Saving 40 minutes in 40 minutes

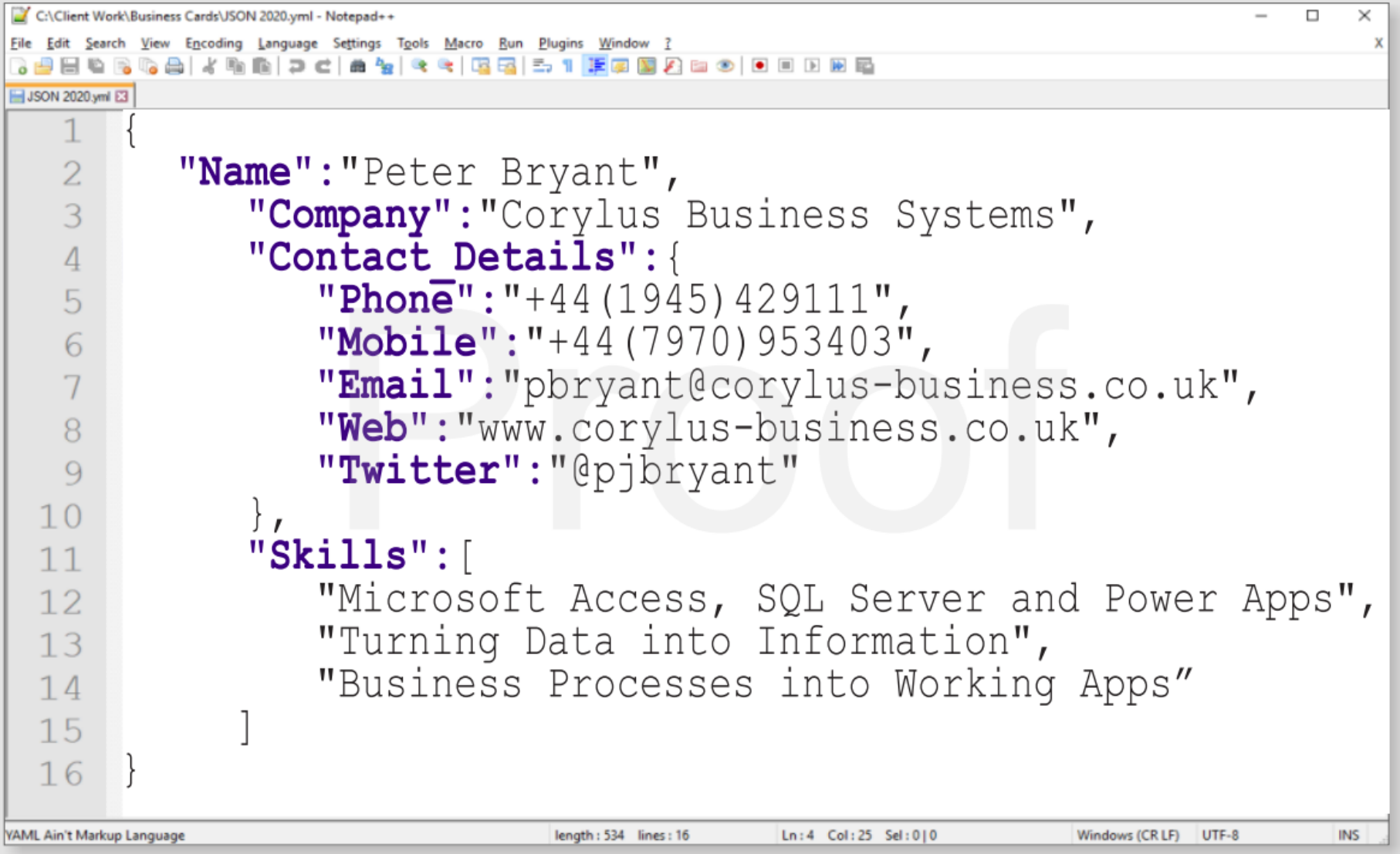
Peter Bryant

Corylus Business Systems

pbryant@corylus–business.co.uk

Me (you can read this later!)

- Started in mainframe/mini development (punched cards and green screens) – COBOL, Algol68R, Fortran
- Mid 80's – came the PC into business
- Used Windows since v2.x and Access developer since the 1st beta (this does not, necessarily, make me an expert!)
- 20+ years of IT Management dabbling in code for internal needs
- Now self–employed, I have three development areas:
  - Turn data into information – long term (pseudo–)agile/prototyping projects
  - Build new systems to replace rubbish or outmoded system
  - Take over maintenance
- That's my excuse list finished ☺


C:\Client Work\Business Cards\JSON 2020.yml - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
JSON 2020.yml
{
"Name":"Peter Bryant",
"Company":"Corylus Business Systems",
"Contact Details":{
"Phone":"+44(1945)429111",
"Mobile":"+44(7970)953403",
"Email":"pbryant@corylus-business.co.uk",
"Web":"www.corylus-business.co.uk",
"Twitter":"@pjbryant"
},
"Skills":[
"Microsoft Access, SQL Server and Power Apps",
"Turning Data into Information",
"Business Processes into Working Apps”
]
}
YAML Ain't Markup Language
length : 534 lines : 16
Ln : 4 Col : 25 Sel : 0 | 0
Windows (CR LF)
UTF-8
INS

Agenda

- Client scenario
- What was needed
- How did I approach it
- What was done
- How does the user like it?
- How much time does it really save – is it 40 minutes?
- A PS – Dealing with the public sector

Client scenario

- This project started pre-Covid
- User is a hospital trust
- Application written by someone a long time ago. Also – they were after a SQL upgrade (thus MUST, ∴ me)
- But wanted improvements to the system too (mostly quick changes or new columns in the database)
- But – their reporting was a nightmare.
  - Their process was a custom code extract to Excel
  - It takes a long time.
- As this is NHS, I'm going to be lighter on demo & data than usual, and heavier on slides, code & discussion (hopefully this betters suits the online world anyway)

The medical process

- Patient put onto one of several possible courses of treatment
- Patient can do more than one course
- They have up to three medical assessments for each course:
  - Initial
  - End of course
  - 6 month follow up
- Each assessment involves a LOT of data.
- The database to support this is used for both auditing of the treatments, and afterwards for research into protocols and outcomes.

The reporting issues

- The database is structured poorly.
  - Patient Information – 79 columns
  - A few reference tables
  - One table for each of the three assessment types, with the following field counts (and all need a few more to extract)
    - Initial – 302 columns
    - End – 238 columns
    - 6 Month – 253 columns
  - Each of the three tables is indexed by Patient id, course number, and date of assessment (in theory assessments can be performed more than once but in practice).
  - There was no identity column in the tables.

How was reporting done?

N.B. the original developer was an enthusiast and competent amateur, who was helping them move from Excel. Nothing below implies or should be used to infer any criticism of that person.

- The database was copied in full using local tables (to ensure consistency)
- Some work was done to work out the references
- Excel was opened and the data dumped programmatically
- This was to overcome the 255 column limit on queries and therefore export to excel.
- The user then used filtering on Excel columns to identify patient/course records of interest – and delete the rest…

Old Code

- This is an approximation and simplified for slides
- Some code to create the excel sheet and some header data
- Each of the four tables was opened in turn.
  - Iterate through all the fields in the record set to get the field names
  - Iterate through every record and every column to get the data value
  - For certain data types, change the formatting in the cell

Old Code (run across 4 tables)

- Code to generate the column headings
- Then code to generate the data
- Then code to format the data

```
'Generate column headings
For intCol = intColStart To intColStart + intNumberOfColumns
  .Cells(1, intCol) = rst.Fields(intCol - intColStart_Init).Name
Next intC

'Generate data
While Not rst.EOF
    intRow = intRow + 1
    For intCol = 1 To intNumberOfColumns
        .Cells(intRow, intCol) = rst(intCol - 1)
        'format the data
        If IsDate(rst(intCol - 1)) Then
          .Cells(intRow, intCol).NumberFormat = "dd/mm/yyyy"
        ElseIf InStr(rst(intCol - 1).Name, "CourseNo") <> 0 Then
          .Cells(intRow, intCol).NumberFormat = "0"
        ElseIf InStr(rst(intCol - 1).Name, "CourseSessionsAttended") <> 0 Then
          .Cells(intRow, intCol).NumberFormat = "0"
        ElseIf IsNumeric(rst(intCol - 1)) Then
          .Cells(intRow, intCol).NumberFormat = "0.00"
        End If
        If IsNumeric(rst(intCol - 1)) Then
          .Cells(intRow, intCol).HorizontalAlignment = xlRight
        End If
    Next intC
    rst.MoveNext
Wend
```

Project Scope

- New columns, new reference tables
- Reporting to be “improved”
  - Faster
  - Fewer records output
  - Fewer columns output

How did I approach it

- To give myself time…
- We phased the project
  - SQL Upgrade
  - New Columns in data base
  - New fields on forms
  - Efficiency improvements on forms (filtered queries, not whole database)
- MUST to the rescue which achieved a lot of the above
- Replaced badly formed patient lookup to a drop down list from a custom (and much faster) SQL view
- Performance improved to sufficiently near instantaneous
- All this was delivered and tested whilst I played with reporting.

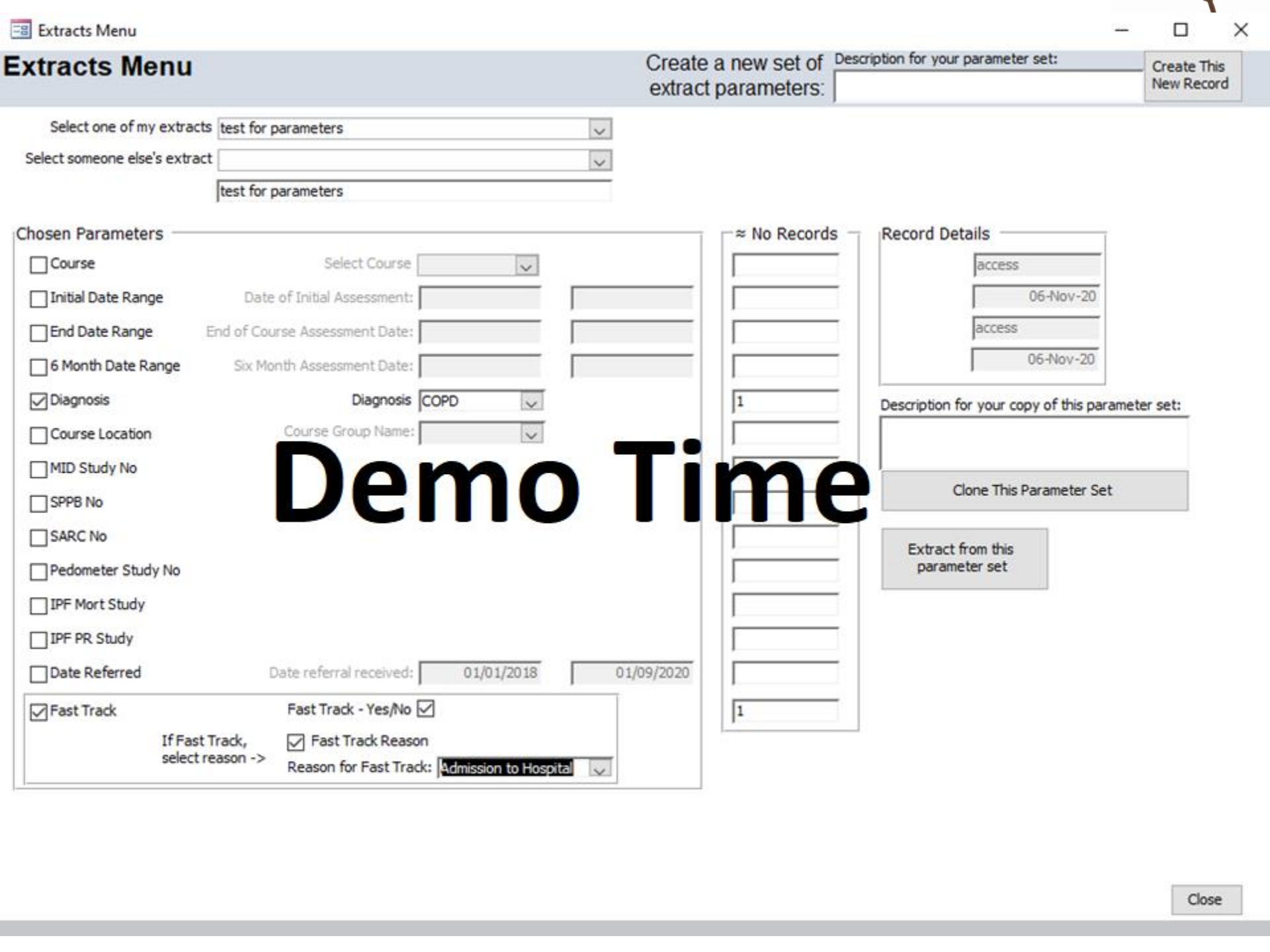
What was done

- Firstly you have to accept the 255 column limit. I'd been hoping to find some snazzy way to get around it. But the original developer was right. You have to iterate through columns in a recordset*
- There were two things
  - Improve the creation of the spreadsheet
  - Filter the data *before* the export – all the users to pre-select the criteria before generating.

*Unless of course – you know better, but the listserver didn't some months ago (if you do – tell me after!!)

Filtering

- The filtering required was across all tables. But a mix and match approach
- The concept of a filter parameters table was created (so users could share criteria).
- They could choose up to 15 criteria on which data was filtered
- These criteria were from across all three assessment data sets and Patient Information. A custom SQL view was used to process this.
- The filter parameters table has two sets of data:
  - Boolean for each parameter – is this used in this extract
  - Values
- A screen grab to explain


Extracts Menu
Extracts Menu
Create a new set of extract parameters:
Description for your parameter set:
Create This New Record
Select one of my extracts
test for parameters
Select someone else's extract
test for parameters
Chosen Parameters
Course
Select Course
Initial Date Range
Date of Initial Assessment:
End Date Range
End of Course Assessment Date:
6 Month Date Range
Six Month Assessment Date:
Diagnosis
Diagnosis
COPD
Course Location
Course Group Name:
MID Study No
SPPB No
SARC No
Pedometer Study No
IPF Mort Study
IPF PR Study
Date Referred
Date referral received:
01/01/2018
01/09/2020
Fast Track
Fast Track - Yes/No
If Fast Track, select reason ->
Fast Track Reason
Reason for Fast Track:
Admission to Hospital
≈ No Records
1
1
Record Details
access
06-Nov-20
access
06-Nov-20
Description for your copy of this parameter set:
Clone This Parameter Set
Extract from this parameter set
Demo Time
Close

End of part 1

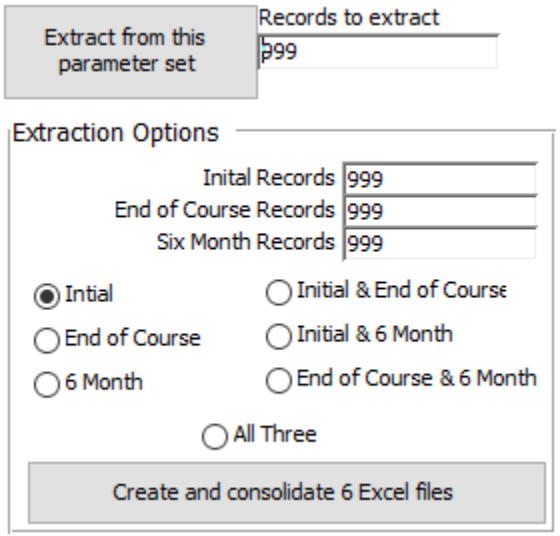
Filtering – now we have criteria…

- What does that extract button do?
- It empties an user filtered temporary extract table which holds ONLY the 15 columns of data being tested for filtering (this may change).
- Uses a custom SQL View to read all those 15 parameter columns records from the underlying database
- Iterates through the SQL View and tests against each extract parameter, allowing me to identify the record as being of interest.
- If a record passes all the tests, it is written out to the temporary extract table. The blnWanted=true is unnecessary, but for future readability.

```
If blnWanted And Me.blnPedometerStudyNo Then
    If Not IsNull(rstSource![PedometerStudyNo]) Then
        blnWanted = True
    Else
        blnWanted = False
    End If
End If
```

Now we have a temp table identifying records of interest

- App displays on the form the number of records from each assessment type that meet the criteria (this allows some sanity checking before pushing the button).
- The users decide if they want to refine the criteria before performing any extract.
- User selects the combination of data they need (the radio checklist).



What does the export do?

- It runs (quickly) up to 6 excel exports that use SQL views with 255 columns or fewer to generate the data we want.
- These views are customised as I use an alias in the SQL View to create the correct column headers.
- An empty spreadsheet is created, and basic header detail created.
- Programmatically I know the column count of each extracted XLSX, and each component has the data and the right column headers.
- Copy and paste all the data into the right location in the new spreadsheet.
- Reuse the original developer's formatting code to get the same output style (keeping the result identical to the old method)
- Quit Access as this prevents Excel staying hidden in Task Manager (see listserver traffic on this)
- Over to second demo and some code!

End of part 2

Does the user like it?

- They are still in final testing, but the initial feedback is..
- WOW that was fast!
- AND I get to filter things before I start with just a few simple clicks????
  - (I think this is going to go down well with the users.)
- The best thing – because we store the parameters in a reusable table – search criteria can be shared and reused by the team (previously no one knew why rows or columns were deleted)
- Their research in the coming years will allow them to focus on the data, not the tiresome task of generating a spreadsheet from which to start.
- Criteria are not complete, but an acceptable compromise

How much time does it really save – is it 40 minutes?

- Actually – no.
- My new extract process itself takes about a minute, the old one currently takes 2 hours
- Remember they then had to filter the data in Excel to subset it to the data they need to examine? This can take many more hours.
- So a conservative estimate is 4-5 hours, sometimes up to a whole day.
- Their research in the coming years will allow them to focus on the data, not the tiresome task of nearly a day's work to get one record set from which to start.

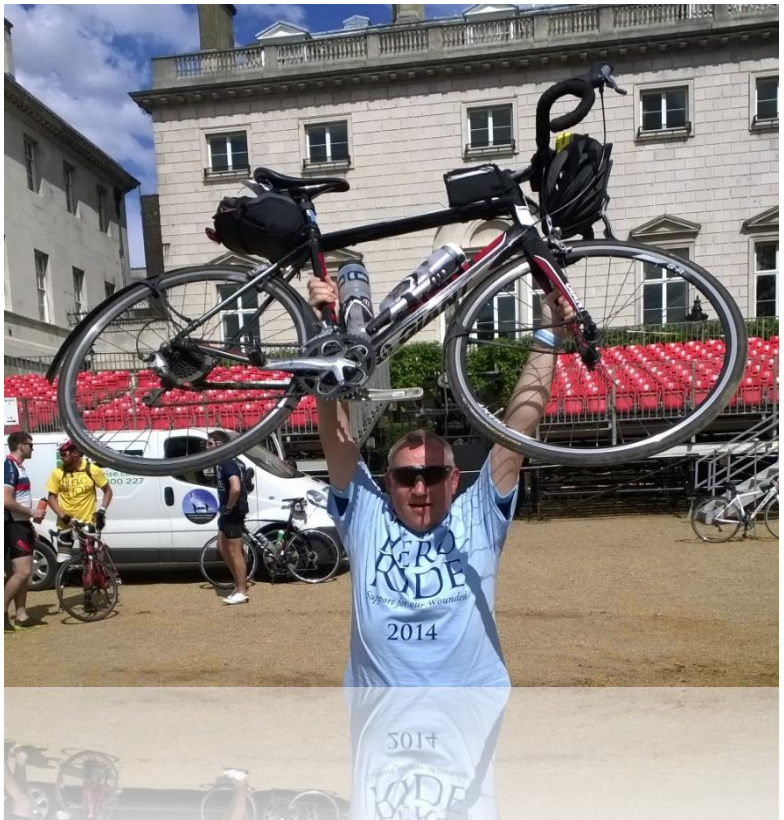
A PS – Dealing with the public sector

- This may been the hardest part of the process!!
- It can take ages to get through the system to get to a "Yes, we want to proceed with the project"
- It can take just as long to get a PO
- It can take just as long to get a hard pressed IT team to do bits for you (SQL changes) that aren't part of their day to day job
- Covid-19 – you may have heard of it, changed things this year, a tad.
- If you want to see my April DevCon presentation on PowerApps and Access:

  https://bit.ly/PowerApps-Access

Help for Heroes

- I’m doing my 9th bike ride (this year’s was cancelled) – 350 miles along the whole of the British Western Front in June next year.
- If you would like to donate – https://bit.ly/PJB2021



Saving 40 minutes in 40 minutes

Business Card

Name
Peter Bryant

Company
Corylus Business Systems

Contact Details
phone: +44 (1945) 429111
mobile : +44 (7970) 953403

email: pbryant@corylus-business.co.uk
web: www.corylus-business.co.uk

linkedin: pjbryant
twitter: @pjbryant

Skills
integration:
Microsoft Access, SQL Server and Power Apps

data:
Turning Data into Information

projects:
Business Processes into Working Apps