Report 1

Clinical Data Management

Megan Pete - [redacted]

Contents

1 Introduction

In this report, we will use a subset of data from the London_Paris study. This data includes a number of health and personal characteristics of 1000 individuals from different countries, as well as genotype data for 500 of those individuals. We aim to analyse this data in order to better understand the distribution of the data in the study, as well as demonstrate certain functionalities of MySQL and R.

2 Creating a database, tables and importing data

Our first goal is to import the data we have from the London_Paris study into SQL. We do this in the following three steps.

2.1 Creating the database (1)

First, we create a database called London_Paris_DB, using the MySQL commands

```
# Question 1
# Create database
CREATE DATABASE London_Paris_DB;
USE London_Paris_DB;
```

which were found on the website [3].

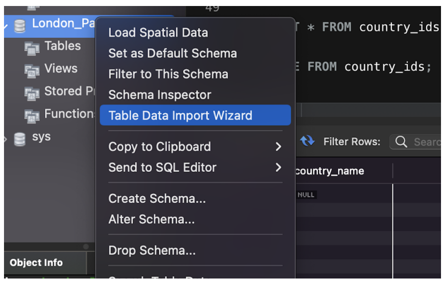
2.2 Creating the tables (2)

Next, we make three tables: 'country_ids', 'sample_characteristics' and 'sample_genotypes'. The primary and foreign keys of these tables are as follows.

Table Name	Primary Key	Foreign Key
country_ids	country	none
sample_characteristics	sample_id	country_ids.country
sample_genotypes	sample_id	sample_characteristics.sample_id

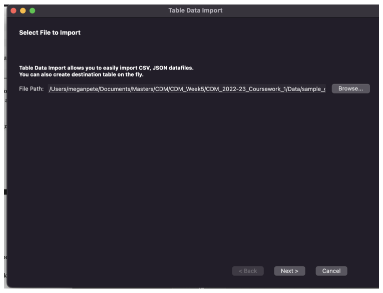
The code to make the tables is given by

```
# Question 2
# Create tables
CREATE TABLE country_ids (
    country char(20) NOT NULL,
    country_name char(20),
    PRIMARY KEY (country)
);

CREATE TABLE sample_characteristics (
    sample_id char(20) NOT NULL,
    dob char(20),
    dor char(20),
    dod char(20),
    dodiag char(20),
    sex int,
    cc_status int,
    smoke_status int,
    country char(20),
    center char(20),
    vit_status int,
    a_sta_smok double,
    a_quit_smok double,
    age_recr double,
    n_cigret_lifetime double,
    PRIMARY KEY (sample_id),
    FOREIGN KEY (country) REFERENCES country_ids(country)
);

CREATE TABLE sample_genotypes (
    sample_id char(20) NOT NULL,
    SNP1 int,
    SNP2 int,
    SNP3 int,
    SNP4 int,
    SNP5 int,
    PRIMARY KEY (sample_id),
    FOREIGN KEY (sample_id) REFERENCES sample_characteristics(sample_id)
);
```



```
* FROM country_ids
FROM country_ids;
```

(a) Select Table Data Import Wizard.

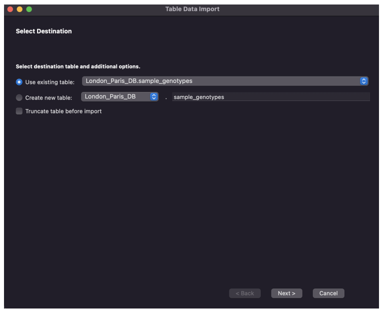


Select File to Import

Table Data Import allows you to easily import CSV, JSON datafiles.
You can also create destination table on the fly.

File Path: /Users/meganpete/Documents/Masters/CDM/CDM_Week5/CDM_2022-23_Coursework_1/Data/sample_

(b) Select file path for the csv file.

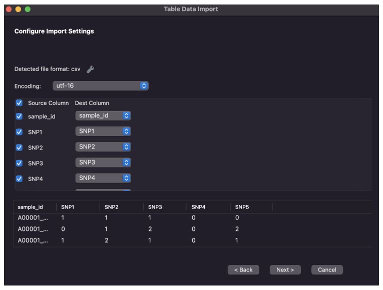


Select Destination

Select destination table and additional options.

[✓ Use existing table:] London_Paris_DB.sample_genotypes
Create new table: London_Paris_DB . sample_genotypes
Truncate table before import

(c) Select sample_genotypes table.



Configure Import Settings

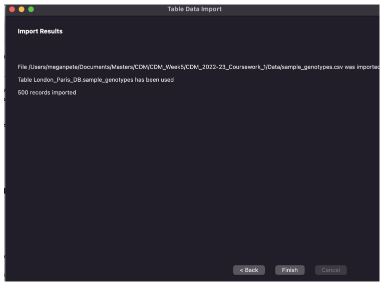
Detected file format: csv

Encoding: utf-16

✓ Source Column	Dest Column
✓ sample_id	sample_id
✓ SNP1	SNP1
✓ SNP2	SNP2
✓ SNP3	SNP3
✓ SNP4	SNP4

sample_id	SNP1	SNP2	SNP3	SNP4	SNP5
A00001_...	1	1	1	0	0
A00001_...	0	1	2	0	2
A00001_...	1	2	1	0	1

(d) Ensure column names in the SQL table and csv files match.



Import Results

File /Users/meganpete/Documents/Masters/CDM/CDM_Week5/CDM_2022-23_Coursework_1/Data/sample_genotypes.csv was importe

Table London_Paris_DB.sample_genotypes has been used

500 records imported

(e) Finish.

Figure 1: Importing the data into sample_genotypes.

2.3 Importing the data (3)

Finally, we import the data from the London_Paris study to the respective tables. This is done using three separate csv files containing the information for the three tables. The process of importing the data to the table sample_genotypes is demonstrated in figure 1.

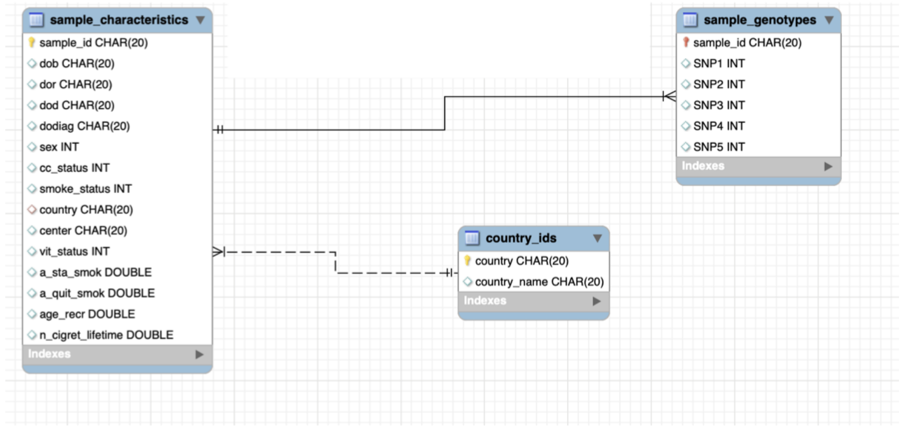


Figure 2: ER diagram for London_Paris_DB.

3 Exploring the data

3.1 Entity Relationship Diagram (4)

Now that we have created our three tables and imported the data into them, we can begin exploring the data. Firstly, we look at the Entity Relationship diagram for our database, shown in figure 2. This figure was made by following the steps in [1]. Notice the difference between the symbols in the ER diagram as follows:

- A yellow coloured key is only a primary key whereas a red coloured key is both a primary and a foreign key.
- A blue diamond is a simple attribute of the table, whereas a red diamond is a foreign key.
- Moreover, a filled diamond means that the values are not null, whereas hollow diamonds mean that the values may be null.
- Hence if we combine the points above we see that a blue filled diamond represents a simple attribute which is not null; a hollow red diamond represents a foreign key which can be null, and so on.

3.2 Dimensions of our tables (5)

To determine the number of records in each of our tables we use the following MySQL code:

```
# Question 5
# Count number of records in each country
SELECT COUNT(country)
FROM country_ids; #13

SELECT COUNT(sample_id)
FROM sample_genotypes; #500

SELECT COUNT(sample_id)
FROM sample_characteristics; #1000.
```

We see that country_ids contains 13, sample_genotypes contains 500 and sample_characteristics contains 1000 records respectively.

3.3 Null values (6)

In order to get a better understanding of the type of data in our tables so that we are able to analyse it, we would like to check which of the attributes contain null values. The following code

```
SELECT
    SUM(CASE WHEN Attribute_1 IS NULL then 1 ELSE 0 END) as Attribute_1,
FROM table1
```

sums up the number of null values in the Attribute_1 column (in table1) and returns it under the Attribute_1 heading. We can use similar code to determine the number of null values that each attribute in sample_characteristics contains.

```
# 6.2 / 6.3:
# This code counts how many null values each column in sample_characteristics contains
SELECT
    SUM(CASE WHEN sample_id IS NULL then 1 ELSE 0 END) as sample_id,
    SUM(CASE WHEN dob IS NULL then 1 ELSE 0 END) as dob,
    SUM(CASE WHEN dor IS NULL then 1 ELSE 0 END) as dor,
    SUM(CASE WHEN dod IS NULL then 1 ELSE 0 END) as dod,
    SUM(CASE WHEN dodiag IS NULL then 1 ELSE 0 END) as dodiag,
    SUM(CASE WHEN sex IS NULL then 1 ELSE 0 END) as sex,
    SUM(CASE WHEN cc_status IS NULL then 1 ELSE 0 END) as cc_status,
    SUM(CASE WHEN smoke_status IS NULL then 1 ELSE 0 END) as smoke_status,
    SUM(CASE WHEN country IS NULL then 1 ELSE 0 END) as country,
    SUM(CASE WHEN center IS NULL then 1 ELSE 0 END) as center,
    SUM(CASE WHEN vit_status IS NULL then 1 ELSE 0 END) as vit_status,
    SUM(CASE WHEN a_sta_smok IS NULL then 1 ELSE 0 END) as a_sta_smok,
    SUM(CASE WHEN a_quit_smok IS NULL then 1 ELSE 0 END) as a_quit_smok,
    SUM(CASE WHEN age_recr IS NULL then 1 ELSE 0 END) as age_recr,
    SUM(CASE WHEN n_cigret_lifetime IS NULL then 1 ELSE 0 END) as n_cigret_lifetime
FROM sample_characteristics;
```

The output of this code shows that there are five attributes in sample_characteristics containing null values. The frequencies of null values are summarised in the table below.

Attribute	dod	dodiag	a_sta_smok	a_quit_smok	n_cigret_lifetime
Frequencies of NULL	329	263	309	749	487

4 Querying the data

We now have a better understanding of the basic properties of our tables, and so we move on to querying our data. We will now perform a number of queries to gather various facts about the distribution of our data.

4.1 Average recruitment age (7.1)

First we determine the average age of recruitment into the study for each country using the following code

```
# 7.1:
# Age of recruitment for each country
SELECT country_ids.country, country_name, ROUND(AVG(age_recr),2) AS avg_age_country
FROM country_ids
INNER JOIN sample_characteristics
ON country_ids.country = sample_characteristics.country
GROUP BY country_ids.country
ORDER BY country_ids.country;
```

The results can be summarised in a table.

country	country_name	avg_age_country
1	UK	54.81
2	USA	53.88
3	France	53.62
4	Sri Lanka	55.00
5	India	54.75
6	China	55.83
7	Portugal	56.20
8	Australia	55.88
9	Italy	53.92
B	Unknown	54.56

4.2 Country with most samples (7.2)

Next, we find the country with the largest number of samples. In order to do so we compute the number of samples in each country, in descending order as follows:

```
# 7.2:
# Country with most samples
SELECT country_ids.country, country_name,
COUNT(sample_characteristics.country) AS counts
FROM country_ids INNER JOIN sample_characteristics
ON country_ids.country = sample_characteristics.country
GROUP BY country_ids.country
ORDER BY counts DESC;
```

and we see that the country with the most samples is **Italy**, with **190** samples.

4.3 Center with fewest samples (7.3)

The center with the fewest samples is **center 25** with **5** samples, which is calculated as follows:

```
# 7.3:
# Center with least samples
SELECT center, COUNT(sample_id) AS counts
FROM sample_characteristics
```

```
GROUP BY center
ORDER BY counts ASC;
```

4.4 Country with youngest recruitment age (7.4)

Using this code

```
# 7.4:
# Country with single youngest age at recruitment

SELECT country_ids.country,country_ids.country_name,
MIN(sample_characteristics.age_recr) AS min_age_country
FROM country_ids
INNER JOIN sample_characteristics
ON country_ids.country = sample_characteristics.country
GROUP BY sample_characteristics.country
ORDER BY min_age_country;
```

we see that the youngest person to be recruited into the study is from **Italy** and was **20.25188** years old at the time of recruitment.

4.5 Country with most male samples (7.5)

Italy also has the largest number of males participating in the study, with **106**.

```
# 7.5
# Country with most males samples
SELECT country_ids.country,country_ids.country_name,
SUM(CASE WHEN sample_characteristics.sex = 2 then 1 ELSE 0 END) AS number_of_males
FROM country_ids
INNER JOIN sample_characteristics
ON country_ids.country = sample_characteristics.country
GROUP BY country
ORDER BY number_of_males DESC;
```

4.6 Country with most distinct centers (7.6)

The **Italy** has **29** distinct centers supplying samples to it, which is the most out of all the countries.

```
# 7.6:
# Country with most distinct centers
SELECT country_ids.country, country_ids.country_name,
COUNT(DISTINCT sample_characteristics.country, sample_characteristics.center)
AS number_distinct_centers
FROM sample_characteristics
INNER JOIN country_ids
ON country_ids.country = sample_characteristics.country
GROUP BY country
ORDER BY number_distinct_centers DESC;
```

4.7 Country-center pair with most samples (7.7)

Finally, we calculate that the country-center pair with the most samples is **Italy-92**, with **28** samples, as follows:

```
# 7.7:
# Country-center pair with the most samples
SELECT country_ids.country, country_ids.country_name, sample_characteristics.center,
COUNT(sample_characteristics.center) AS number_samples_per_center
FROM sample_characteristics
INNER JOIN country_ids
ON country_ids.country = sample_characteristics.country
GROUP BY sample_characteristics.country, sample_characteristics.center
ORDER BY number_samples_per_center DESC;
```

This matches up with our code in Q7.2 and Q7.3.

5 Creating new data from old data

5.1 Creating a new table (8.1)

Now that we have analysed a number of characteristics of the London_Paris study, we will make a new table which summarises important attributes from our existing tables called sample_char_genotypes. The attributes that will be included are:

sample_id, dob, sex, cc_status, age_recr, country, country_name, SNP1, SNP2, SNP3, SNP4 and SNP5.

In order to do so, we use the CREATE TABLE AS statement [4]:

```
# 8.1:
# Create new table
CREATE TABLE sample_char_genotypes
AS (SELECT sample_genotypes.sample_id, sample_characteristics.dob,
sample_characteristics.sex, sample_characteristics.cc_status,
sample_characteristics.age_recr, country_ids.country,
country_ids.country_name, sample_genotypes.SNP1,
sample_genotypes.SNP2, sample_genotypes.SNP3,
sample_genotypes.SNP4, sample_genotypes.SNP5
FROM sample_genotypes
INNER JOIN sample_characteristics
ON sample_genotypes.sample_id = sample_characteristics.sample_id
INNER JOIN country_ids
ON sample_characteristics.country = country_ids.country);
```

5.2 Primary and foreign key (8.2)

We would like to find out whether sample_char_genotypes has inherited a primary or foreign key from any of the old tables. In order to do this, we can check the ER diagram of our new database, which is shown in figure 3

This shows us that sample_char_genotypes does not have a primary or foreign key, as there are only **blue diamonds** (simple attributes). In order for it to have a primary key, we would need to see a red key icon, and in order for it to have a foreign key, we would need to see either a yellow key or a red diamond. Therefore, it has not inherited either a primary or a foreign key from our tables.

5.3 Exporting the table (8.3)

To export the sample_char_genotypes table to csv form, we follow the steps in figure 4.

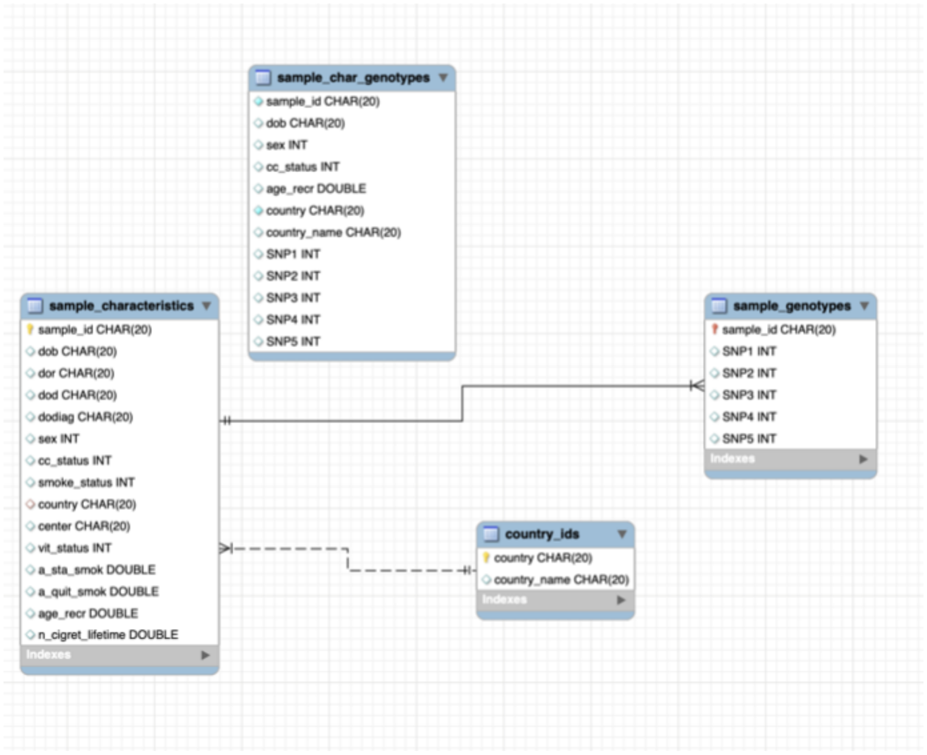
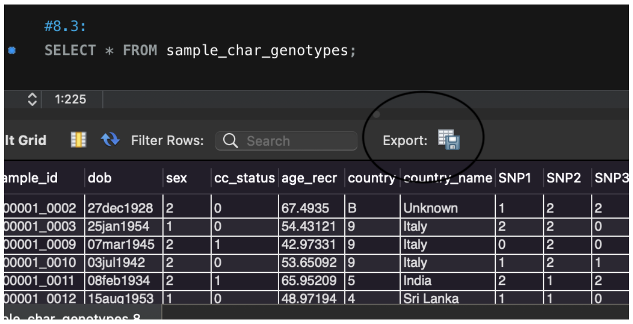


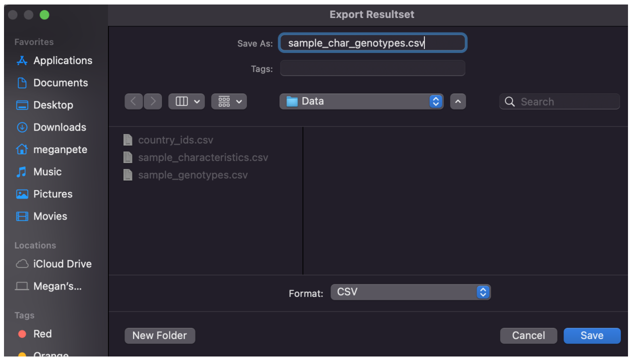
Figure 3: ER diagram of the database including the new table sample_char_genotypes



```
#8.3:
SELECT * FROM sample_char_genotypes;
```

ample_id	dob	sex	cc_status	age_recr	country	country_name	SNP1	SNP2	SNP3
00001_0002	27dec1928	2	0	67.4935	B	Unknown	1	2	2
00001_0003	25jan1954	1	0	54.43121	9	Italy	2	2	0
00001_0009	07mar1945	2	1	42.97331	9	Italy	0	2	0
00001_0010	03jul1942	2	0	53.65092	9	Italy	1	2	1
00001_0011	08feb1934	2	1	65.95209	5	India	2	1	2
00001_0012	15aug1953	1	0	48.97194	4	Sri Lanka	1	1	0

(a) Select everything (*) from the sample_char_genotypes table, and press 'export'.



Export Resultset

Save As: sample_char_genotypes.csv

Tags:

Format: CSV

(b) Save the csv file.

Figure 4: Exporting the sample_char_genotypes table.

6 Using R

6.1 Connecting MySQL to R (9.1)

Our final task is to connect our MySQL server to R so that we are able to perform queries from R. We do this by installing the RMySQL package on R. [2] The following code is used to do this:

```
install.packages("RMySQL")
library(RMySQL)

sqlconnect = dbConnect(RMySQL::MySQL(),
                            dbname='London_Paris_DB',
                            host='localhost',
                            port=3306,
                            user='root',
                            password='mypassword') # Omitting my password
```

6.2 Importing data to R (9.2)

Now that R is connected to the MySQL server, we are able to query our database London_Paris_DB. We can import our sample_genotypes table in R as follows:

```
query = dbSendQuery(sqlconnect, "SELECT * FROM sample_genotypes")
sample_genotypes = fetch(query)
print(sample_genotypes)
View(sample_genotypes).
```

7 Conclusion

We have now analysed a range of data from the London_Paris study. We have created a database using this data, and investigated both its structure as well as the distribution of the data within it. Apart from gaining a better understanding of the data, which is necessary for further analysis, we have also gained experience querying with MySQL, and learnt how to transfer these methods to R.

8 References

[1] *Create ER Diagram of a Database in MySQL Workbench*. https://medium.com/@tushar0618/how-to-create-er-diagram-of-a-database-in-mysql-workbench-209fbf63fd03. Accessed: 2022-11-14.

[2] *How to connect to mysql using R*. https://www.projectpro.io/recipes/connect-mysql-r. Accessed: 2022-11-14.

[3] *SQL CREATE DATABASE Statement*. https://www.w3schools.com/sql/sql_create_db.asp. Accessed: 2022-11-13.

[4] *SQL: CREATE TABLE AS Statement*. https://www.techonthenet.com/sql/tables/create_table2.php. Accessed: 2022-11-14.