canada

Bits and Bytes – A Mix for High Volume of Data

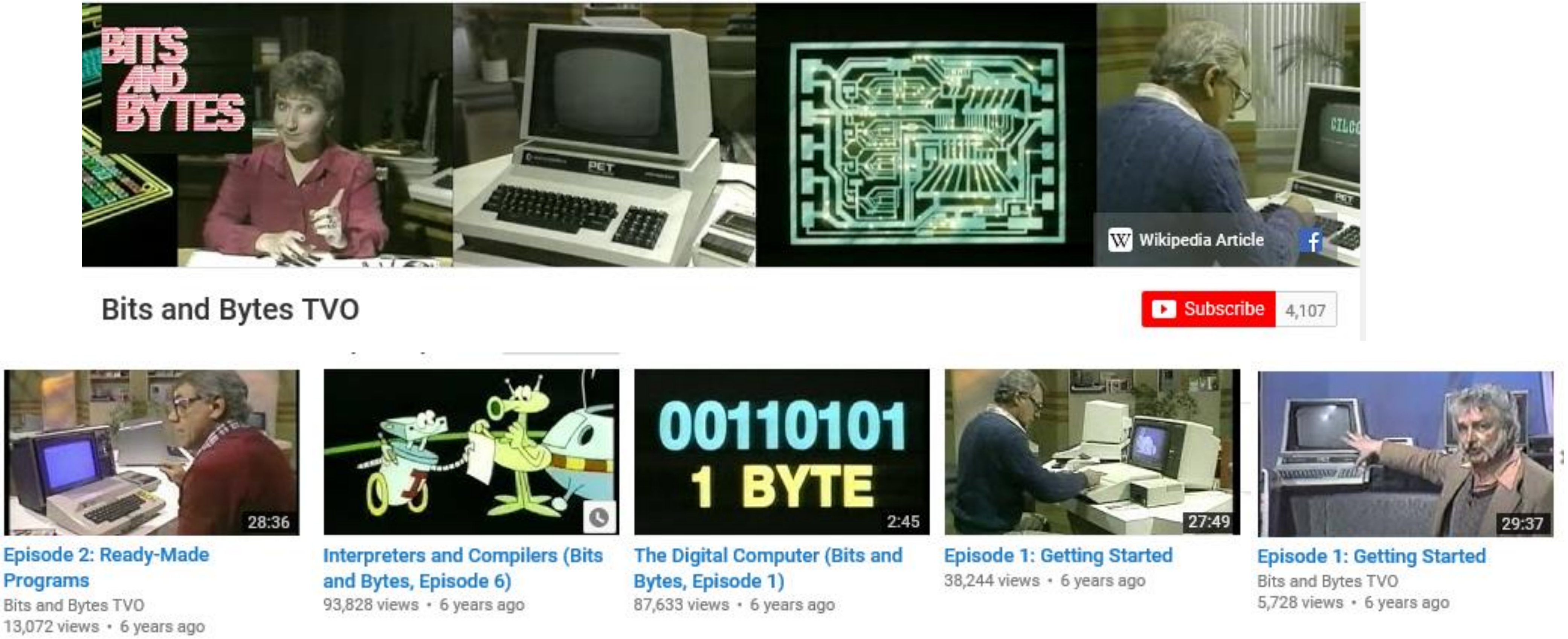
Karine Désilets

Senior Quantitative Analyst

Research and Innovation LAB Directorate
Service, Innovation and Integration Branch
Canada Revenue Agency
Government of Canada

Originally co-presented and co-created with Yves Deguire, Alumni, Statistics Canada

Bits and Bytes: The TV Show



Bits and Bytes TVO

Episode 2: Ready-Made Programs
Bits and Bytes TVO
13,072 views • 6 years ago

Interpreters and Compilers (Bits and Bytes, Episode 6)
93,828 views • 6 years ago

The Digital Computer (Bits and Bytes, Episode 1)
87,633 views • 6 years ago

Episode 1: Getting Started
38,244 views • 6 years ago

Episode 1: Getting Started
Bits and Bytes TVO
5,728 views • 6 years ago

Original Show Episodes (83-84) – The Agenda

- Program 1: Getting Started **(Introduction)**
- Program 2: Ready-Made Programs **(Recipe for Large Datasets)**
- Program 3: How Programs Work? **(General Principles)**
- Program 4: File & Data Management **(Infrastructure and Disk Space)**
- Program 5: Communication Between Computers **(I/O Processing)**
- Program 6: Computer Languages **(SAS Specific)**
- Program 7: Computer-Assisted Instruction **(Divide and Conquer: Multi-Threading or Parallel Processing)**
- Program 8: Games & Simulations **(# of Variables or # of Records)**
- Program 9: Computer Graphics **(Memory Processing)**
- Program 10: Computer Music **(Dataset Compression)**
- Program 11: Computers at Work **(Indexes)**
- Program 12: What Next? **(Conclusion)**

Working with High Volume of Data?

Infrastructure

SAS Specific

Algorithm Order O(.)

Number of Variables

Memory Space / Processing

KEEP CALM AND TAKE A DEEP BREATH

Multi-Threading or Parallel Processing

Disk Space

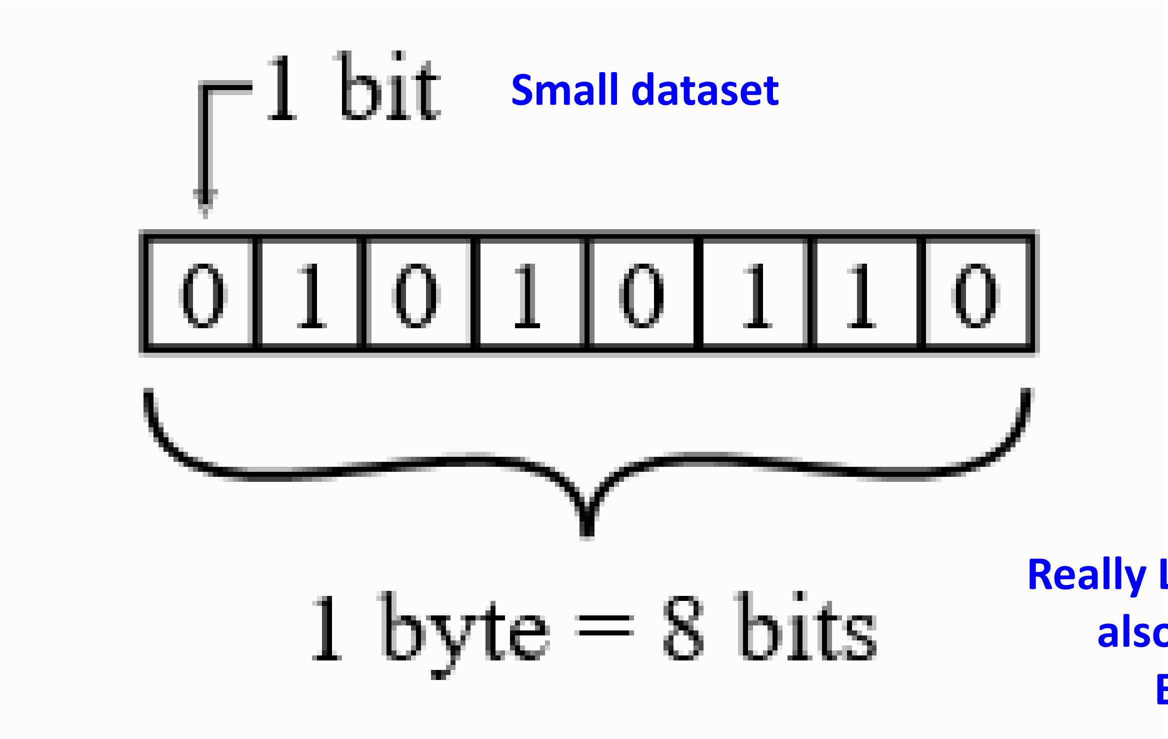
Number of Records

I/O processing

Compression

Indexes

Bits or Bytes ?

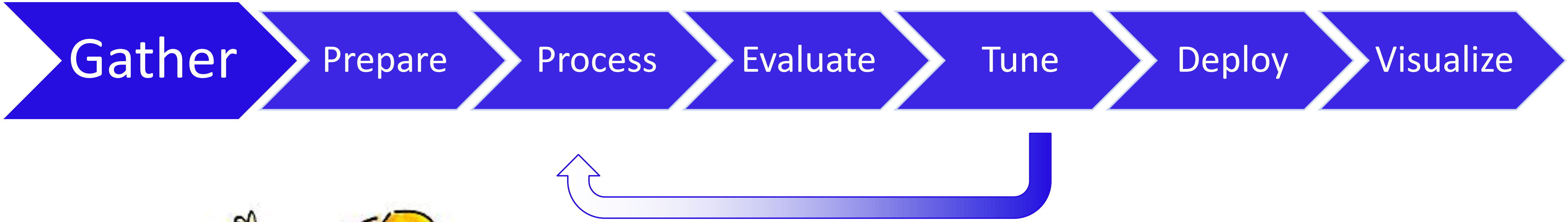


Small dataset

Really Large Datasets
also known as
Big Data

Processing ToolKit Accelerator

" With high volume of Data, avoid unnecessary steps, datasteps or reports"



Gather

Prepare

Process

Evaluate

Tune

Deploy

Visualize

HEY IT always has the same steps but with different naming depending where you find the info.

CRA Data : The Pot of GOLD!

Income Tax

GST/HST

Payroll

- Agency Mandate
- Flagship Goals
- Agency Metadata Repository
- Agency Data Lakes

How many data SAVVY Analysts are in this room ?

Business Number and Related

Savings and Pension Plan

Child Benefits

Charities and Giving

Excise Taxes, Duties, and Levies

A Recipe – KISS Principle

Keep It Simple Stupid!

Techniques for Reducing the Amount of Data Processed

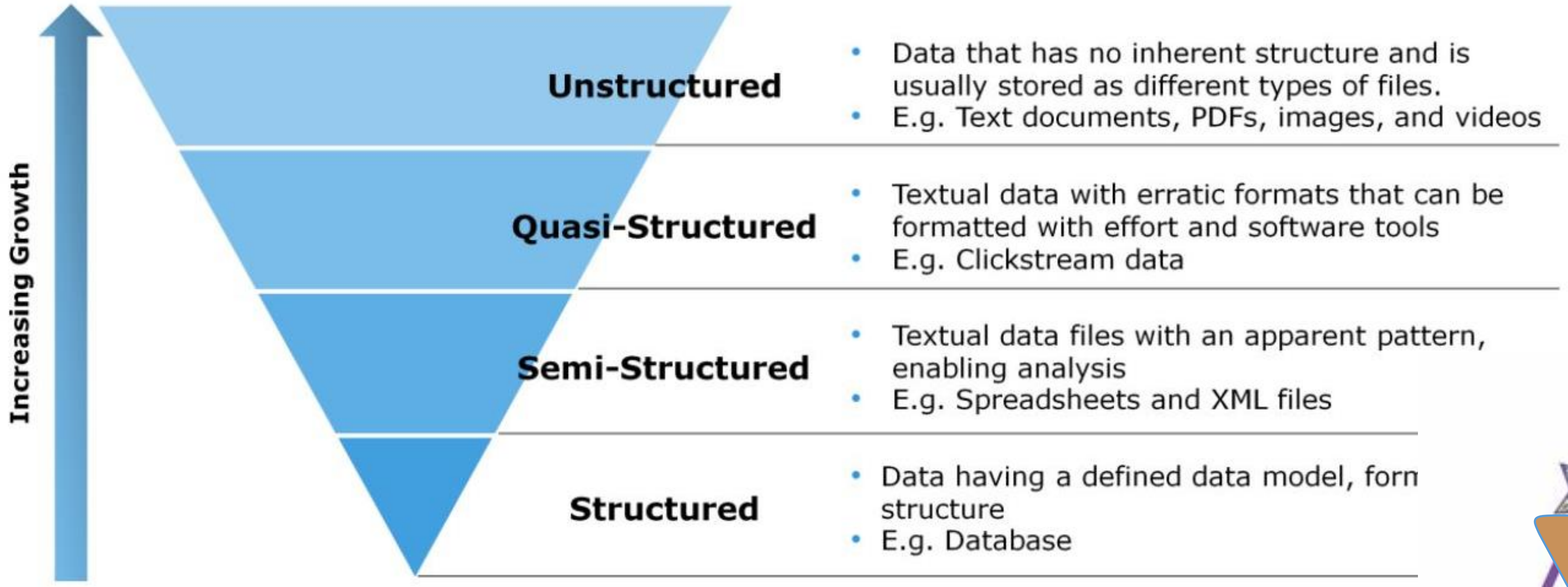
Techniques for Reducing the Amount of Data Stored on Disk

Techniques for Processing Large Datasets Efficiently

The Location of Data

Confused about Data Lakes, Data Warehouse and Data Mart ? At the end of the day, **IT is all SAS Datasets!**

The Nature of Data



Unstructured

- Data that has no inherent structure and is usually stored as different types of files.
- E.g. Text documents, PDFs, images, and videos

Quasi-Structured

- Textual data with erratic formats that can be formatted with effort and software tools
- E.g. Clickstream data

Semi-Structured

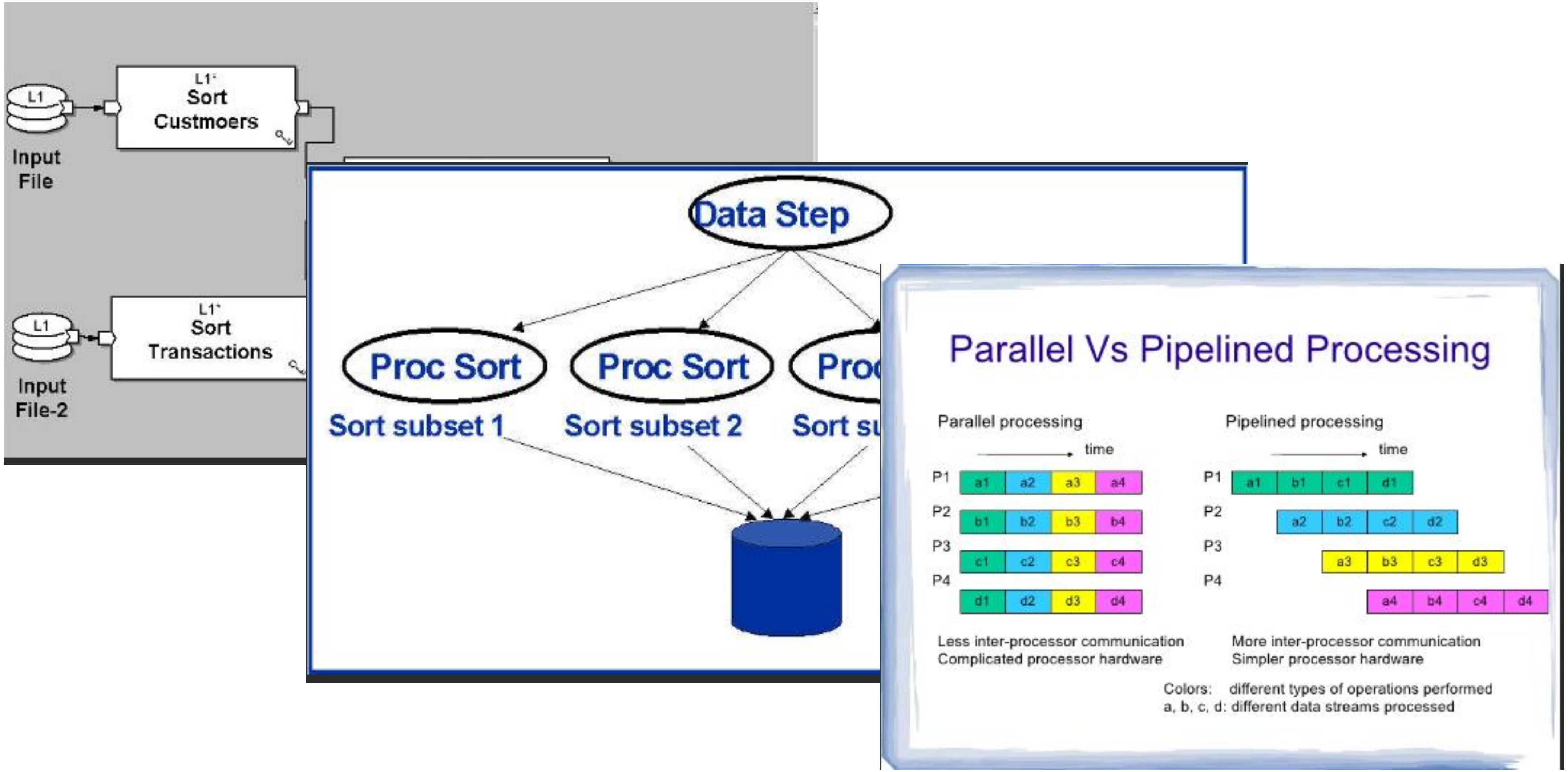
- Textual data files with an apparent pattern, enabling analysis
- E.g. Spreadsheets and XML files

Structured

- Data having a defined data model, forn structure
- E.g. Database

Source : https://4.bp.blogspot.com/-pUjc76RbEiE/V01K_GzWc1I/AAAAAAAAC4U/KAfRqWCv3GYMlH7esHhGgP0r98q_aqDVwCLcB/s1600/1.JPG

Parallel and Sequential Processing / Pipelines



Parallel Vs Pipelined Processing

Parallel processing

a1 a2 a3 a4
b1 b2 b3 b4
c1 c2 c3 c4
d1 d2 d3 d4

Pipelined processing

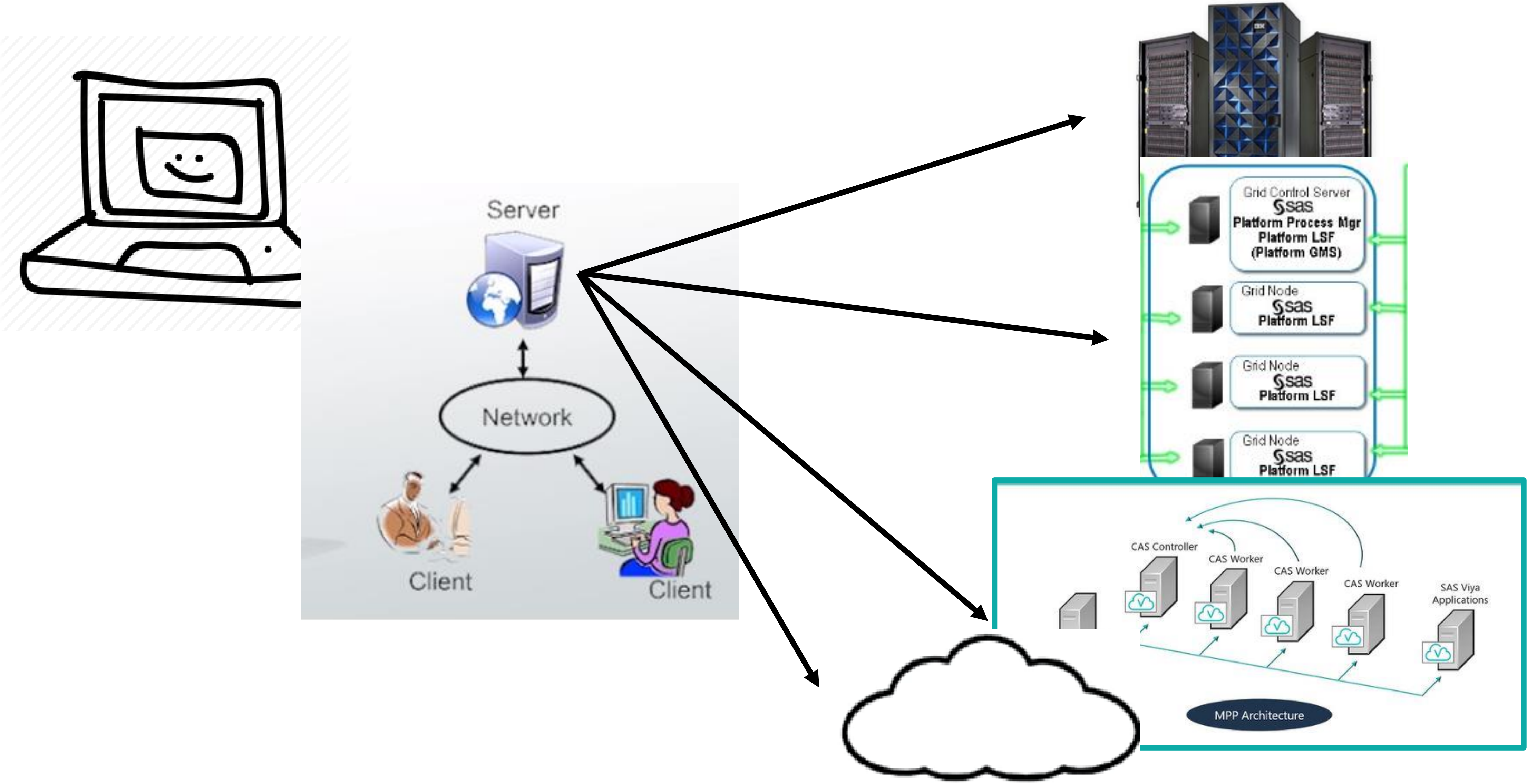
a1 b1 c1 d1
a2 b2 c2 d2
a3 b3 c3 d3
a4 b4 c4 d4

Less inter-processor communication
Complicated processor hardware

More inter-processor communication
Simpler processor hardware

Colors: different types of operations performed
a, b, c, d: different data streams processed

Infrastructure – The Options



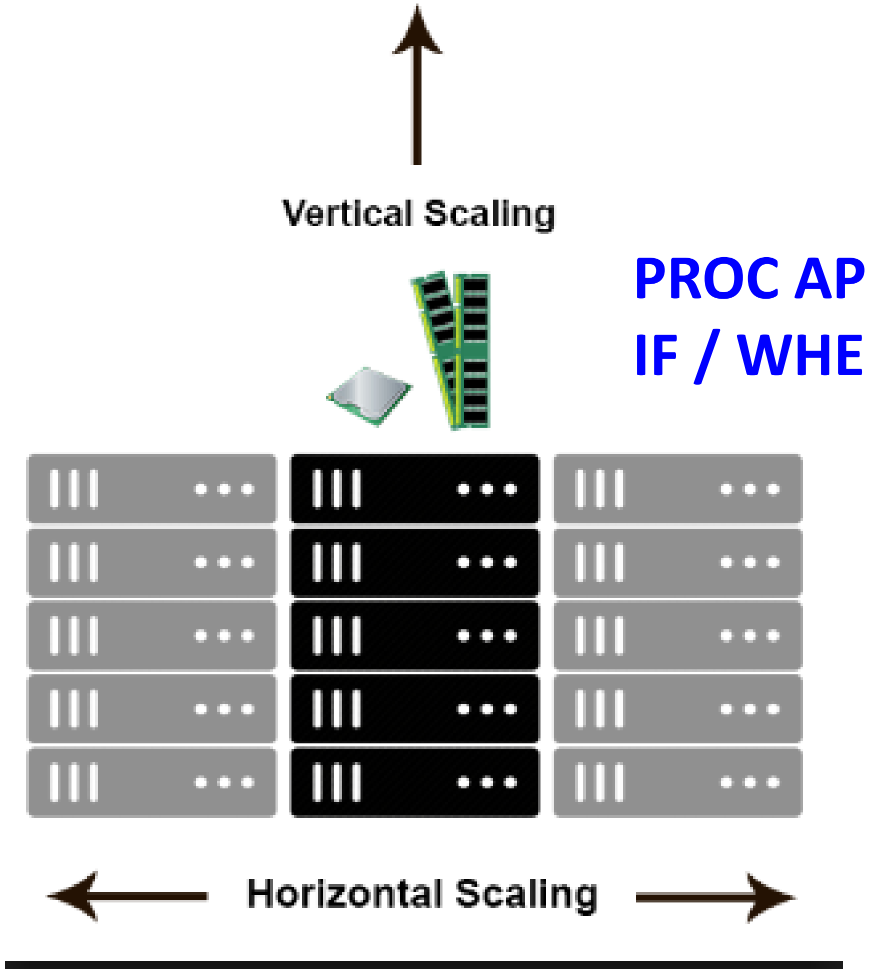
Techniques for Reducing the Amount of Data Processed

ONLY PROCESS THE DATA YOU NEED TO PROCESS

Attack it as a Data Scientist Approach:

“From ETL to data preparation, with model training or data processing up to deployment and data visualisation”

Scale Up! Scale Down!



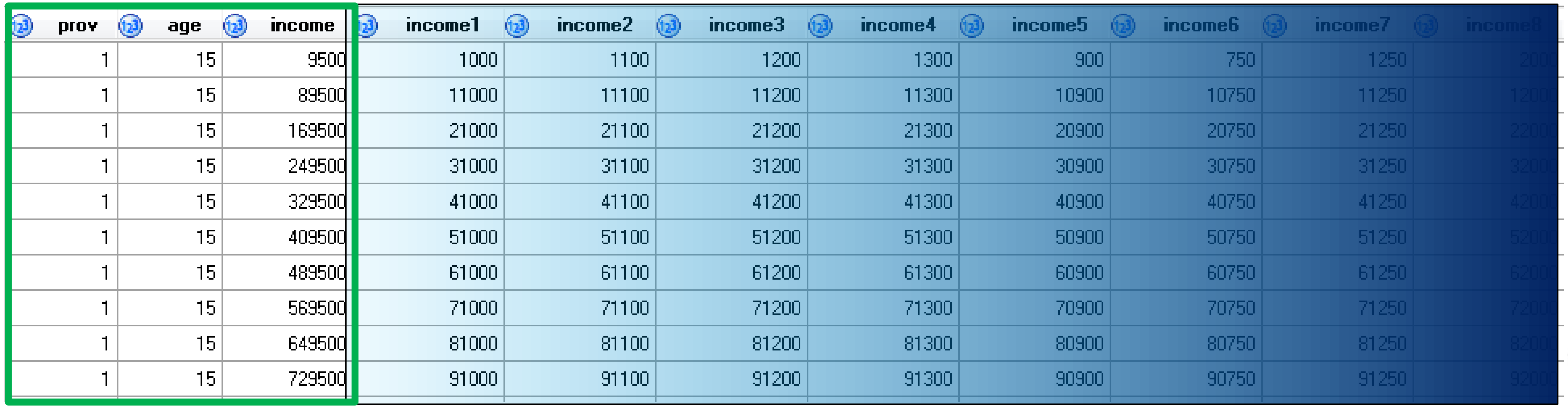
PROC APPEND versus
IF / WHERE statements

New variables versus **DROP / KEEP**

Keep Only the Needed Ones!

```
data verticalReduce (drop=income1-income100);
    set myBigFile (keep=prov age income1-income100);
    income = sum(of income1-income100);
run;
```

Vertical Reduction (KEEP/DROP)



prov	age	income	income1	income2	income3	income4	income5	income6	income7	income8
1	15	9500	1000	1100	1200	1300	900	750	1250	2000
1	15	89500	11000	11100	11200	11300	10900	10750	11250	12000
1	15	169500	21000	21100	21200	21300	20900	20750	21250	22000
1	15	249500	31000	31100	31200	31300	30900	30750	31250	32000
1	15	329500	41000	41100	41200	41300	40900	40750	41250	42000
1	15	409500	51000	51100	51200	51300	50900	50750	51250	52000
1	15	489500	61000	61100	61200	61300	60900	60750	61250	62000
1	15	569500	71000	71100	71200	71300	70900	70750	71250	72000
1	15	649500	81000	81100	81200	81300	80900	80750	81250	82000
1	15	729500	91000	91100	91200	91300	90900	90750	91250	92000

Data Scientists would say:

- Hyper-Parameters or Features Reduction (PCA)

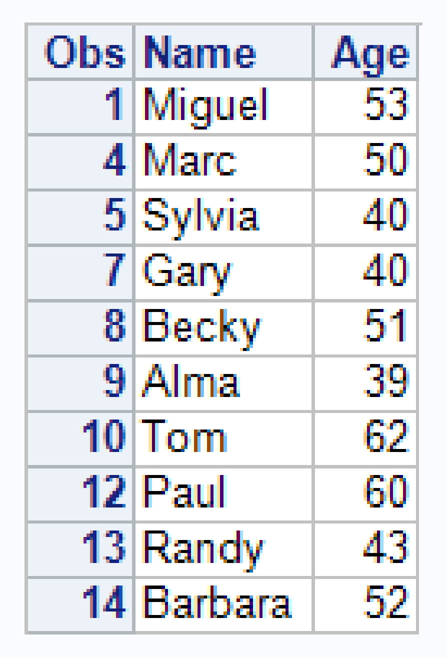
Subset Them!

```
data Ages;
   input Name $ Age;
   datalines;
Miguel 53
Brad 27
Willie 69
Marc 50
Sylvia 40
Arun 25
Gary 40
Becky 51
Alma 39
Tom 62
Kris 66
Paul 60
Randy 43
Barbara 52
Virginia 72
;
```

Horizontal Reduction (WHERE/IF)

```
proc print data=Ages;
     WHERE (30<=Age<=65);
run;
```

```
data SelectAges;
      set Ages;
     if 30<=Age<=65;
run;
```



Obs	Name	Age
1	Miguel	53
4	Marc	50
5	Sylvia	40
7	Gary	40
8	Becky	51
9	Alma	39
10	Tom	62
12	Paul	60
13	Randy	43
14	Barbara	52

“Delete irrelevant observations as early as possible”

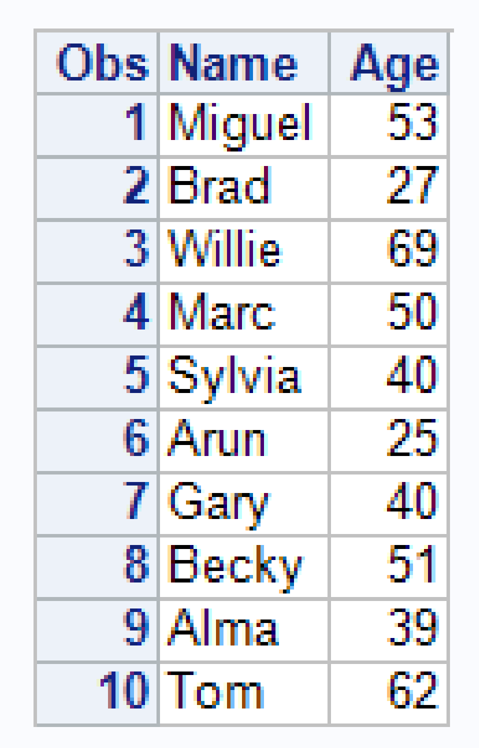
Process in Small Chunks!

- **Select a Subset**
  - ✓ Set OBS= n to specify a number to indicate when to stop processing observations in a DATA step or PROC.
  - ✓ For testing functionality with a smaller subset of data.
- **Use Sampling Strategy**
  - ✓ Proc SurveySelect
- **Data Scientists would sometimes refer to:**
  - ✓ Training Dataset
  - ✓ Test Dataset

Process a Small Number of Observations

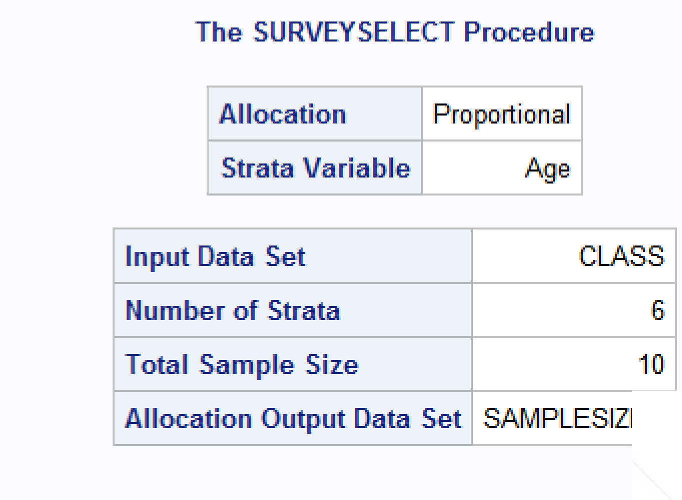
```
data Ages;
    input Name $ Age;
    datalines;
Miguel 53
Brad 27
Willie 69
Marc 50
Sylvia 40
Arun 25
Gary 40
Becky 51
Alma 39
Tom 62
Kris 66
Paul 60
Randy 43
Barbara 52
Virginia 72
;
```

```
proc print data=Ages (obs=10);
run;
```



Obs	Name	Age
1	Miguel	53
2	Brad	27
3	Willie	69
4	Marc	50
5	Sylvia	40
6	Arun	25
7	Gary	40
8	Becky	51
9	Alma	39
10	Tom	62

```
proc surveyselect data= Ages
      n = 10 out = SampleSizes;
    strata Age / alloc= prop nosample;
 run;
```



The SURVEYSELECT Procedure

Allocation	Proportional
Strata Variable	Age

Input Data Set	CLASS
Number of Strata	6
Total Sample Size	10
Allocation Output Data Set	SAMPLESIZI

Clean Them in Early Stage

“If possible, clean the datasets in parallel and with pipelines processes.”

- Remove Duplicates
- Derived Variables Creation
- Sparse Data and Imputation
- Outliers Processing
- Standardization

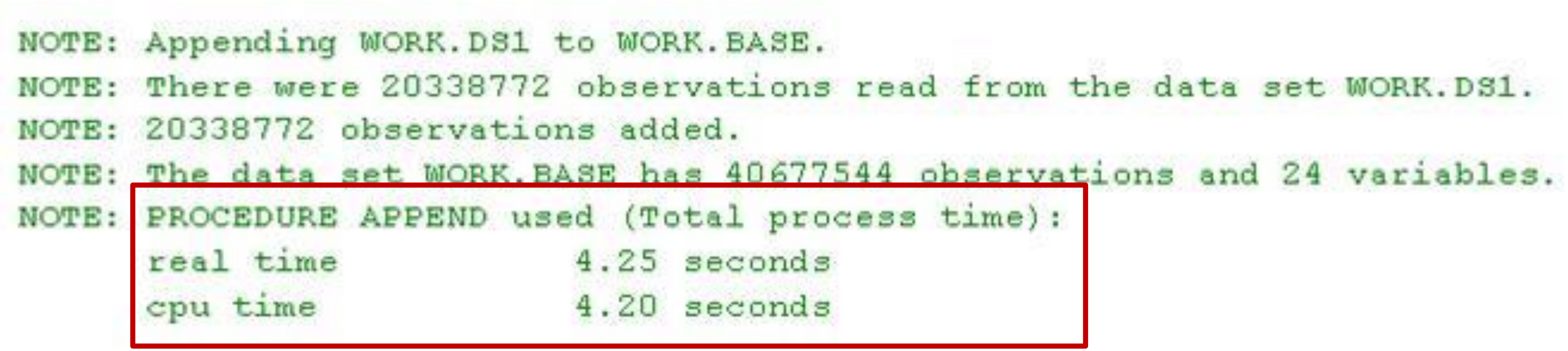
Incomparable Append

```
proc append base=base data=ds1;
run;
```

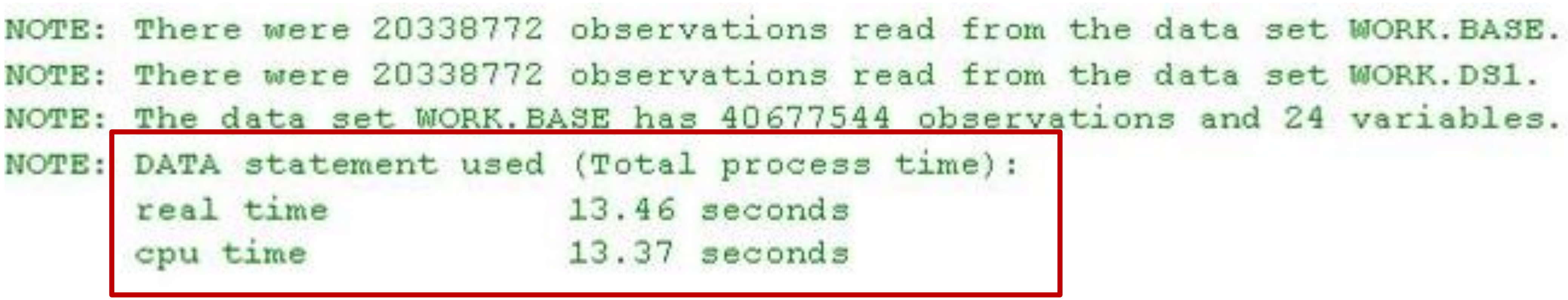
VS

```
data base;
    set base ds1;
run;
```

PROC APPEND VS Data Step



```
NOTE: Appending WORK.DS1 to WORK.BASE.
NOTE: There were 20338772 observations read from the data set WORK.DS1.
NOTE: 20338772 observations added.
NOTE: The data set WORK.BASE has 40677544 observations and 24 variables.
NOTE: PROCEDURE APPEND used (Total process time):
      real time           4.25 seconds
      cpu time            4.20 seconds
```



```
NOTE: There were 20338772 observations read from the data set WORK.BASE.
NOTE: There were 20338772 observations read from the data set WORK.DS1.
NOTE: The data set WORK.BASE has 40677544 observations and 24 variables.
NOTE: DATA statement used (Total process time):
      real time           13.46 seconds
      cpu time            13.37 seconds
```

What About PROC SQL?

```
proc sql;
     create table work.Append_Table as
          select * from work.base
               outer union corr
                    select * from work.ds1
     ;
     drop table base;
quit;

proc datasets library=work;
     change Append_Table=Base;
run;
```

Techniques for Reducing the Amount of Data Stored on Disk

REDUCE THE SIZE OF YOUR SAS DATA SETS AS MUCH AS YOU CAN

Keep that Length under Control!

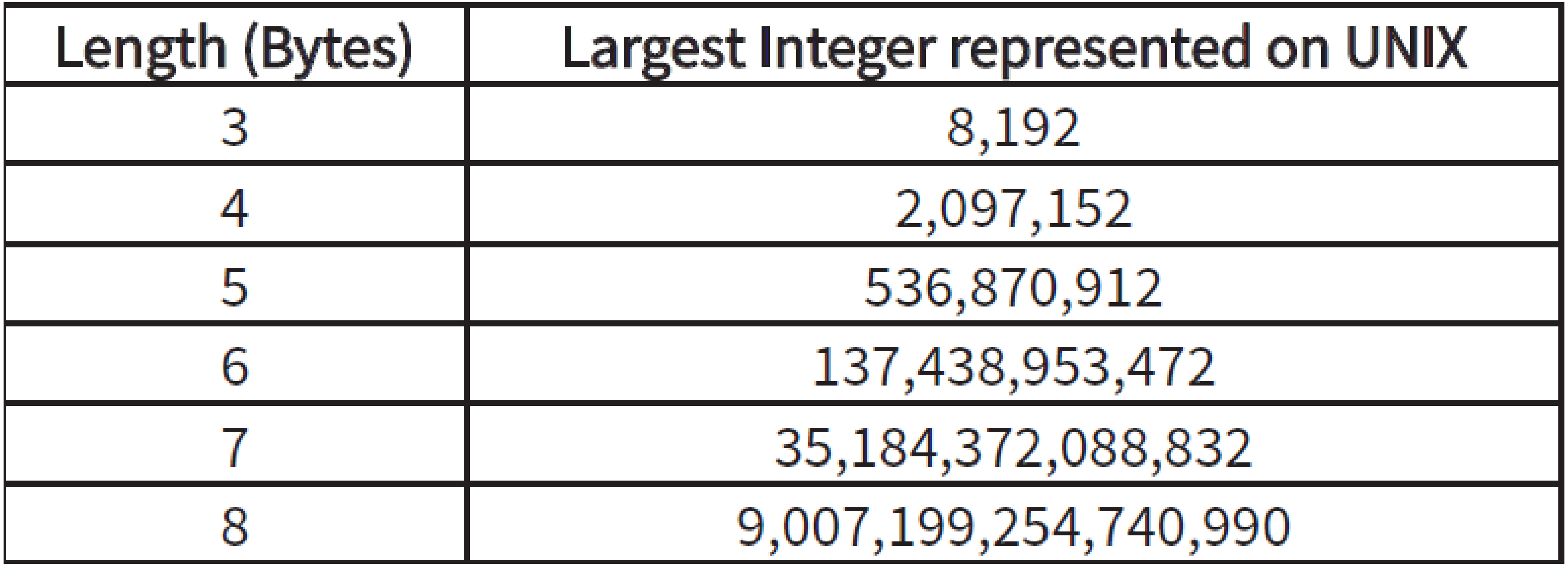
- Numeric variables:
  - Default /maximum length is 8.
  - Length can be reduced down to 3.
- Character variables:
  - The first assigned value determines the length.
  - Length ranges from 1 to 32767.

Examples

```
Data DATA1;
    length varn1 varn2 varn3 3
           varc1 varc2 $4;
    set DATA1;
run;
```

```
data DATA1;
    attrib varn1 varn2 varn3 length=3;
    attrib varc1 varc2  length=$4;
    set DATA1;
run;
```

You May not Need all Those Numbers



Length (Bytes)	Largest Integer represented on UNIX
3	8,192
4	2,097,152
5	536,870,912
6	137,438,953,472
7	35,184,372,088,832
8	9,007,199,254,740,990

CAREFUL:

1) **DATES need 4 bytes**
2) **Fractions should be left with 8 bytes**

A Special Case: Flag Variables

- Flags are variable that can be set to 0 or 1
- Typically defined as numeric and occupy at least 3 bytes.
- Use character variables with length of 1 instead!

```
data mynewsds(drop=var1);
       length flag $1.;
       set myds;

       if ((var1 = 123) or (var1=456)) then
              flag=0;
       else flag=1;
run;
```

Squeeze that Dataset

- The **%SQUEEZE** macro created by Ross Bettinger macros can find the minimum lengths required by numeric or character variables for a SAS data set and use these lengths to reduce the size of the dataset.
- The source code is available at

  http://support.sas.com/kb/24/804.html

To Compress or Not to Compress

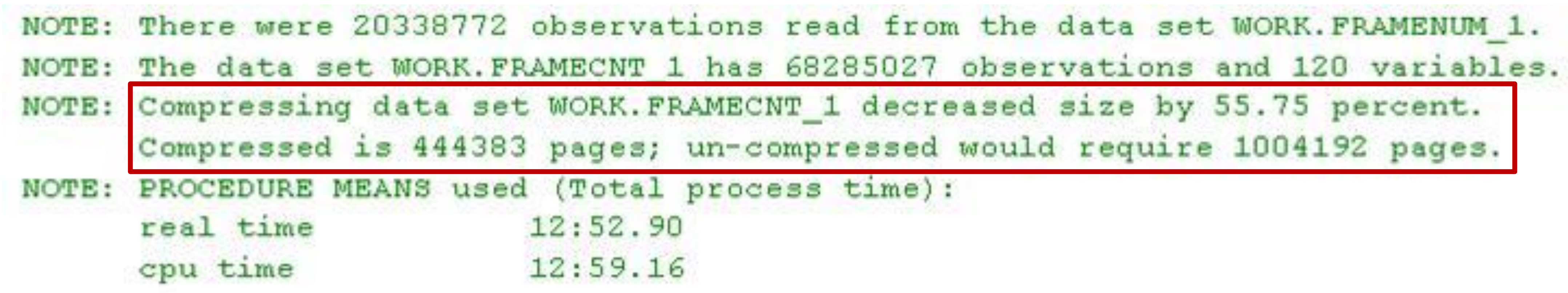
- Goal: Reduce the number of bytes each observation required
- A dataset option or a system option
    - **COMPRESS = NO | YES | CHAR | BINARY**
- Not free: CPU cycles are required!

```
/* System option */
options compress = binary;
```

```
/* Dataset option */
data mydata(compress = yes);
        set mylib.bigdata;
run;
```

When to Compress

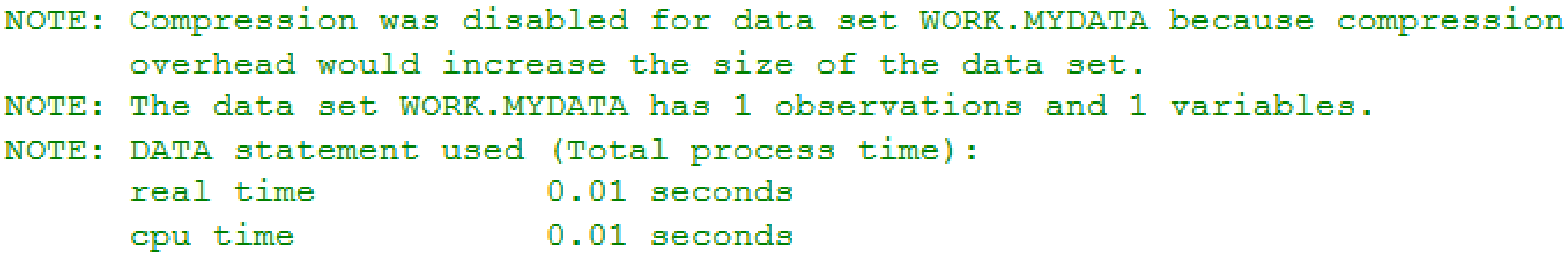
- ✓ Large dataset (millions of records)
- ✓ Large character variables
- ✓ Many numeric variables
- ✓ Lots of repetitions: common patterns, empty spaces and numeric variables



```
NOTE: There were 20338772 observations read from the data set WORK.FRAMENUM_1.
NOTE: The data set WORK.FRAMECNT_1 has 68285027 observations and 120 variables.
NOTE: Compressing data set WORK.FRAMECNT_1 decreased size by 55.75 percent.
      Compressed is 444383 pages; un-compressed would require 1004192 pages.
NOTE: PROCEDURE MEANS used (Total process time):
      real time           12:52.90
      cpu time            12:59.16
```

When NOT to Compress

- Small datasets
- Few variables
- Few repetitions



```
NOTE: Compression was disabled for data set WORK.MYDATA because compression
      overhead would increase the size of the data set.
NOTE: The data set WORK.MYDATA has 1 observations and 1 variables.
NOTE: DATA statement used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds
```

Such a Nice View!

- A view is a virtual table defined by a query.
- 2 type of views:
  - DATA Step views
  - SQL views
- Can help performance by reducing the amount of data written to disk.

SQL View

```
proc sql noprint;
        create view myview as select * from mydata
        where (myvar ne .)
        order by frame_id;
quit;
```

```
data myview / view = myview;
        set mydata;
        where (myvar ne .);
        order by frame_id;
quit;
```

Delete all of Them ?

PROC DELETE

```
proc delete;
        data = mylib.datasetName;
run;
```

PROC DATASET

```
proc datasets lib = mylib;
        delete datasetName;
 run;
```

SAS ENTERPRISE GUIDE DROP DATASET

```
%eg_conditional_dropds(work.myDataset);
```

DROP TABLE

```
proc sql;
        drop table mylib.datasetName;
quit;
```

Clean, Clean and Clean!

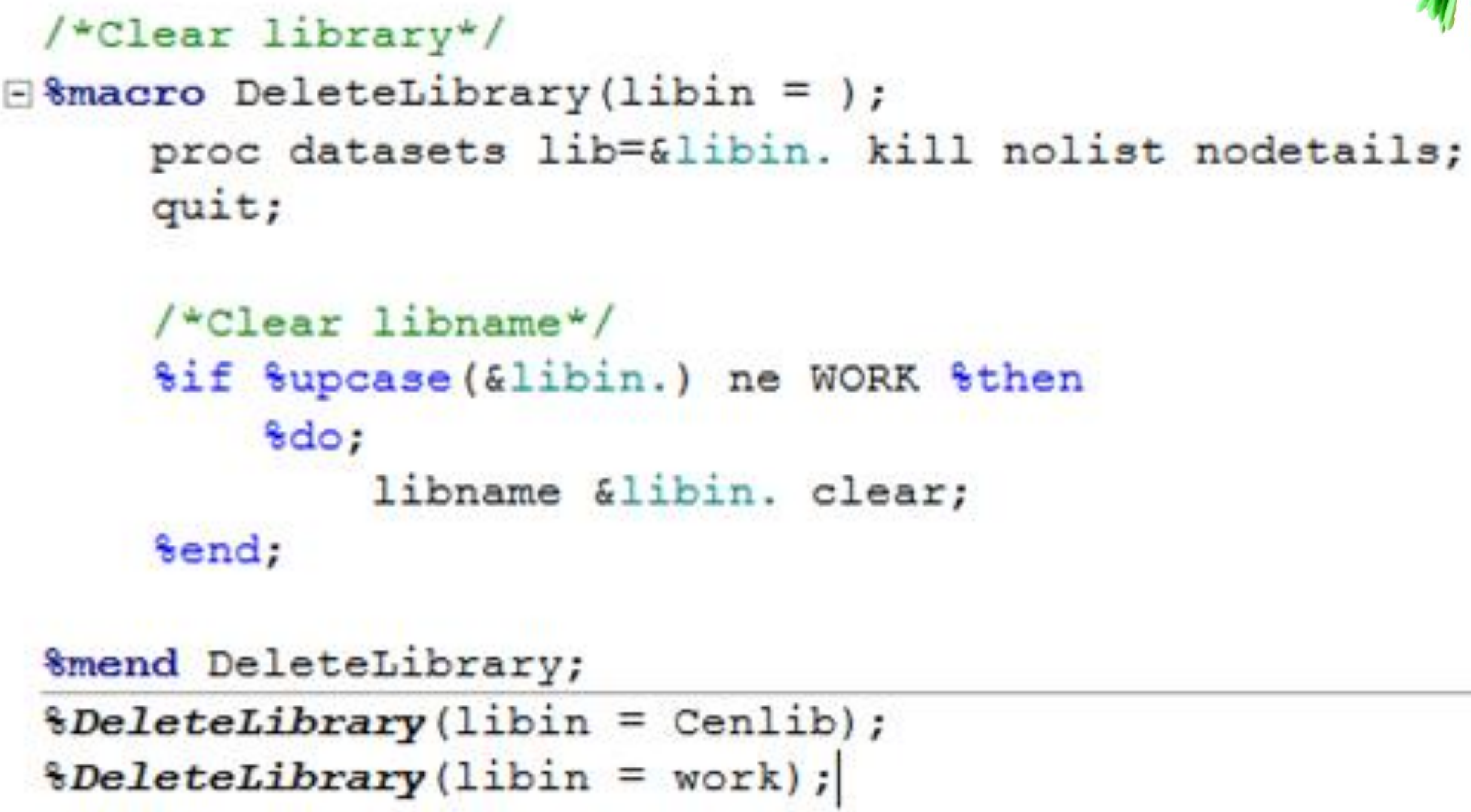
Why Cleaning ? Servers management is more than important

- Deletion of high volume of data take some processing time
- Not deleting high volume of data take disk space

What is the solution ? Where are the Guidelines ?

- Delete a SAS dataset as soon as it is not used, it clear up the SAS data libraries and free some space.
- A SAS session has an amount of WORK space associated to it, with high volume of data, that specific amount could be reached – you can increase it but, it is **Much Better** to delete the no longer needed datasets.

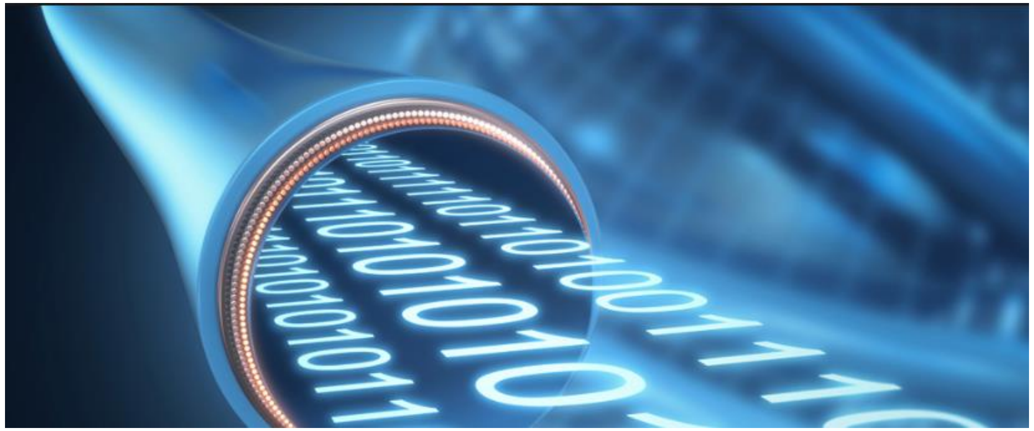
How to Clean Everything ?



```
/*Clear library*/
%macro DeleteLibrary(libin = );
    proc datasets lib=&libin. kill nolist nodetails;
    quit;

    /*Clear libname*/
    %if %upcase(&libin.) ne WORK %then
        %do;
            libname &libin. clear;
    %end;

%mend DeleteLibrary;
%DeleteLibrary(libin = Cenlib);
%DeleteLibrary(libin = work);
```

Techniques for Processing Large Datasets Efficiently

THE MORE DATA YOU HAVE TO PROCESS, THE MORE YOU PAY ATTENTION TO EFFICIENCY



Notation Big O - O(.)

Algorithm Order

Mathematical notation that:

- Describe the performance and complexity of an algorithm.
- Describes the worst-case scenario
- Can be used to describe the execution time required or the space used by an algorithm

Notation: $O(f(n))$

It provide a useful approximation to the actual number of steps in the computation.

The parameter *n* is often referred to as the "size of the problem,"

The function $f(n)$ can be read as the time it takes to solve a problem of size n – a simple representation of the dominant part of f(n)

EXAMPLE # 1

```
x = 1;
for i = 1 to n then do;
            x = x + i ;
end;
```

$O(n)$

EXAMPLE # 2

```
x = 1;
for i = 1 to n then do;
    for j = 1 to m then do
            x = x + i + j ;
    end;
end;
```

$O(n^2)$

A HUGE NO to O(6n)

```
data myBigFile;
            set myBigFile;
run;
```

10 minutes

```
data myBigFile;
            set myBigFile;
            x = 1;
run;
```

10 minutes

```
data myBig
                myBigF
                  + 2
run;
```

10 minutes

```
data myBigFile;
            se          e;
            a
run;
```

10 minutes

```
data myB    le;
            set myBigFile;
            c = x + a;
run;
```

10 minutes

```
data myBigFile;
            set myBigFile;
            d = c / 3;
run;
```

10 minutes

```
data myBigFile;
            set myBigFile;
            x = 1;
            x = 1 + 2;
            a = 3;
            c = x + a;
            d = c / 3;
run;
```

10.2 minutes

HE. OH. Use the PROCS.

“Do not Reinvent the wheels!”

SQL join vs DATA Step Merge

OMG! – Which ones to use ? Let’s look at that.

A Join or a Merge? Give it a Try.

DATA STEP MERGE

- ✓ Need explicit sort before merging tables.
- ✓ DATA step set operators can handle more data sets at a time than PROC SQL outer joins.
- ✓ Non-SQL techniques can open files for read and write at the same time.
- ✓ Customized DATA step report writing techniques (DATA _NULL_) are more versatile than using PROC SQL SELECT clauses to learn SQL constructs.
- ✓ Input of non-RDBMS external sources is easier.

PROC SQL JOIN

- ✓ Does not require explicit code to pre-sort tables (**_Method**) when merging tables
- ✓ Use **Feedback** Options may execute faster for smaller tables.
- ✓ More portable for non-SAS programmers and non-SAS applications.
- ✓ Does not require common variable names to join on
- ✓ Does require same type and length
- ✓ Knowledge of relational data theory opens the power of SQL for many additional tasks.

A Datastep Simple Merge

```
proc sort data = cars out = cars_one ;
          by make model type origin;
run;

proc sort data = cars out = cars_two ;
          by  make model type origin;
run;
```

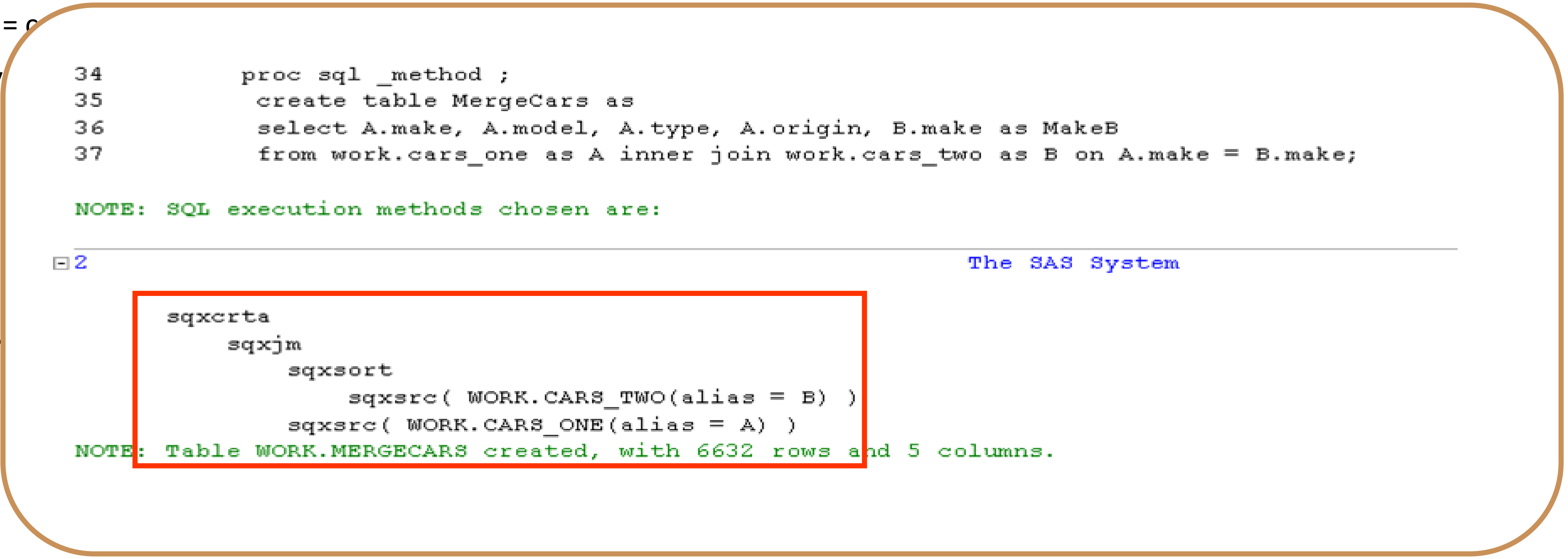
Want to merge datasets? Sort all of them first!

```
data dataStepMerge;
          merge cars_one (in = a ) cars_two (in = b);
          by make model type origin;
          if a and b then output;
run;
```

SQL INNER JOIN EQUIVALENT

That’s the _Method!

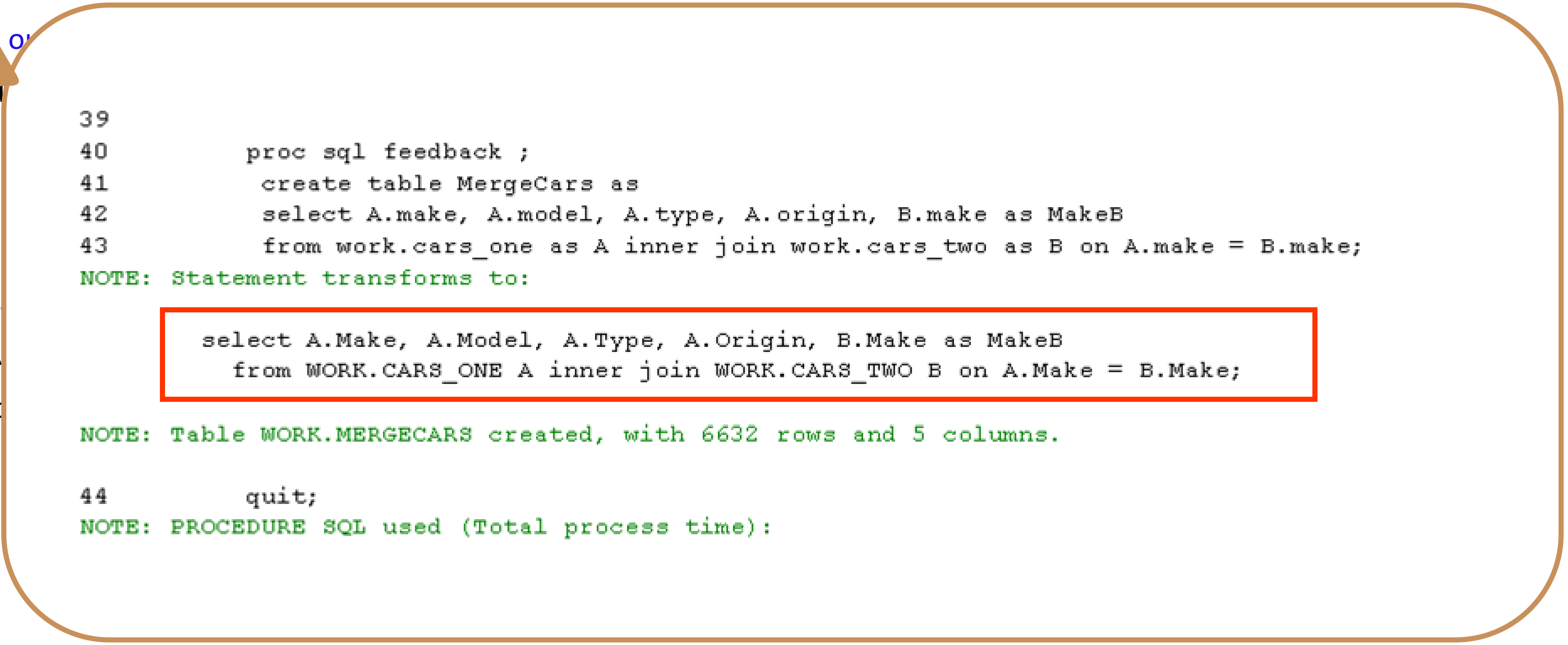
```
proc sort data = cars out = cars_one ;
        by make model type origin;
run;

proc sort ta cars out = c
        by odel ty
run;

proc sql _method ;
        create table
        select A.mak
        from work.c
quit;
```



```
        proc sql _method ;
         create table MergeCars as
         select A.make, A.model, A.type, A.origin, B.make as MakeB
         from work.cars_one as A inner join work.cars_two as B on A.make = B.make;

NOTE: SQL execution methods chosen are:

2                                                         The SAS System

     sqxcrta
         sqxjm
             sqxsort
                 sqxsrc( WORK.CARS_TWO(alias = B) )
             sqxsrc( WORK.CARS_ONE(alias = A) )
NOTE: Table WORK.MERGECARS created, with 6632 rows and 5 columns.
```

Give me some Feedback!

```
proc sort data = cars out = cars_one ;
          by m  e model type origin;
run;

proc sort   ta = c  s o
          by  mod
run;

proc sql feedback;
          create t
          select A
          from wo
quit;
```



```
39
40          proc sql feedback ;
41           create table MergeCars as
42           select A.make, A.model, A.type, A.origin, B.make as MakeB
43           from work.cars_one as A inner join work.cars_two as B on A.make = B.make;
NOTE: Statement transforms to:

        select A.Make, A.Model, A.Type, A.Origin, B.Make as MakeB
          from WORK.CARS_ONE A inner join WORK.CARS_TWO B on A.Make = B.Make;

NOTE: Table WORK.MERGECARS created, with 6632 rows and 5 columns.

44          quit;
NOTE: PROCEDURE SQL used (Total process time):
```

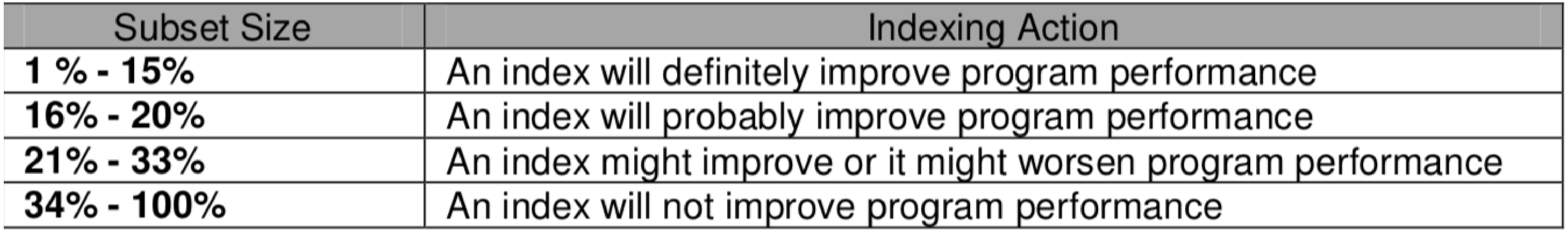
Indexes, Indexes, Indexes

- A SAS index is a physical file that is associated with a data file.
- It is based on the value of or many variables which are known as key or index variables.
- An index can be composed of:
  - One unique key variable
  - Many unique key variables (called composite keys)
  - One or many non unique key variables
- Multiple indexes can be created against the same data file.
- It speeds up the location of records in the data file.
- If indexed properly, no sorting the data file require.

SAS decides whether or not an index will be used!

When to Create an Index?

- For variables frequently used in WHERE clauses, WHERE data set options, or sub-setting IF statements (For specific SIN Search as example)
- Use SAS index only when the dataset is very large in size.



Subset Size	Indexing Action
1 % - 15%	An index will definitely improve program performance
16% - 20%	An index will probably improve program performance
21% - 33%	An index might improve or it might worsen program performance
34% - 100%	An index will not improve program performance

Use MSGLEVEL= (I) option to determine if an index is used.

Examples

```
proc sql;
       create unique index id on cands;
       create index prov on cands;
       create index geo on cands(prov county);
quit;
```

```
proc datasets library = work;
       modify cands;
              index create id / unique;
              index create prov;
              index create geo=(prov county);
run;
```

```
/* Create indexes via the Index dataset option */
data cands(index = (id /unique prov geo = (prov county)));
```

Just Passing Through

✓ The **PROC SQL PASS-THROUGH** facility allows you to send statements directly to a DBMS (in SQL syntax) rather than being executed by PROC SQL.

✓ Data movement is minimized when it is subset by the DBMS before being sent to PROC SQL!

✓ A good example is **the interaction of SAS with IBM Pure Data Analytics (PDA) – Netezza**

SQL Implicit Pass-Through *

```
LIBNAME NZ netezza  server = XXXX    Database = XXXX
                    schema = XXXX    authdomain = XXXX;
```

```
proc sql ipassthru;
   create table work.IncomeData_SQL as
   select *
   from NZ.IncomeDat
   where province = 'SK'
   order by taxyear, province, sin ;
quit;
```

```
data IncomeData_DS;
   set NZ.IncomeData
                    (where = (province = 'SK'));
proc sort;
   by taxyear province sin;
run;
```

?

```
data IncomeData_DS;
   set NZ.IncomeData ;
   where province = 'SK';
proc sort;
   by taxyear province sin;
run;
```

```
proc sql passtrough;
   connect to netezza as db(server = XXXX
   database = XXXX authdomain = XXXX);

   execute (
      create table IncomeData_EPT as (
      SELECT * FROM IncomeData *PDA SQL GOES HERE */)
   ) by db;
quit;
```

Table created directly on the PDA

[* SAS Online Documentation](#)

What's Next - Conclusion

Is there a ROADMAP for High Volume of Data ?

- In-memory with SAS Viya and MPP ? Hadoop Cluster ? Kubernetes ?
- **Clean**! **Clean**! **Clean**! Make sure your System/Code are cleaned
- Use **simple tips / code** to work effectively and intelligently – Keep It Simple Stupid!
- Avoid redundancy
- Make a good dosage of **Elegance** and **Functionality**
- Do not reinvent the wheel – **Productivity**!!!
- Think of **Maintenance** for SAS programs
- Use **Google** as your Best Friend.

Re-read that presentation and find out how you can apply SOME of those TIPS

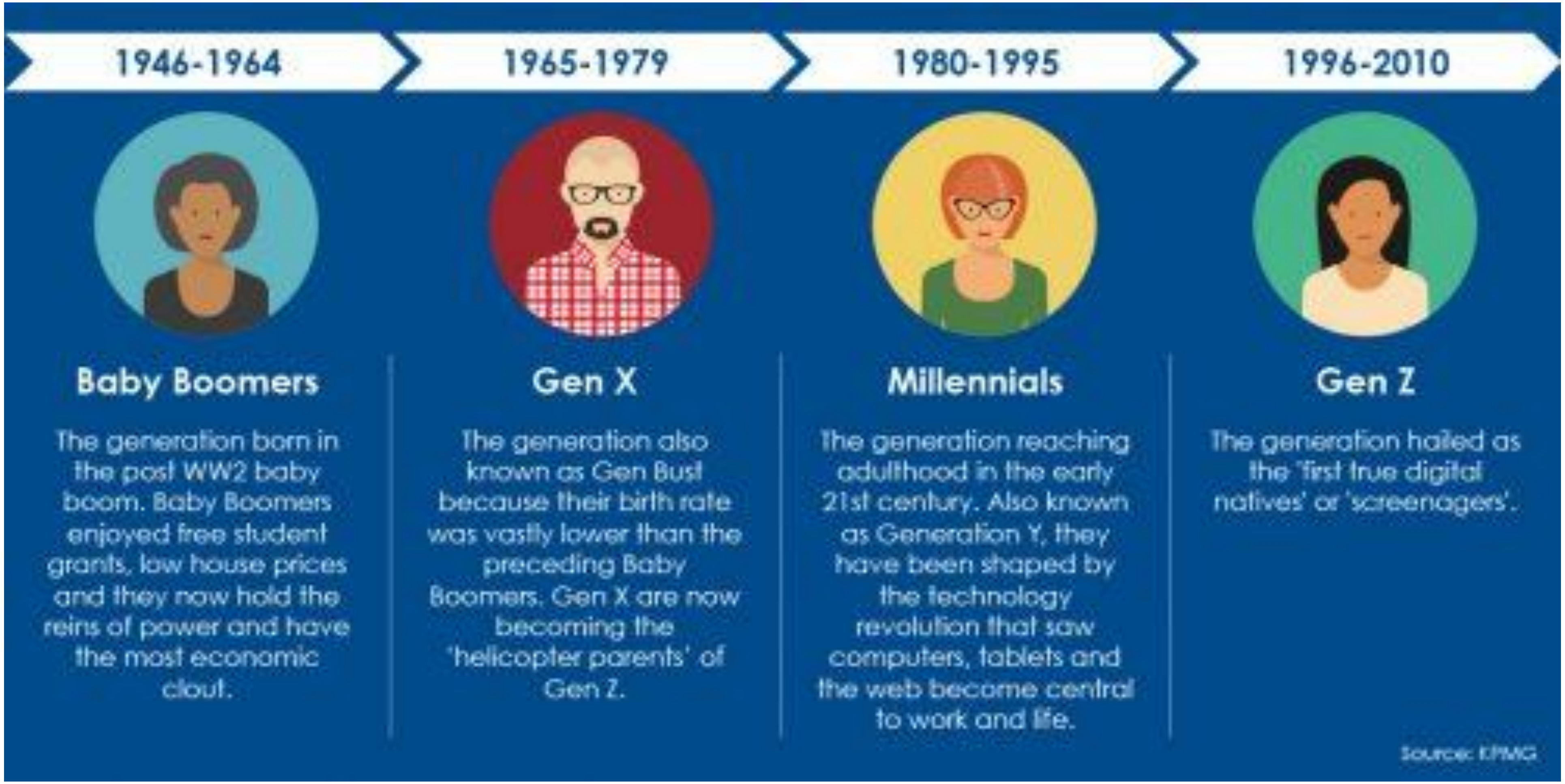
Bits and B(i)ytes : The Snack

A PREVIEW OF NEXT OTTAWA USER GROUP

BITS N' BYTES – TOPIC # 1:

DATA SCIENCE, PERSONAS AND ETHICS

The Workforce Composition



1946-1964

Baby Boomers

The generation born in the post WW2 baby boom. Baby Boomers enjoyed free student grants, low house prices and they now hold the reins of power and have the most economic clout.

1965-1979

Gen X

The generation also known as Gen Bust because their birth rate was vastly lower than the preceding Baby Boomers. Gen X are now becoming the 'helicopter parents' of Gen Z.

1980-1995

Millennials

The generation reaching adulthood in the early 21st century. Also known as Generation Y, they have been shaped by the technology revolution that saw computers, tablets and the web become central to work and life.

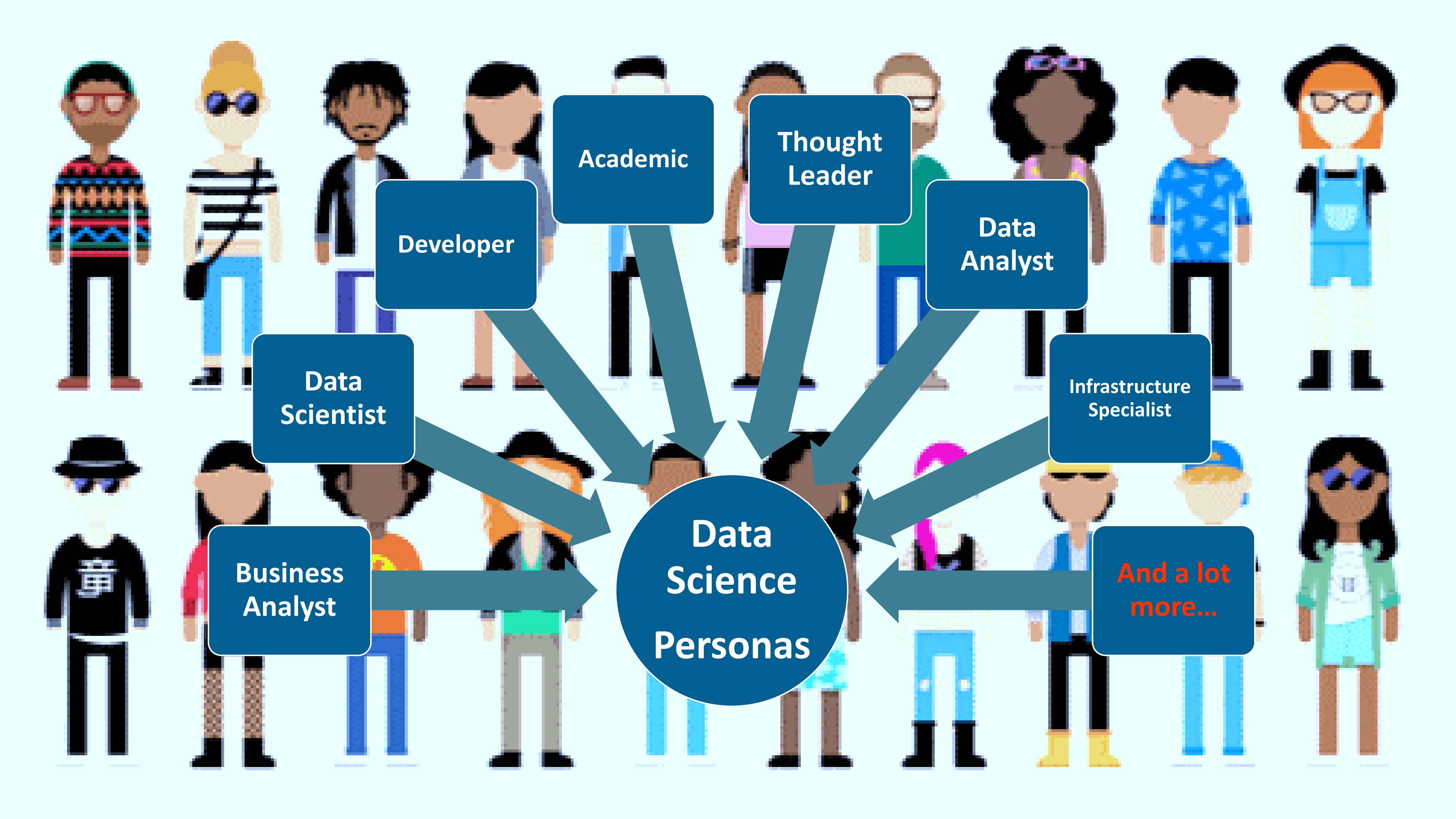
1996-2010

Gen Z

The generation hailed as the 'first true digital natives' or 'screenagers'.

Source: KPMG

Source : https://www.jcinstitute.com/top-fashion-brands-that-millennials-gen-x-gen-z-trust/



Data Science Personas

Academic

Thought Leader

Developer

Data Analyst

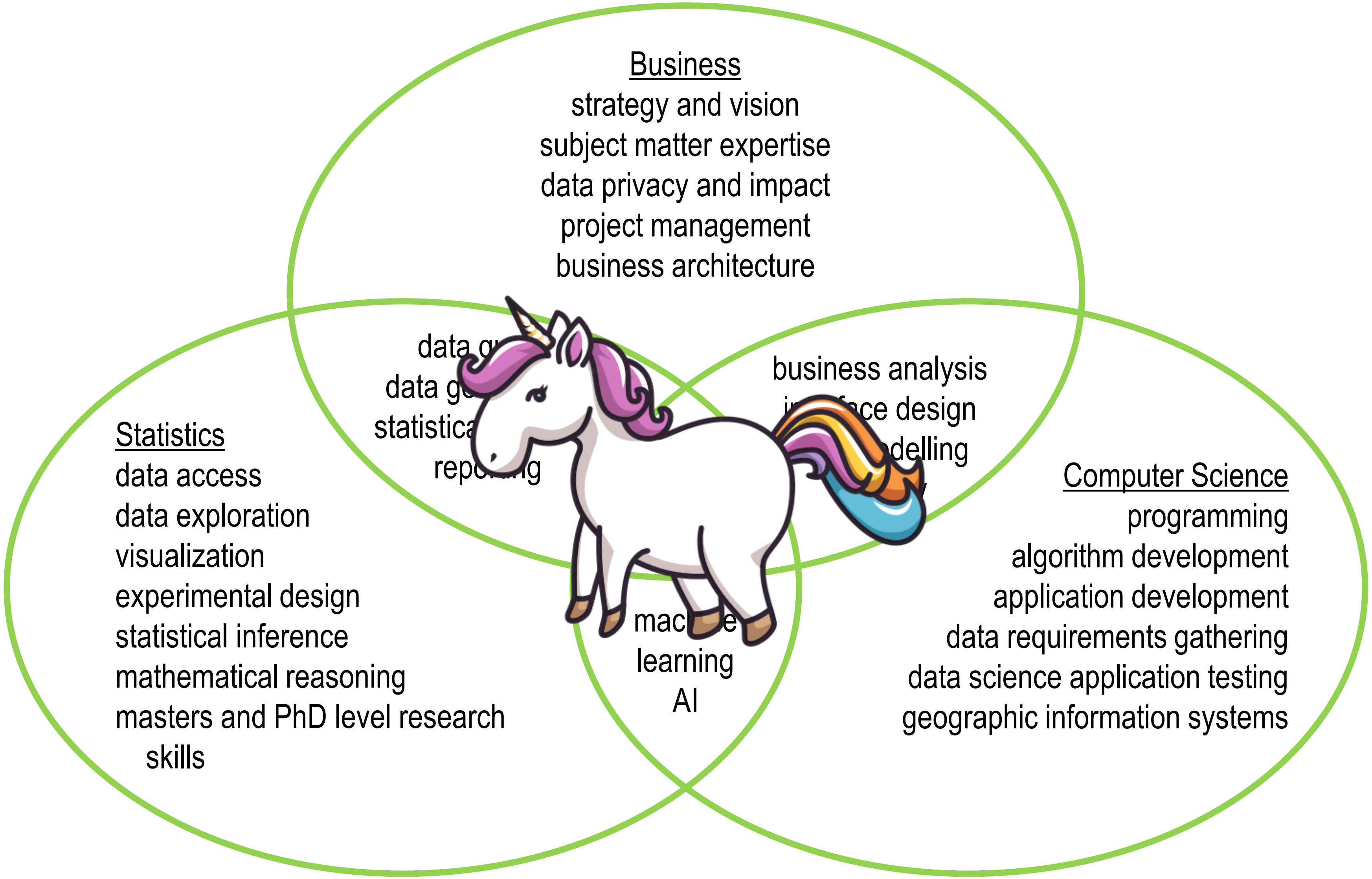
Data Scientist

Infrastructure Specialist

Business Analyst

And a lot more…

Data Science Skills



<u>Business</u>
strategy and vision
subject matter expertise
data privacy and impact
project management
business architecture

data g
data g
statistica
reporting

business analysis
interface design
modelling
y

<u>Statistics</u>
data access
data exploration
visualization
experimental design
statistical inference
mathematical reasoning
masters and PhD level research skills

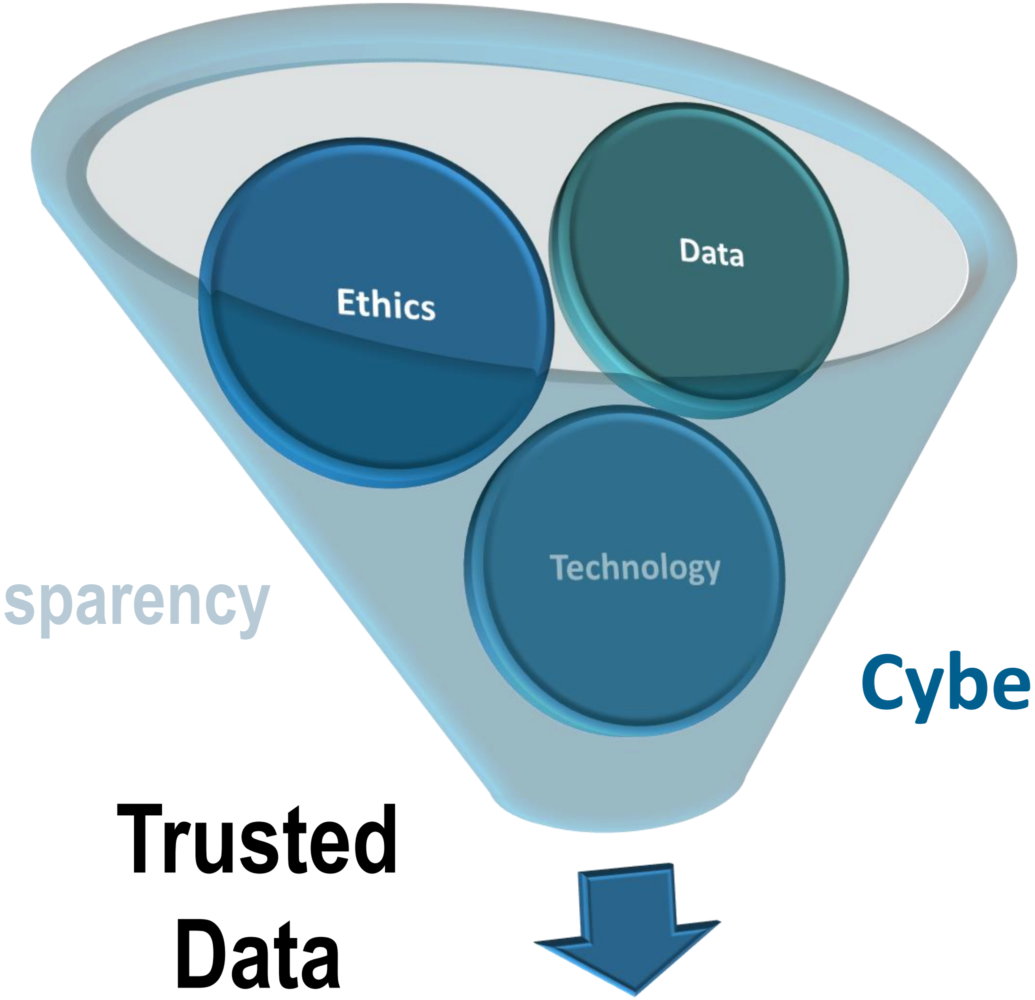
machine
learning
AI

<u>Computer Science</u>
programming
algorithm development
application development
data requirements gathering
data science application testing
geographic information systems

Trusted
Employees

Data Accuracy

Impact of
Assessment



Practical
Concerns

Trusted
Algorithms

Transparency

Cybersecurity

Transparency

Laws and
Regulation

Trusted
Data

Property Rights and
Patents

Accidental
Disclosure

Let's Talk about
Integrity

Data
Privacy

Technology, Ethics and Human Rights

What is Technology and Ethics?

Technology Ethics is the application of ethical thinking to the practical concerns of **technology**. The reason **technology ethics** is growing in prominence is that new **technologies** give us more power to act, which means that we have to make choices we didn't have to make before.

Brian Patrick Green

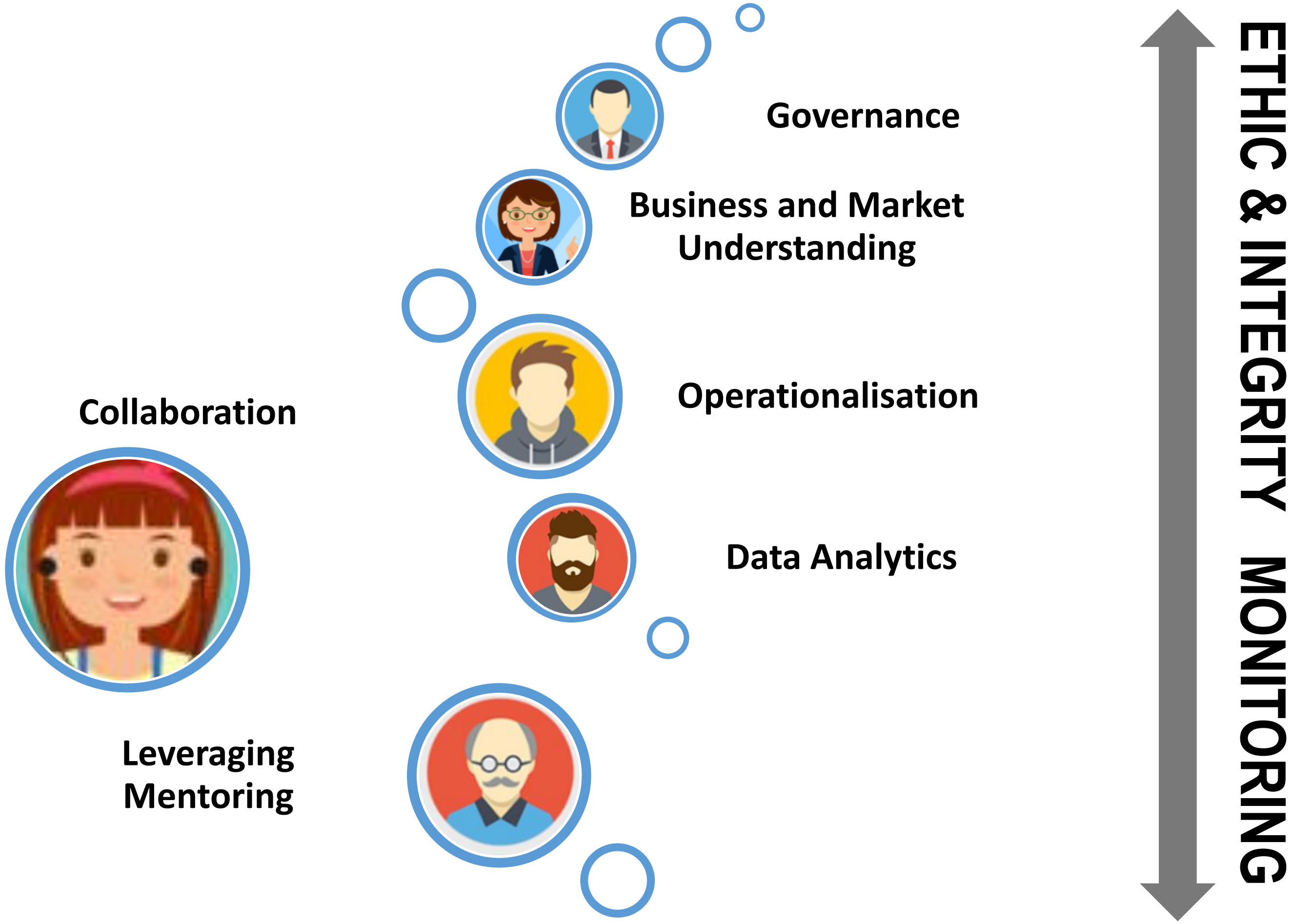
Director of Technology Ethics

Markkula Center for Applied Ethics

Santa Clara University

> "Inventions like pesticides or GMOs can reduce hunger but can also cause unexpected harm to people and the environment."
>
> **Sheila Sen Jasanoff**
>
> Pforzheimer Professor of Science and Technology Studies,
> Harvard University

Ethic & Integrity as a Daily Panacea



Governance

Business and Market Understanding

Operationalisation

Collaboration

Data Analytics

Leveraging Mentoring

ETHIC & INTEGRITY MONITORING

Organisational Trusted Results VS Imperfect Modelling

Questions

Acknowledgments

Silvana Kairouz

Mireille Éthier

Sheri Albers

Philippe Bélanger

Anne-Marie Johnson

André Patry

References

- Droogendykb H. and Dosani F, Joining Data: Data Step Merge or SQL?, paper 178-2008, SGF 2008
- First, S., & Schudrowitz, T. Arrays Made Easy: An Introduction to Arrays and Array Processing. SUGI 30, Paper 242-30.
- Lafler, K. P. Top Ten SAS® Performance Tuning Techniques. *MWSUG 2012.*
- Landge, R. (2016). SAS Techniques for Managing Large Datasets. *PharmaSUG, Paper QT23.*
- Malachy J. Foley, MERGING vs. JOINING: Comparing the DATA Step with SQL, paper 249-30, SUGI 30
- Novotny, J., & Bramley, M. Stuff We All Do: Mistakes We've Made That You Won't Have To. *MWSUG-2010, Paper 119-2010.*
- Palmer L, Using SAS Views and SQL Views, State of California, Richmond, CA
- Richardson B., Optimize Your Delete, paper 022-2013, SAS Institute Inc., Cary, NC, SGF 2013
- Raithel, M. A. Creating and Exploiting SAS® Indexes. *SUGI 29, Paper 123-29.*
- Samudral, K., & Giddings, G. M. BY'S NOTSORTED OPTION. *NESUG 2006.*
- Squillace, J. *Flexibility by Design: A Look at New and Updated.* SAS Technical Support, Cary, NC.
- Sridharma, S. How to Reduce the Disk Space Required by a SAS® Data Set. *NESUG 2006.*
- Troy Martin Hughes, Sorting a Bajillion Records: Conquering Scalability in a Big Data World, paper-11888-2016, SGF 2016
- *Working Efficiently with Large SAS® Datasets.* Quanticate The Clinical Data Experts
- https://www.sas.com/en_us/insights/data-management/what-is-etl.html#etl-importance
- https://communities.sas.com/t5/SAS-Data-Management/How-To-Work-Efficiently-with-Very-Big-SAS-Datasets/td-p/141832
- https://www.sas.com/en_us/insights/articles/data-management/data-lake-and-data-warehouse-know-the-difference.html

Contact

KARINE DESILETS

Senior Quantitative Analyst
Innovation and Analytics Division
Canada Revenue Agency
555 Mackenzie Avenue
Ottawa ON K1A 0L5
karine.desilets@cra-arc.gc.ca