Wrangling and Pipelines for Data Handling

8th - 9th April

Hello!

Richard Strange

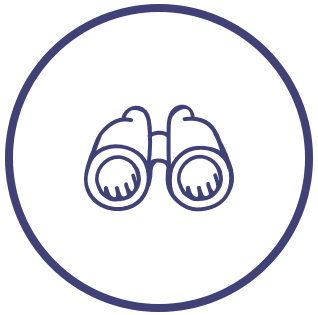
Environmental Research DTP

Volcanic Seismology and AI

Data Scientist and Data Architect

Day 1 Themes

Pipelines



Profiling

Cleaning

Day 1 outline

1. What do data pipelines look like?
2. Where do pipelines go wrong?
3. Breakout session: data flow challenges

Lunch!

4. Building blocks
5. Data Profiling and Cleaning
6. Data wrangling lab

Course Objectives

Understand the basics of data pipeline design

Understand the options available in building a pipeline

Understand the likely errors in data, how to spot them and fix them

Understand the advantages of code that pre-emptively reports failures

Understand the purpose and function of new technologies for handling and finding data

Able to begin constructing pipelines

Able to write functional ELT code

Able to interact with online data sources

Course Objectives

Understand the basics of data pipeline design

Understand the options available in building a pipeline

Understand the likely errors in data, how to spot them and fix them

Understand the advantages of code that pre-emptively reports failures

Understand the purpose and function of new technologies for handling and finding data

Able to begin constructing pipelines

Able to write functional ELT code

Able to interact with online data sources

You can make an informed decision with your data

House rules…

House Rules - I expect you to...

Stop me whenever you want

No such thing as a silly question, and there is no such thing as a badly timed question

Stop underestimating your knowledge

It's impossible to avoid working with data. Just because you might not recognise the terminology doesn't mean you don't already have experience

Stop worrying

This course is a whistle-stop tour of the topic, not a assessed lecture series.

Section 1

Data Pipelines

Modularity

- There are a thousand ways to achieve the same thing
- No one way is correct...

Modularity

- There are a thousand ways to achieve the same thing
- No one way is correct...
- ... but some ways are more correct than others

What is a data pipeline?

"a set of data processing elements connected in series, where the output of one element is the input of the next one [...] some amount of buffer storage is often inserted between elements"

-WIKIPEDIA

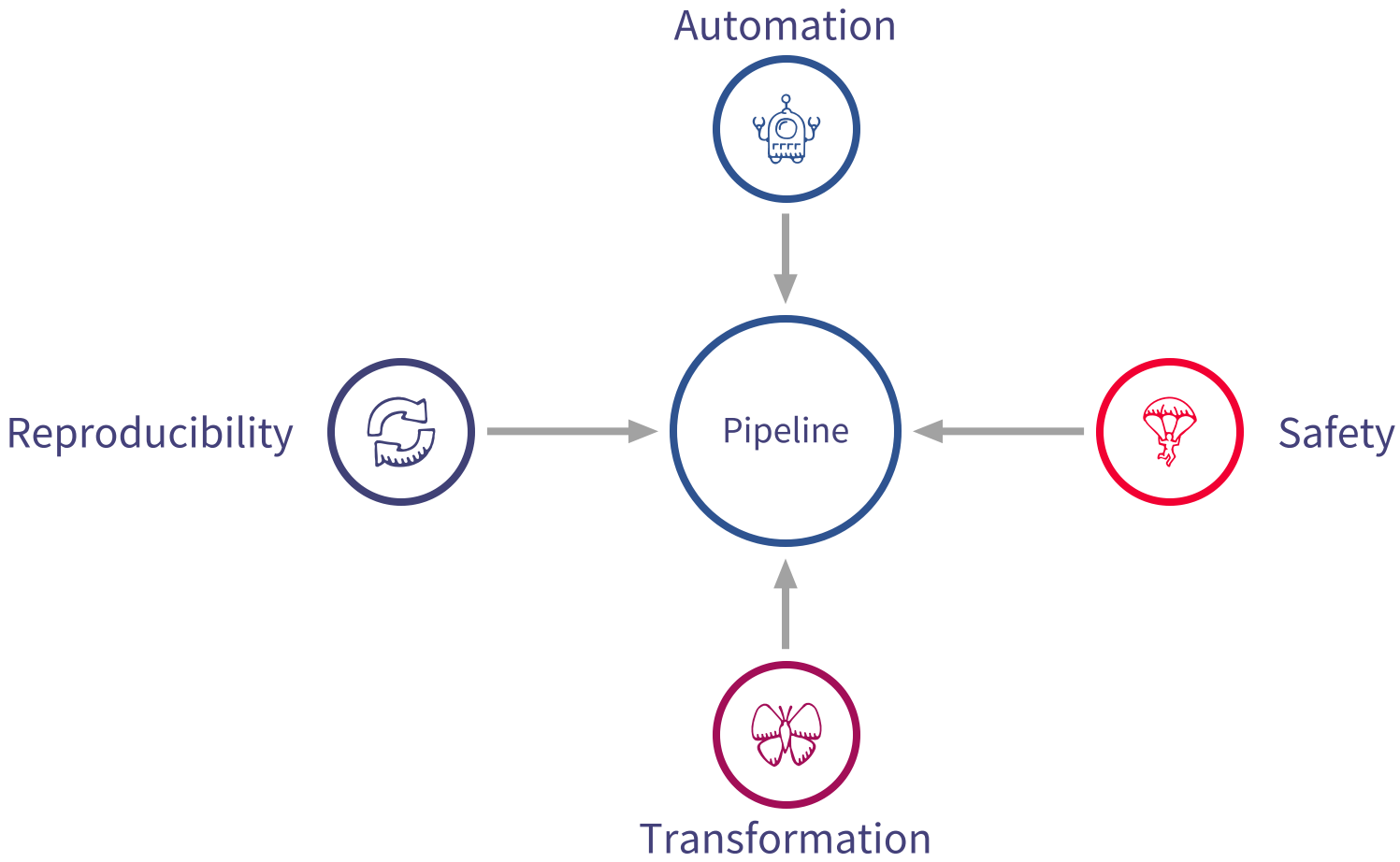
"eliminates many manual steps from the process and enables a smooth, automated flow of data from one station to the next. It starts by defining what, where, and how data is collected. It automates the processes involved in extracting, transforming, combining, validating, and loading data for further analysis and visualization. It provides end-to-end velocity by eliminating errors and combatting bottlenecks or latency."

-ALOOMA

"A data pipeline is a set of actions that extract data (or directly analytics and visualization) from various sources. It is an automated process: take these columns from this database, merge them with these columns from this API, subset rows according to a value, substitute NAs with the median and load them in this other database"

-MEDIUM

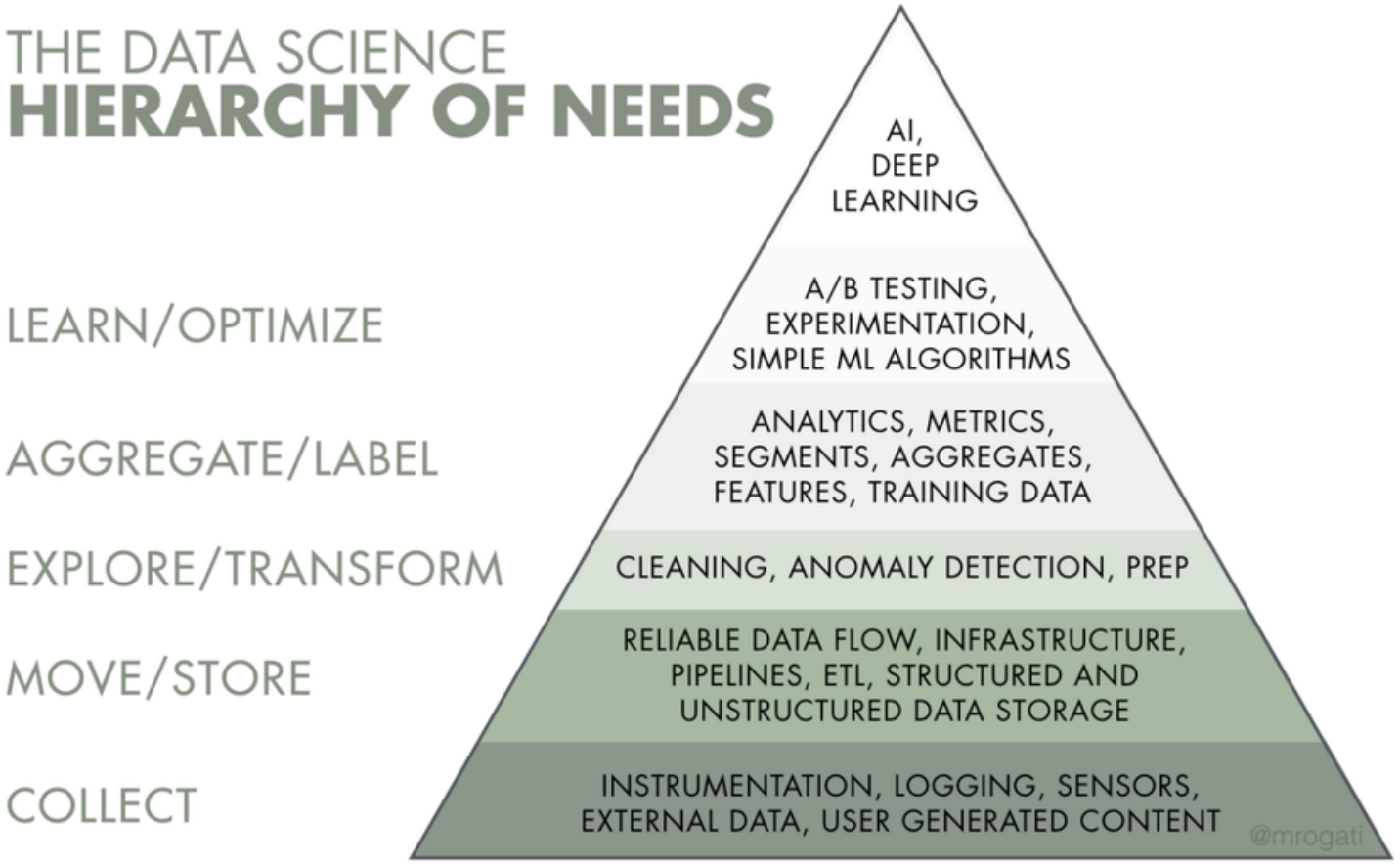
What's the point?



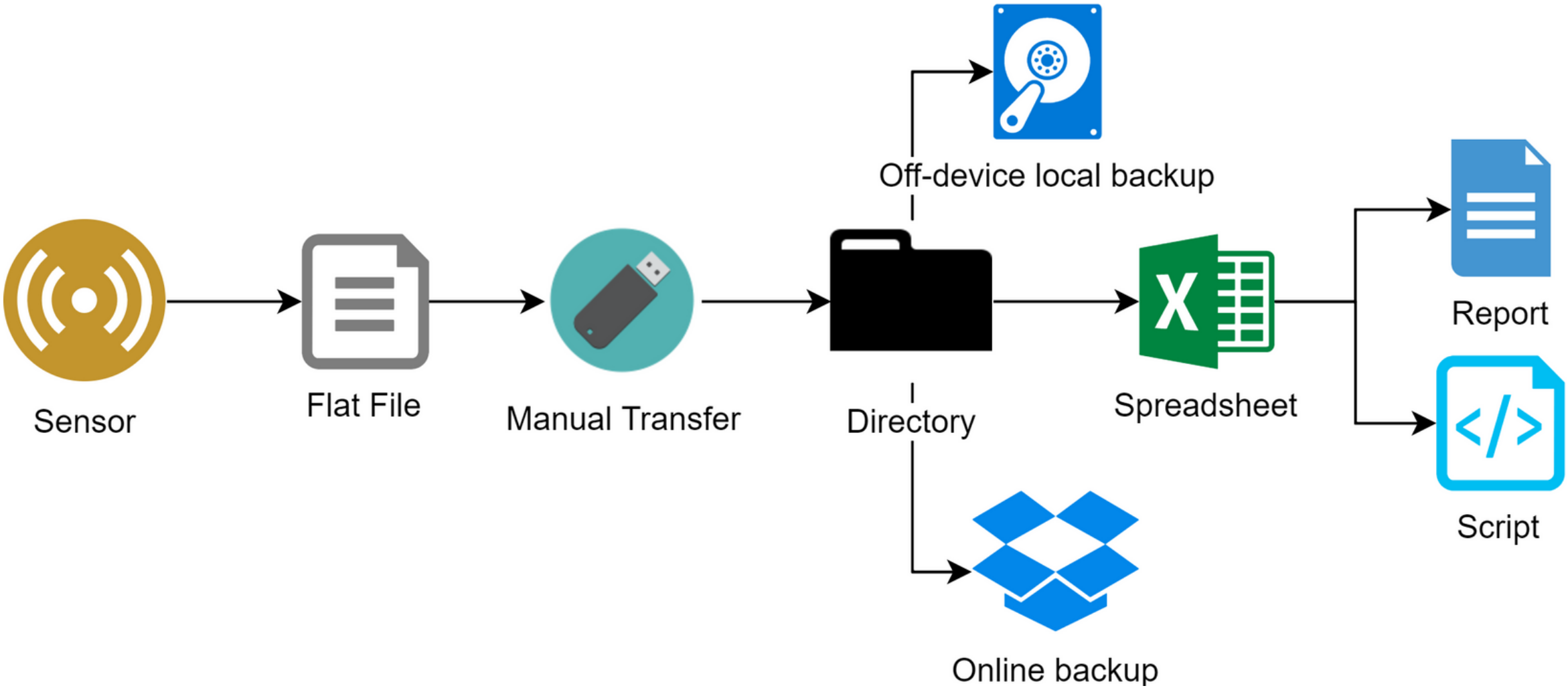
Pipeline Reliability

- Data should not arrive duplicated
- Data should not arrive incomplete
- Updated data can be distinguished
- Nothing fails quietly
- Staging catches data, even in partial failure
- Different sources can merge safely

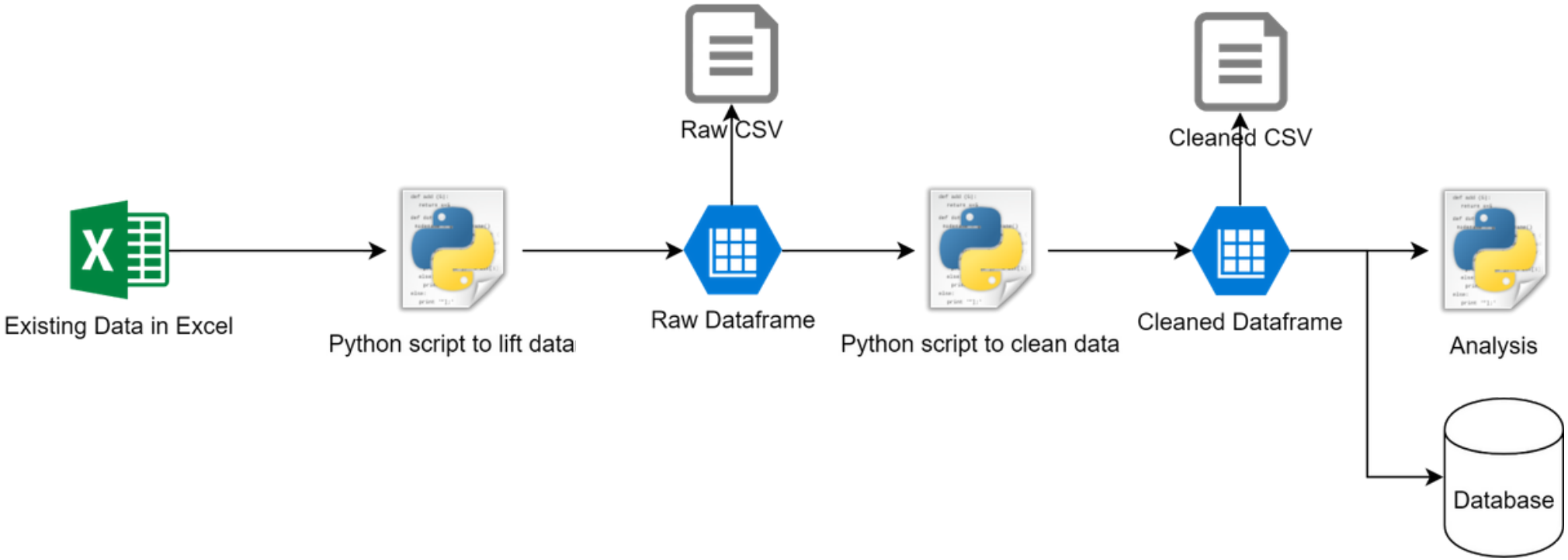
Work your way upwards



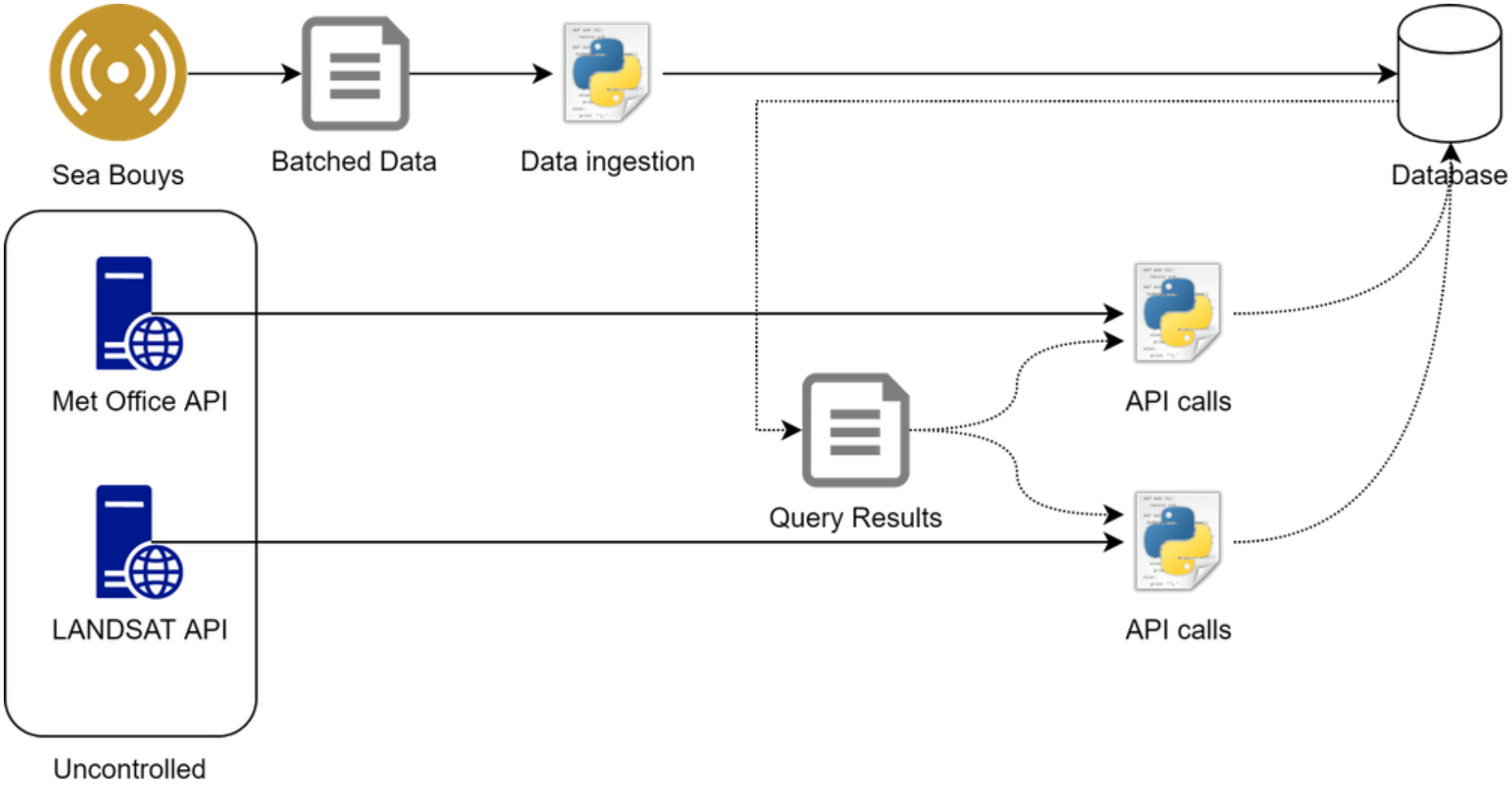
Basic Data flow



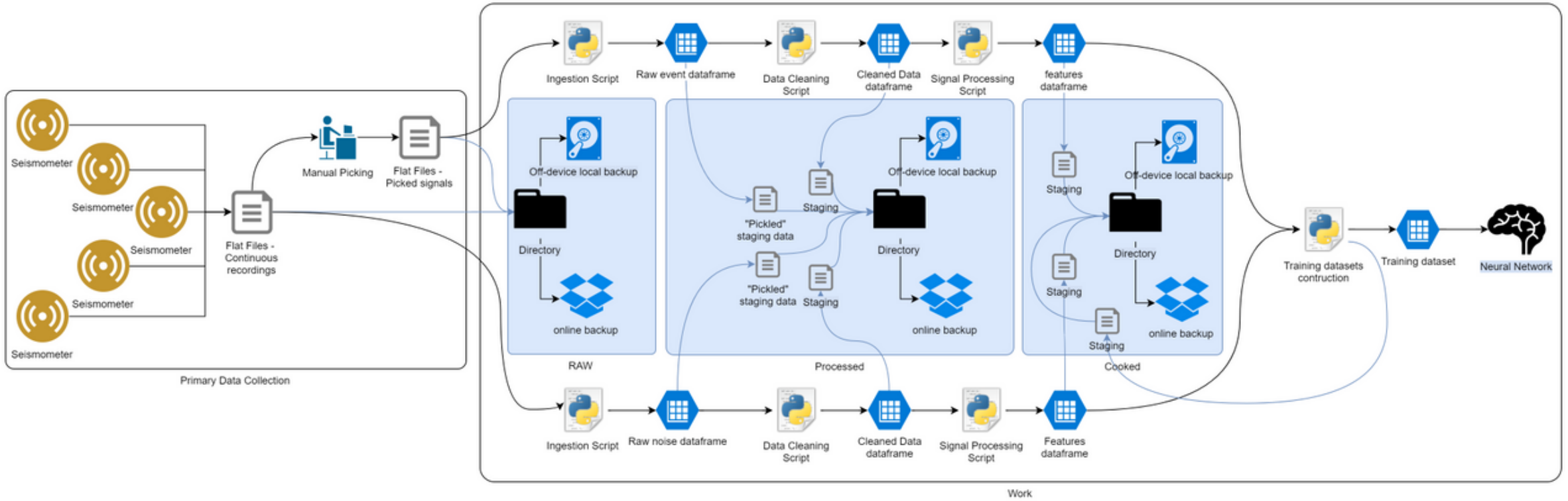
Excel Workflow



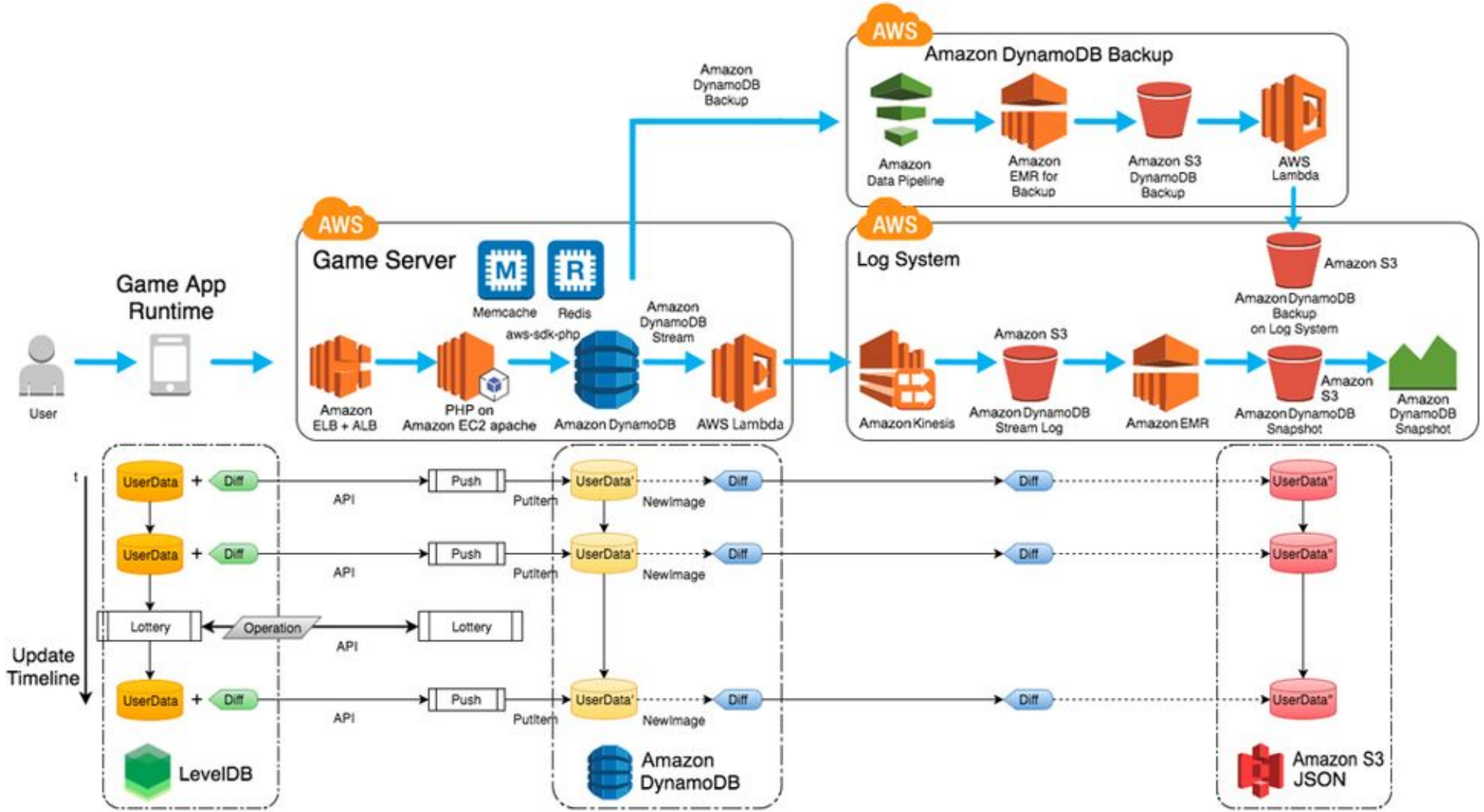
API Workflow



Working seismology pipeline

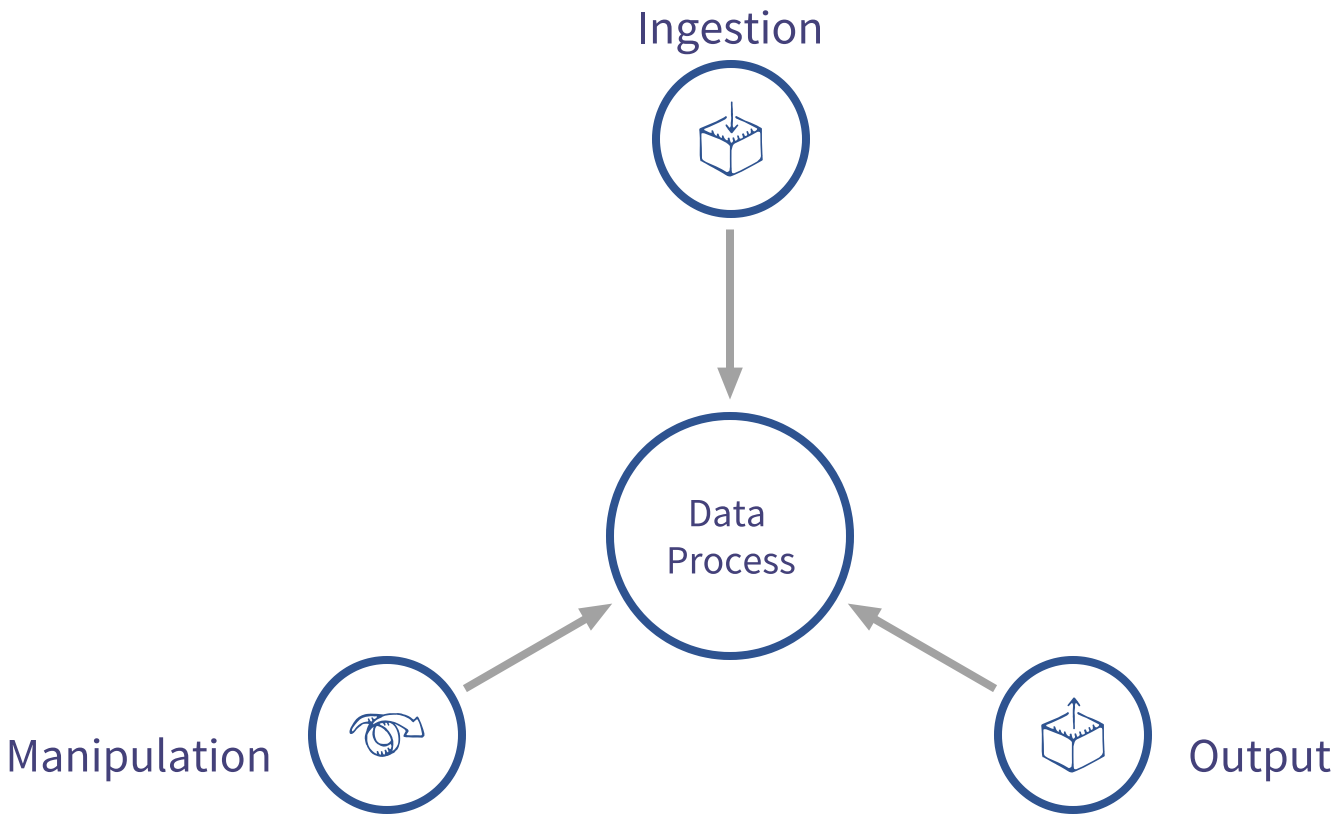


Simple commercial pipeline

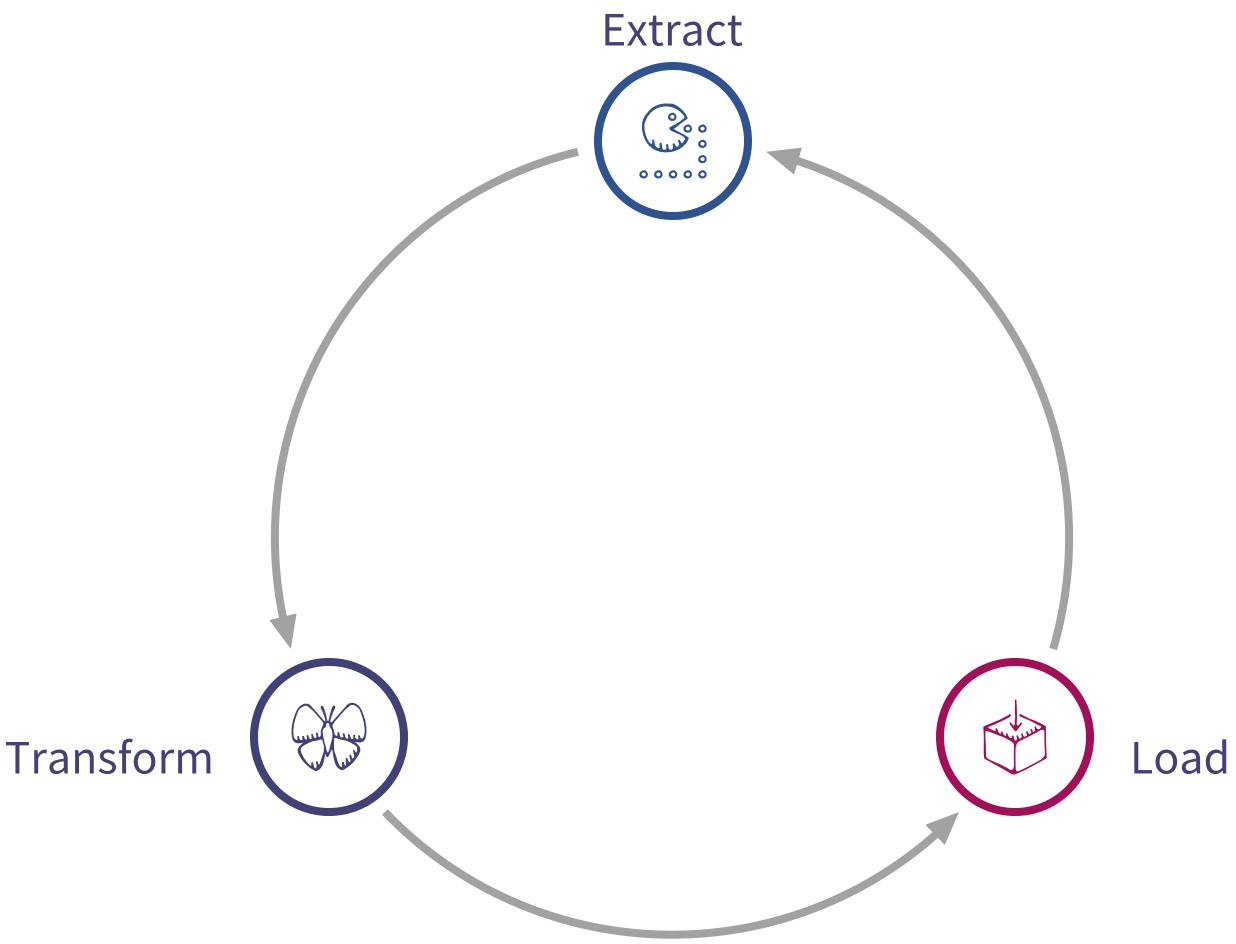


What do data processes do?

Purposes of data handling



ETL

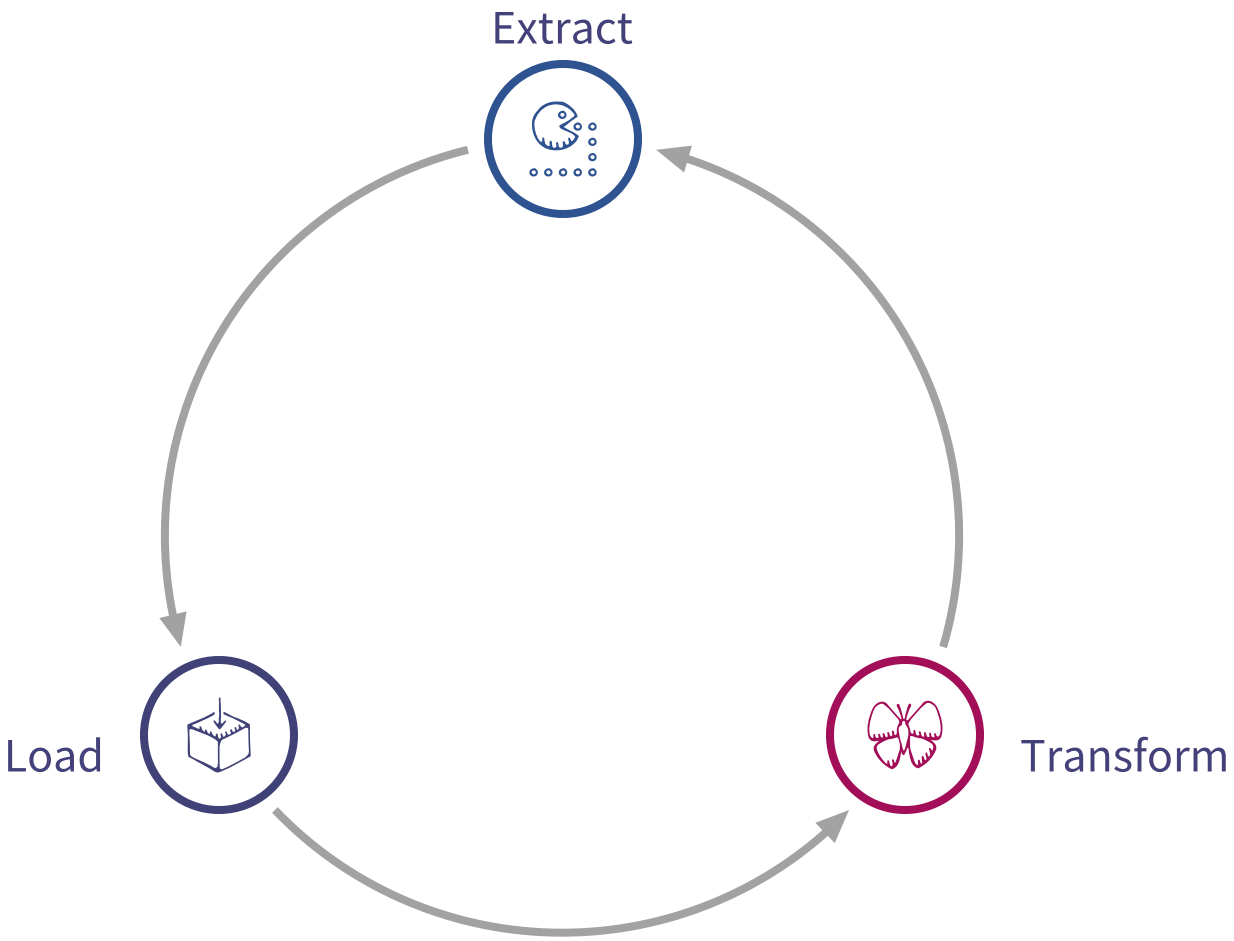


Extract - pull data from its source

Transform - change the data to a suitable state

Load - put the fixed data into an accessible location

ELT



ETL - Raw data exists in the source

ELT - Raw data exists in your system

Tenets

Keep it raw

The rawest form of your data should always be kept stored, even if it is a duplicate of data elsewhere.

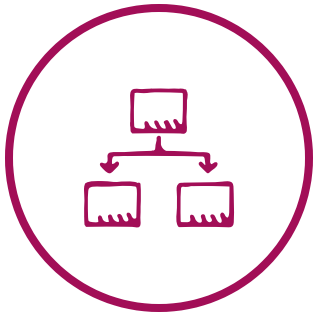
If anything goes wrong the rest of your data can be worked from first principles from the raw store.

Keep it simple

Every step in a flow should aim to only do one major task.

Every step in a script should aim to do one minor task.

Every staging file should aim to record a change to the underlying data.



Keep it copied

Intermediate data from script or spreadsheet stages can be dumped into flat files.

Being able to evaluate every output makes diagnosing issues and testing steps simple, keeping you sane.

Break

What do you call a group of stormtroopers playing monopoly?

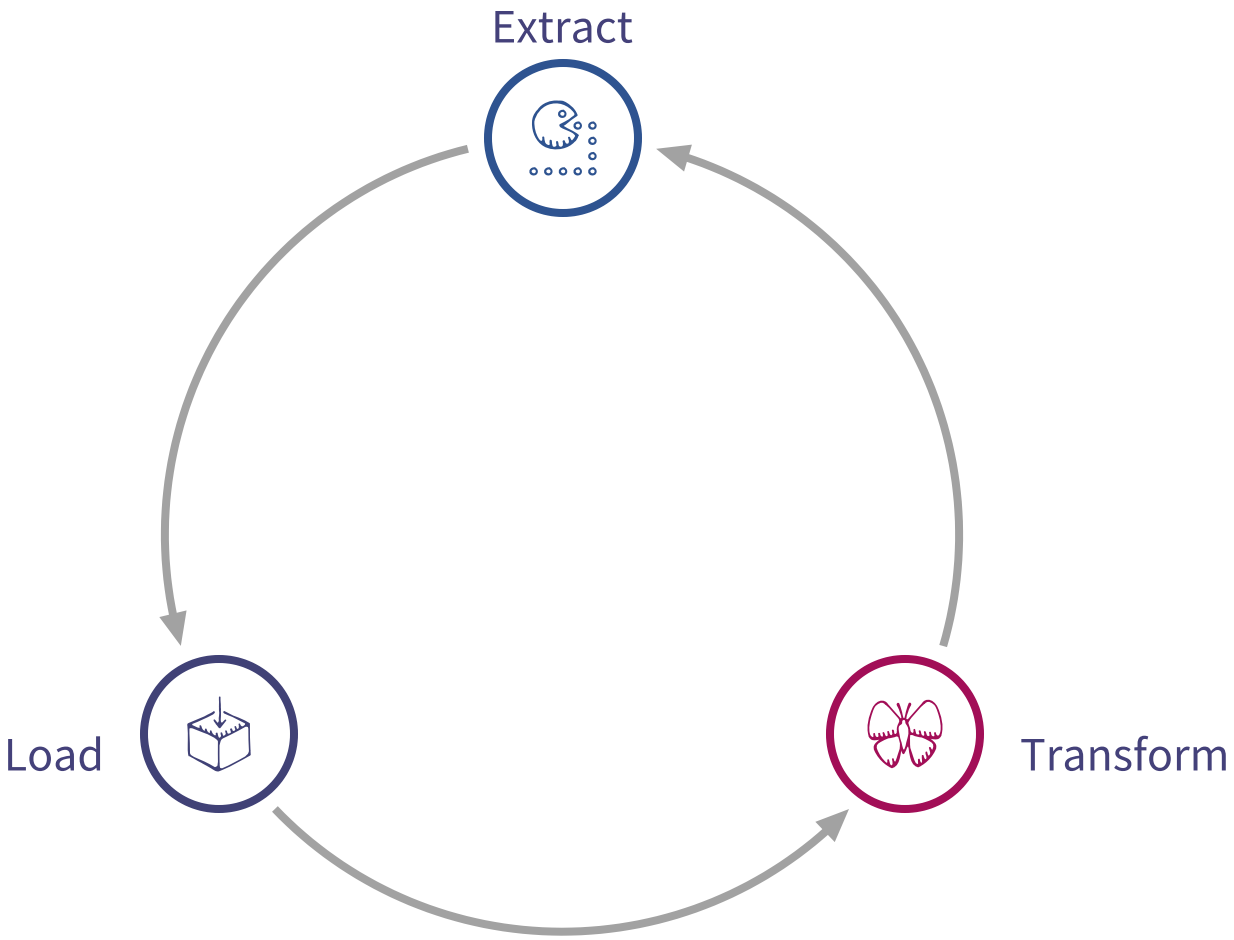
What do you call a group of stormtroopers playing monopoly?

Game of Clones

Section 2

Pitfalls in data management

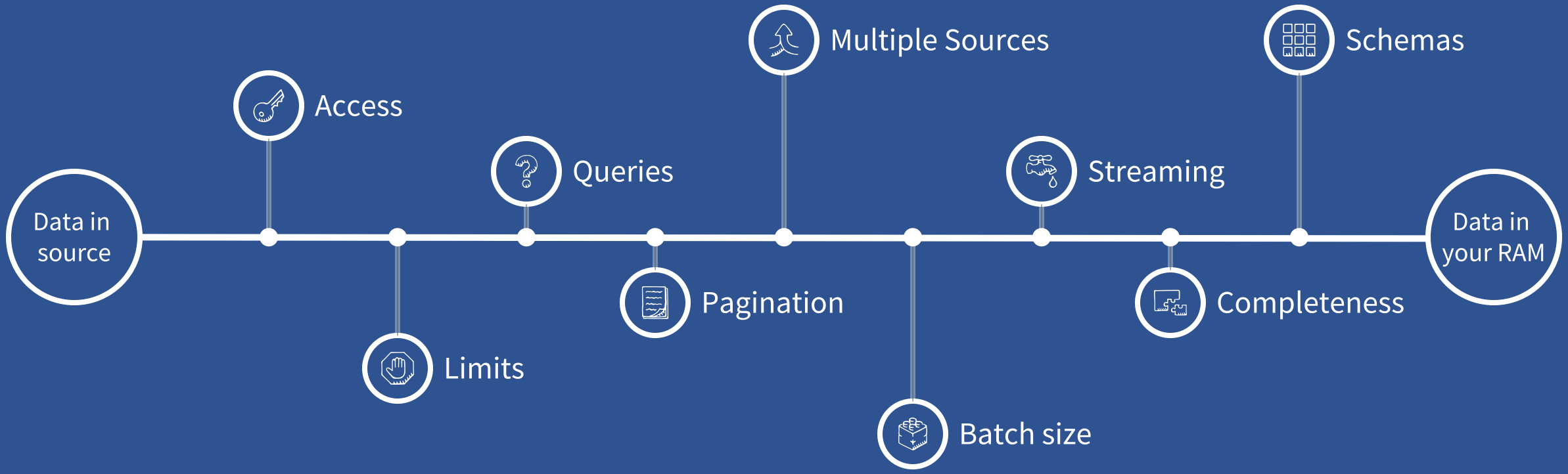
ELT



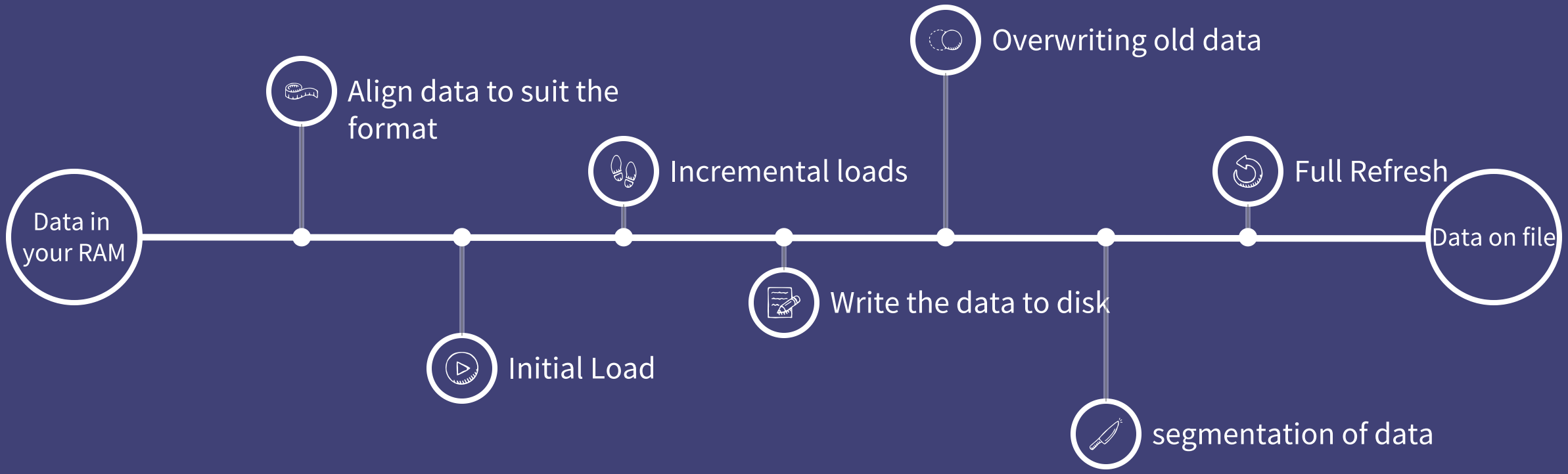
ETL - Raw data exists in the source

ELT - Raw data exists in your system

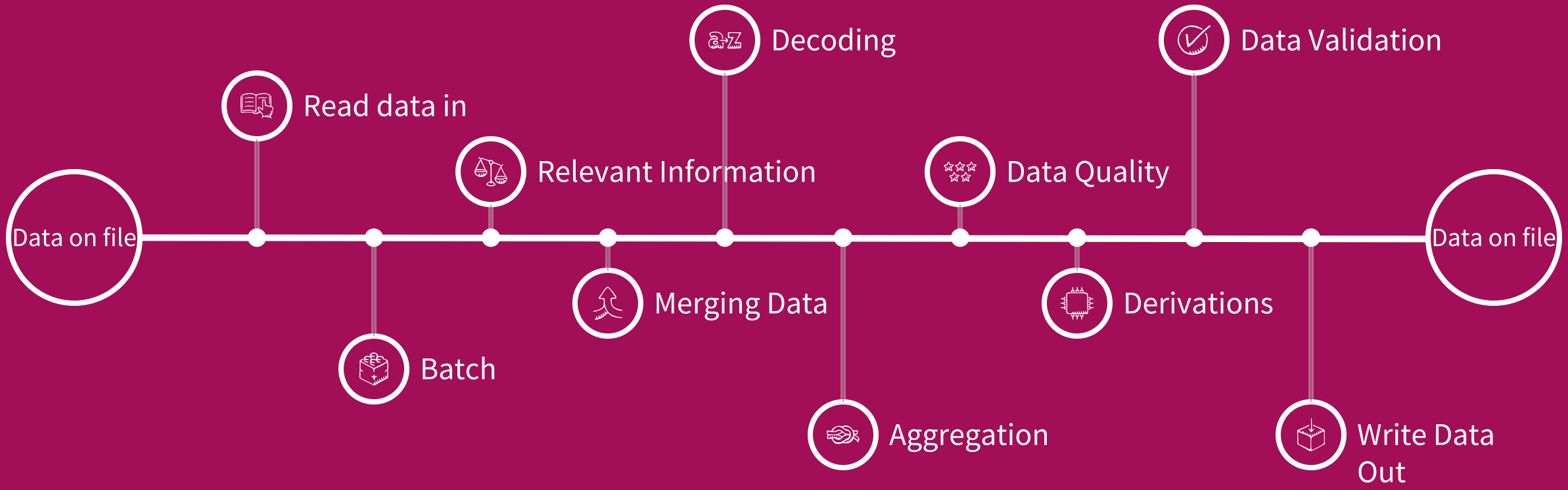
Extract


Data in source
Access
Limits
Queries
Pagination
Multiple Sources
Batch size
Streaming
Completeness
Schemas
Data in your RAM

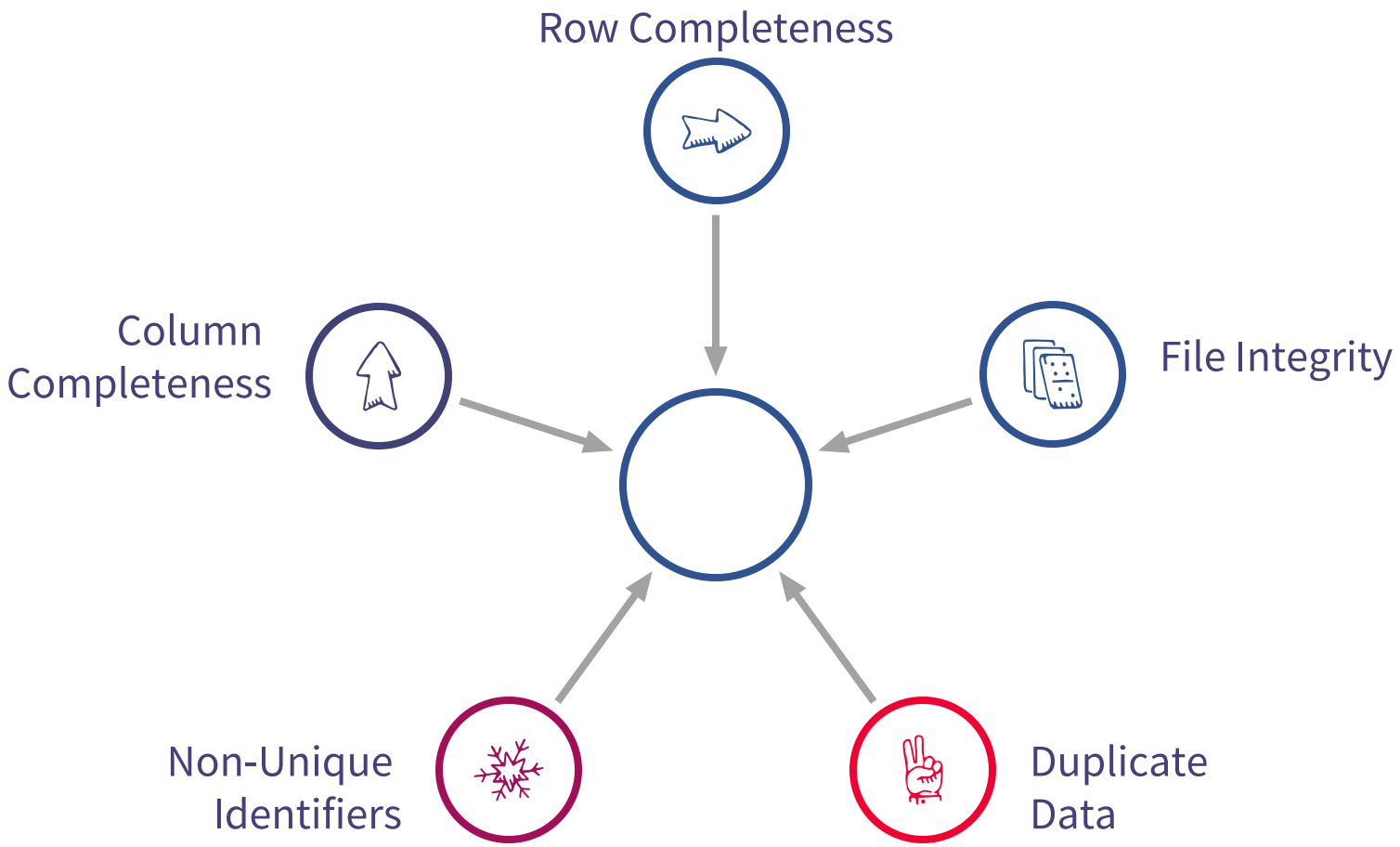
Load


Data in your RAM
Align data to suit the format
Initial Load
Incremental loads
Write the data to disk
Overwriting old data
segmentation of data
Full Refresh
Data on file

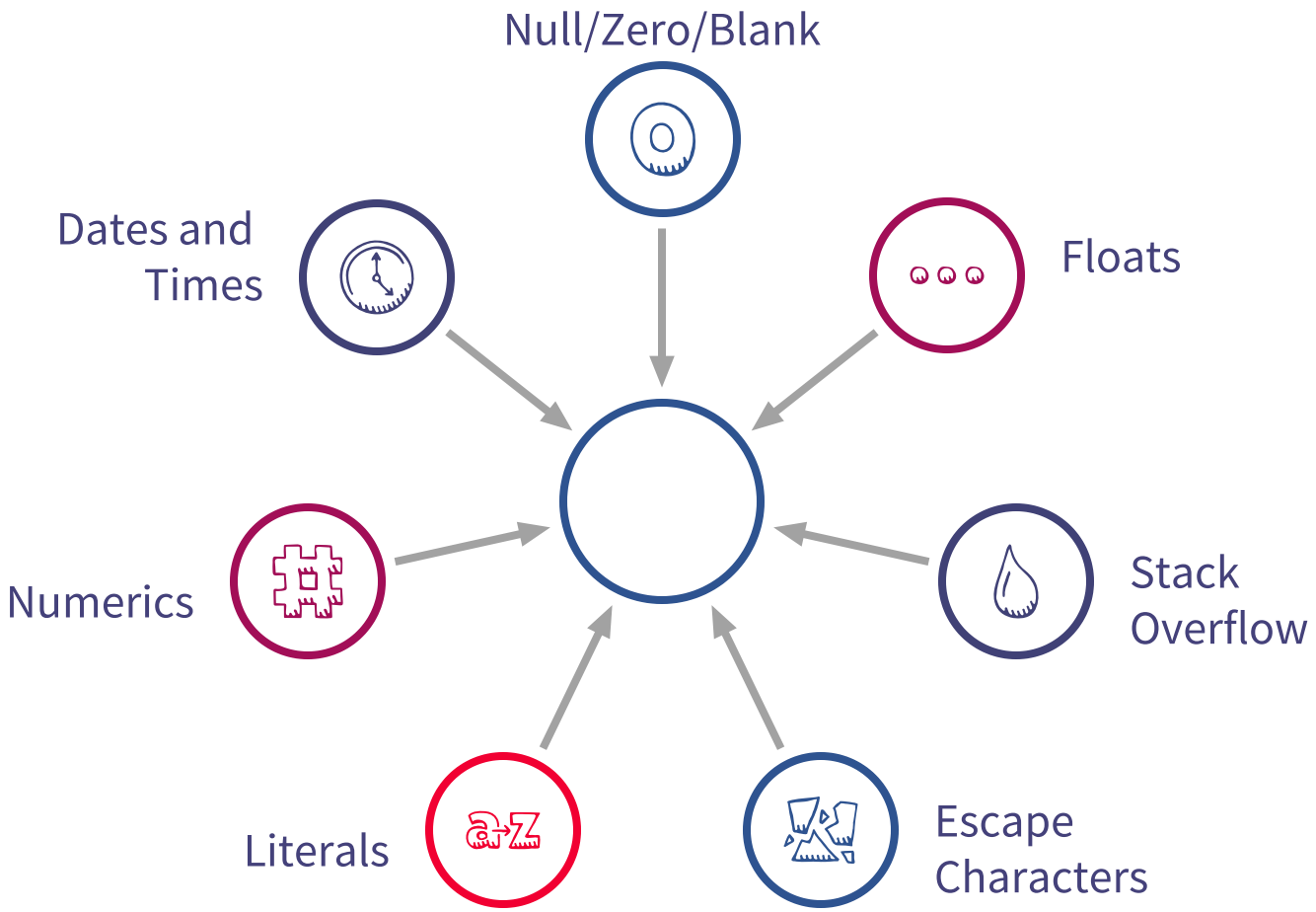
Transform


Data on file
Read data in
Batch
Relevant Information
Merging Data
Decoding
Aggregation
Data Quality
Derivations
Data Validation
Write Data Out
Data on file

Out-of-cell issues

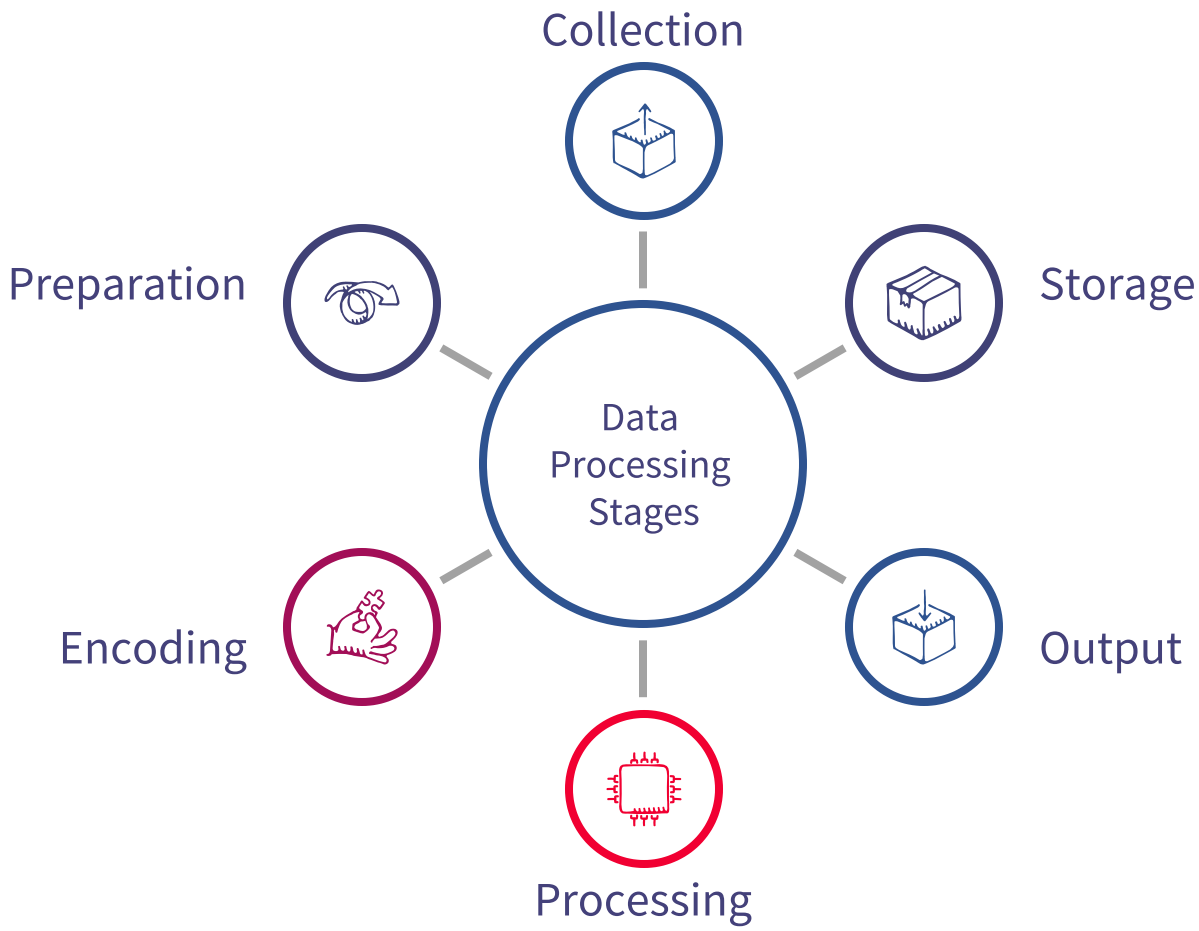


In-cell issues



Your turn!

Areas to consider...



Lunch

What did the Lego Alien say?

What did the Lego Alien say?

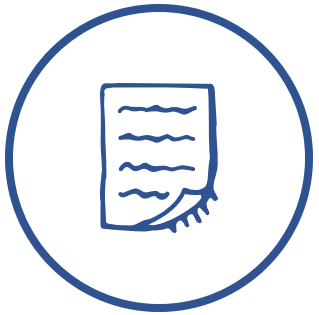
I come in pieces

Section 3

ELT Building Blocks


INHERITED
TRANSFORM
EXTRACT
VALIDATE
SHELL
LOAD
SERVER
CLOUD
SCHEMA
PRIMARY
CYCLE
PUBLISH
SQL
PANDAS
FOREIGN
VALUE
HYBRID
AZURE
DATA
SORTING
BUILD
JOINS
PIPING
API
PYTHON
INTEGRITY
TESTING
LOGGING
R
LEFT
MERGES
OUTER
SCALABILITY
KEY
ROLLUP
TABLE
SILO
WAREHOUSE
STAGING
SCHEMA
BASH
TRANSPOSE
ROW
GCP
PIVOT
AUDIT
CELL
LOOKUP
AWS
INNER
AGGREGATES
QUERY
DATABASE
DEDUPLICATION
RULES
BUSINESS
INHERITED
ARCHITECTURE
RIGHT
LOADTIME
WAREHOUSE
DISAGGREGATES
COLUMN
AUTOMATED

Technique Areas



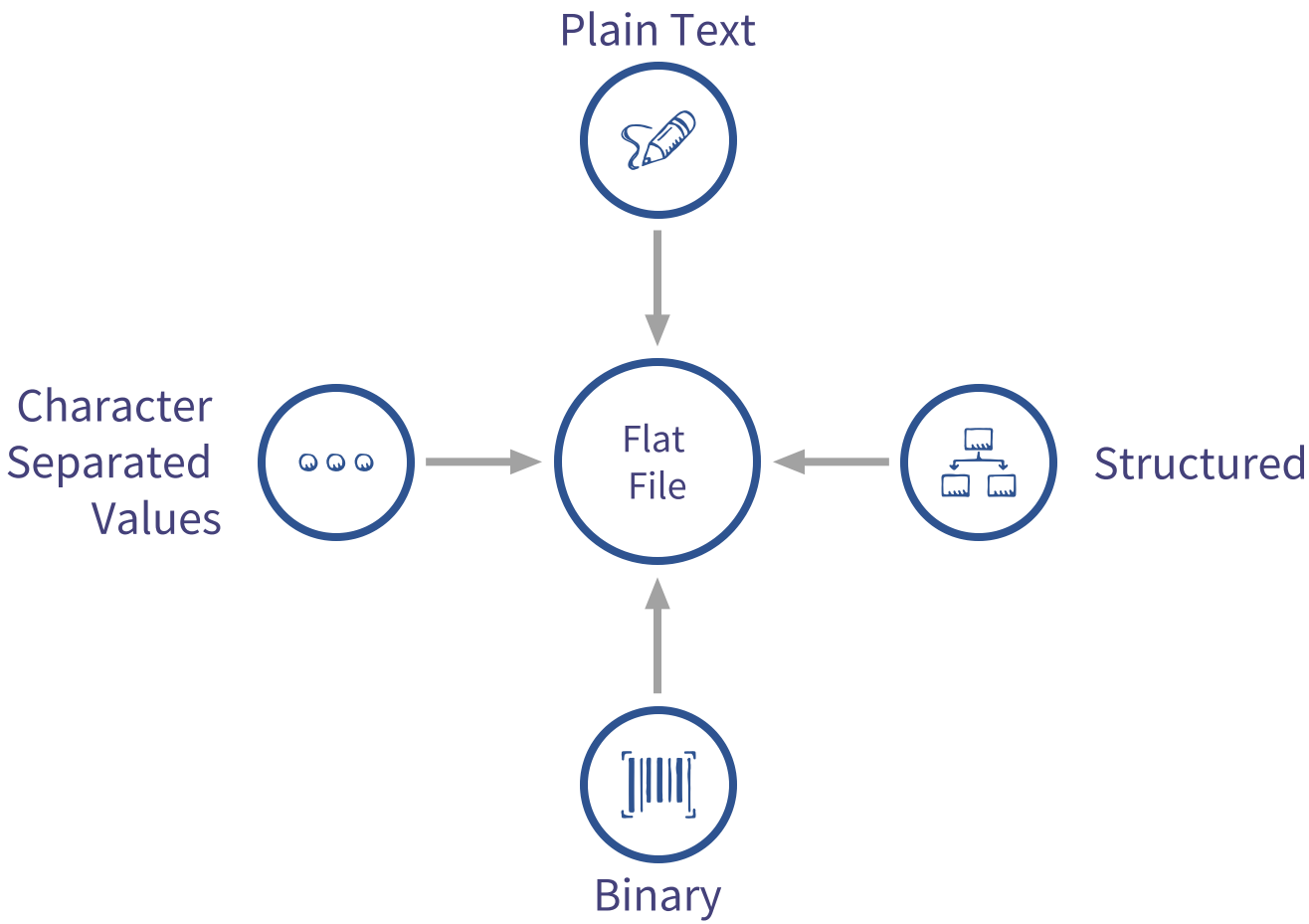
Files

Shell

Language

Database

Flat Files


Plain Text
Character Separated Values
Flat File
Structured
Binary

Plain Text

```
                VARIABLE : Mean TS from clear sky composite (kelvin)
                FILENAME : ISCCPMonthly_avg.nc
                FILEPATH : /usr/local/fer_dsets/data/
                SUBSET   : 93 points (TIME)
                LONGITUDE: 123.8W(-123.8)
                LATITUDE : 48.8S
                           123.8W
                            23
16-JAN-1994 00 /  1:  278.9
16-FEB-1994 00 /  2:  280.0
16-MAR-1994 00 /  3:  278.9
16-APR-1994 00 /  4:  278.9
16-MAY-1994 00 /  5:  277.8
16-JUN-1994 00 /  6:  276.1
```

Simplest type of file.

Often difficult to ingest, and may rely on header codes to define a schema.

Can be fixed-width.

Character Separated Values , | ; \n ' " / \ .

```
Year,Make,Model,Description,Price
1997,Ford,E350,"ac, abs, moon",3000.00
1999,Chevy,"Venture ""Extended Edition""","",4900.00
1999,Chevy,"Venture ""Extended Edition, Very Large""",,5000.00
1996,Jeep,Grand Cherokee,"MUST SELL!
air, moon roof, loaded",4799.00
```

marks the end of cells, rows, decimals and text with a character

Easier to ingest as a table of data

Vulnerable to escape

Can also include Excel Spreadsheets

Binary Files

- Not human-readable
- Program-specific dump of information
- Often intermediary data formats
- Used to encode or protect information

Structured


```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 27,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    },
    {
      "type": "mobile",
      "number": "123 456-7890"
    }
  ],
  "children": [],
  "spouse": null
}
```


Hold structured hierarchical data and metadata

Can hold class data well

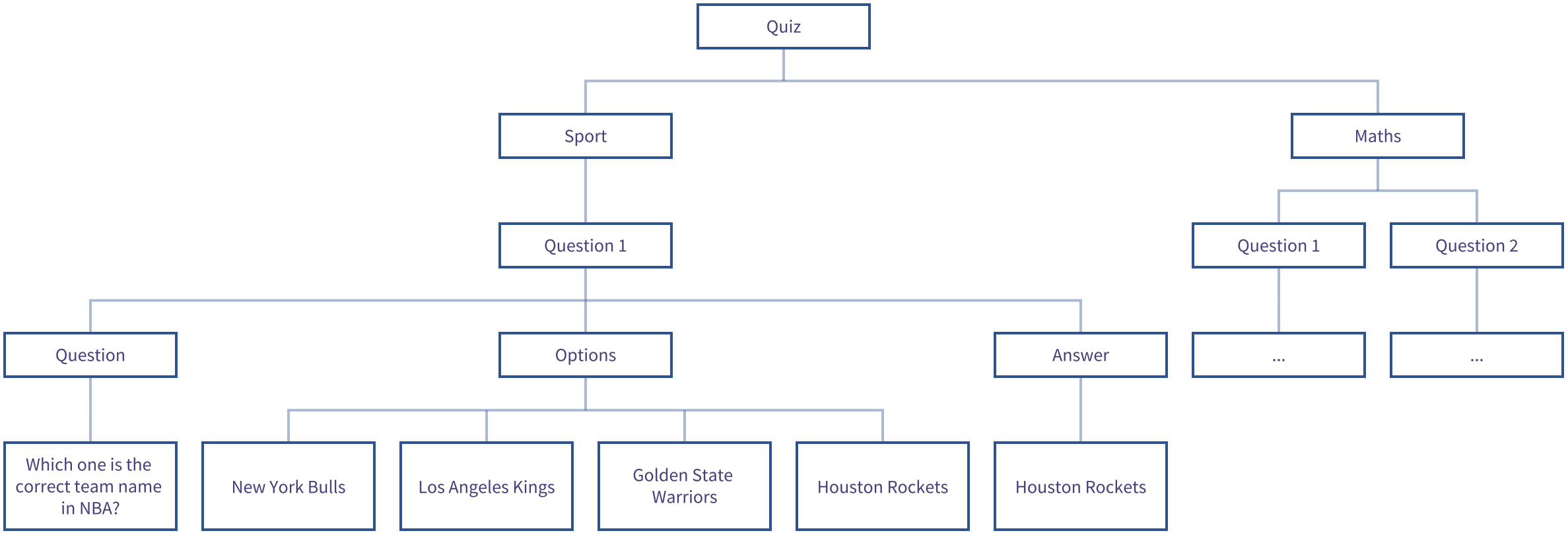
requires a schema to ingest

capable of holding data that does not fit within a two dimensional table structure well

Example JSON file


```
{
    "quiz": {
        "sport": {
            "q1": {
                "question": "Which one is correct team name in NBA?",
                "options": [
                    "New York Bulls",
                    "Los Angeles Kings",
                    "Golden State Warriros",
                    "Huston Rocket"
                ],
                "answer": "Huston Rocket"
            }
        },
        "maths": {
            "q1": {
                "question": "5 + 7 = ?",
                "options": [
                    "10",
                    "11",
                    "12",
                    "13"
                ],
                "answer": "12"
            },
            "q2": {
                "question": "12 - 8 = ?",
                "options": [
                    "1",
                    "2",
                    "3",
                    "4"
                ],
                "answer": "4"
            }
        }
    }
}
```

Structure


Quiz
Sport
Maths
Question 1
Question 1
Question 2
Question
Options
Answer
...
...
Which one is the correct team name in NBA?
New York Bulls
Los Angeles Kings
Golden State Warriors
Houston Rockets
Houston Rockets

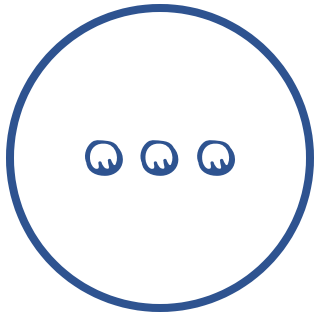
Filename Extensions

Plain Text

.txt

.log

no extension



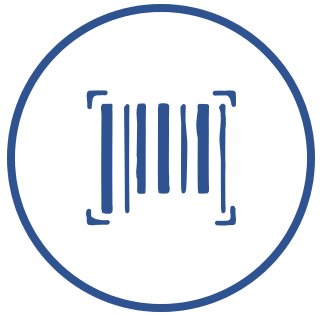
Character Separated Value

.csv

.psv

.tsv

.log



Binary

.feather

.pickle

.exe

.dat

.jpg

.gif

.png

Structured

.xml

.json

.avro

.parquet

.orc

File Overview

	Structure	Ease of use	Size
Plain Text	Poor	Fair to Poor	Fair to Good
CSV	Fair	Fair to Good	Fair to Good
Binary	Poor to Good	Good	Poor to Fair
Structured	Good	Fair to Good	Poor

Terminal (Shell) scripting

Good for simple piping

Historically easier in the UNIX terminal, but Windows command-line is quite capable now.

Many shells have been developed over the last 50 years, but the predominant linux shell is the Bourne Again Shell (bash).

Alternatives include the C Shell (csh), and tenex c shell (tcsh), Hamilton c shell and KornShell (ksh)

Windows runs on its own terminal, but can now support bash, and offers powershell (but beware!)

MacOS typically runs with bash

if in doubt - bash it

common data manipulation bash commands

- **cat**
  concatenate - print each row of all specified files
- **wget**
  web-get. Download data from a URL
- **sort**
  sorts data. use -n for a numeric sort
- **uniq**
  assimilates duplicated values in a list. only sorts adjacent rows so sort first!
- **head**
  prints the top of the file (default top 10 rows)
- **tail**
  prints the bottom of the file (default 10 rows)
- **wc**
  word count. Can count by file and row.
- **grep**
  Global Regular Expression Print:filter to select specific rows by a search pattern
- **awk**
  file pattern scanning and processing
- **touch**
  update the modification date on a file
- **sed**
  Stream Editor: text transformations on a file or input stream

Data flow in Bash

| pipe

connects the output of one command to the next (STDOUT to STDIN)

> insert

Writes the output of one command to a text file, typically, but can also be used to insert parameters to a further command.

|& pipe error

connects the error and output of one command to the next (STDOUT and STDERR to STDIN)

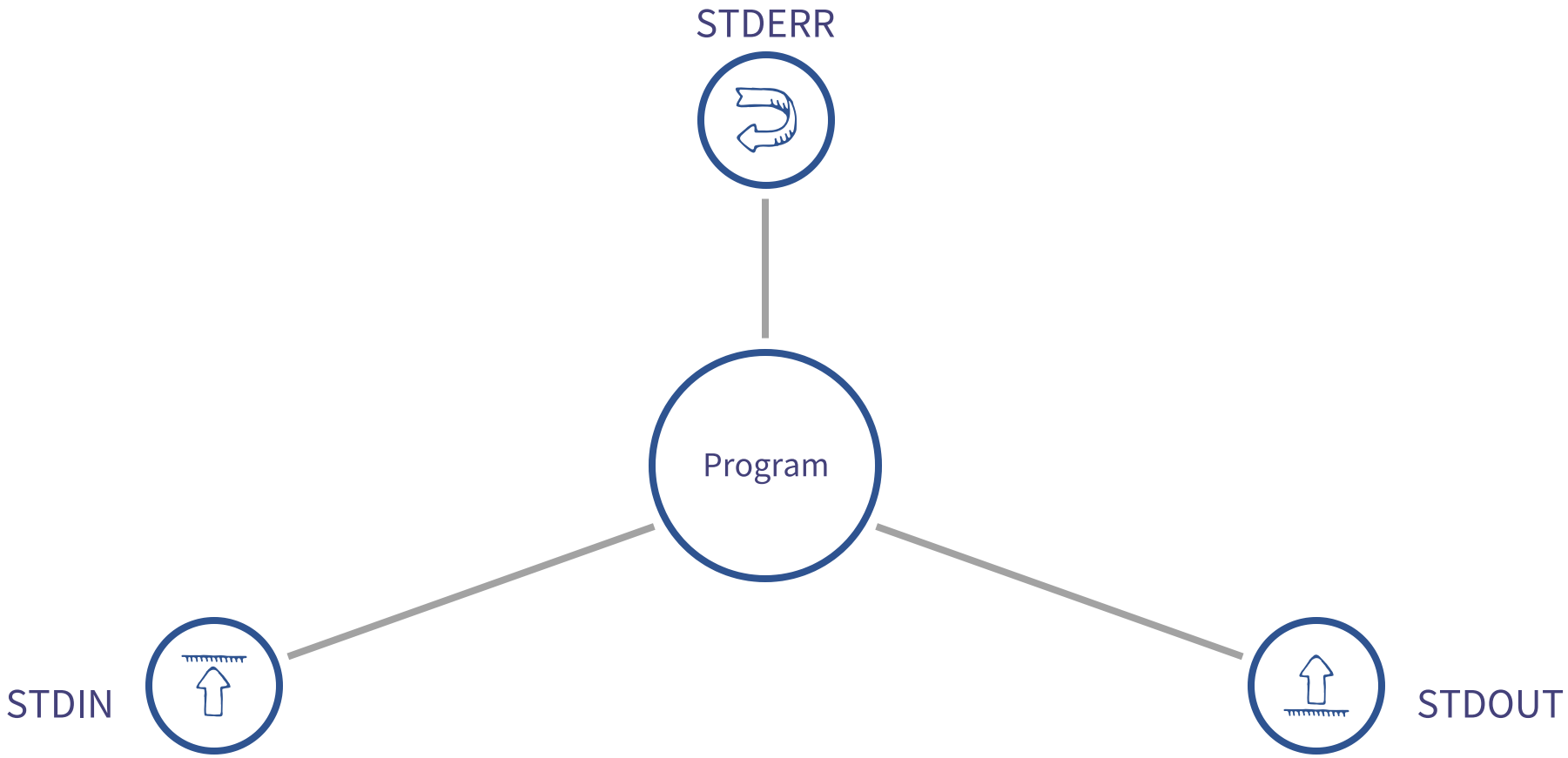
< insert back

Applied the output of one command back to the previous step. Can be used to insert data, parameters or to write to the previous file.

>> insert append

When data is streamed into a text file, any new data will append not overwrite.

POSIX compliance



Shell (BASH)

```
cat ./data_directory/datafile.csv | head -5 | grep 'search_string' > results.txt
```

Shell scripts allows simple files to be manipulated from the command line.

allows simple error handling

runs quickly, and compatible with a wide range of programs

Difficult for complex operations

Advanced shell wizardry

```
file_list = os.listdir(".")

f = FloatProgress(min=0, max=len(file_list))
display(f)
total_no_files = len(file_list)
for item in file_list:
    awk_command="""awk 'NR  > 30 {}' {}| tr -d '\\n' | awk 'gsub(" +", "\\n") {}' > './headless_only/'{}"""".format('{ print }', item, '{ print }', item)
    os.system(awk_command)
    f.value += 1
```

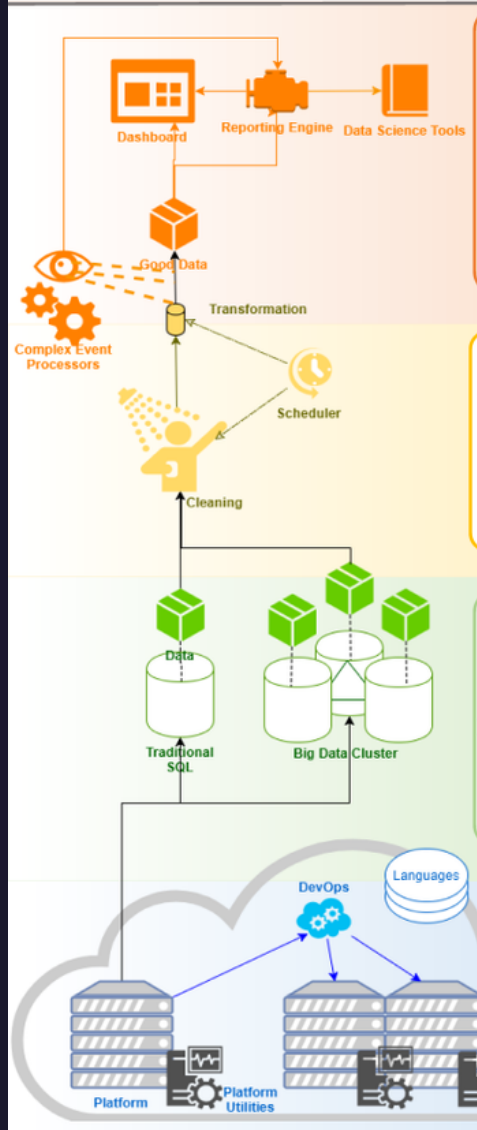
Common Data management Languages

More Languages

- Python
- R
- TSQL, PL/SQL, pgSQL...
- SCALA
- Java
- Julia
- Perl
- MatLab
- Octave
- C/C++

Data Technology Landscape

Winter 2018/2019



Dashboards and Reporting Engines

Data Science (including AI and ML)

ETL/ELT

Cleaning

Scheduler

Streaming

Reporting Systems

Integration

Databases/File systems

Data Formats

SQL Engines

Database Systems

Platform

DevOps

Platform Utilities

Version Control

Backend Languages

Platform and DevOps

Working with python

- **Use notebooks**

  Store notes and narrative in addition to code during development.

- **Use Pandas**

  There are wide-ranging data profiling and cleaning tools available

- **Use Numpy**

  NumPy changes how python handles variables for scientific purposes. No reason not to use it

- **Use Python 3.x**

  Major libraries now exist in Python 3. Python 3 runs faster and cleaner.

- **Watch out for old code**

  Python 2 code =/= Python 3 code

- **Love your libraries**

  There are thousands of libraries available. Explore PyPi, PIP, Conda, Github.... to find them

File handling in python

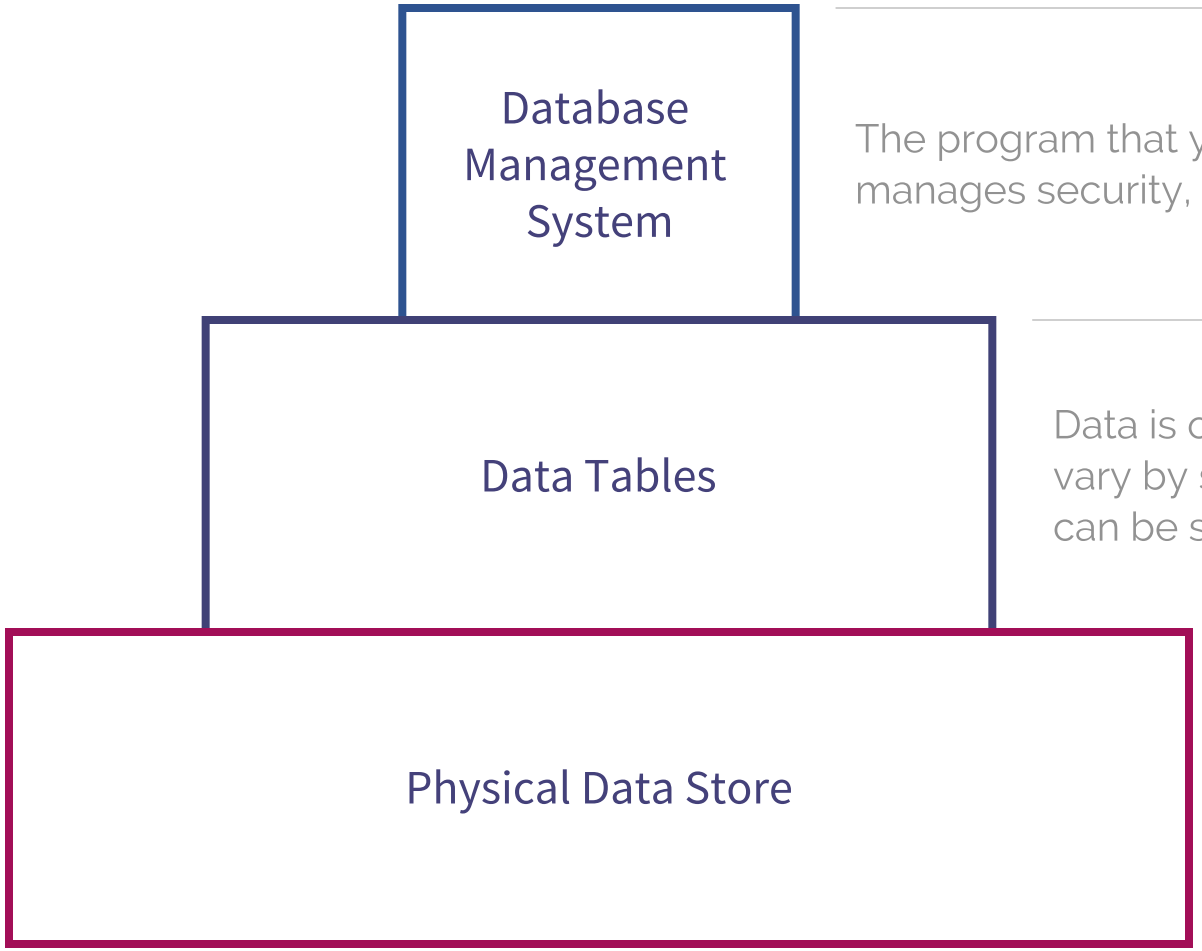
In [ ]:
```
file_list = os.listdir("../../../DATA/ASCII/AllData/headless_only/TimeseriesAndAmplitude/")[:-1] #to remove the FFT_dB directory
at the end of the list
input_file_list = map(lambda x: "../../../DATA/ASCII/AllData/headless_only/TimeseriesAndAmplitude/"+x, file_list)
output_file_list = map(lambda x: "../../../DATA/ASCII/AllData/headless_only/TimeseriesAndAmplitude/FFT_dB/"+x[:-3]+"csv",file_li
st)

i = 0
for file_path in tqdm(file_list):
    input_filepath = input_file_list[i]
    output_filepath = output_file_list[i]

    fft_kaiser_db(input_filepath=input_filepath, output_filepath=output_filepath, kaiser_betamax_value=14, time_interval=0.01)
    i+=1
```

What is a database?

Database Layers



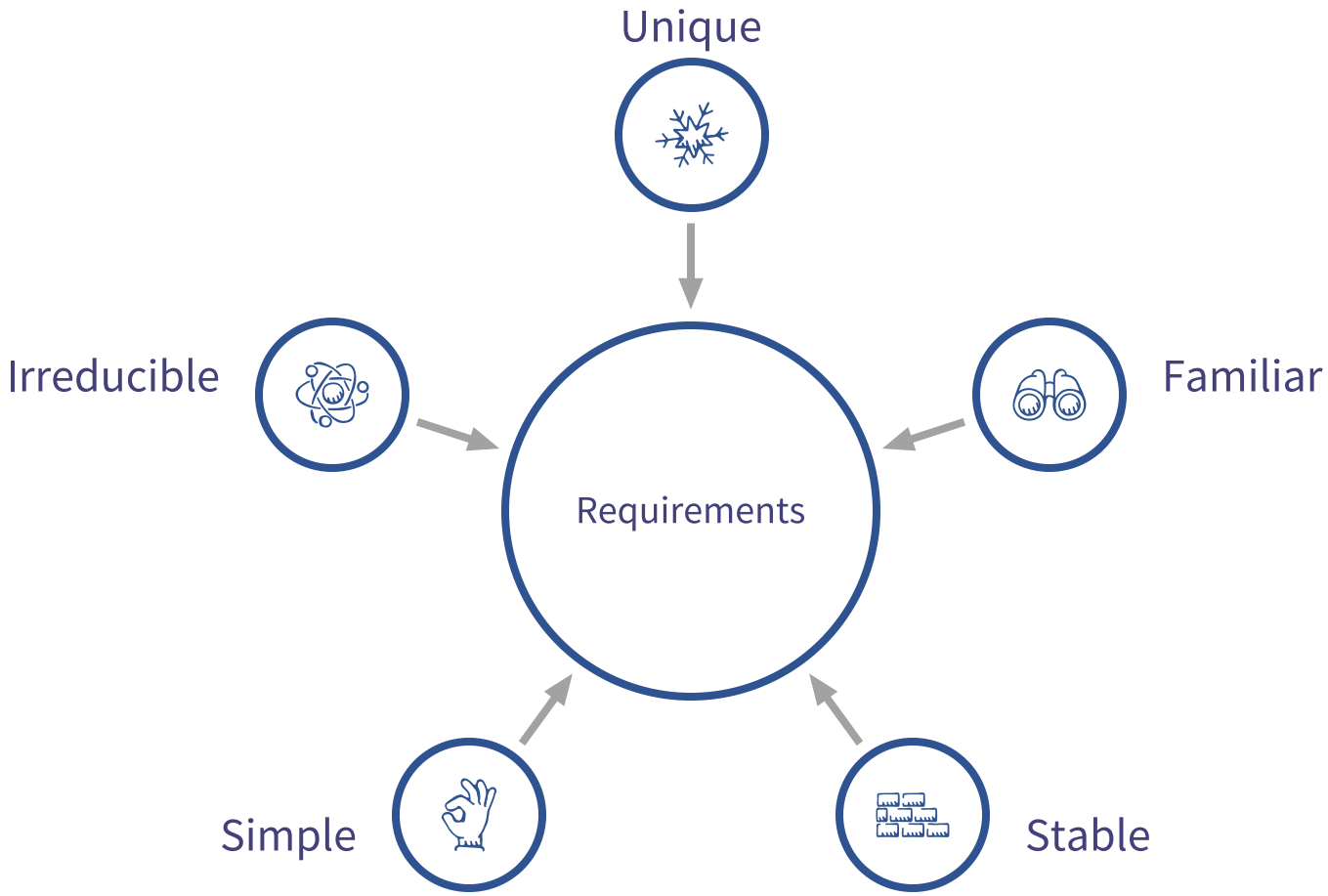
The program that you use to manage and interact with the database. It manages security, data updates, backups, roll backs...

Data is organised in tables, The way they are organised can vary by style, database type and design philosophy. This can be similar to a spreadsheet in structure.

Data is written to a disk, in different ways depending on the type of database. You won't need to interact with this layer. Data tables pull data from the store to populate their rows.

What do you need in a table?

Keys - Requirements



Keys - Types

Primary Key

the principal identifier for a table. Every piece of information should be associated with a PK. Values must be unique and NonNull

Foreign Key

The Primary Key of another table, used to identify related data across a database. Helps with joins.

Natural Key

A type of identifier that uses a meaningful attribute to identify each row. Examples include NI number, Student Number, ISBN. Can reduce the number of joins needed.

Surrogate Key

As opposed to a natural key, a surrogate key is a randomly generated unique value that acts as a key. Usually requires a new column to be added, but changing the data structure later won't break the key.

Normal Forms

1st Normal Form

Data is in a table

Values in the table are “atomic” - they can't be subdivided

Columns do not repeat

Data is being organised into meaningful structures!

Normal Forms

2nd Normal Form

All the columns in the table have to rely on the Primary Key

Each table holds information about only one thing!

Normal Forms

3rd Normal Form

No columns of data are transitively dependent on the Primary Key

Each piece of information about the subject has no hidden links!

UnNormalised Data

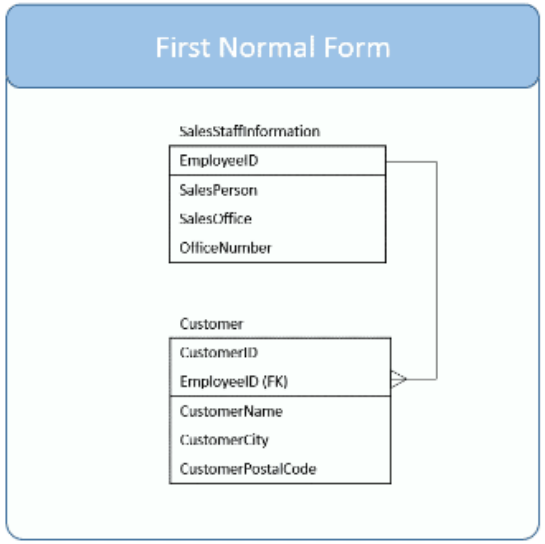
SalesStaff						
EmployeeID	**SalesPerson**	**SalesOffice**	**OfficeNumber**	**Customer1**	**Customer2**	**Customer3**
1003	Mary Smith	Chicago	312-555-1212	Ford	GM	
1004	John Hunt	New York	212-555-1212	Dell	HP	Apple
1005	Martin Hap	Chicago	312-555-1212	Boeing		

1st Normal Form

SalesStaffInformation			
EmployeeID	**SalesPerson**	**SalesOffice**	**OfficeNumber**
1003	Mary Smith	Chicago	312-555-1212
1004	John Hunt	New York	212-555-1212
1005	Martin Hap	Chicago	312-555-1212

Note: Primary Key: EmployeeID

Customer				
CustomerID	**EmployeeID**	**CustomerName**	**CustomerCity**	**PostalCode**
C1000	1003	Ford	Dearborn	48123
C1010	1003	GM	Detroit	48213
C1020	1004	Dell	Austin	78720
C1030	1004	HP	Palo Alto	94303
C1040	1004	Apple	Cupertino	95014
C1050	1005	Boeing	Chicago	60601



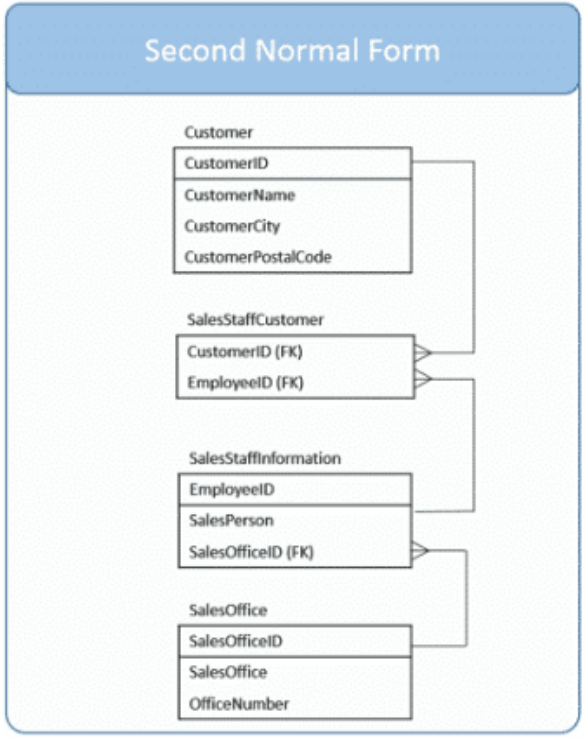
2nd Normal Form

Customer			
CustomerID	**CustomerName**	**CustomerCity**	**CustomerPostalCode**
C1000	Ford	Dearborn	48123
C1010	GM	Detroit	48213
C1020	Dell	Austin	78720
C1030	HP	Palo Alto	94303
C1040	Apple	Cupertino	95014
C1050	Boeing	Chicago	60601

SalesStaffCustomer	
CustomerID	**EmployeeID**
C1000	1003
C1010	1003
C1020	1004
C1030	1004
C1040	1004
C1050	1005

SalesStaffInformation		
EmployeeID	**SalesPerson**	**SalesOffice**
1003	Mary Smith	S10
1004	John Hunt	S20
1005	Martin Hap	S10

SalesOffice		
SalesOfficeID	**SalesOffice**	**OfficeNumber**
S10	Chicago	312-555-1212
S20	New York	212-555-1212



3rd Normal Form

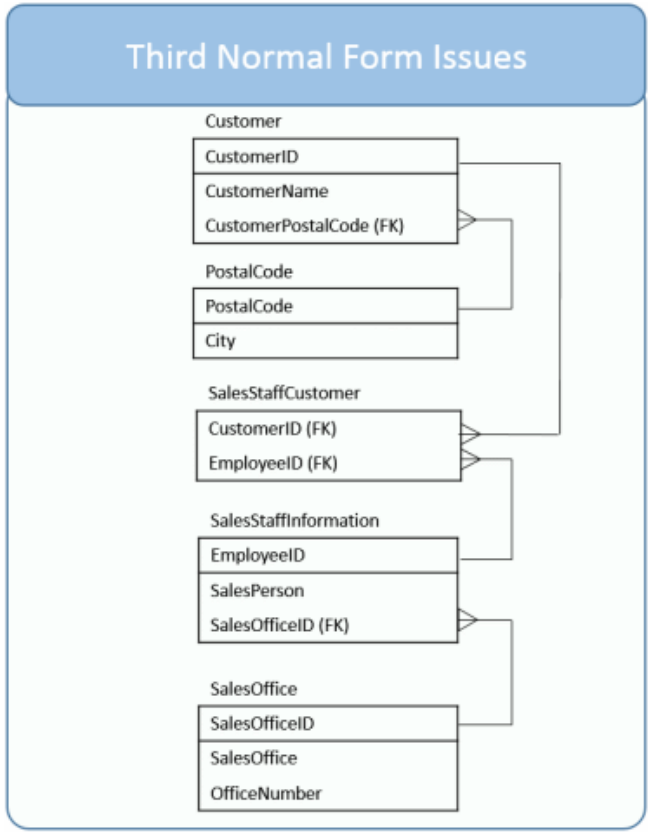
Customer		
CustomerID	**CustomerName**	**CustomerPostalCode**
C1000	Ford	48123
C1010	GM	48213
C1020	Dell	78720
C1030	HP	94303
C1040	Apple	95014
C1050	Boeing	60601

PostalCode	
PostalCode	**City**
48123	Dearborn
48213	Detroit
60601	Chicago
78720	Austin
94303	Palo Alto
95014	Cupertino

SalesStaffCustomer	
CustomerID	**EmployeeID**
C1000	1003
C1010	1003
C1020	1004
C1030	1004
C1040	1004
C1050	1005

SalesStaffInformation		
EmployeeID	**SalesPerson**	**SalesOffice**
1003	Mary Smith	S10
1004	John Hunt	S20
1005	Martin Hap	S10

SalesOffice		
SalesOfficeID	**SalesOffice**	**OfficeNumber**
S10	Chicago	312-555-1212
S20	New York	212-555-1212



Structured Query Language

- Lets you modify data (Read, Write, Update, Delete)
- Lets you query data
- Human Readable
- Different Dialects! TSQL, PL/SQL...

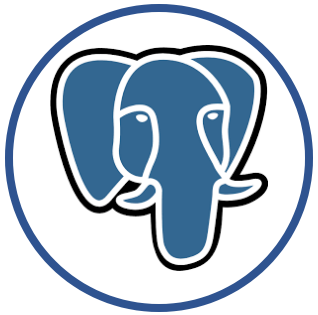
Common Relational Database Programs

MySQL

Basic but functional database

uses SQL/PSM

Free



PostgreSQL

aka pgSQL and Postgres

uses PL/pgSQL

Stricter relational database

Free

Oracle DB

Enterprise grade database

uses T/SQL and PL/SQL

widely supported

Pricey!

Does have free variants (Oracle DB Express and more)

example Queries

- SELECT * FROM Customer;

  Display all data from the Customer table

- SELECT EmployeeID, SalesOffice FROM SalesStaffInformation

  Display the data from the Employee ID and The Sales Office columns ONLY from the Sales Staff Information table

- SELECT * FROM SalesStaffInformation WHERE SalesPerson LIKE 'Mary*';

  Display data from all columns of the Sales Staff Information, but only the rows where the sales person is called Mary

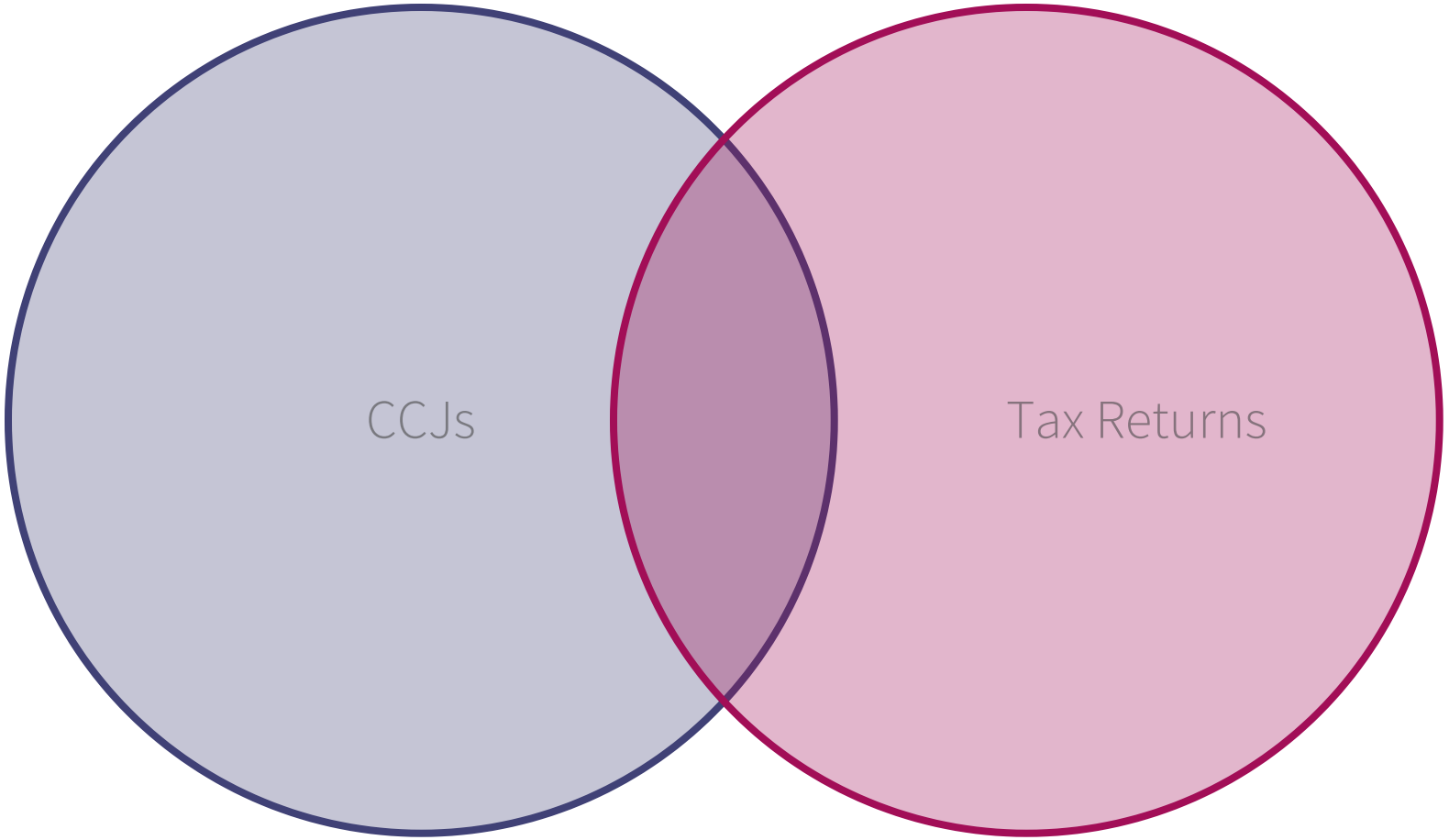
- SELECT DISTINCT CustomerName FROM Customer;

  Display unique customer data from the customer table

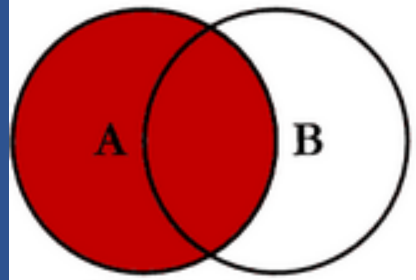
- SELECT COUNT (DISTINCT CustomerName) FROM Customer;

  Returns the number of distinct customers in the Customer table

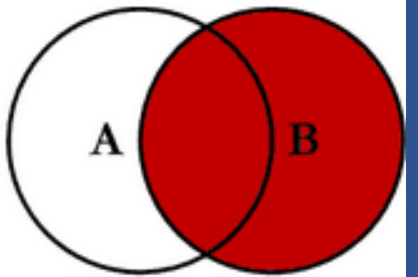
Joins



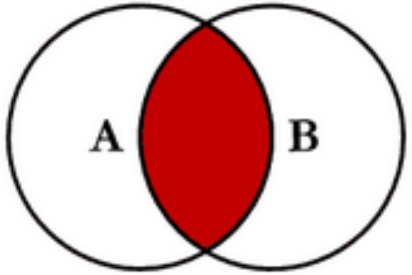
SQL JOINS



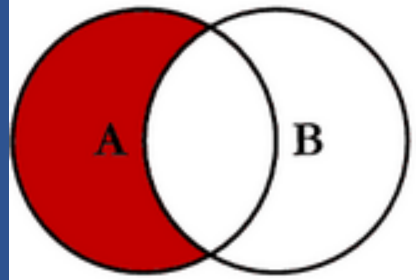

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



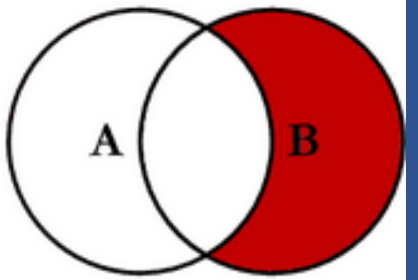

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



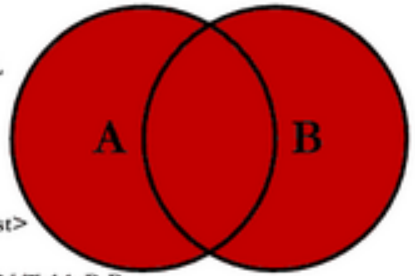

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



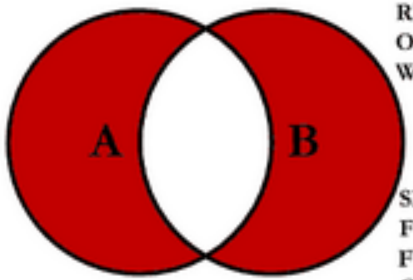

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

Common scenarios

Same Data, Different Name

Data gets duplicated, and inflates the row count. This might encourage artificial trends in the data

Different Data, Same Name

Data might be over-written and lost. The identifier may be failing, and would cause issues if not considered when summing columns

Data only found in one table

Data may be missing in other tables, or simply does not exist. Can cause issues when joining tables

Different Keys, Same Data

Similar to the first scenario, but worse: the keys are the baseline identifier which makes differentiating between them difficult

General Tips

- **Use Snake Case**

  when tables or column names cannot use spaces, complicated_column_name is easier to read than complicatedcolumnname

- **make names (and variables) easy to understand**

  sgl_arr_tme is difficult to understand. signal_arrival_time is easier.

- **Append dates to table names if you are exporting or copying them**

  table_name_vfinal_FINAL means less than table_name_190329

- **Be careful where you leave your credentials**

  Hard coding your access credentials into your scripts is a quick way to test your code and an even faster way to broadcast your keys

Break

Did you hear about the elf that lost his toys?

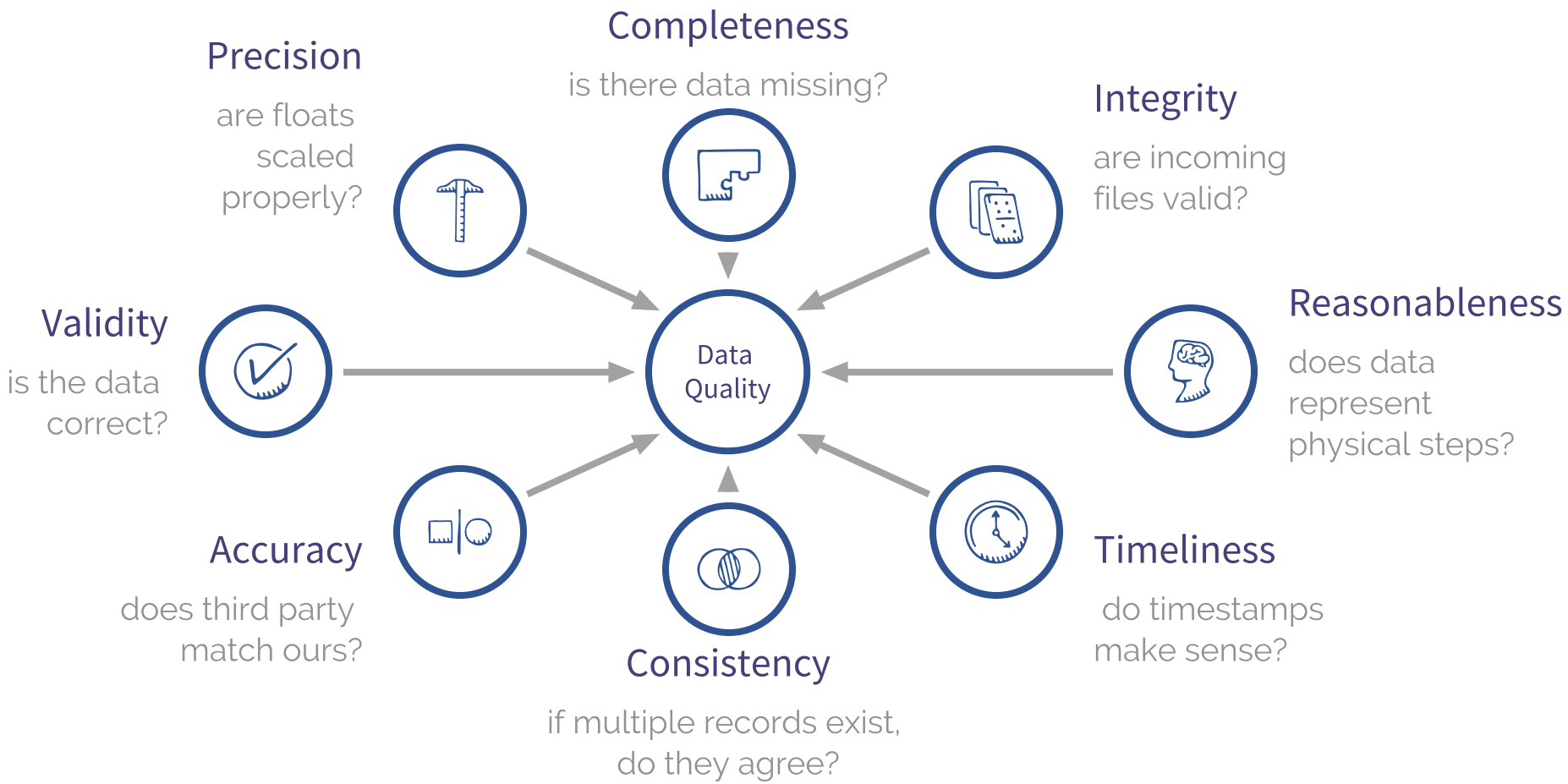
Did you hear about the elf that lost his toys?

He was lego-less

Section 4

Data Profiling

Data Quality



DAMA Definition

Adapted from DAMA-DMBOK

1. **Timeliness**

   Does the data represent reality from the required point in time?

2. **Completeness**

   Does all the data for the event exist?

3. **Uniqueness**

   Is anything recorded more than once?

4. **Validity**

   Does the data conform to its syntax?

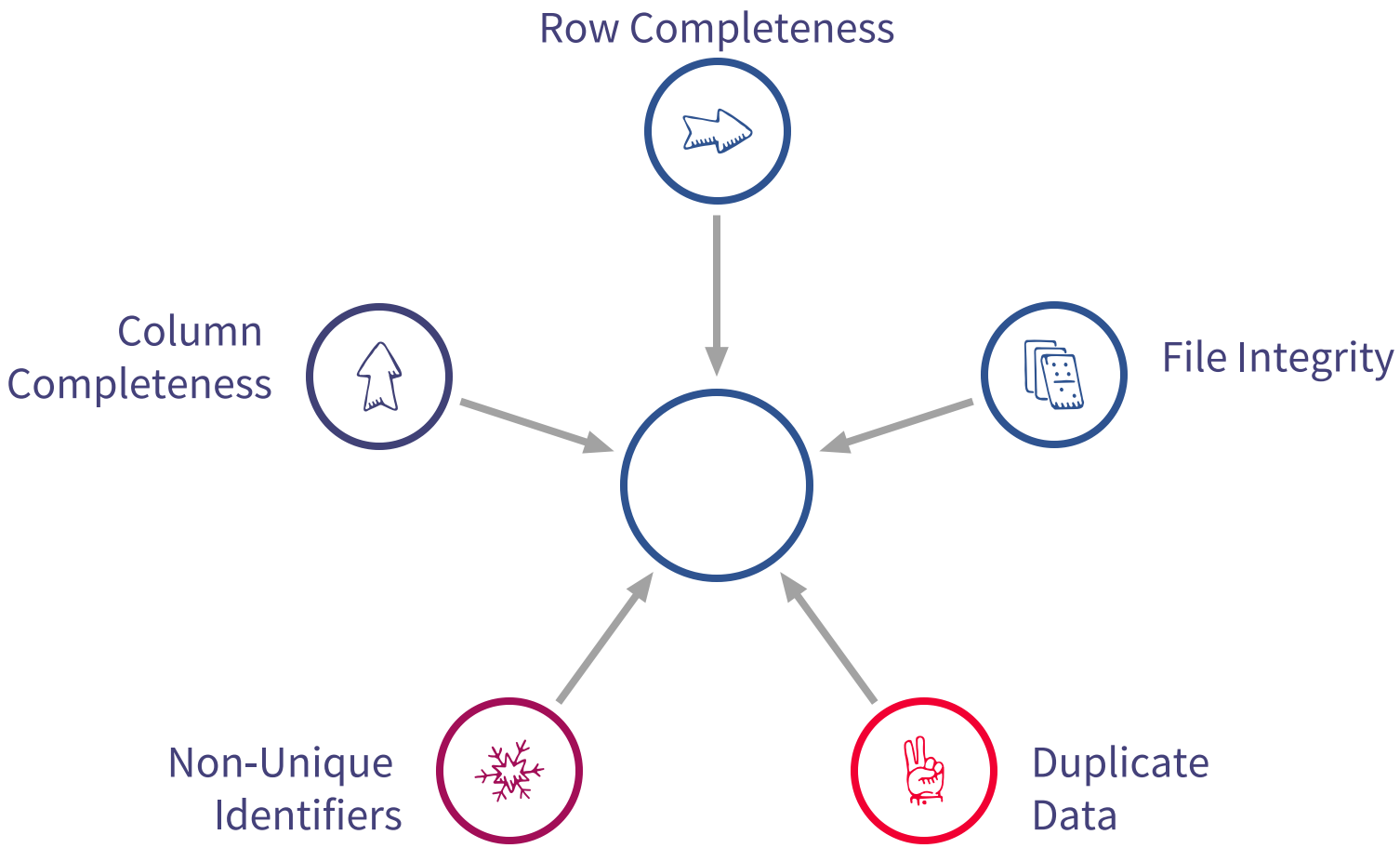
5. **Consistency**

   Do multiple representations of a record differ?

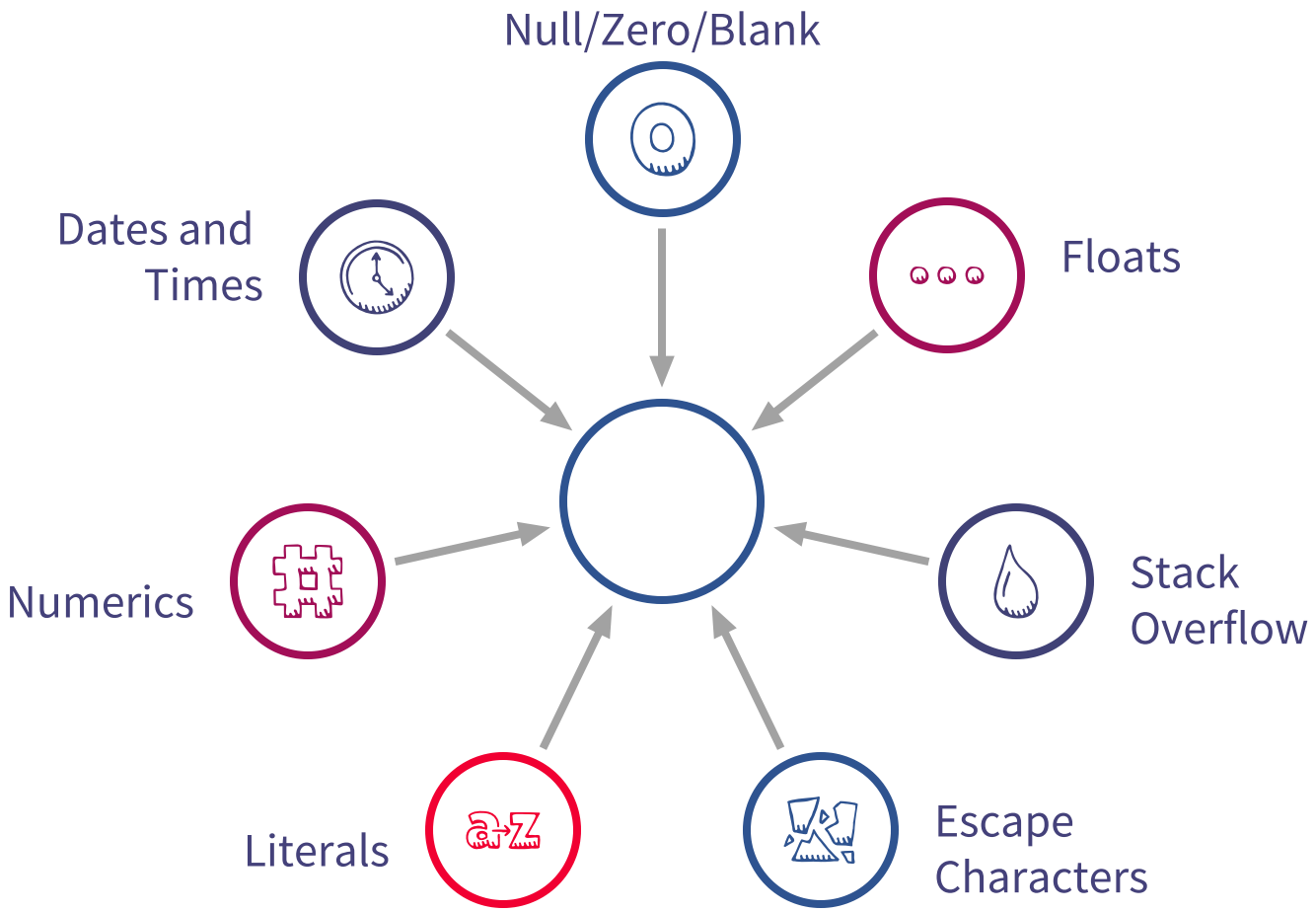
6. **Accuracy**

   Does the data accurately describe the real world event or object?

Out-of-cell issues

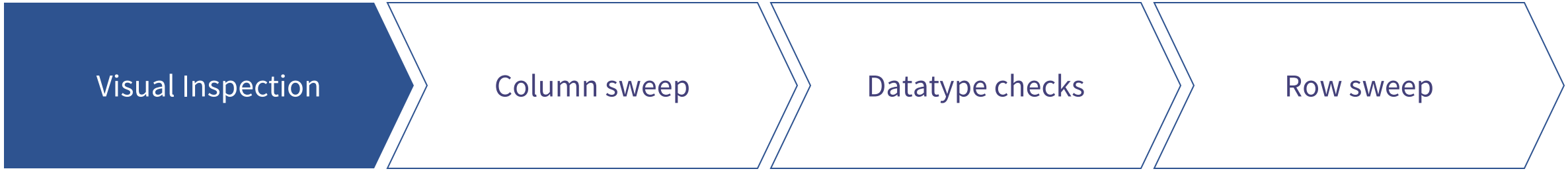


In-cell issues



Data Profiling

sample_df.head()



Data Profiling

```
sample_df.head()
```


```
sample_df.describe(include='all')
```

```
for column in
sample_df.columns()
    print(column.count())
```

Data Profiling

sample_df.head()

sample_df.dtypes

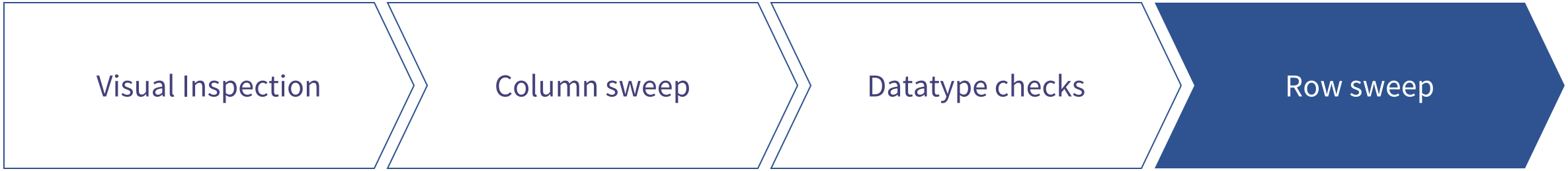
sample_df.describe(include='all')

```
for column in
sample_df.columns()
    print(column.count())
```

Data Profiling

sample_df.head()

sample_df.dtypes



sample_df.describe(include='all')

for column in
sample_df.columns()
    print(column.count())

dataframe.isnull()
.values.ravel().sum()

Identifiers - unique?

- Pandas: series.is_unique
- Python: len(x) == len(set(x))
- SQL: SELECT CASE WHEN count(distinct col)= count(col) THEN 'pass' ELSE 'fail' END FROM table;

Null|NaN|Zero|None|“”[]{}

- **Zero**

  is a value. Can be computed.

- **Null**

  is not a value - nothing there

- **NaN**

  Not a Number: can be a null or a dtype issue

- **None**

  Python object for Null.

- **“” “ ”**

  Empty - string. May be a space.

- **[]**

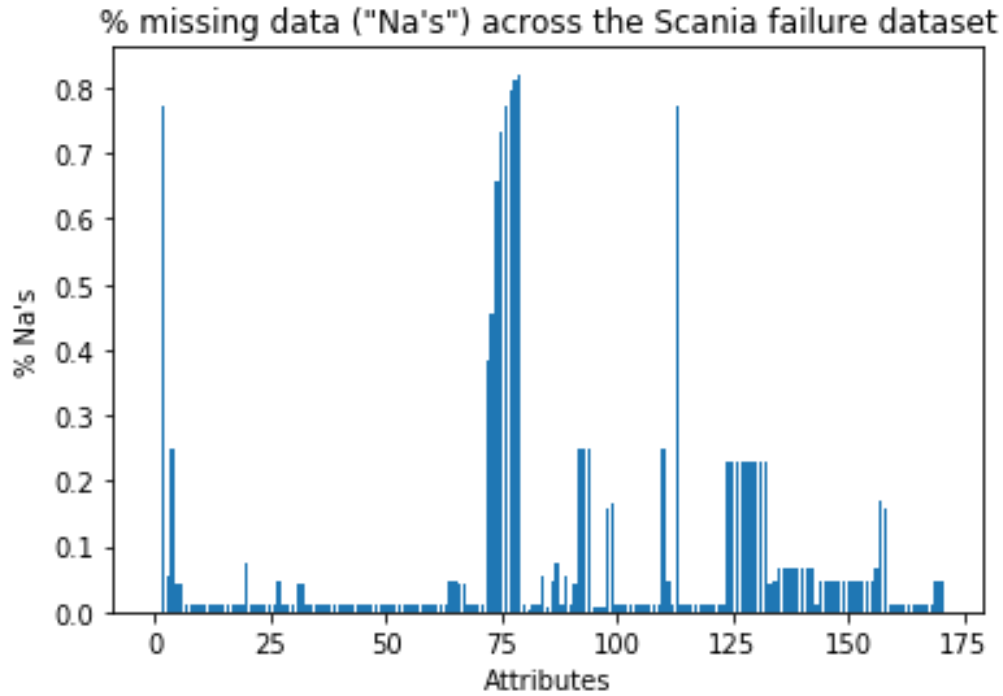
  empty - list.

- **{}**

  empty - dict.

Boolean Test

Element	True	False
0		✓
Null		✓
NaN		✓
None		✓
“”		✓
“ ”	✓	
[]		✓
{}		✓


% missing data ("Na's") across the Scania failure dataset
% Na's
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
25
50
75
100
125
150
175
Attributes

Profiling Crib Sheet

Issue	When you know	how to check
Non Unique Identifiers/Duplicate Data	failure on load failure on join	count distinct
Completeness	failure on join failure on computation	count distinct count null values
Null/Zero	failure on computation failure to derive datatype	count distinct count null values targeted query
DateTime	failure on load failure on computation	check datatype regex for wrong format
Numerics	failure on load failure on computation	check datatype sample and plot values inspect
Escapes	failure on load failure on write	check for column length and row completeness regex/targeted query
Stack Overflow	failure on computation	plot range of values by column describe statistics/boxplot check datatype

Section 5

Data Cleaning

Casting - numerics

pushing one datatype into another

- pandas: pd.to_numeric(obj)
- python: int(), decimal(), float()
- SQL: CAST (x AS numeric(y))

Casting - datetimes

pushing one datatype into another

- pandas: pd.to_datetime(obj)
- python: strptime()
- SQL: TO_DATE (x, “format”)

YYYY-MM-DD HH:MM:SS:UU.....

2018-04-08 10:30:00:00

2018-04-08 10:30:00:00 GMT

2018-04-08 05:30:00:00 -5

2018-04-08 05:30:00:00 EST

Missing Values

- Pandas: df.skipna(axis="0 or 1", how = "any")
- Python: Boolean test

- SQL: SELECT *
  FROM table
  WHERE column IS NOT NULL;

Dropping missing/bad data

Drop by column

you do not lose rows of data

good for continuous time data

limits the information available for analysis

Drop by row

you do not lose information on each point

good for more complex/wide analysis

you have fewer datapoints

Use both

Drop very poor columns first

Follow with a row-wide sweep

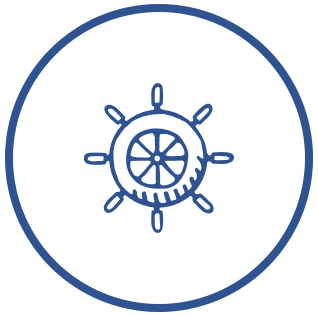
Identifiers - unique

- Pandas: df.drop_duplicates
- Python: set(x)
- SQL: SELECT DISTINCT

note

Database connectors

Connectors



Admin console

Terminal

Script

Connecting to an online database

host, username, password, port etc...

SQLalchemy

Psycopg2

Uniform Resource Identifier: a string that id's exactly the server, role and data you need

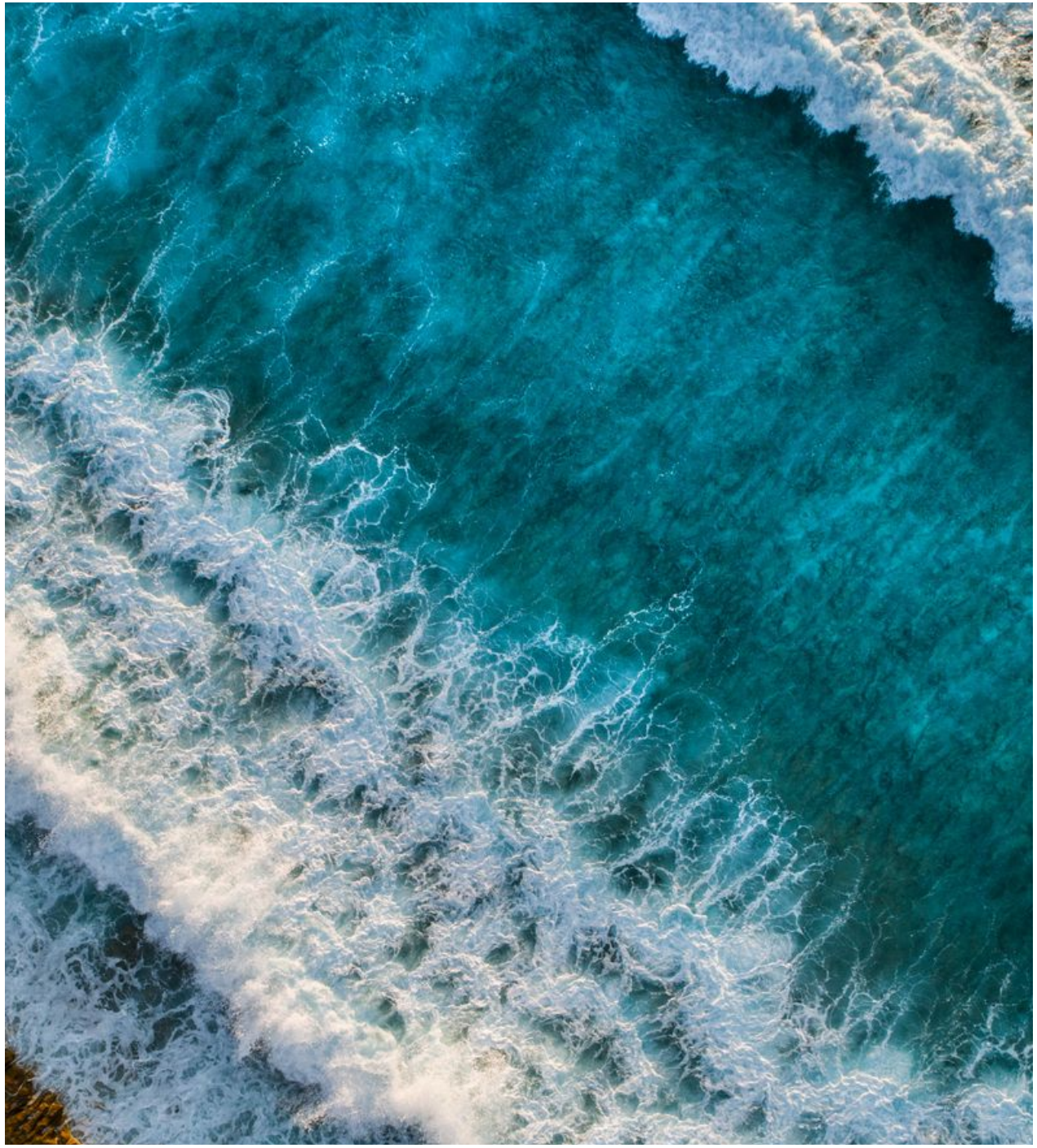
pull data down

Your turn!

Objectives

- access a dataset from an online database
- profile the data
- if you find any issues, clean the data

https://github.com/R-Strange/Data_Wrangling_Course_Oxford



Recap

- What do pipelines look like?
- What do pipelines do?
- Extract-Load-Transform
- Flat Files
- Shell Scripting
- SQL Queries
- Database options
- Normal Forms
- Joins
- Profiling
- Cleaning
- DB Connectors

Fin

Go home - while you still can

Wrangling for Pipelines and Data Handling

8th April

Sorry!

Day outline

1. Data Flows Revisited
2. Testing

Lunch

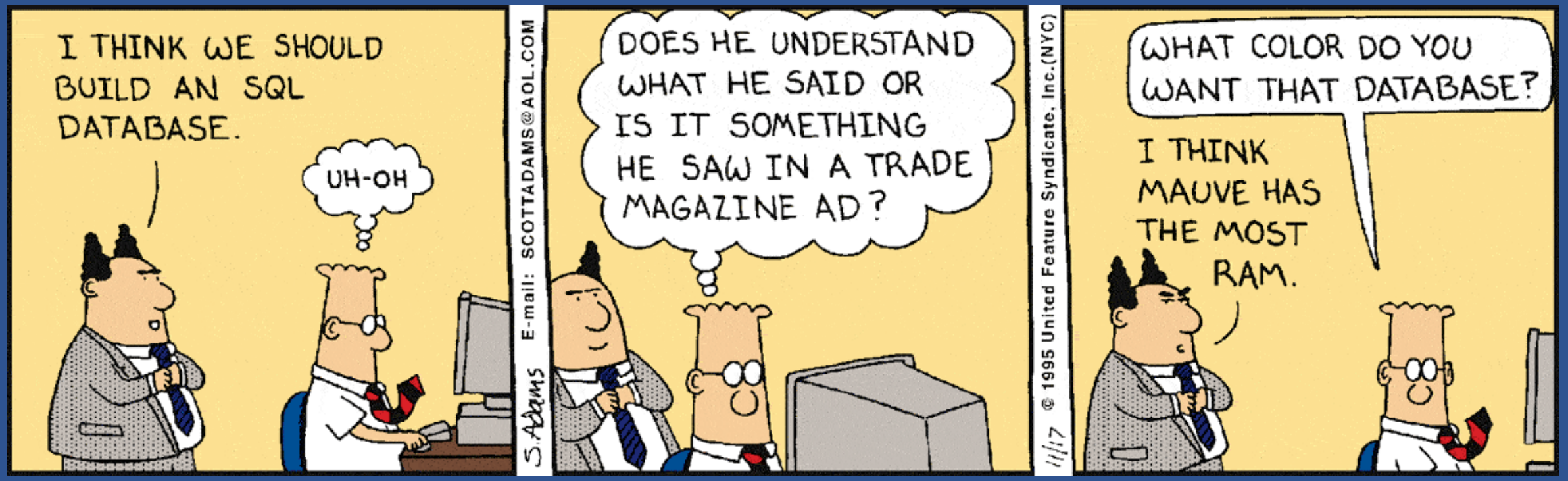
3. Cloud
4. Big Data
5. IoT
6. APIs
7. API Lab

Section 1

Data Pipelines Revisited

Yesterday's Pipelines...

Break

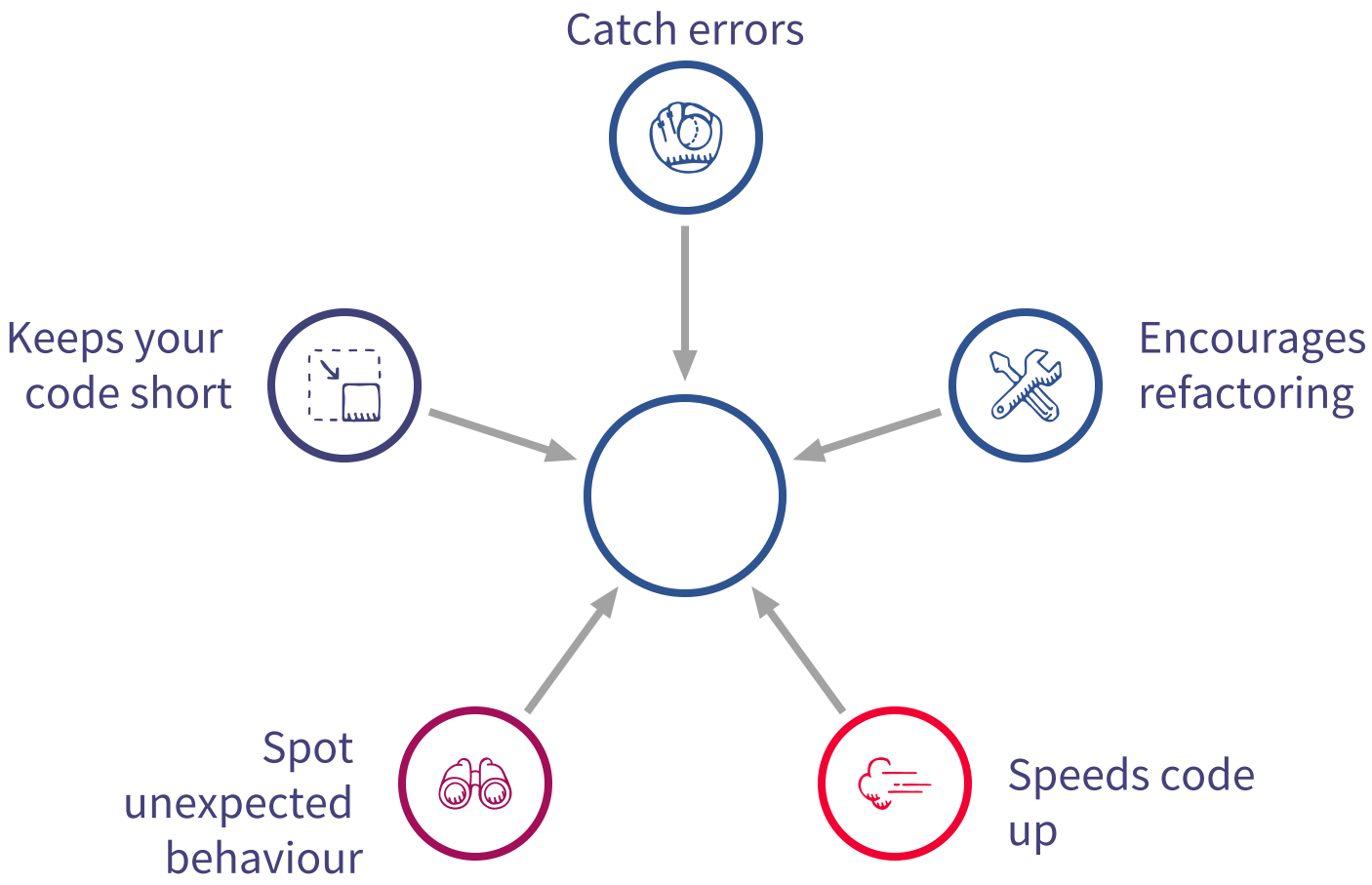

I THINK WE SHOULD BUILD AN SQL DATABASE.
UH-OH
E-mail: SCOTTADAMS@AOL.COM
S. Adams
DOES HE UNDERSTAND WHAT HE SAID OR IS IT SOMETHING HE SAW IN A TRADE MAGAZINE AD?
© 1995 United Feature Syndicate, Inc.(NYC)
11/17
WHAT COLOR DO YOU WANT THAT DATABASE?
I THINK MAUVE HAS THE MOST RAM.

Section 2

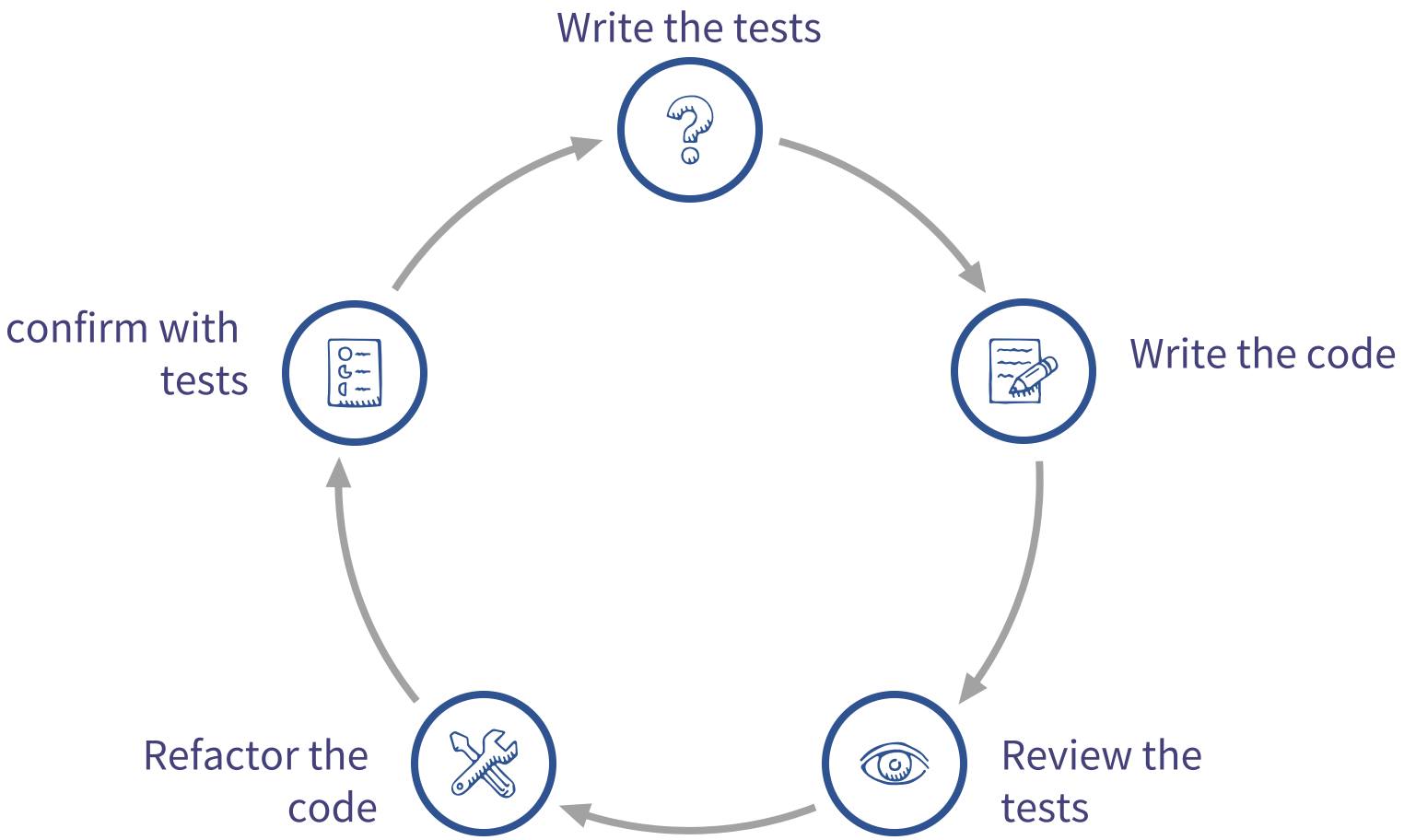
Testing


WE CAN ONLY AFFORD TO FIX THE HIGH-PRIORITY BUGS.
IF WE DON'T FIX 100% OF THE BUGS, THE SOFTWARE WILL BE 100% USELESS.
SO OUR PLAN IS TO FAIL?
MORE SLOWLY.
www.dilbert.com scottadams@aol.com
6-22-09 © 2009 Scott Adams, Inc./Dist. by UFS, Inc.

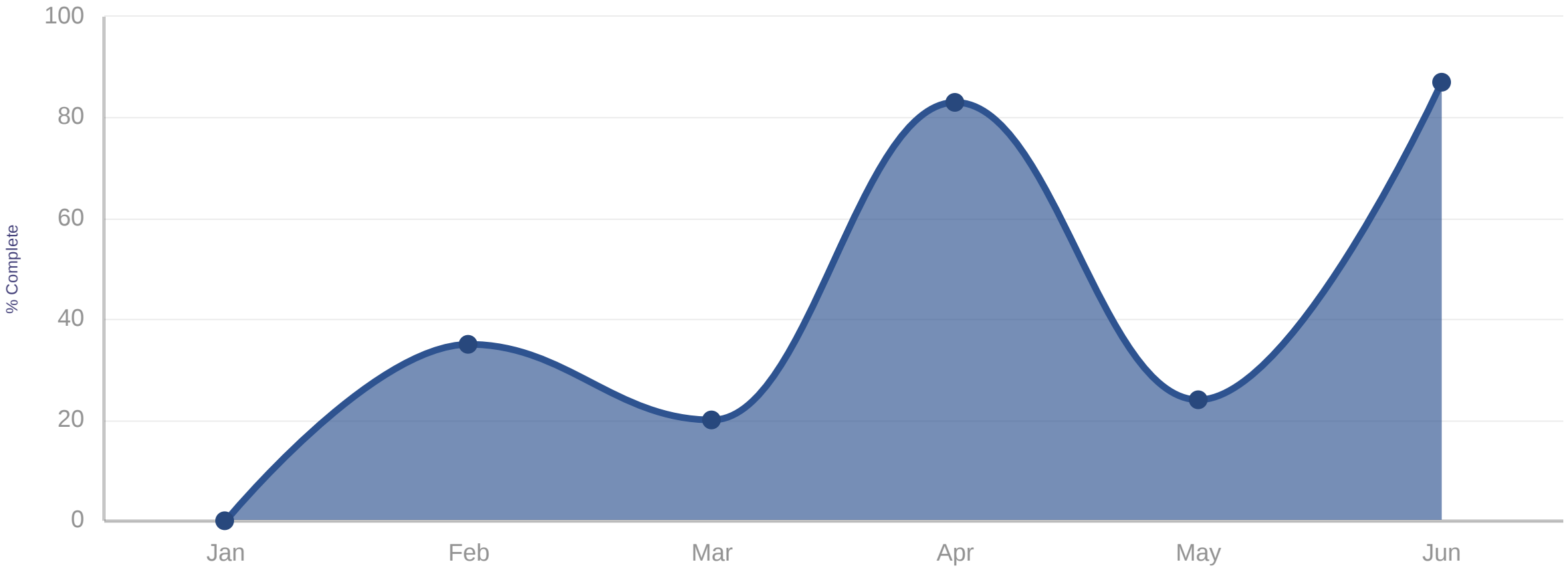
Why bother testing code?



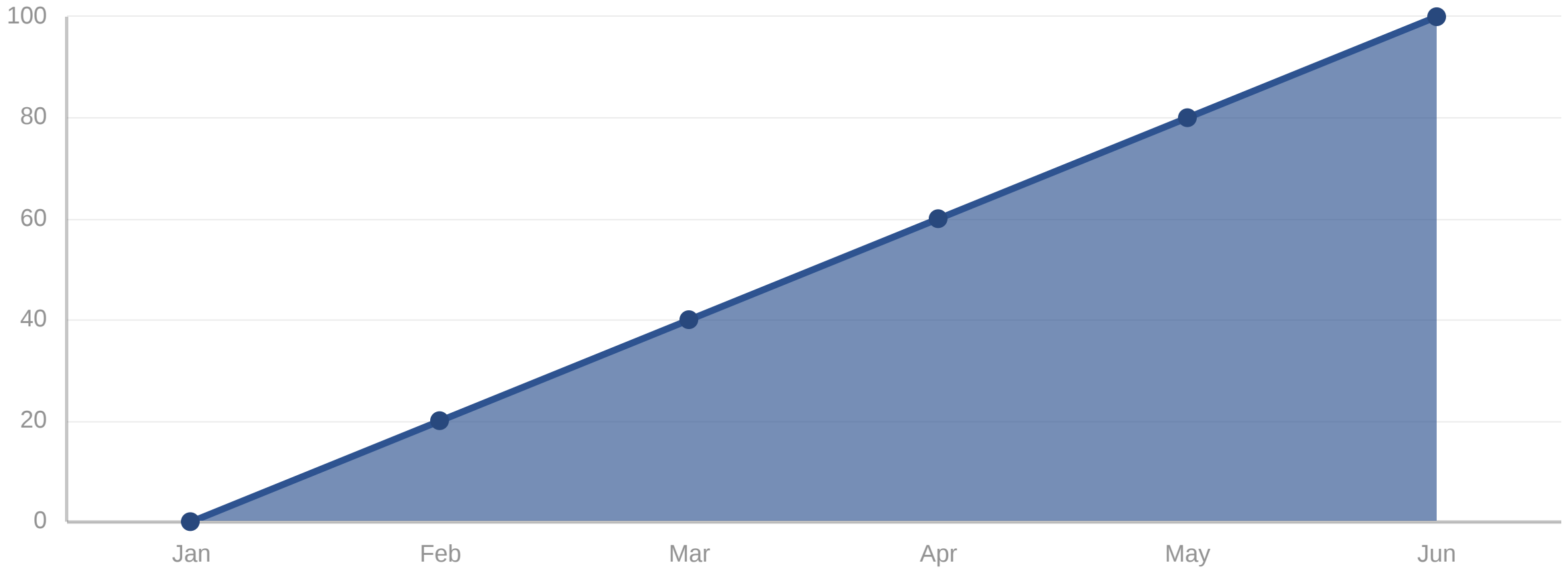
Test driven development?



Typical development



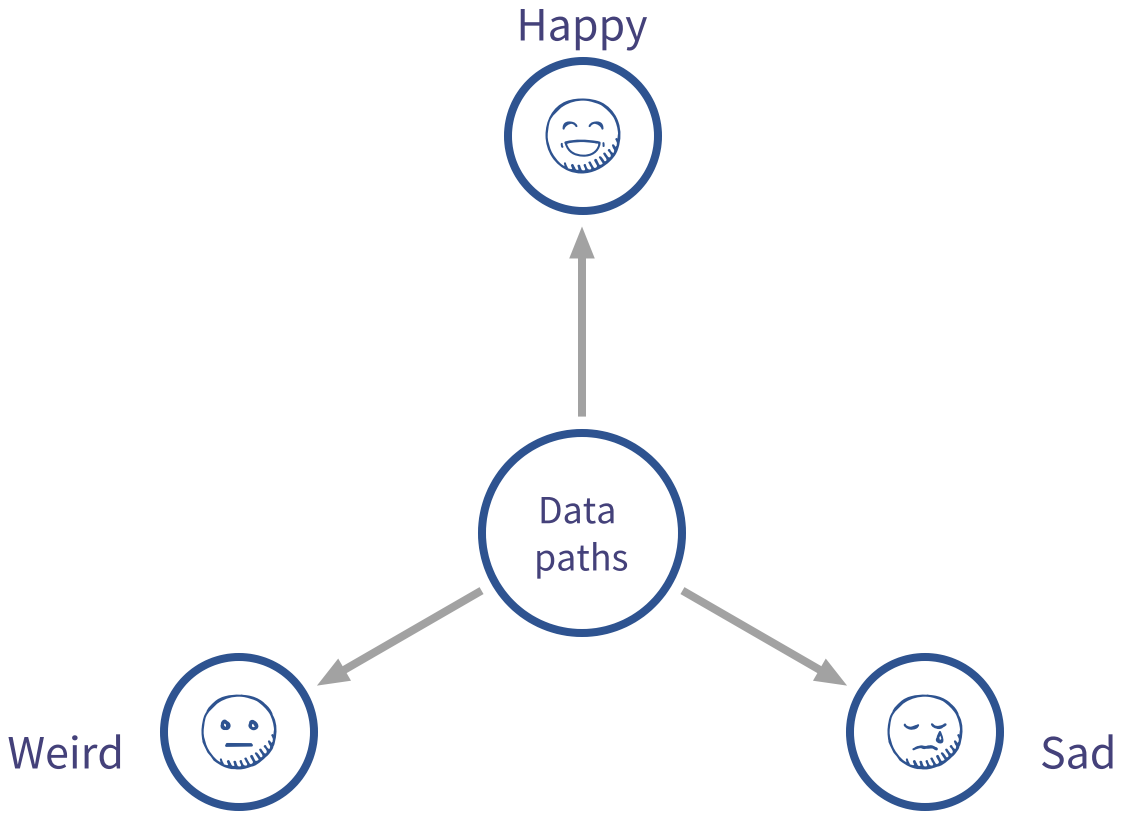
Test Driven



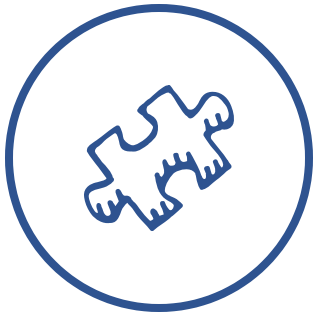
Benefits

- **Define the purpose of your code**
  - Defines inputs, outputs and functions before you write
  - Helps you design how the code fits together
  - Helps you keep code clean and simple
- **Noisy failure**
  - Fewer surprise failures down the line
  - More reliable code
  - Confidence in results
- **Granular progress**
  - You can track code progress more effectively
  - Easier to predict completion time
- **Better code**
  - Timing tests point out processing bottlenecks
  - Stops feature bloat/creep
  - Safer refactoring

Paths



Further reading



Unit Testing

Function-level testing of the smallest chunks of code

Automated code health checks

Code is tested outside of its environment

Looks for unneeded dependencies

Behavioural Testing

Tests pieces of code on their “behaviour”

Tests by what the program should produce, not just on individual “units” of code

Based on a framework with natural language tests

Test Driven Development

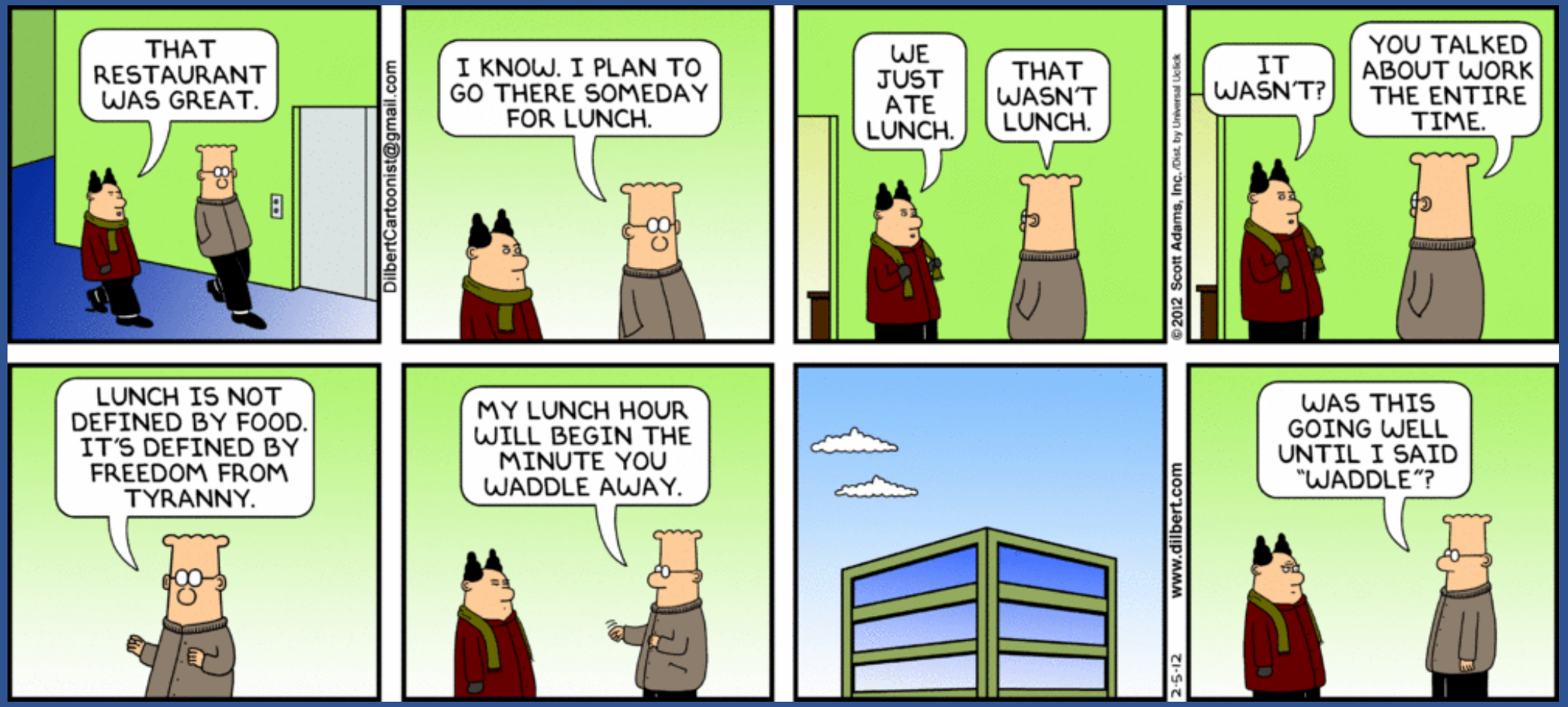
Framework for building code with tests

Everything the code must do is represented by a series of tests

All code should fit the tests

The program is complete when all tests pass

Lunch


THAT RESTAURANT WAS GREAT.
DilbertCartoonist@gmail.com
I KNOW. I PLAN TO GO THERE SOMEDAY FOR LUNCH.
WE JUST ATE LUNCH.
THAT WASN'T LUNCH.
© 2012 Scott Adams, Inc. /Dist. by Universal Uclick
IT WASN'T?
YOU TALKED ABOUT WORK THE ENTIRE TIME.
LUNCH IS NOT DEFINED BY FOOD. IT'S DEFINED BY FREEDOM FROM TYRANNY.
MY LUNCH HOUR WILL BEGIN THE MINUTE YOU WADDLE AWAY.
www.dilbert.com
2-5-12
WAS THIS GOING WELL UNTIL I SAID "WADDLE"?

This Afternoon - “Buzzword” Techs

Cloud

Big Data

Internet of Things

Section 3

Cloud

What is the cloud?

Cloud resources:

- Online servers
- You rent machines
- You subscribe to services
- You store information remotely
- You compute remotely
- Prices and resources change dynamically

Platform Providers

Amazon Web Services

Google Cloud Platform

Azure

Market Share


4
3
3
2
3
3
6
3
3
3
3
3
3
2
3
2
2
4
1
2
3
3

Compute
EC2
Lightsail
ECR
ECS
EKS
Lambda
Batch
Elastic Beanstalk
Serverless Application Repository

Storage
S3
EFS
FSx
S3 Glacier
Storage Gateway
AWS Backup

Database
RDS
DynamoDB
ElastiCache
Neptune
Amazon Redshift
Amazon DocumentDB

Migration & Transfer
AWS Migration Hub
Application Discovery Service
Database Migration Service
Server Migration Service
AWS Transfer for SFTP
Snowball
DataSync

Networking & Content Delivery
VPC
CloudFront
Route 53
API Gateway
Direct Connect
AWS App Mesh
AWS Cloud Map
Global Accelerator

Developer Tools
CodeStar
CodeCommit
CodeBuild
CodeDeploy
CodePipeline
Cloud9
X-Ray

Robotics
AWS RoboMaker

Blockchain
Amazon Managed Blockchain

Satellite
Ground Station

Management & Governance
AWS Organizations
CloudWatch
AWS Auto Scaling
CloudFormation
CloudTrail
Config
OpsWorks
Service Catalog
Systems Manager
Trusted Advisor
Managed Services
Control Tower
AWS License Manager
AWS Well-Architected Tool
Personal Health Dashboard

Media Services
Elastic Transcoder
Kinesis Video Streams
MediaConnect
MediaConvert
MediaLive
MediaPackage
MediaStore
MediaTailor

Machine Learning
Amazon SageMaker
Amazon Comprehend
AWS DeepLens
Amazon Lex
Machine Learning
Amazon Polly
Rekognition
Amazon Transcribe
Amazon Translate
Amazon Personalize
Amazon Forecast
Amazon Textract

Analytics
Athena
EMR
CloudSearch
Elasticsearch Service
Kinesis
QuickSight
Data Pipeline
AWS Glue
MSK

Security, Identity, & Compliance
IAM
Resource Access Manager
Cognito
Secrets Manager
GuardDuty
Inspector
Amazon Macie
AWS Single Sign-On
Certificate Manager
Key Management Service
CloudHSM
Directory Service
WAF & Shield
Artifact
Security Hub

Mobile
AWS Amplify
Mobile Hub
AWS AppSync
Device Farm

AR & VR
Amazon Sumerian

Application Integration
Step Functions
Amazon MQ
Simple Notification Service
Simple Queue Service
SWF

AWS Cost Management
AWS Cost Explorer
AWS Budgets
AWS Marketplace Subscriptions

Customer Engagement
Amazon Connect
Pinpoint
Simple Email Service

Business Applications
Alexa for Business
Amazon Chime
WorkMail

End User Computing
WorkSpaces
AppStream 2.0
WorkDocs
WorkLink

Internet Of Things
IoT Core
Amazon FreeRTOS
IoT 1-Click
IoT Analytics
IoT Device Defender
IoT Device Management
IoT Events
IoT Greengrass
IoT SiteWise
IoT Things Graph

Game Development
Amazon GameLift

Core services

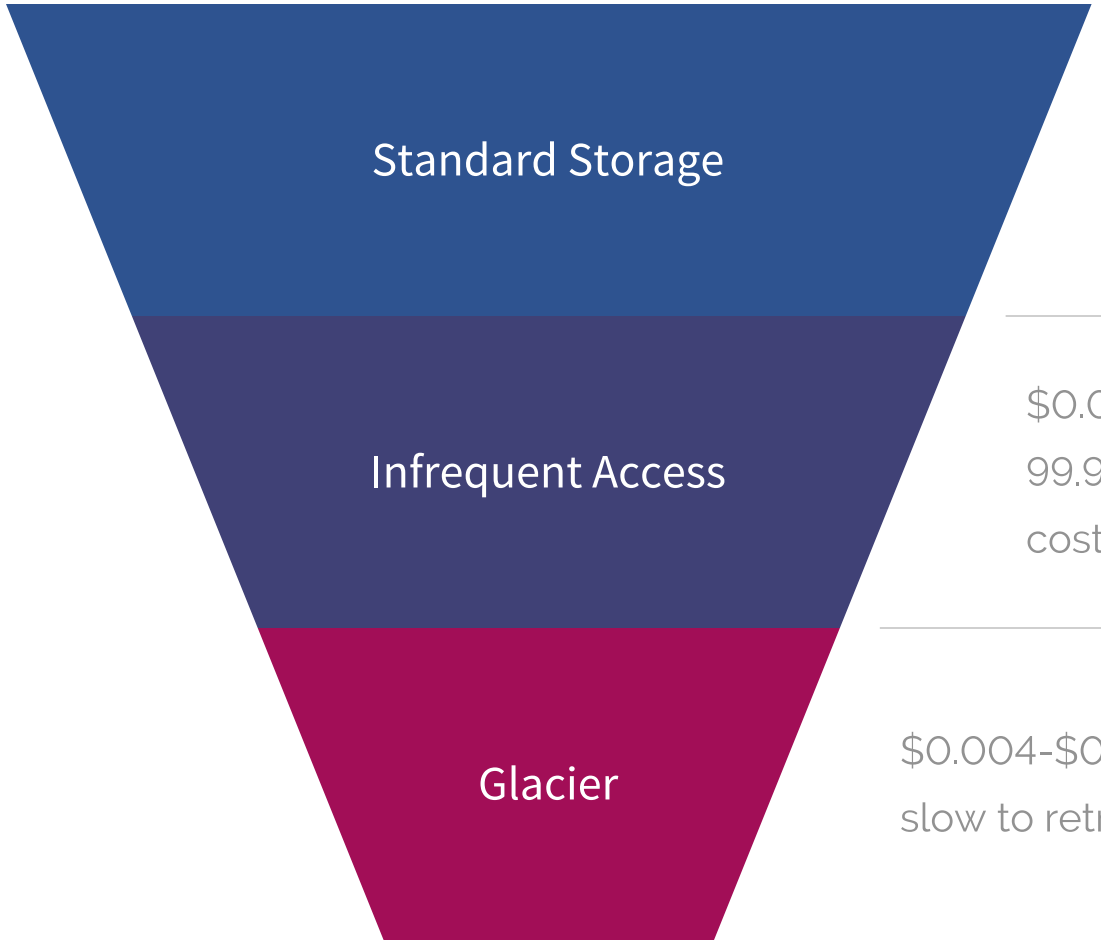
EC2

Elastic Compute Cloud

S3

Simple Storage Solution

S3



$0.021-$0.023 per GB per month
99.999999999% durability across 3 zones
costs more to store and less to access

$0.01-$0.0125 per GB per month
99.999999999% durability across 3 zones
costs more to access, less to store

$0.004-$0.00099 per GB per month
slow to retrieve (5 hours for a request?)

EC2

- virtual server
- predefined builds
- various hardware options
- reserved/hourly/spot pricing
- replicable

EC2 instance types

Instance	Use	Notes	Cost (on demand hour)
T3	**General Purpose**	**General Balanced Server**	**$0.0052 - $0.3341**
M5	General Purpose	T, with higher network bandwidth	$0.096-$4.608
A1	General Purpose	ARM edge processing	$0.0255 - $0.408
C5	**Compute Optimised**	**HPC - good for number crunching**	**$0.085-$3.06**
X1	Memory Optimised	Extreme in-memory needs	$6.669-$26.688
R5	Memory Optimised	For in-memory calculation (e.g. in-memory DB or stream)	$0.126-$6.048
P3	**Accelerated Computing**	**General Purpose GPU instances**	**$3.06-$31.212**
G3	**Accelerated Computing**	**Graphics intensive instances**	**$1.14-$4.56**
F1	Accelerated Computing	allows low-level GPU logic alteration	$1.65-$13.20
I3	Storage Optimised	Resource intensive databases	$0.156-$4.992
D2	Dense-store Optimised	Massively-Parallel Processing	$0.69-$5.52

prices

On demand

Most expensive

Run your server whenever you want

Only pay for the hours that your server is online

Spot-Price

Name a price per hour

When server demand drops, “spot” price falls. If you meet the spot price you get compute time.

Reserved Price

You rent a server whole-hog for a year or three years

Cheaper than on-demand

reliable

Some other services...

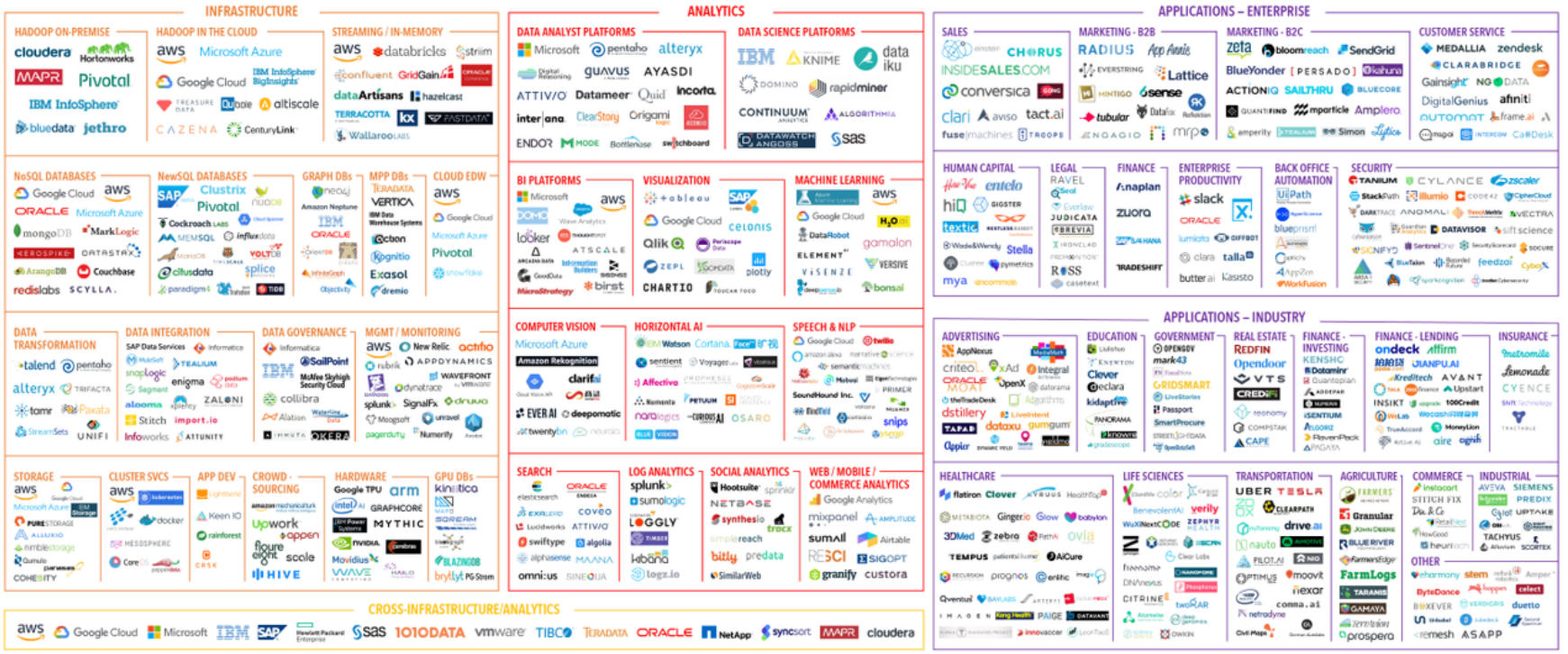
- **RDS**
  Relational Data Store - automanaged Databases
- **Lambda**
  Serverless computation
- **QuickSight**
  Dashboard Analytics
- **Kinesis**
  Data streaming
- **Data Pipeline**
  Pipeline management and scheduling
- **Sumerian**
  VR and AR development
- **Cloud9**
  online IDE
- **GreenGrass**
  IoT device management
- **Polly**
  Text to Speech
- **SageMaker**
  AI development
- **Rekognition**
  Pre-built computer vision
- **GroundStation**
  Satellite Management

Section 4

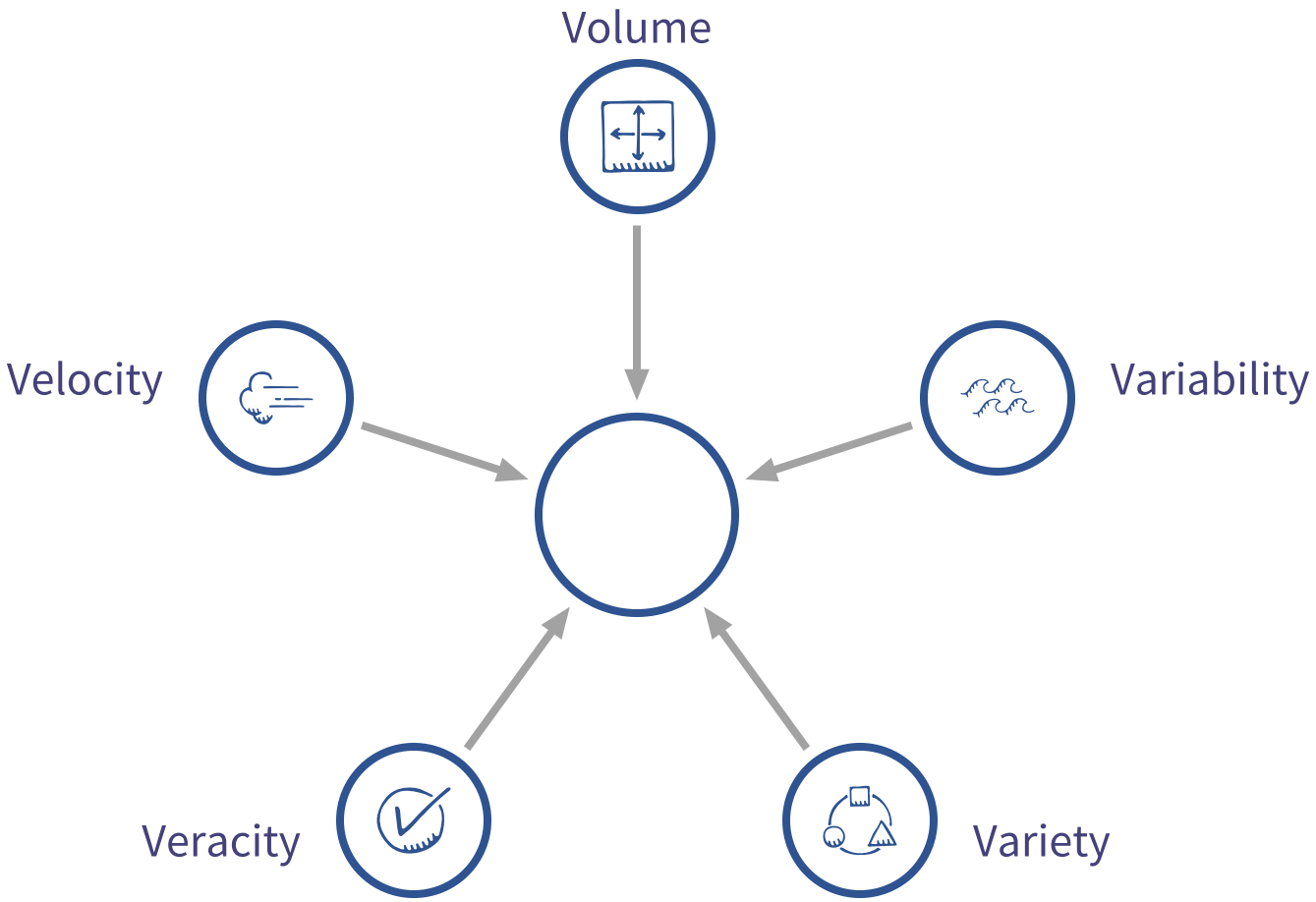
Big Data

Major Technologies

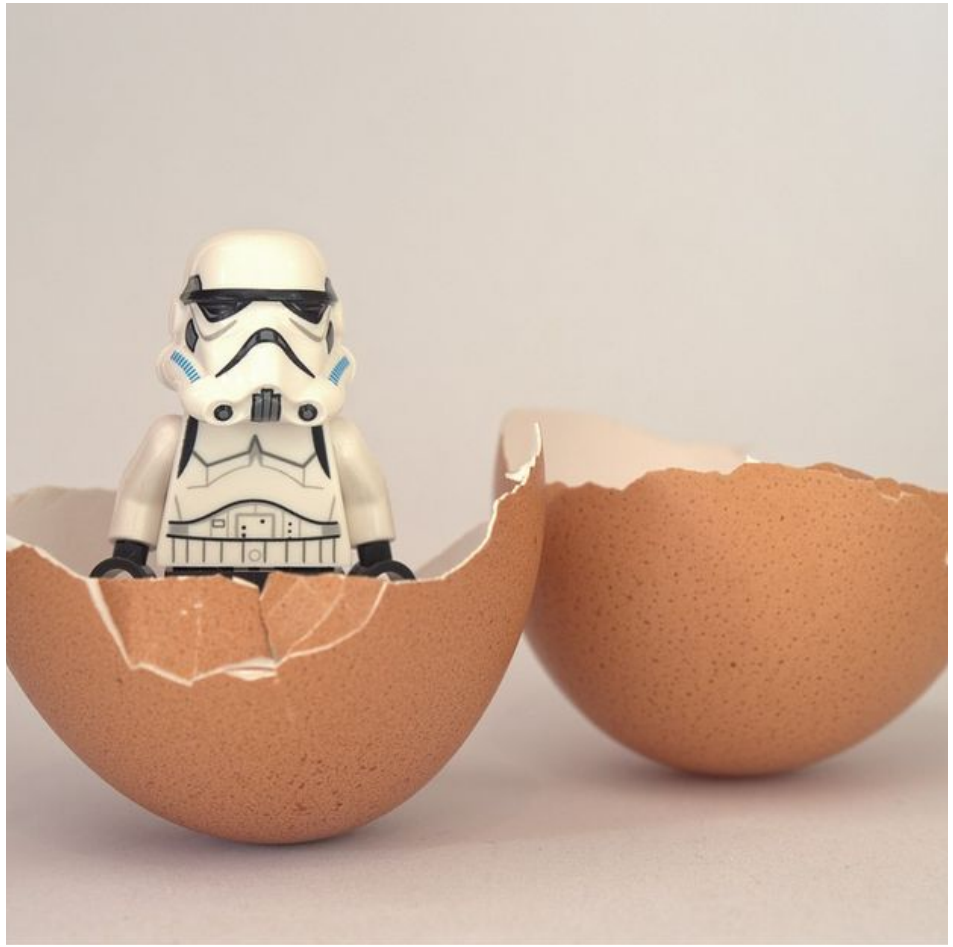
BIG DATA & AI LANDSCAPE 2018



The V's



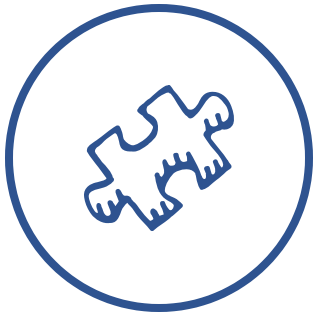
What's different?



Traditional approaches break down. No matter how big and powerful your servers are they can't keep up.

Instead of processing your data through big servers as fast as possible, split the data across hundreds of servers and process different bits in different places

Big Data Steps



Distributed file system

Data is kept fragmented across a cluster of systems

Each node in a cluster has an incomplete copy of the data

The same piece of information will be stored in multiple servers

Computational mapping

Any queries need to be sent out to every node in a cluster

Data should be distributed to ensure even compute load

Parallel computation speeds up processing

Results reduced down

Each node returns different results from their dataset

They need to be reduced to a single result

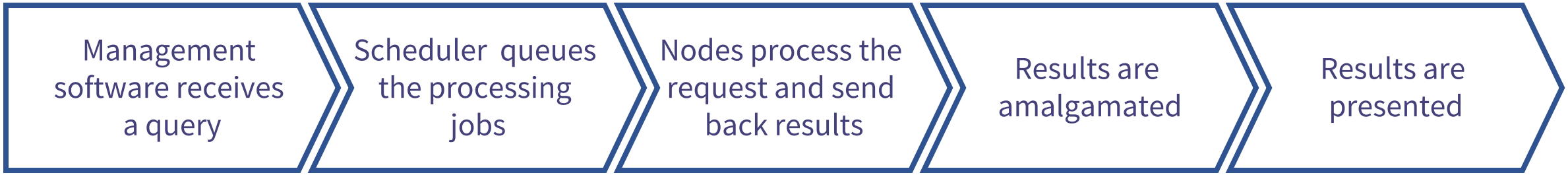
Beware! there be dragons here

Stack Flow

Hive
SparkSQL
Impala
Athena

Docker/Kubernetes
Hadoop

Hive
SparkSQL
Impala
Athena



Yarn
Tez (DAG)
Map(reduce)
SparkEngine

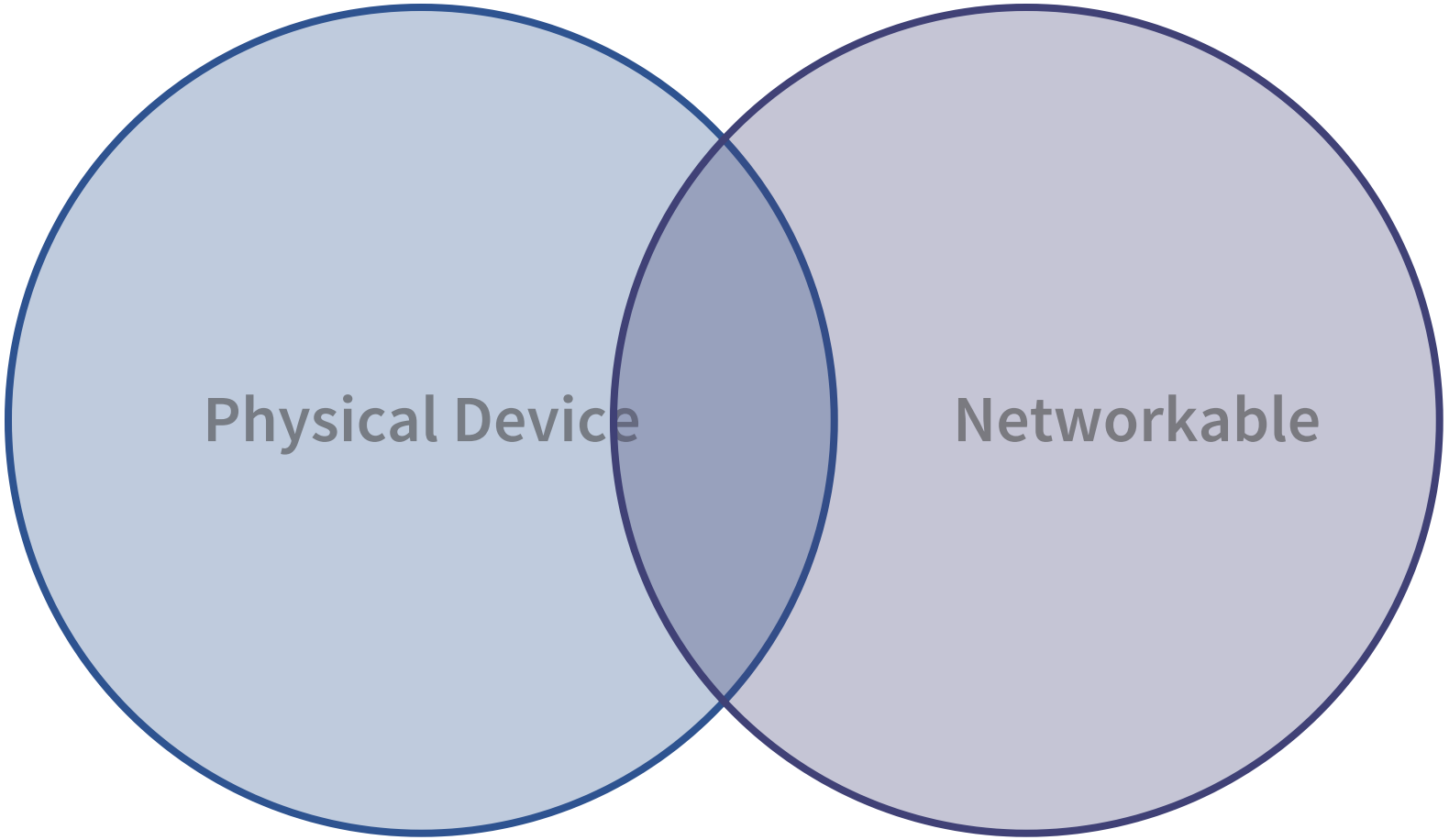
(Map)reduce
SparkEngine

Section 5

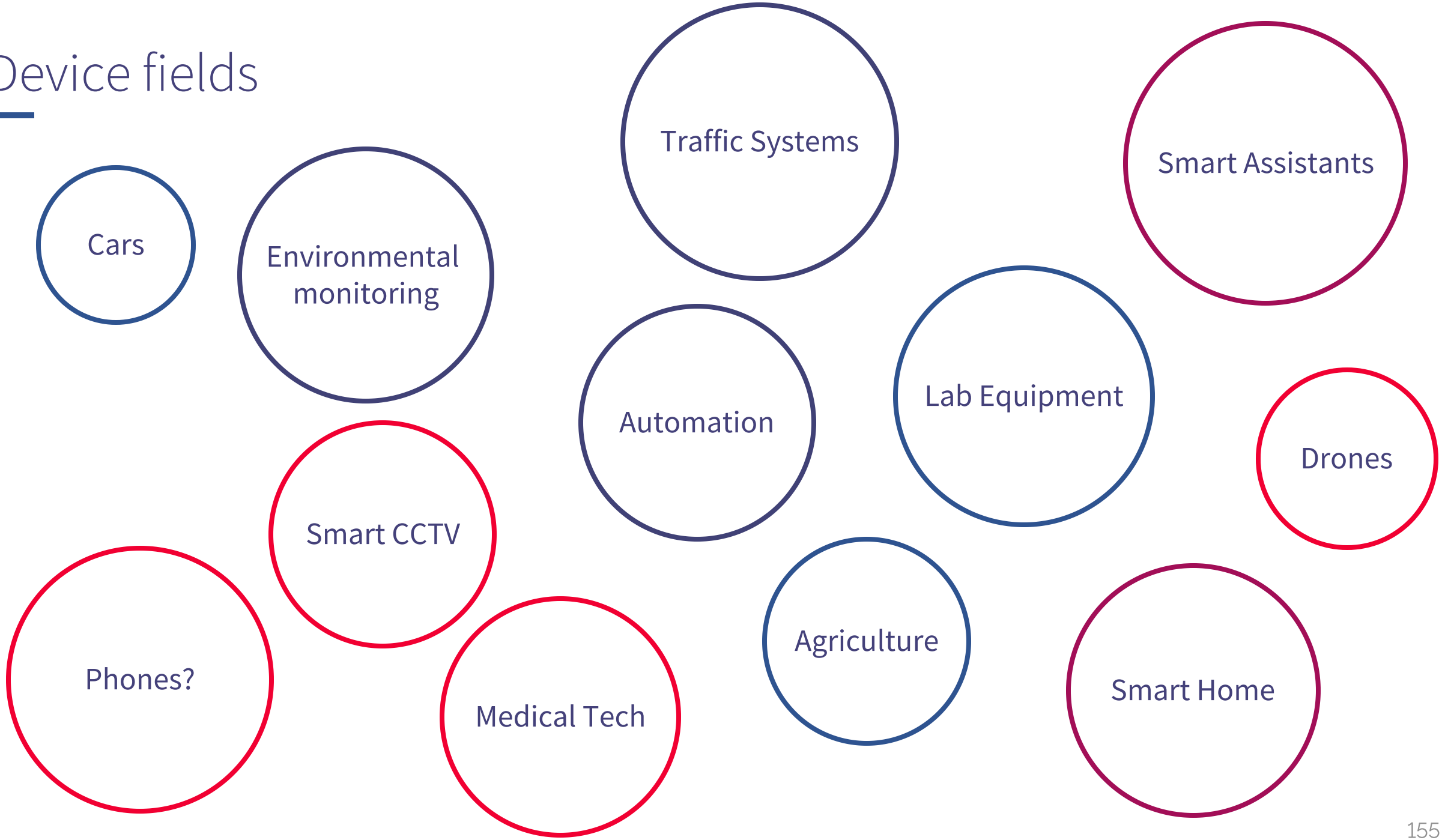
Internet of Things

What is IoT?

IoT Devices



Device fields



Transmission

Collected

Batch

Stream

Received Data

Location Data

RTK

GPS/GLONASS/Galileo

Data Transmission

WiFi

GSM

User Input

Voice commands

Interaction

Break


WEAR THIS BIOSENSOR SO MANAGEMENT CAN MONITOR YOUR HEALTH DURING THE DAY.
DilbertCartoonist@gmail.com
Dilbert.com
WOW. I DIDN'T KNOW YOU CARED SO MUCH ABOUT MY HEALTH.
OH, I DO.
EMPLOYEE 479 DOESN'T HAVE SHALLOW BREATHING. YOU CAN GIVE THAT ONE SOME MORE WORK.
8-15-13 ©2013 Scott Adams, Inc./Dist. by Universal Uclick

Section 6

APIs

Application Program Interface

- Platform for providing and receiving information
- Other applications don't have to interact with a service on a code level
- Only exposes part of a system outwards
- can be used to transfer data or action commands

API examples

Phone Camera

Rather than make their app work with each phone's camera code, apps instead connect to an API that provides the image data.

Arduino

Much of the servo/stepped motor control runs through an API that accepts simple code requests that are converted into more assembly/complex machine code

Web service

To access data held on an online server, programs can call to a web API that provides the data they need in a structured way, and manages the service.

Web (Services) API

- **RESTful (Representational State Transfer) APIs**

  The most common type of API for larger data providers

  Requests can all be driven through HTTP web links

  Often these providers have their own libraries for requests

  Resources sit ready for being visited by the right URI (Unique Resource Identifier)

  Often dumps data in JSON format

  e.g. Met Office data

- **SOAP (Simple Output Application Protocol)**

  Less common for data providers

  More complex and less uniform than REST

  The protocol allows operations to be driven, rather than just simple data recall

  Older style of API

  Can be ACID compliant - better for transactional data, and can communicate during data transfer

  Supports better security measures

  e.g. Mobile banking application talking to a banking server

Twitter API

Query data

You can search by user, hashtag, geolocation etc...

Control user accounts

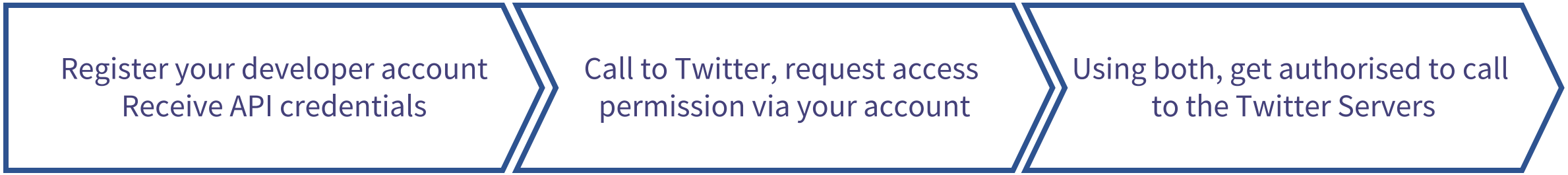
Interact with DMs, anaylse follower networks, post for a user...

Mine data

Pull data by the terrabytes...

Twitter API - OAuth2

Consumer key and secret



Access token and secret

Your turn!

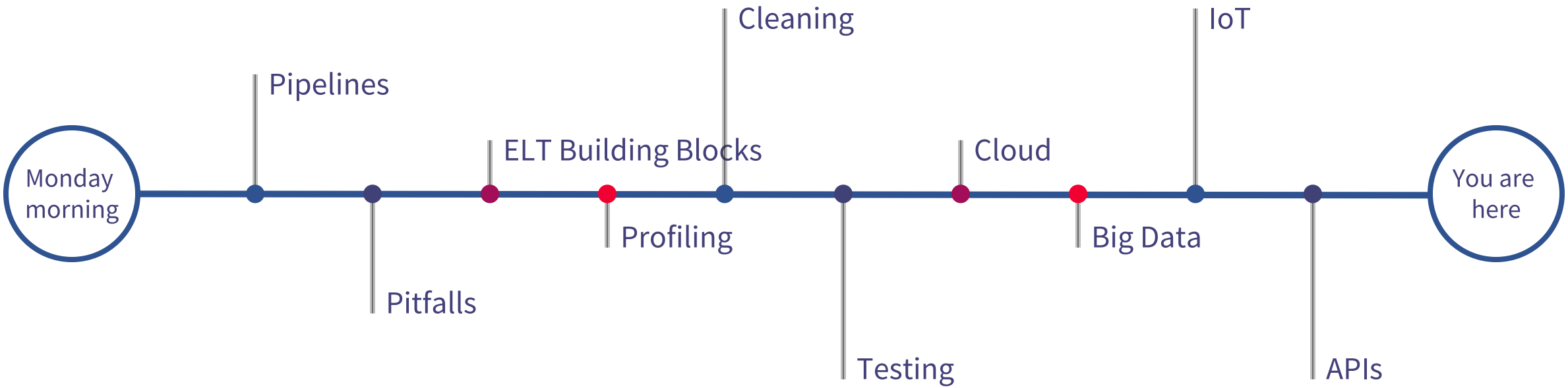
Twitter API lab

- Register with Twitter
- Establish token access
- Run queries
- Pull results
- Analyse!
- **https://github.com/R-Strange/Data_Wrangling_Course_Oxford**

Day 2 Summary

- Pipelines revisited
- Testing your Code
- Cloud Computing
- Cloud Storage
- Big Data
- MapReduce
- Internet of Things
- APIs
- REST
- SOAP

Course


Monday morning
Pipelines
Pitfalls
ELT Building Blocks
Profiling
Cleaning
Testing
Cloud
Big Data
IoT
APIs
You are here

What's next?

eBooks

Manning

O'Reilly

Pragmatic Programmers

Head First

eLearning

Coursera

Udemy

Cloud Guru (AWS Solutions Architect)

Testing

Behavioural Driven Development

Unit Testing

Experiment!

A year of free AWS micro instances

write a few tests in existing code

Build your skills outwards and upwards

You can make an informed decision with your data

https://bit.ly/2IgY3gw

Richard Strange

Earth Sciences 30.17

richard.strange@linacre.ox.ac.uk

075684132854

https://www.linkedin.com/in/richardpstrange/

www.github.com/R-Strange


SPRINTER