Drug prescription analysis by NHS practices

Ricard Solé Casas

June 16, 2017

Summary

This report contains a brief analysis on drug prescription by NHS practices during the period of January and February of 2016.

Declaration

I confirm that the submitted coursework is my own work and that all material attributed to others (whether published or unpublished) has been clearly identified and fully acknowledged and referred to original sources. I agree that the College has the right to submit my work to the plagiarism detection service. TurnitinUK for originality checks.

Acknowledgements

I'd like to thank my partner Shannon for her continued support and challenges that help me grow, both professionally and personally. I would also like to thank all of you who also helped me get here.

Contents

Chapter 1

Introduction

1.1 Goal

The goal of this analysis is to get a better grasp on drug prescriptions per NHS practices and expenditure per practice over the months of January and February of the year 2016.

1.2 Tasks

We will download and load CSV files containing data for registered chemicals, practices, prescriptions, and patients over the specified time periods.

The analysis will cover:

- The amount of practices in a specific area, along with its registered patients.
- Prescription amounts for some specific medicines/chemicals.
- Estimated expenditure per practice in prescriptions.

1.3 Further materials and remarks

The entire source code for this report in markdown and latex format and the SQL queries to load data, create views and analyze the data that used in the report are publicly published on Github[1] under the MIT License[2].

1.3.1 Running

I provided a utility script in the Github repository to `setup` and`run-queries` on your local machine:

Disclaimer: *Assumes bash CLI, MariaDB w/ password and a working internet connection.*

```
./setup && ./run-queries
```

[1] https://github.com/rcsole/coursework-db
[2] https://choosealicense.com/licenses/mit/

Chapter 2

Starting up

For consistency with the lectures, the RDMS used is MariaDB[1], a fork of MySQL[2]. The OS is Apple's OSX[3], and the package manager to install MariaDB we used for this exercise is homebrew[4]. I chose MariaDB over MySQL because I'll be working at Google and that seems to be their fork.

2.1 Installing MariaDB

Using the `brew` command, in our terminal:

```
brew --version
Homebrew 1.2.2
Homebrew/homebrew-core (git revision 73a8655; last commit 2017-06-12)
```

```
brew install mariadb
```

We might be prompted to *secure* the `MySQL/MariaDB` (we will use both names interchangeably from now on), this is accomplished by following the steps asked in this command:

```
mysql_secure_installation
```

2.2 Creating the database

With some really simple steps, we'll be able to create a database. Fortunately, I've created a bash script that will automate the setup on a UNIX system. Here, however, we'll describe it step by step.

```
mysql -uroot -p
```

```
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 78
Server version: 10.2.6-MariaDB Homebrew

Copyright (c) 2000, 2017, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> CREATE DATABASE prescriptionsdb;
Query OK, 1 row affected (0.01 sec)
```

[1] https://mariadb.org/
[2] https://mysql.com
[3] https://www.wikiwand.com/en/MacOS
[4] https://brew.sh

2.3 Modeling the data

Upon inspection of the data, there are **four** CSV files. Three of them found here[5] and the last one here[6]. This maps to also four tables in our database. `practices` —containing the address, names and postcodes of GPs—, `prescriptions` —actual prescribed medicine, cost, GP, and quantity—, `gppatients` —the number oh patients, per age, per GP—, and `chemicals` —extended information on medicaments that are registered by the NHS.

Everything builds on the `practices` table: both `prescriptions` and `gppatients` have a `practiceid` field which will be a `FOREIGN KEY` reference to the `practices` table.

2.3.1 Create practices table

```
CREATE TABLE IF NOT EXISTS practices(
  period INT,
  id VARCHAR(6) NOT NULL PRIMARY KEY,
  practicename TEXT,
  address1 TEXT,
  address2 TEXT,
  city TEXT,
  state TEXT,
  postcode TEXT
);
```

2.3.2 Create prescriptions table

```
CREATE TABLE IF NOT EXISTS prescriptions(
  id BIGINT NOT NULL AUTO_INCREMENT,
  sha TEXT,
  pct TEXT,
  practiceid VARCHAR(6),
  bnfcode VARCHAR(15),
  bnfname TEXT,
  items INT,
  ingredientcost FLOAT,
  actualcost FLOAT,
  quantity INT,
  period INT,
  PRIMARY KEY (id)
);
```

2.3.2.1 Add FOREIGN KEY reference

```
ALTER TABLE prescriptions ADD FOREIGN KEY (practiceid) REFERENCES practices(id);
```

2.3.3 Create gppatients table

```
CREATE TABLE IF NOT EXISTS gppatients(
  practiceid VARCHAR(6),
  patientcount INT
);
```

[5] https://goo.gl/zC3afI
[6] https://goo.gl/n8XbX7

2.3.3.1 Add FOREIGN KEY reference

```
ALTER TABLE gppatients ADD FOREIGN KEY (practiceid) REFERENCES practices(id);
```

2.3.4 Create chemicals table

```
CREATE TABLE IF NOT EXISTS chemicals(
  bnfcodesub VARCHAR(9) PRIMARY KEY,
  chemicalname TEXT
);
```

2.4 Wrapping up

This does it for scaffolding our database and tables. In the next chapter, we'll load the CSV files, and create some useful views for future queries.

Chapter 3

Setting up our data

As described in the previous chapter we'll detail step by step how to load the data from the CSV to our `MariaDB` instance, and into `prescriptionsdb`.

3.1 Fetching the data

The data we'll be working with is from 2016, January and February. It is provided by the government. The easiest way to fetch it is to `cURL` it from the terminal:

```
#!/bin/bash

function fetch_chemicals() {
  cd data
  curl -s -L -o chemicals-01-2016.csv \
    http://datagov.ic.nhs.uk/presentation/2016_01_January/T201601CHEM+SUBS.CSV
  curl -s -L -o chemicals-02-2016.csv \
    http://datagov.ic.nhs.uk/presentation/2016_02_February/T201602CHEM+SUBS.CSV

  echo "Dowloaded chemicals csv files."
  echo "Concatenating chemicals files."

  cp chemicals-01-2016.csv chemicals.csv
  tail -n+2 chemicals-02-2016.csv >> chemicals.csv

  echo "Concatenated chemicals files."

  rm -rf chemicals-{01,02}-2016.csv
  cd ..
}

function fetch_practices() {
  cd data
  curl -s -L -o practices-01-2016.csv \
    http://datagov.ic.nhs.uk/presentation/2016_01_January/T201601ADDR+BNFT.CSV
  curl -s -L -o practices-02-2016.csv \
    http://datagov.ic.nhs.uk/presentation/2016_02_February/T201602ADDR+BNFT.CSV

  echo "Dowloaded practices csv files."
  echo "Concatenating practices files."

  cp practices-01-2016.csv practices.csv
  cat practices-02-2016.csv >> practices.csv
```

```
  echo "Concatenated prescriptions files."

  rm -rf practices-{01,02}-2016.csv
  cd ..
}

function fetch_prescriptions() {
  cd data
  curl -s -L -o prescriptions-01-2016.csv \
    http://datagov.ic.nhs.uk/presentation/2016_01_January/T201601PDPI+BNFT.CSV
  curl -s -L -o prescriptions-02-2016.csv \
    http://datagov.ic.nhs.uk/presentation/2016_02_February/T201602PDPI+BNFT.CSV

  echo "Dowloaded prescriptions csv files."
  echo "Concatenating prescriptions files."

  cp prescriptions-01-2016.csv prescriptions.csv
  tail -n+2 prescriptions-02-2016.csv >> prescriptions.csv

  echo "Concatenated prescriptions files."
  echo "Adding index column to prescriptions."

  rm -rf prescriptions-{01,02}-2016.csv

  awk -v OFS=',' '
    NR == 1 {print "ID", $0; next}
    {print (NR-1), $0}
  ' prescriptions.csv > tmp && mv tmp prescriptions.csv

  "Done fetching prescriptions."

  cd ..
}

function fetch_gppatients() {
  cd data
  curl -s -L -o gppatients.csv \
    http://www.hscic.gov.uk/catalogue/PUB19775/gp-reg-patients-prac-quin-age.csv
  cd ..
}

checksum_chemicals=1b898694c32b96ecd74d85c201cc178e
checksum_practices=3a57b4a1b45a75fdd1eea19676c65691
checksum_prescriptions=bcf3aa7e0a53b5ff11f41e0e6e51dca5
checksum_gppatients=bba2a3aaa86d727dc53beff60007347a

FILES="
chemicals
practices
prescriptions
gppatients
"

mkdir -p data

for f in $FILES
do
```

```
  if [ ! -f "data/$f.csv" ]
  then
    echo ""
    echo "data/$f.csv - NOT FOUND. Downloading."
    echo "Depending on the size of the file(s) this might take a while."
    eval "fetch_$f"
  else
    file_checksum=$(md5 -q data/$f.csv)
    checksum_name="checksum_$f"
    correct_checksum=${!checksum_name}

    if [ $file_checksum == $correct_checksum ]
    then
      echo ""
      echo "data/$f.csv checksum - OK. Skipping download."
    else
      echo ""
      echo "data/$f.csv checksum - NOT OK. Re-downloading."
      echo "Depending on the size of the file(s) this might take a while."
      eval "fetch_$f"
    fi
  fi
done
```

3.2 How to load a CSV file

MariaDB provides an instruction, `LOAD DATA [LOCAL] INFILE`, which lets us dump a file into a table. You can find more information on how that command works on the MariaDB documentation website for `LOAD DATA`[1].

3.2.1 Usage

```
LOAD DATA LOCAL INFILE './data/practices.csv'
INTO TABLE practices
FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n'
(@period, @id, @name, @address1, @address2, @city, @state, @postcode)
SET
  period = @period,
  id = @id,
  practicename = @name,
  address1 = @address1,
  address2 = @address2,
  city = @city,
  state = @state,
  postcode = @postcode
;

LOAD DATA LOCAL INFILE './data/prescriptions.csv'
INTO TABLE prescriptions
FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n' IGNORE 1 LINES
(@id, @sha, @pct, @practice, @bnfc, @bnfn, @items, @nic, @actcost, @qty, @period)
SET
  id = @id, pct = @pct, sha = @sha, practiceid = @practice,
  bnfcode = @bnfc, bnfname = @bnfn, items = @items,
```

[1] https://mariadb.com/kb/en/mariadb/load-data-infile/

```
  ingredientcost = @nic, actualcost = @actcost, quantity = @qty,
  period = @period
;

LOAD DATA LOCAL INFILE './data/chemicals.csv'
INTO TABLE chemicals
FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n' IGNORE 1 LINES
(@bnf, @name)
SET
  bnfcodesub = @bnf,
  chemicalname = @name
;

LOAD DATA LOCAL INFILE './data/gppatients.csv'
INTO TABLE gppatients
FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n' IGNORE 1 LINES
(@practiceid, @i1, @i2, @i3, @i4, @i5, @i6, @i7, @totalpatients)
SET
  practiceid = @practiceid,
  patientcount = @totalpatients
;
```

3.3 Creating views

Before moving on to actually querying our data, I found it useful to look at the specified tasks and possibly create views that would aid said queries. From Wikipedia:

> *"In database theory, a view is the result set of a stored query on the data, which the database users can query just as they would in a persistent database collection object. This pre-established query command is kept in the database dictionary."*

You can find more information on what views are on the MariaDB documentation website about views[2], or on Wikipedia[3].

3.3.1 Beta-blockers

One of the first things asked is to identify beta-blocker[4] prescriptions. The NHS Provides with some information in what some common beta-blockers are. I found it particularly useful to create a view of what prescriptions actually match the chemicals commonly identified as beta-blockers.

```
CREATE OR REPLACE VIEW bb AS
SELECT bnfcodesub
FROM chemicals
WHERE chemicalname REGEXP 'atenolol|bisoprolol|carvedilol|metoprolol|nebivolol|propranolol';


CREATE OR REPLACE VIEW bbprescriptions AS
SELECT practiceid,
       quantity
FROM prescriptions
INNER JOIN bb ON prescriptions.bnfcode LIKE concat(bb.bnfcodesub, '%');


CREATE OR REPLACE VIEW bbpgp AS
```

[2] https://mariadb.com/kb/en/mariadb/views/
[3] https://www.wikiwand.com/en/View_(SQL)
[4] http://bit.ly/2reDxBL

```
SELECT SUM(quantity) AS total,
       practiceid,
       practices.practicename
FROM bbprescriptions
LEFT JOIN practices ON bbprescriptions.practiceid = practices.id
GROUP BY practiceid;
```

3.3.2 Prescriptions-per-x

Other tasks ask for common patterns such as prescriptions per practice, and prescriptions per chemical.

```
CREATE OR REPLACE VIEW ppgp AS
SELECT COUNT(*) AS prescriptioncount,
       practiceid
FROM prescriptions
GROUP BY practiceid
ORDER BY prescriptioncount DESC;


CREATE OR REPLACE VIEW ppc AS
SELECT COUNT(*) AS dispensedamount,
       substring(bnfcode, 1, 9) AS bnfcodesub
FROM prescriptions
GROUP BY substring(bnfcode, 1, 9);
```

3.3.3 Other views

There are some other views that proved to be necessary/useful as queries were being constructed. These will be discussed further with their corresponding queries.

```
CREATE OR REPLACE VIEW spentpergp AS
SELECT SUM(actualcost) AS spent,
       practiceid
FROM prescriptions
GROUP BY practiceid;


CREATE OR REPLACE VIEW ssri AS
SELECT bnfcodesub
FROM chemicals
WHERE chemicalname REGEXP 'citalopram|dapoxetine|escitalopram|fluoxetine|fluvoxamine|paroxetine|se


CREATE OR REPLACE VIEW ssriprescriptions AS
SELECT practiceid,
       quantity,
       period,
       bnfcodesub
FROM prescriptions
INNER JOIN ssri ON prescriptions.bnfcode LIKE concat(ssri.bnfcodesub, '%');


CREATE OR REPLACE VIEW metformin AS
SELECT bnfcodesub
FROM chemicals
WHERE chemicalname LIKE '%metformin%';
```

```
CREATE OR REPLACE VIEW metforminprescriptions AS
SELECT practiceid,
       quantity,
       period,
       bnfcodesub
FROM prescriptions
INNER JOIN metformin ON prescriptions.bnfcode LIKE concat(metformin.bnfcodesub, '%');
```

3.4 Wrapping up

That's all there is to model our data. To recap we have:

- Downloaded the CSV files into a data folder using `cURL`.
- Loaded the CSV files using `LOAD DATA`
- Created some `VIEW`s that we'll be using later to abstract complexity from our queries.

Chapter 4

Data Analysis

4.1 Practices in a particular area

We are particularly interested in *how* many practices —and how many registered patients in said practices— there are in a specific area covered by the NHS —in N17.

Getting the number of practices is an easy task to accomplish with the provided data. From the `practices` table, we want to retrieve the rows whose `postcode` column start with `N17`. However, we also want the patients from that area. That's in the `gppatients` table which relates to `practices` through the `practiceid` column. In this case, we will need a `LEFT JOIN` when combining both tables. The reason we need it to be `LEFT` is that we might find some `practices` which do not have any registered patients but **are** in the *N17* postcode.

```
SELECT COUNT(*) AS 'practices in n17',
       SUM(patientcount) AS 'patients in n17'
FROM practices
LEFT JOIN gppatients ON gppatients.practiceid = practices.id
WHERE postcode LIKE 'N17%';
```

```
-- +------------------+-----------------+
-- | practices in n17 | patients in n17 |
-- +------------------+-----------------+
-- |                7 | 49358           |
-- +------------------+-----------------+
```

From the result of the query we can conclude there are **7 practices in N17** and a total of **49,358** patients are registered in that postcode.

4.2 Beta-blockers

Taking advantage of the views created in section 3.3.1 the query to find the GP which prescribed the most beta-blockers turns out to be a rather simple `JOIN` from that `VIEW` to the `gppatients` table:

```
SELECT bbpgp.practiceid AS 'practice id',
       bbpgp.total / gppatients.patientcount AS betablockersperpatient,
       bbpgp.practicename AS 'practice name'
FROM bbpgp
INNER JOIN gppatients ON gppatients.practiceid = bbpgp.practiceid
ORDER BY betablockersperpatient DESC
LIMIT 1;
```

```
-- +-------------+----------------------+-----------------------+
```

```
-- | practice id | betablockersperpatient | practice name          |
-- +-------------+------------------------+------------------------+
-- | G82651      | 161.0000               | BURRSWOOD NURSING HOME |
-- +-------------+------------------------+------------------------+
```

The practice that has prescribed the most beta-blockers per patient registered (in that specific practice) is **Burrswood Nursing Home**, ID **G82651**, prescribing **161** beta-blockers per patient.

4.3 Prescriptions per medication

The relevant `VIEW` here is `ppc`. It contains how many prescriptions there are per chemical. Doing a simple `JOIN` with the `chemicals` table, sorting it by the dispensed amount and `LIMIT`ing it to `1` result:

> **Disclaimer**: *The assumption is we are looking for the chemical that's been dispensed most often, not necessarily the one that's been dispensed the most as far as actual quantity goes. With the data we could gather that's a rather difficult task (more quantity does not equal more substance).*

```
SELECT chemicalname,
       dispensedamount
FROM chemicals
INNER JOIN ppc ON ppc.bnfcodesub = chemicals.bnfcodesub
ORDER BY dispensedamount DESC
LIMIT 1;
```

```
-- +----------------+-----------------+
-- | chemicalname   | dispensedamount |
-- +----------------+-----------------+
-- | Colecalciferol |          280495 |
-- +----------------+-----------------+
```

The most prescribed medication in January & February of 2016 was **Colecalciferol**, which was prescribed **280495** times.

4.4 Expenditure per practice

Like in every other query we also have a relevant view here, `spentpergp`. All we need to do is join `spentpergp` with `practices` to get the name, and we are done! `spentpergp` is the result of `SUM`ing all `prescriptions` `GROUP`ed `BY` their `practiceid`:

```
SELECT spentperpatient,
       practicename
FROM
  (SELECT spent / patientcount AS spentperpatient,
          spentpergp.practiceid
   FROM spentpergp
   INNER JOIN gppatients ON gppatients.practiceid = spentpergp.practiceid) AS spp
INNER JOIN practices ON spp.practiceid = practices.id
ORDER BY spentperpatient DESC
LIMIT 1;
```

```
-- +------------------+------------------------+
-- | spentperpatient  | practicename           |
-- +------------------+------------------------+
-- | 7609.050047278404 | BURRSWOOD NURSING HOME |
-- +------------------+------------------------+
```

```
SELECT spentperpatient,
       practicename
FROM
  (SELECT spent / patientcount AS spentperpatient,
          spentpergp.practiceid
   FROM spentpergp
   INNER JOIN gppatients ON gppatients.practiceid = spentpergp.practiceid) AS spp
INNER JOIN practices ON spp.practiceid = practices.id
ORDER BY spentperpatient ASC
LIMIT 1;
```

```
-- +---------------------+-----------------------------------------+
-- | spentperpatient     | practicename                            |
-- +---------------------+-----------------------------------------+
-- | 0.013433942387335699 | SCHOOL LANE PMS PRACTICE                |
-- +---------------------+-----------------------------------------+
```

The **biggest spender** was **Burrswood Nursing Home** at **7609.05£** per patient. The **cheapest practice** was **School Lane PMS Practice** at **0.013£** per patient.

4.5 SSRI prescriptions change

There are similar `VIEW`s for SSRI[1] as there were for beta-blocker. To find the difference in the amount of times SSRIs have been prescribed between January and February we simply `COUNT(*)` and `GROUP BY` the `period` when they were prescribed.

To know what qualifies as SSRI we've referred to the NHS website. The relevant excerpt:

- Types of SSRIs
  - citalopram
  - dapoxetine
  - escitalopram
  - fluoxetine
  - fluvoxamine
  - paroxetine
  - sertraline

```
SELECT period,
       COUNT(*) AS 'SSRI prescriptions'
FROM ssriprescriptions
GROUP BY period;
```

```
-- +--------+--------------------+
-- | period | SSRI prescriptions |
-- +--------+--------------------+
-- | 201601 |              99797 |
-- | 201602 |              99311 |
-- +--------+--------------------+
```

The difference between February and January is minimal, at only **486** fewer SSRIs prescribed.

4.6 Metformin per practice

You guessed it right, we are using `VIEW`s! In this case, `metforminprescriptions` which builds on `metformin`. The latter gets all the `bnfcodesubs` belonging to `chemicals` containing `metformin`. We

[1] http://www.nhs.uk/conditions/SSRIs-(selective-serotonin-reuptake-inhibitors)/Pages/Introduction.aspx

use the former to `JOIN` it with `practices` on the `practiceid = id` and `GROUP BY` the `id`, while also `ORDER`ing `BY` the amount of `prescriptions` and finally `LIMIT` it to `10` elements:

```
SELECT COUNT(*) AS prescribed,
       practicename
FROM metforminprescriptions
INNER JOIN practices ON id = practiceid
GROUP BY practiceid
ORDER BY prescribed DESC
LIMIT 10;
```

```
-- +------------+----------------------------------+
-- | prescribed | practicename                     |
-- +------------+----------------------------------+
-- |         45 | DR GHAFOOR & ABBASI              |
-- |         44 | PARADISE MEDICAL CENTRE          |
-- |         44 | SHANTI MEDICAL CENTRE            |
-- |         40 | MIDLANDS MEDICAL PARTNERSHIP     |
-- |         39 | BUXTED MEDICAL CENTRE            |
-- |         39 | WHITE HORSE HEALTH CENTRE        |
-- |         38 | EAGLE HOUSE SURGERY              |
-- |         38 | OLD TRAFFORD MEDICAL PRACTICE    |
-- |         38 | HALL GREEN HEALTH                |
-- |         37 | EAST NORWICH MEDICAL PARTNERSHIP |
-- +------------+----------------------------------+
```

The above comment shows a nicely formatted table of the **top 10** `practices` per metformin prescribed.

4.7 Wrapping up

This covers all the analysis we are doing on drug prescription by NHS practices.

Chapter 5

Conclusion

There a few takeaways from the analysis:

- In order to better understand drug prescription quantity how prescriptions are stored needs to be rethought. It's hard to conclude which medicine has been most prescribed overall, or which practice has prescribed the highest **actual** amount of [insert medicine here]. The chemicals do not carry any information regarding dosage in a way that is easily parsable.
- There aren't as many patients in *N17* as I expected.
- Whatever *beta-blockers* actually are, *Burrswood Nursing Home* must have a lot of people needing them. At a whooping 161 bb/p they are leading the charge on that type of drug.
- We can see massive gaps on money spent per patient, it ranges from *0.013£* to over *7609£*.
- As far as SSRIs go, the fluctuation between months seems to be minimal (less than 1%).

Chapter 6

References

- Health and Social Care Information Centre, Prescribing (2015): *General Practice Prescribing Data Frequently Asked Questions*. Available at: http://bit.ly/2tifJNU
- NHS (2016): *Beta-blockers*. Available at: http://bit.ly/2reDxBL
- Wikipedia (2016): *View (SQL)*. Available at: https://en.wikipedia.org/wiki/View_(SQL)
- NHS (2015): *Selective serotonin reuptake inhibitors (SSRIs)*. Available at: http://bit.ly/1Uym01X