University of Sheffield

Wrangling big health data in R

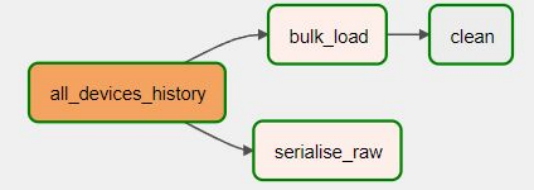
Data engineering for sensitive health data

22nd April 2024

My first R project

Why R there so many puns?

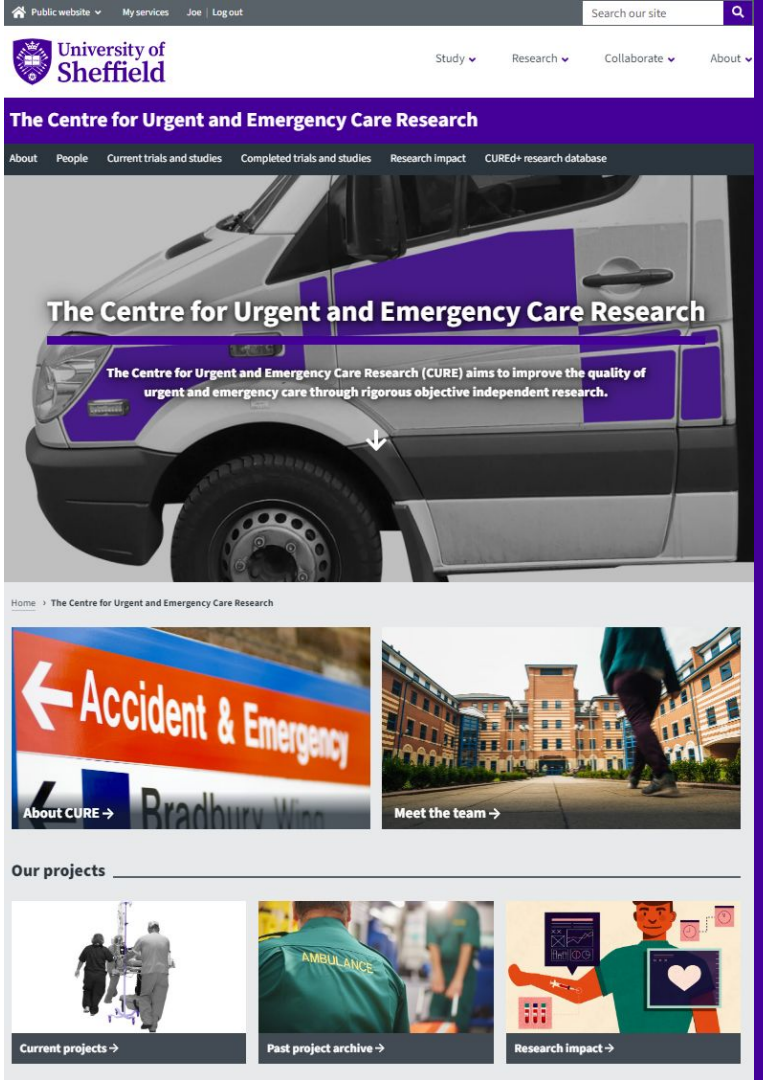
- My background:
  - Physics research
  - Data engineering & analytics
  - C++/C# and Python
  - Research & Innovation IT [Data Analytics Service]
- New skills:
  - R packaging
  - Documenting R code
  - Different ecosystem of data processing libraries, etc.



CUREd+ research database

Improving the quality of urgent and emergency care

- CUREd+ research database run by Sheffield Centre for Health and Related Research (SCHARR)
- Making **anonymous health data available** to researchers
- Patient cohort: **all England** covering 12 years of health events
- **Data sources**:
  - Yorkshire Ambulance Service (YAS)
  - NHS England Digital
- **Data extracts** made available in a secure research environment



A WORLD TOP 100 UNIVERSITY

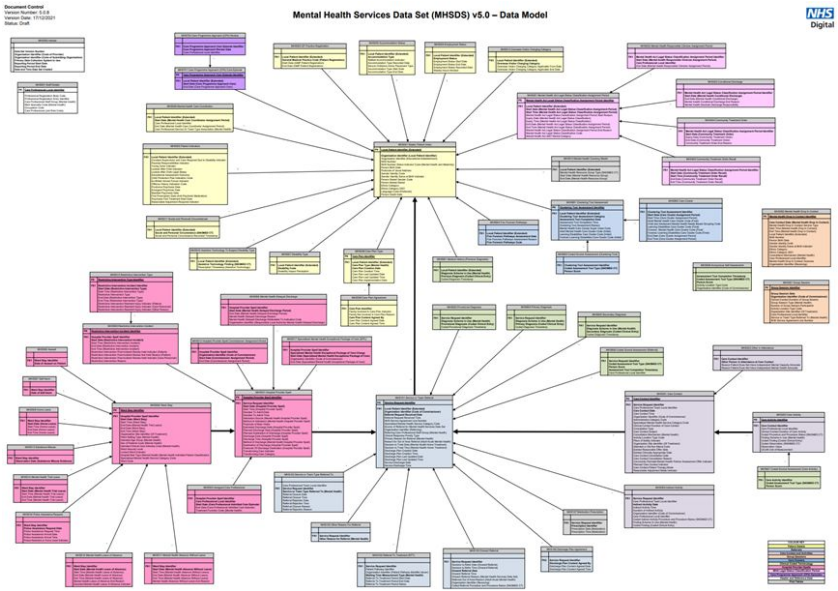
One challenge—lots of raw data

What are we working with?

- **15 data sets** containing information about various aspects of the health service
- Over **3.2 billion rows** of ambulance calls, A&E visits, outpatient stays, etc.
- Over **1,108 columns of data** of various types, including medical codes and identifiers
- **1,465 GB of raw data** in CSV and RDS format
- Largest data set (hospital outpatient records)
  - 586 GB of CSV files covering 12 years
  - 1,256,416,191 outpatient visits/consultations (1.3 billion)

Another challenge—complexity

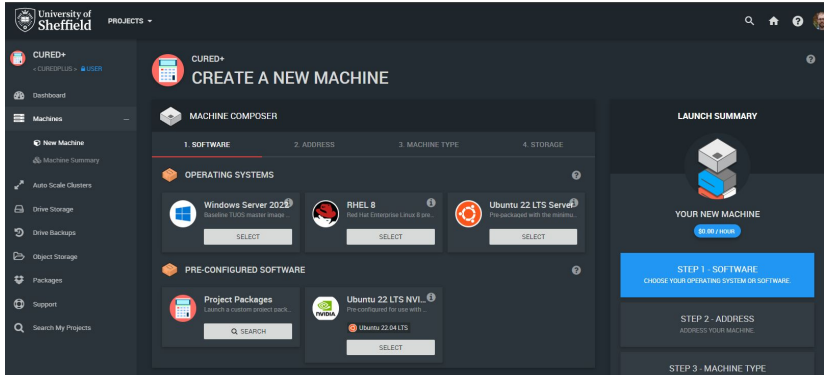
- Expert **medical knowledge** required
  - (I focus on the technical side)
- Messy data
- Linking different data sources
- Lots of interrelated tables
- Anonymous identifiers
- Strict regulations on removing personal information
- Example: [Mental Health Services Data Set (MHSDS)]



Trusted Research Environments

Secure Data Service at the University of Sheffield

- Secure Data Service
- Secure **cloud platform**—RONIN
  - VPN, MFA, etc.
  - Semi-isolated cloud environments (with internet) with virtual machines
- No data egress



Secure Data Service

Find out how to keep sensitive research data safe.

Doing research with sensitive data

If you have any concerns about the sensitivity or security of data on your research project, email **research-it@sheffield.ac.uk** to discuss this or book an appointment.

We can provide advice on how to complete agreements with data providers, and we can direct you to the most appropriate platform to host your research project.

Secure Data Service cloud platform

The Secure Data Service maintains a secure, cloud-based platform for researchers working with sensitive data.

The platform is accredited to the NHS Data Protection Security Toolkit and certified to ISO.IEC 27001:2022. If your project requires additional security controls, please let us know and we may be able to help.

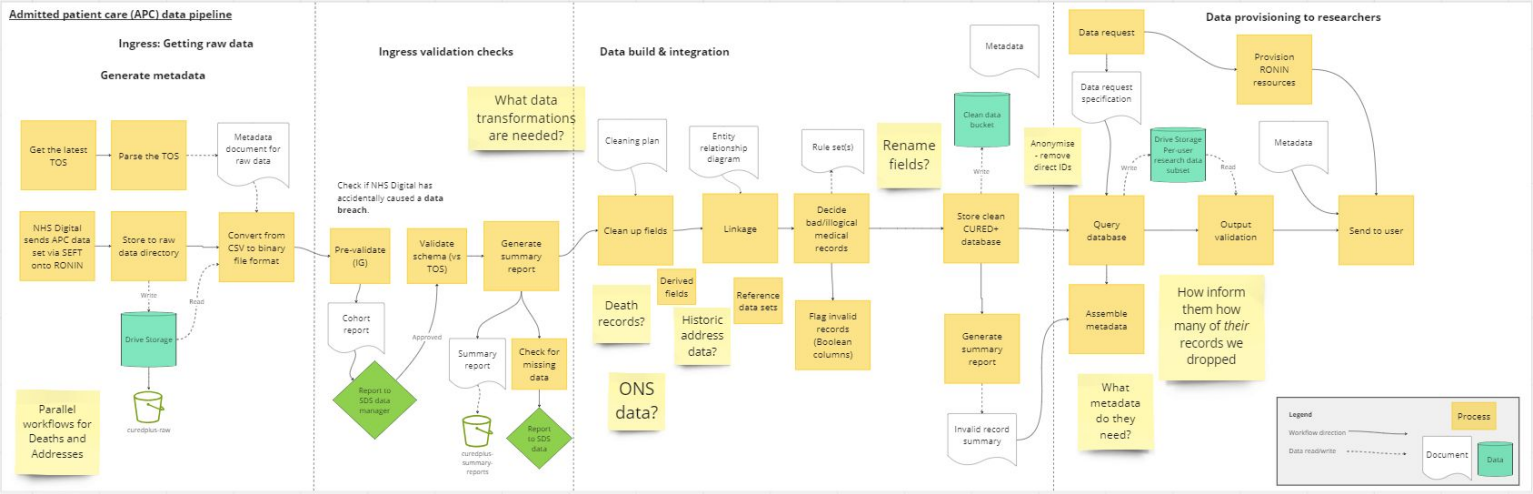
Who can use the Secure Data Service

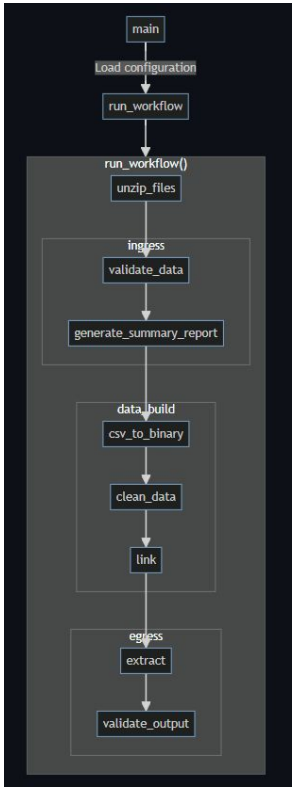
You can apply for support from the Secure Data Service if you're a researcher at any level (including postgraduate researchers) at the University of Sheffield, and you are doing research with sensitive data.

Data pipelines in R

The general approach

- Simple approach—it's **just functions**
  - (Not a workflow management system)
- Functions take **directories** and **metadata files** as inputs and outputs
- Linear(ish) sequence of data operations



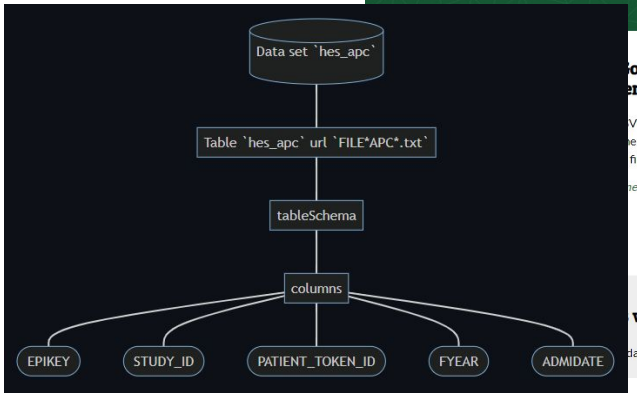


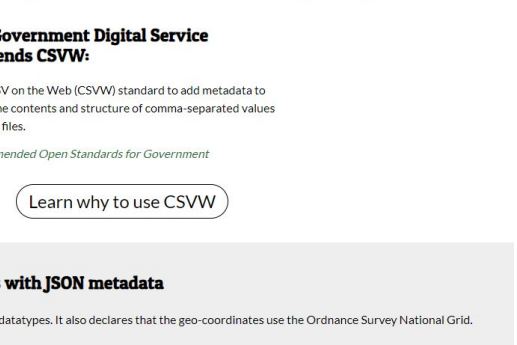
CSV on the Web

Open metadata standard

- Raw data described using CSV on the Web (CSVW) format
  - JSON documents
- [Adopted by gov.uk]
- Data sets with multiple tables with different schemas

```
{
    "@context": [
        "http://www.w3.org/ns/csvw",
        {
            "@language": "en"
        }
    ],
    "dialect": {
        "header": true,
        "delimiter": "|"
    },
    "id": "hes_apc",
    "notes": "Hospital Episide Statistics (HES) Admitted Patient Care (APC) data set",
    "tables": [
    {
      "id": "hes_apc",
      "notes": "https://digital.nhs.uk/data-and-information/data-tools-and-services/data-services/hospital-episode-statistics",
      "url": "*HES_APC*.txt",
      "tableSchema": {
        "primaryKey": "EPIKEY",
        "columns": [
        {
            "name": "ACSCFLAG",
            "datatype": "integer"
```





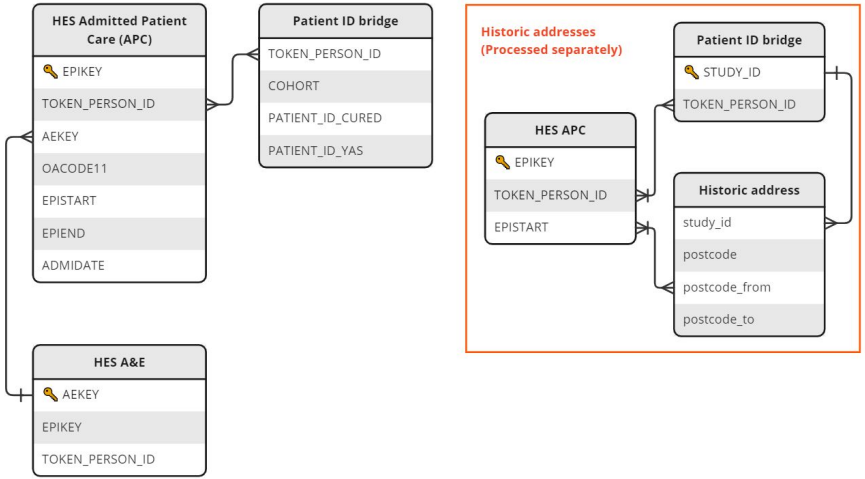
Data modelling

Entity relationship diagrams

- Each table has columns
- Each column has a data type
- Some columns are *primary keys* (unique)
- ERDs describe the relationships between tables

One

Many



R Packaging

"The fundamental unit of shareable, reusable, and reproducible R code" — Wickham

- R Packages (2e) book by Hadley Wickham and Jennifer Bryan
- Dependencies
- Version numbers and GitHub releases
  - Used to specify data build version

```
Package: cuRed
Title: CUREd+: Linking Urgent and Emergency Care Data in Yorkshire and
    Humber Region and Nationwide
Version: 1.1.3
```

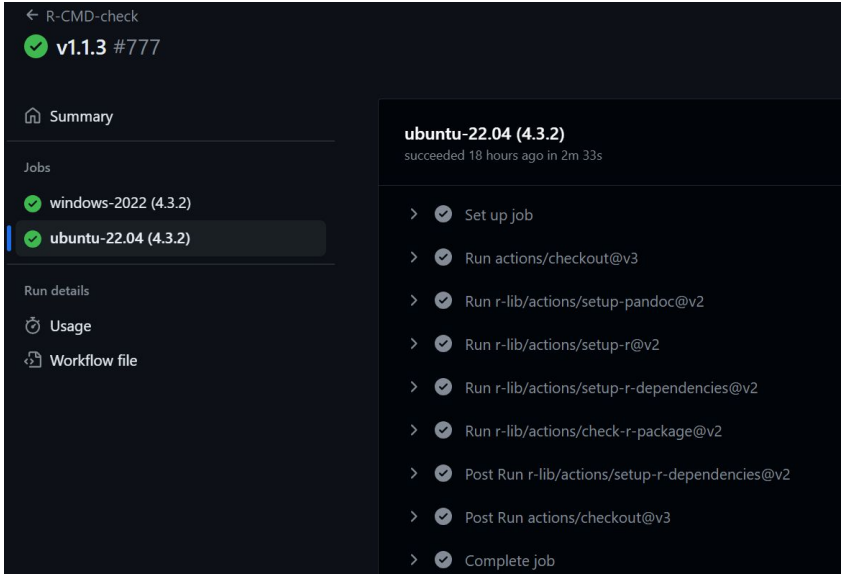

```
> remotes::install_github("CUREd-Plus/cuRed@v1.1.3")

> packageVersion("cuRed")
[1] '1.1.3'
```

Automated testing

Testthat & GitHub actions

- Define unit tests for each function using [testthat]
  - Gives confidence to make changes without breaking things
- Run the tests as part of R CMD CHECK
- GitHub Actions workflow
  - Checks each PR

```r
library(fs)


test_that("link", {
  # Test the link() function

  # Set file paths for the dummy data in this package
  input_path <- extdata_path("data/hes_apc/FILE_HES_APC.parquet", mustWork = TRUE)
  patient_path <- extdata_path("data/patient_id_bridge/patient_id_bridge.parquet", mustWork = T
  demographics_path <- extdata_path("data/pd/pd.parquet", mustWork = TRUE)

  # Count the number of rows in the input data set
  expected_apc_rows <- count_rows(input_path)

  # Create a temporary working directory for this test
  test_dir <- temp_dir()
  # Tidy up (delete temporary files) on failure or exit
  on.exit(unlink(test_dir, recursive = TRUE, force = TRUE), add = TRUE, after = FALSE)

  # Run the data linkage workflow step
  expect_no_error(
```



Practical walk-through

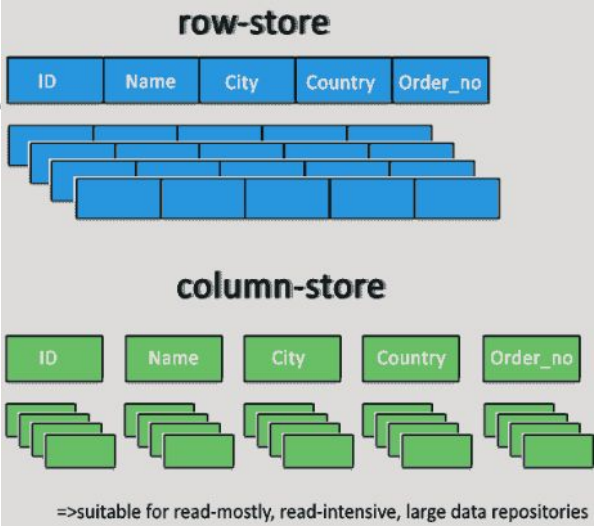
A billion outpatient hospital stays in R

- Data engineering meme "a billion NYC taxi rides" → "a billion outpatient consultations"
- The problem:
  - 12 partitions in CSV format
  - Total size 586 GB
  - No readily-available machine-readable metadata
- A solution: convert to a suitable database format

Columnar storage

File formats designed for analytics databases

- CSV is simple but inefficient at this scale
  - All data stored as unencrypted strings (no data type
  - Row-based storage
  - No indexes
- Apache Parquet column-oriented data storage format
  - Efficient for big data
  - Easy to use
  - Widely supported
  - Compression



https://www.heavy.ai/technical-glossary/columnar-database

Artificial data pilot

NHS England Digital

Artificial data pilot - NHS England Digital

- Statistically realistic dummy data
- Useful for testing (but not fully accurate)
- Data sets broken up into partitions (chunks)

```
joe@TEN6C2B59ED981F:/mnt/c/Users/cs1jsth/Downloads/artificial_hes_op_202302_v1_sample/artificial_hes_op_202302_v1_sample$ ls -lh
total 89M
-rwxrwxrwx 1 joe joe 4.1M Apr 19 15:40 artificial_hes_op_0304.csv
-rwxrwxrwx 1 joe joe 4.1M Apr 19 15:40 artificial_hes_op_0405.csv
-rwxrwxrwx 1 joe joe 4.1M Apr 19 15:40 artificial_hes_op_0506.csv
-rwxrwxrwx 1 joe joe 4.1M Apr 19 15:40 artificial_hes_op_0607.csv
-rwxrwxrwx 1 joe joe 4.2M Apr 19 15:40 artificial_hes_op_0708.csv
-rwxrwxrwx 1 joe joe 4.3M Apr 19 15:40 artificial_hes_op_0809.csv
-rwxrwxrwx 1 joe joe 4.3M Apr 19 15:40 artificial_hes_op_0910.csv
-rwxrwxrwx 1 joe joe 4.6M Apr 19 15:40 artificial_hes_op_1011.csv
-rwxrwxrwx 1 joe joe 5.0M Apr 19 15:40 artificial_hes_op_1112.csv
-rwxrwxrwx 1 joe joe 5.0M Apr 19 15:40 artificial_hes_op_1213.csv
-rwxrwxrwx 1 joe joe 5.0M Apr 19 15:40 artificial_hes_op_1314.csv
-rwxrwxrwx 1 joe joe 5.0M Apr 19 15:40 artificial_hes_op_1415.csv
-rwxrwxrwx 1 joe joe 5.1M Apr 19 15:40 artificial_hes_op_1516.csv
-rwxrwxrwx 1 joe joe 5.1M Apr 19 15:40 artificial_hes_op_1617.csv
-rwxrwxrwx 1 joe joe 5.1M Apr 19 15:40 artificial_hes_op_1718.csv
-rwxrwxrwx 1 joe joe 5.1M Apr 19 15:40 artificial_hes_op_1819.csv
-rwxrwxrwx 1 joe joe 5.1M Apr 19 15:40 artificial_hes_op_1920.csv
-rwxrwxrwx 1 joe joe 5.1M Apr 19 15:40 artificial_hes_op_2021.csv
-rwxrwxrwx 1 joe joe 5.1M Apr 19 15:40 artificial_hes_op_2122.csv
```

Arrow

Big data processing library

- Work with **multi-file datasets** *without* loading them into memory
- Runs in parallel
- Guesses data types automatically (⚠️ caution)
- Uses dplyr syntax to define data operations (from tidyverse)

```
install.packages(c("arrow", "dplyr"))

data_set <-
arrow::open_dataset("artificial_hes_op_202302_v1_sample",
format="csv")
```

Aggregate queries

dplyr syntax to count the number of values

```
data_set %>%
  group_by(SEX) %>%
  summarise(n()) %>%
  collect()
```

Data cleaning

Replace numeric codes with text values

```
data_set <- data_set %>%
  mutate(SEX = case_when(
    SEX == 1 ~ "Male",
    SEX == 2 ~ "Female",
    SEX == 9 ~ "Not Specified",
    SEX == 0 ~ "Not Known",
    TRUE ~ SEX  # Keep other values unchanged
  ))
```

File format conversion

```
data_set %>%
  arrow::write_dataset("output_data", format="parquet")
```

DuckDB

In-process database

About DuckDB

- Doesn’t require a server
- Data operations are defined using Structured Query Language (SQL)
- Parallelisation is handled for you

Advantages for this project

- SQL is relatively simple
- It can read handle our data (read all 12 CSVs and convert them to Parquet format)

Inspect the data

View table schema

- Automatically detect data types

Non-standard syntax

```
C:\Users\cs1jsth\Downloads\artificial_hes_op_202302_v1_sample\artificial_hes_op_202302_v1_sample>duckdb
v0.10.2 1601d94f94
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
D describe select * from '*.csv';
```

column_name varchar	column_type varchar	null varchar	key varchar	default varchar	extra varchar
FYEAR	VARCHAR	YES			
PARTYEAR	BIGINT	YES			
PSEUDO_HESID	VARCHAR	YES			
ATTENDKEY	BIGINT	YES			
ATTENDKEY_FLAG	BIGINT	YES			
ADMINCAT	VARCHAR	YES			
APPTAGE	DOUBLE	YES			
APPTAGE_CALC	DOUBLE	YES			
APPTDATE	DATE	YES			

Count the rows

```
D SELECT COUNT(*) FROM '*.csv';
100% ?                                                                    ?

┌──────────────┐
│ count_star() │
│    int64     │
├──────────────┤
│       190000 │
└──────────────┘
```

Aggregate queries

Summarise

- Group
- Count rows

```
D select sex, count(*) from '*.csv' group by 1;
100% ?                                                                    ?
```

SEX int64	count_star() int64
0	759
9	1052
2	104408
1	83781

Data processing

- Various SQL functions
  - Dates & times
  - Strings
  - Maths
  - Etc, etc.

```
D SELECT YEAR(apptdate), COUNT(*) AS appointment_year FROM '*.csv' GROUP BY 1 ORDER BY 1;
100% ?                                                                                    ?
```

year(apptdate) int64	appointment_year int64
2003	7447
2004	10072
2005	9848
2006	10146
2007	9934
2008	10007
2009	9936
2010	10099
2011	10016
2012	9963
2013	10025
2014	10022
2015	9940
2016	10005
2017	10092
2018	9937
2019	10006
2020	9751
2021	12754

19 rows 2 columns