MySQL Report

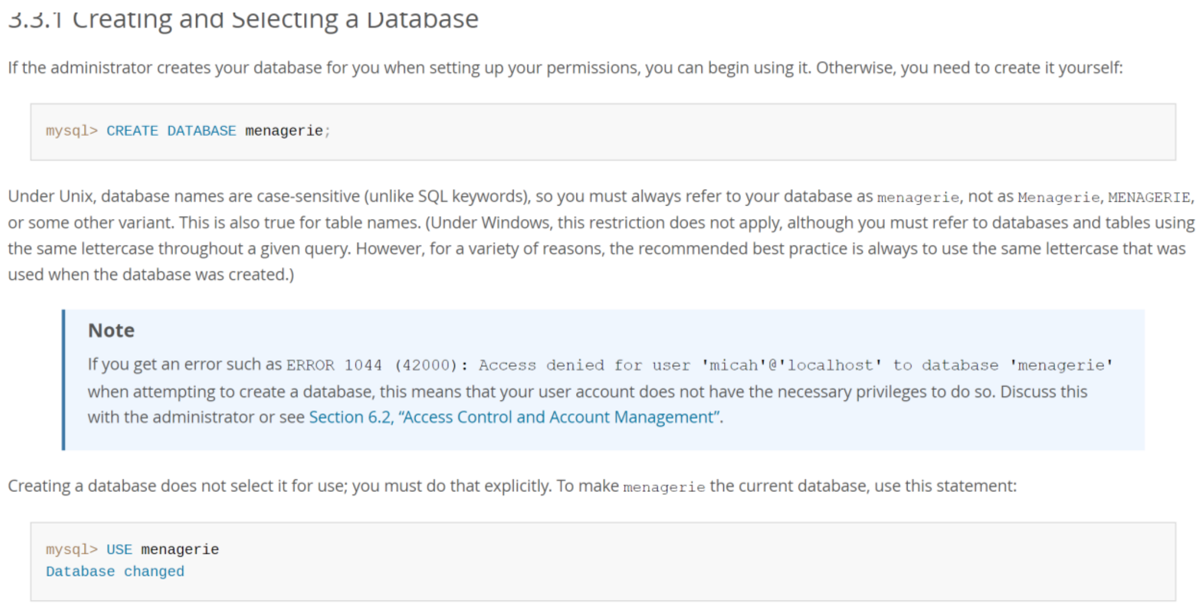
Abstract

Using data detailing information of lifestyle habits and genomic data from samples taken form 13 countries a database of three tables were created as per the data given. The tables were then analysed to look at factor such as the average and minimum age of recruitment and where most of the samples were taken with Italy seeming to have the most amount samples and centres supplying it with samples.

Contents

Creating a database (1)

Here a database has been created to be used. Creating a database has allowed to create a database environment that is able to be completely edited an added to with tables and data required. The SQL reference manual has and will continue to be a reference and a guide for this process (MySQL,2022).



3.3.1 Creating and Selecting a Database

If the administrator creates your database for you when setting up your permissions, you can begin using it. Otherwise, you need to create it yourself:

```
mysql> CREATE DATABASE menagerie;
```

Under Unix, database names are case-sensitive (unlike SQL keywords), so you must always refer to your database as `menagerie`, not as `Menagerie`, `MENAGERIE`, or some other variant. This is also true for table names. (Under Windows, this restriction does not apply, although you must refer to databases and tables using the same lettercase throughout a given query. However, for a variety of reasons, the recommended best practice is always to use the same lettercase that was used when the database was created.)

Note

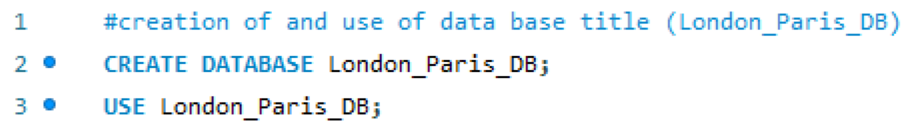
If you get an error such as `ERROR 1044 (42000): Access denied for user 'micah'@'localhost' to database 'menagerie'` when attempting to create a database, this means that your user account does not have the necessary privileges to do so. Discuss this with the administrator or see Section 6.2, "Access Control and Account Management".

Creating a database does not select it for use; you must do that explicitly. To make `menagerie` the current database, use this statement:

```
mysql> USE menagerie
Database changed
```

Figure 1 : Screenshot showing MySQL page where instruction to create database where garnered

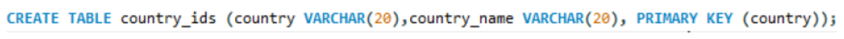
Following this instruction, accessed Localhost server through MySQL workbench that had previously created using MySQL server to create the database within. Allowed to test and modify database before introduction to server accessible for other people. The database was then creating using these commands in SQL.



```
#creation of and use of data base title (London_Paris_DB)
CREATE DATABASE London_Paris_DB;
USE London_Paris_DB;
```

Forming tables within database (2)

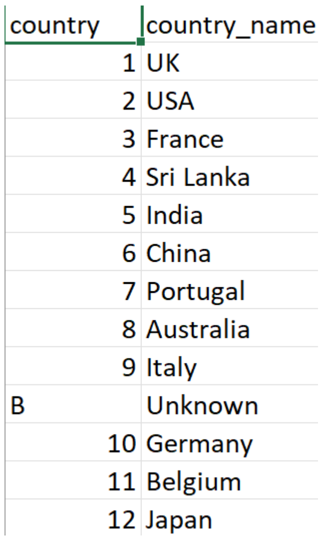
Tables with the variables clearly established where created before importing the data into the London schema. Establishing the characteristic of these tables is essential to create the correct variable for the data. Firstly, looking at the data through excel helped to analyse what column names and class type of data was required. For example country_ids in the excel sheet looked like (figure2). Therefore, a table for country_ids was made in which two columns one for country and for country_name where created through this line of code (figure 3):



```
CREATE TABLE country_ids (country VARCHAR(20),country_name VARCHAR(20), PRIMARY KEY (country));
```

Figure 3: code in MySQL workbench used to create table. VARCHAR allows ability to add string value of any length with max of 20 characters. Primary key set as 'country'

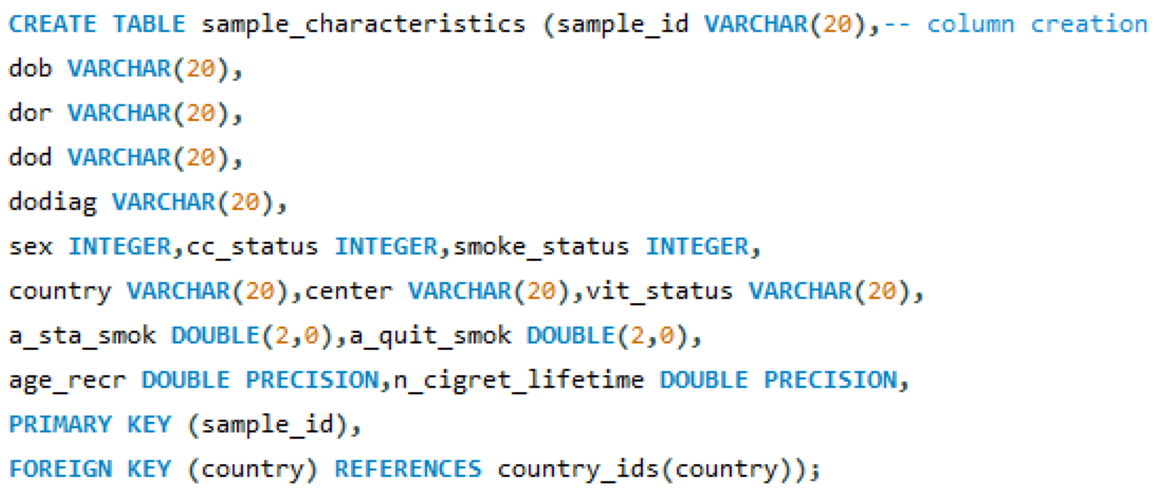
The country_ids after CREATE TABLE enabled labelling of the table. The columns are added in the brackets and data was classed as characters of max 20 characters for both as they act as string variables and the numeric values act as variable string-based labels rather than integers. This was subsequently done for the other two tables establishing data type for columns in sample_ characteristics and sample_ genotypes tables. Primary key was set up as the 'country' column



country	country_name
1	UK
2	USA
3	France
4	Sri Lanka
5	India
6	China
7	Portugal
8	Australia
9	Italy
B	Unknown
10	Germany
11	Belgium
12	Japan

Figure 2: Table from country_ids csv shown in excel

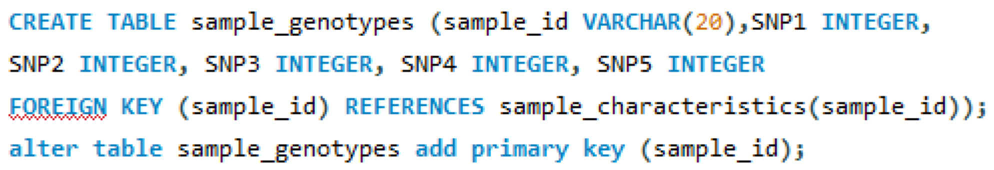
Creating sample_characteristics table



```
CREATE TABLE sample_characteristics (sample_id VARCHAR(20),-- column creation
dob VARCHAR(20),
dor VARCHAR(20),
dod VARCHAR(20),
dodiag VARCHAR(20),
sex INTEGER,cc_status INTEGER,smoke_status INTEGER,
country VARCHAR(20),center VARCHAR(20),vit_status VARCHAR(20),
a_sta_smok DOUBLE(2,0),a_quit_smok DOUBLE(2,0),
age_recr DOUBLE PRECISION,n_cigret_lifetime DOUBLE PRECISION,
PRIMARY KEY (sample_id),
FOREIGN KEY (country) REFERENCES country_ids(country));
```

figure 4:Code used to form the sample characteristic table. The use of integer is used to set up binary data. Double is used to limit numeric values of these columns to two digits for ages and no decimal places are used with age values. DOUBLE PRECISION function used to include all decimal places in numeric results. Primary key set as sample_id colum and foreign key set as country referencing country variable in country_ids linking sample_characteristics and country_ids table through the country values.

Creating gene id list

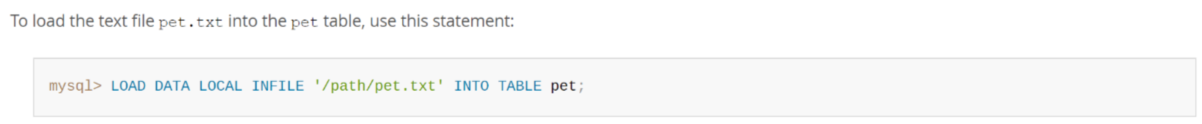


```
CREATE TABLE sample_genotypes (sample_id VARCHAR(20),SNP1 INTEGER,
SNP2 INTEGER, SNP3 INTEGER, SNP4 INTEGER, SNP5 INTEGER
FOREIGN KEY (sample_id) REFERENCES sample_characteristics(sample_id));
alter table sample_genotypes add primary key (sample_id);
```

Figure 5: code used to create sample_genotypes table with sample_id treated as variable strings and SNP values treated as Integers. Foreign id set as sample_id with referencing to the sample_id variable in sample_characteristics table linking both sample_genotypes and sample_characterstics by sample_ids.

Importing data into the tables(3)

Now that database and table is set up it was time to import the data into the tables. The method to do this was again provided by the SQL manual (MySQL,2022). This seen in code below(figure 6).



To load the text file `pet.txt` into the `pet` table, use this statement:

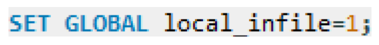
```
mysql> LOAD DATA LOCAL INFILE '/path/pet.txt' INTO TABLE pet;
```

Figure 6 :The example porvided by SQL manual provided a template as to how appload the data. The first segment of code locates the fild required to be imported and INTO TABLE function dictates what table the data will be imported in.

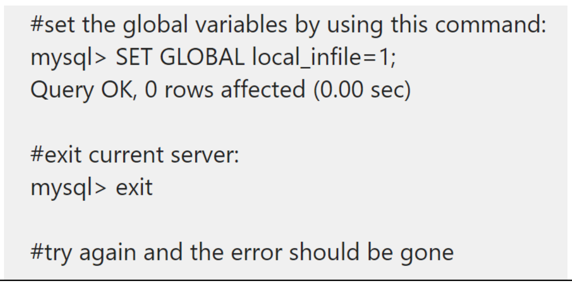
Doing this initially presented with an error claiming local data loading was disabled:

Error Code: 3948. Loading local data is disabled; this must be enabled on both the client and server sides

By consulting an article (Error 3948 (42000) At Line 1: Loading Local Data Is Disabled; This Must Be Enabled On Both The Client And Server Sides With Code Examples. 2022)there is a command to enable local data to be loaded:



```
SET GLOBAL local_infile=1;
```



```
#set the global variables by using this command:
mysql> SET GLOBAL local_infile=1;
Query OK, 0 rows affected (0.00 sec)

#exit current server:
mysql> exit

#try again and the error should be gone
```

Figure 7: screenshot from Folkstalk blog detailing solution to Error code :3948

After this was done another error was encountered :

Error Code: 2068. LOAD DATA LOCAL INFILE file request rejected due to restrictions on access.

Per advice given in the sql forum steps detailed in figure below were used to deal with error (Baxter, 2020):

The manage connection tab was selected form the home menu

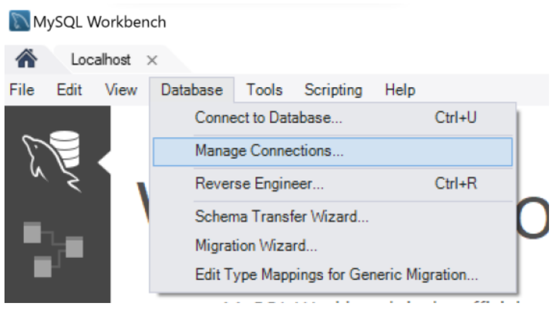
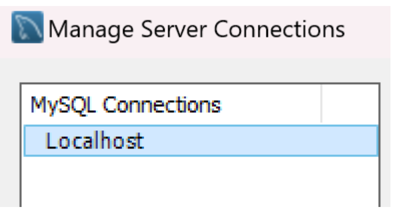

MySQL Workbench
Localhost
File
Edit
View
Database
Tools
Scripting
Help
Connect to Database... Ctrl+U
Manage Connections...
Reverse Engineer... Ctrl+R
Schema Transfer Wizard...
Migration Wizard...
Edit Type Mappings for Generic Migration...

Figure 8: screenshot of where to find the manage connection setting in MySQL workbench

Secondly Selected the localhost server:



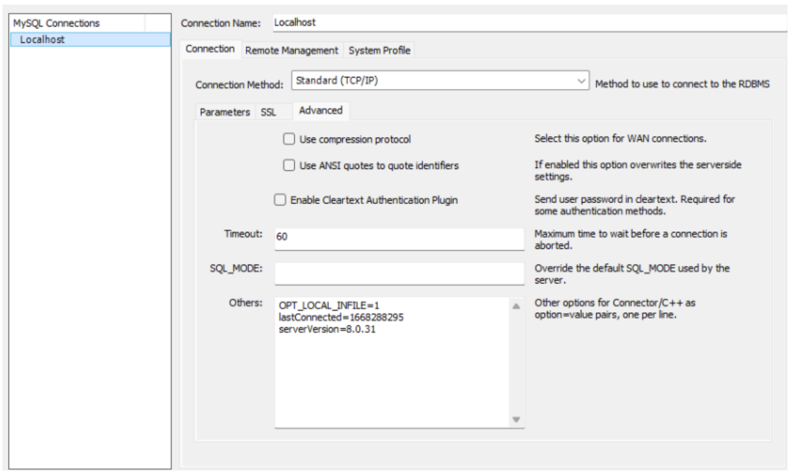
Manage Server Connections

MySQL Connections

Localhost

Figure 9: screenshot where to find and select the local host connection with the manage connections setting

Thirdly added the OPT_LOCAL_INFILE =1 command in the others section. This was accessed through the 'advanced' tab



Connection Name: Localhost

Connection Method: Standard (TCP/IP) — Method to use to connect to the RDBMS

☐ Use compression protocol — Select this option for WAN connections.

☐ Use ANSI quotes to quote identifiers — If enabled this option overwrites the serverside settings.

☐ Enable Cleartext Authentication Plugin — Send user password in cleartext. Required for some authentication methods.

Timeout: 60 — Maximum time to wait before a connection is aborted.

SQL_MODE: — Override the default SQL_MODE used by the server.

Others: — Other options for Connector/C++ as option=value pairs, one per line.

```
OPT_LOCAL_INFILE=1
lastConnected=1668288295
serverVersion=8.0.31
```

Figure 10 : screenshot of showing where the opt_local_infile=1 command should be added within the MySQL connection settings tab.

After this the data was available to be imported. Due to issues with the local data local infile command mentioned above another menthod was used to import data into table utilising the GUI features provided with mySQL workbench. The steps are detailed and pictured in the figures below:

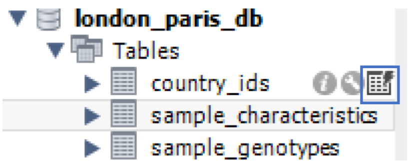
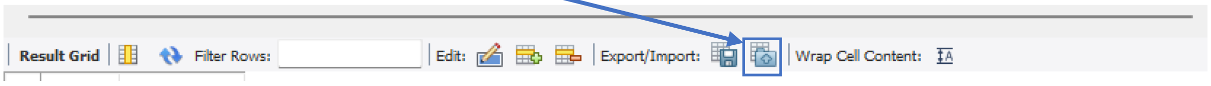
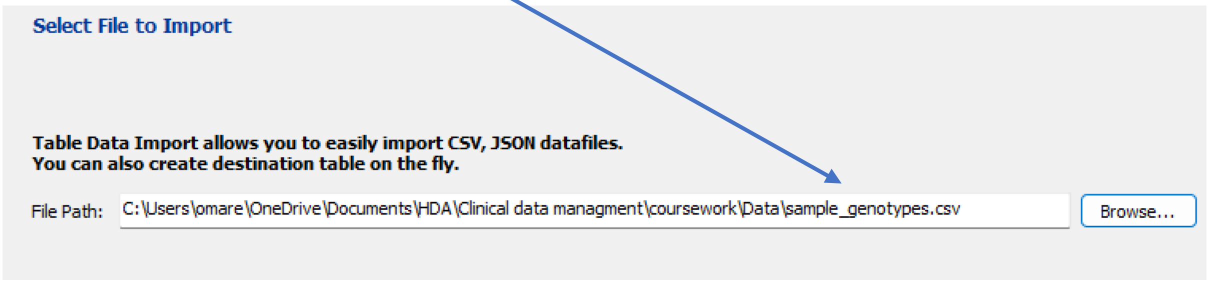


Figure 11: screenshot showing the location of the database and how the tables where viewed using the select table function (symbol in blue square)within the schema tab in MySQL workbench.

a.clicked on the import data option (symbol within blue square) within the results grid accessed by clicking the select table function (figure 11):



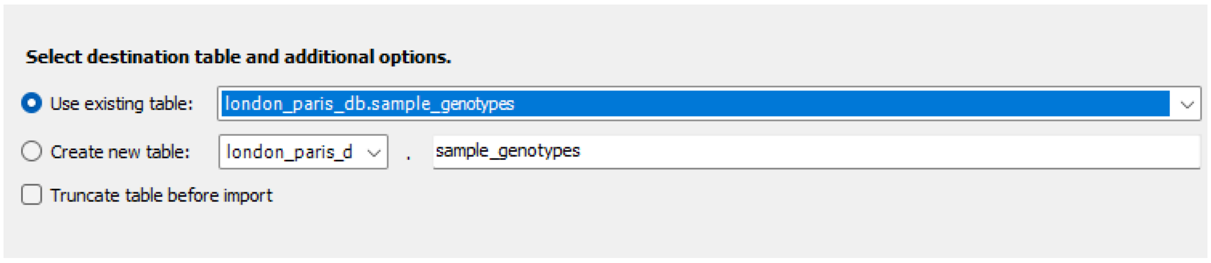
b.Selected the corresponding table to the data I was importing in



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: C:\Users\omare\OneDrive\Documents\HDA\Clinical data managment\coursework\Data\sample_genotypes.csv c.Selected the table for the data to be imported into



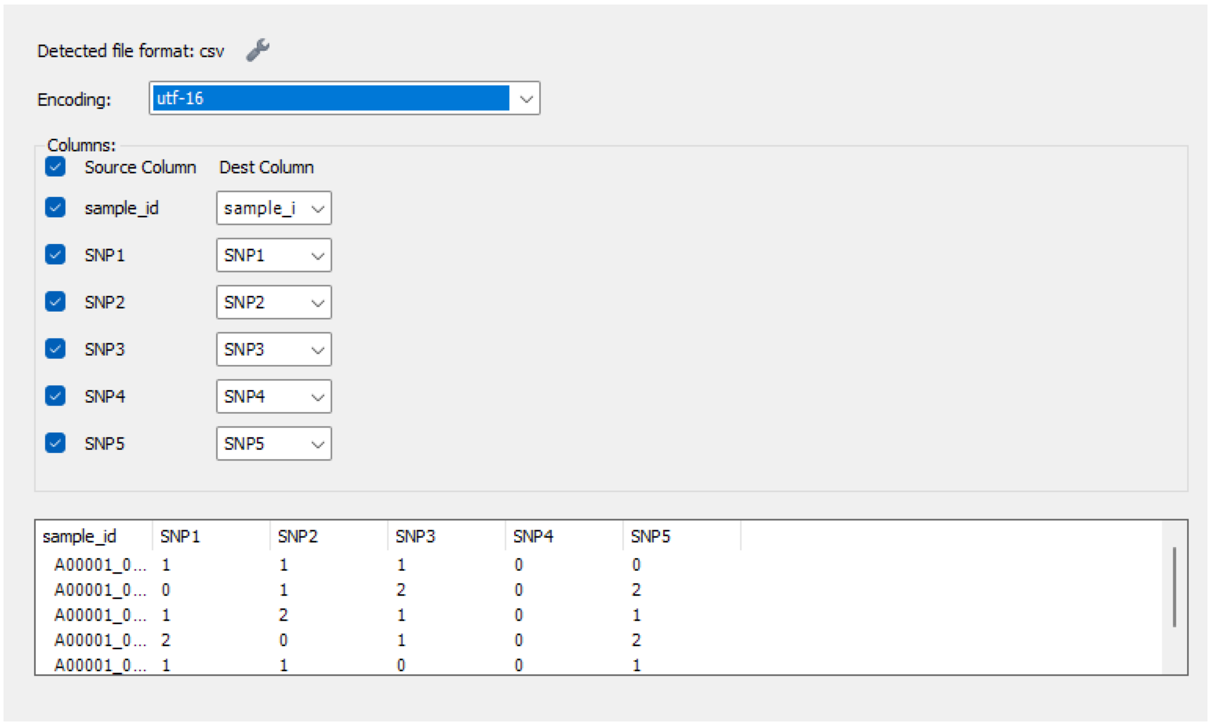
Select destination table and additional options.

[✓ Use existing table:] london_paris_db.sample_genotypes

Create new table: london_paris_d . sample_genotypes

Truncate table before import

d.Finally made sure that all the columns had the right corresponding data being imported into them ensuring that utf-16 encoding was being used



Detected file format: csv

Encoding: utf-16

Columns:

Source Column	Dest Column
✓ sample_id	sample_i
✓ SNP1	SNP1
✓ SNP2	SNP2
✓ SNP3	SNP3
✓ SNP4	SNP4
✓ SNP5	SNP5

sample_id	SNP1	SNP2	SNP3	SNP4	SNP5
A00001_0...	1	1	1	0	0
A00001_0...	0	1	2	0	2
A00001_0...	1	2	1	0	1
A00001_0...	2	0	1	0	2
A00001_0...	1	1	0	0	1

Figure 12 :series of screenshots indicating the steps taken to import data for sample_genotypes table A) screen shot showing how and where the Import data function was found. B) screenshot showing how relevant data was selected in which each table had the csv data with the same title imported within it C) screenshot showing how the correct table was picked per corresponding data to be imported D) screenshot showing the layout import data tab before clicking import in which utf-16 encoding is selected.

EXPLORING THE DATA

ER diagram of schema (4)

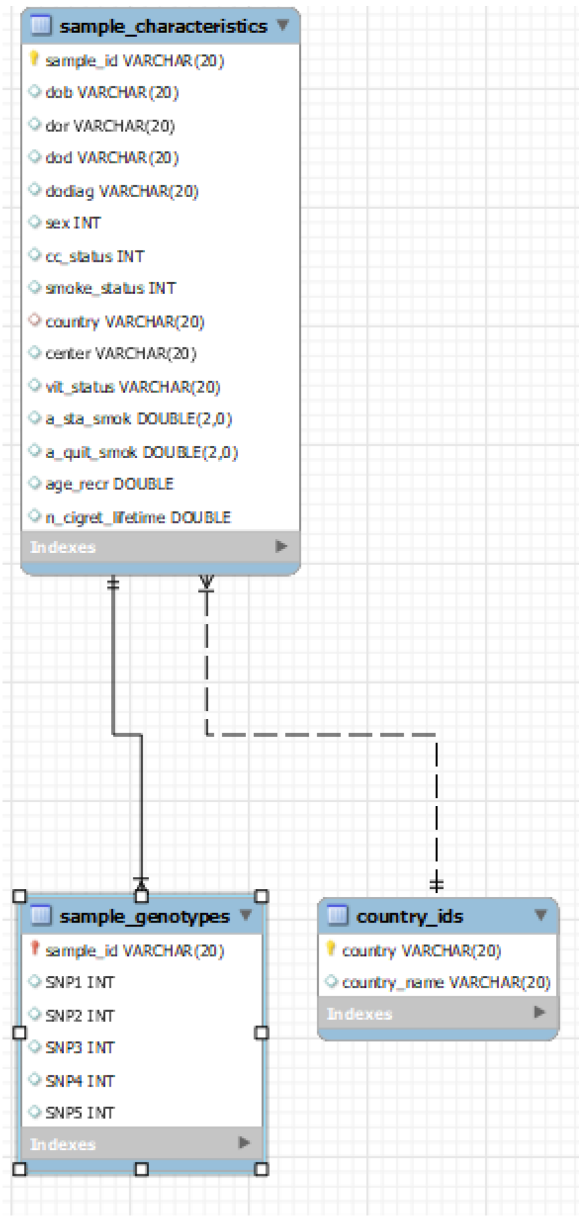


Figure 13: EER (enhanced entity relation) diagram showing datatypes of attributes and links between tables

🔑 Primary key : acts as unique attribute for each table to distinguish items within the data

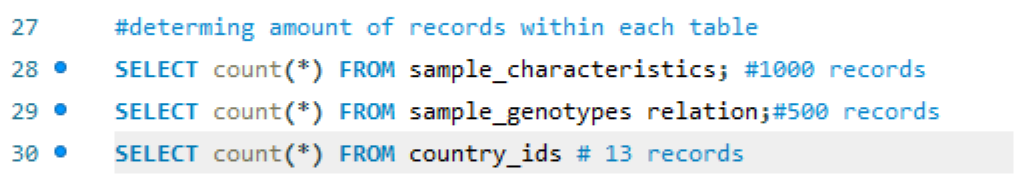
🔑 Primary key that is also part of foreign key and therefore acts to add unique distinguishable attribute to table but also to link table in this case to the sample_id primary key in the sample_characteristics table.

◇ simple attribute can be null.

◇ Symbol represents a foreign key that can have null values in this case country in sample_characteristics is linked to the primary key country in country_ids table.

Counting the amount of record in each table (5)

To do this the count function needs to be used to count all records for each table.



```
#determing amount of records within each table
SELECT count(*) FROM sample_characteristics; #1000 records
SELECT count(*) FROM sample_genotypes relation;#500 records
SELECT count(*) FROM country_ids # 13 records
```

Figure 14: code used to count amount of record per table

Explaining code (6.1)

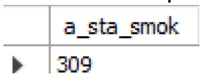
Another question as to what function this line of code seen below has:

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

Function of code investigated by testing the line of code on the sample_characteristics tables using code:

```
SELECT SUM(CASE WHEN a_sta_smok IS NULL then 1 ELSE 0 END) as a_sta_smok
FROM sample_characteristics ;
```

Allowed to deduce that it is used to calculate the number of null values in the data base. Hence the statement 1 IS NULL then 1 ELSE 0 which gave the null values the values of 1 and valid values 0 so that the SUM function can count the number of NULL values that gave a value of 1 and ignores the valid values which had value = 0. The 'as' function can temporarily name this SUM of NULLs as the attribute being calculated for the duration of the query (*SQL AS.* 2022).so when the query is finished it shows the specific column and how man NULL values appeared as seen here:

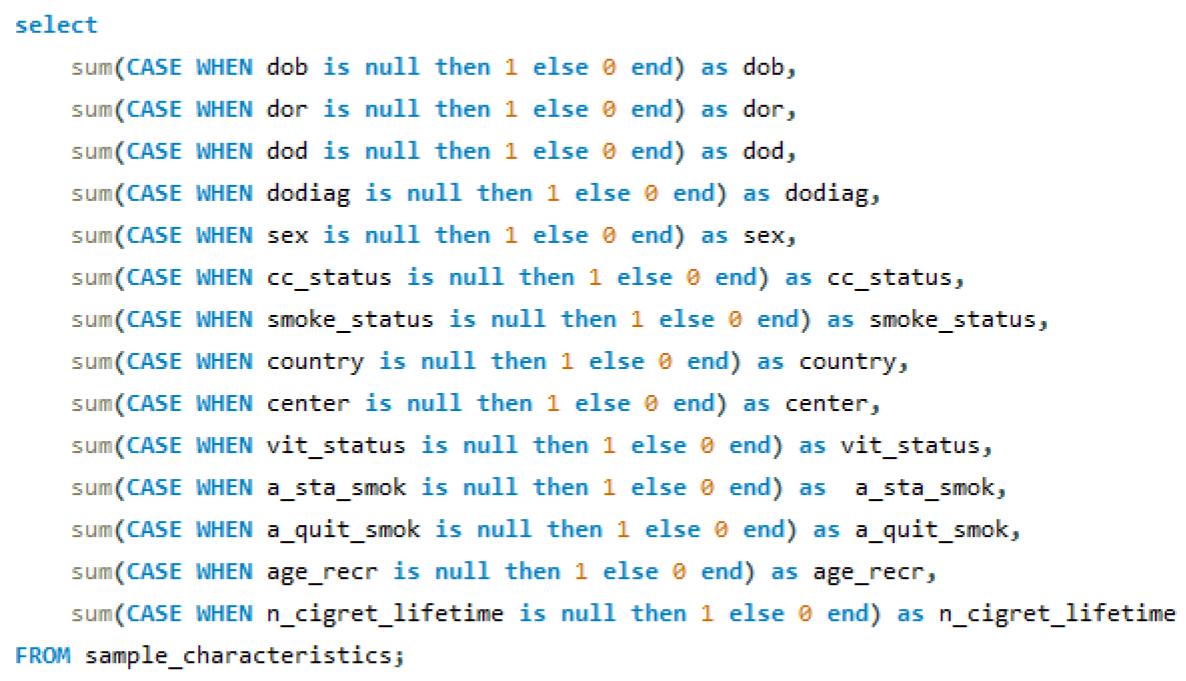


	a_sta_smok
▶	309

Finally, the FROM function is used to dictate what table the attribute to be calculated will be pulled from which is useful as some attributes share the same name between tables.

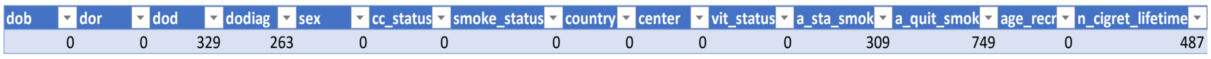
Identifying Null values in each column(6.2,6.3)

Using the code mentioned above allowed to form a table showing the amount of null values in each column for sample_characteristics table by applying code to each column (Figure 16).



```
select
    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;
```

Figure 15: Code used to generate table counting nulls. CASE WHEN function mentioned in (6.1) used to sum number of values who have been given a value = 1 as mentioned in the methodology detailed in section 6.1 above.



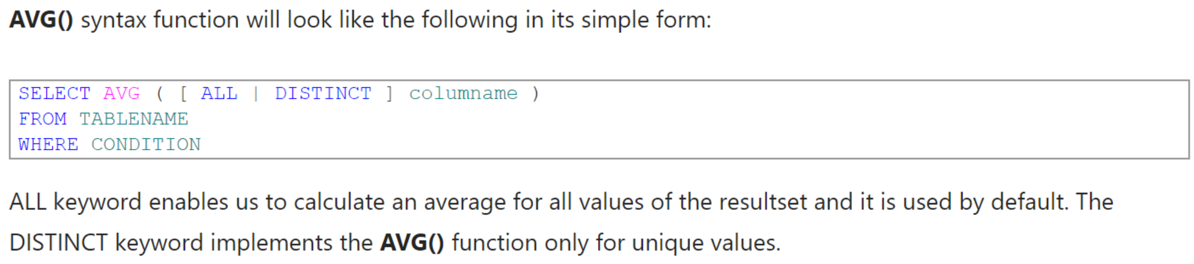
dob	dor	dod	dodiag	sex	cc_status	smoke_status	country	center	vit_status	a_sta_smok	a_quit_smok	age_recr	n_cigret_lifetime
0	0	329	263	0	0	0	0	0	0	309	749	0	487

Table 1: Table showing the column names of the sample characteristics table and the number below indicates the number of null samples per above corresponding column

Querying the data

Finding the average age of recruitment per country (7)

To form table with the average age of recruitment per country decided to consult blog that detailed how to form a table with averages (Erkec, 2020). This produced a useful template



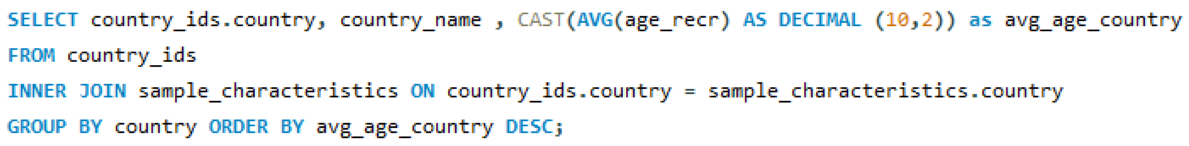
AVG() syntax function will look like the following in its simple form:

```
SELECT AVG ( [ ALL | DISTINCT ] columname )
FROM TABLENAME
WHERE CONDITION
```

ALL keyword enables us to calculate an average for all values of the resultset and it is used by default. The DISTINCT keyword implements the **AVG()** function only for unique values.

Figure 16: template of code used to form basis for creating table with averages

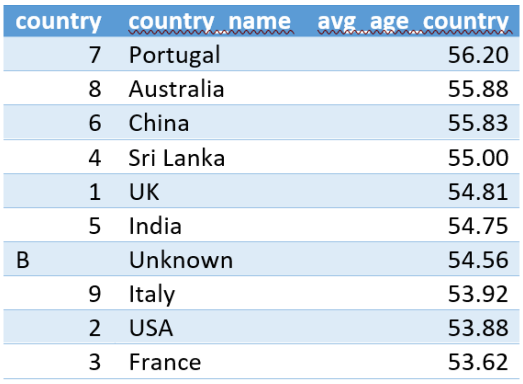
The final code was converted into :



```
SELECT country_ids.country, country_name , CAST(AVG(age_recr) AS DECIMAL (10,2)) as avg_age_country
FROM country_ids
INNER JOIN sample_characteristics ON country_ids.country = sample_characteristics.country
GROUP BY country ORDER BY avg_age_country DESC;
```

Figure 17: code used to create (table 2). SELECT was used to select the correct column. AVG used to get average of recruitment age (age_recr) and CAST and AS DECIMAL (10,2) will mean the result show up with the max of 10 figures before the decimal point and two decimal points (SQL AVG() and CAST() inside Avg() for decimal value., 2022). *FROM picks out the table country_ids and INNER JOIN links the table so that the country and country name have the same corresponding values between the tables. GROUP BY grouped the data by country and ORDER and DESC function allowed to order the table from highest to lowest by average value for age_recr.*

This provided a sufficient data table (table 2). It is worth mentioning that when using the LEFT JOIN we are able to identify that there is no data for the average age of recruitment as Germany, Belgium and Japan return null values hence they were removed through the INNER JOIN function.



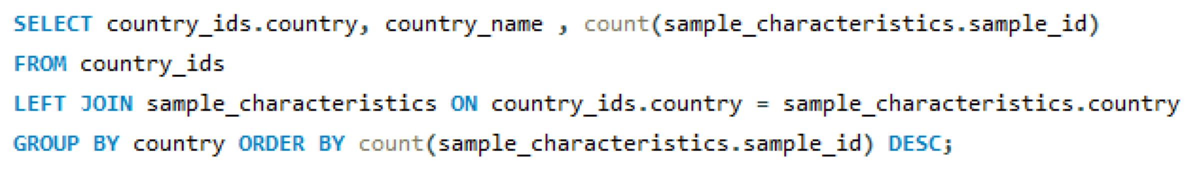
country	country_name	avg_age_country
7	Portugal	56.20
8	Australia	55.88
6	China	55.83
4	Sri Lanka	55.00
1	UK	54.81
5	India	54.75
B	Unknown	54.56
9	Italy	53.92
2	USA	53.88
3	France	53.62

Table 2: Table showing the average age of recruitment represented by column (ave_age_country) of country value with the corresponding country name.

Analysing the samples

Analysing number of samples per country (7.2)

The number of samples were analysed to see what country had the largest amount and smallest number of samples. The same logic applied to averaging the age of recruitment above was used here to supply a useful table. The code is supplied below (figure 17). The table showed the country with the most Samples being ITALY at 190 samples and Belgium Germany and Japan having the lowest amount with 0 samples taken.



```
SELECT country_ids.country, country_name , count(sample_characteristics.sample_id)
FROM country_ids
LEFT JOIN sample_characteristics ON country_ids.country = sample_characteristics.country
GROUP BY country ORDER BY count(sample_characteristics.sample_id) DESC;
```

Figure 18: code to form table to show number of samples per country. code follow the same logic of the average age code but with the count of the samples replacing the average age. Also LEFT JOIN was deemed more suitable to determine the countries with the lease amount of samples.

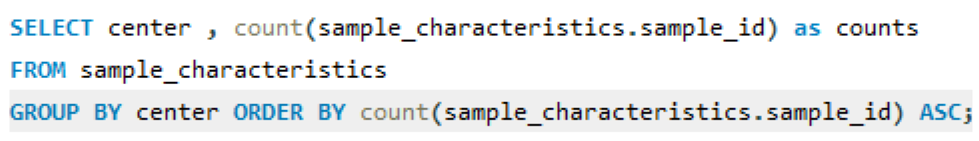
Belgium,Germany and Japan had the least amount samples.

country	country_name	counts
9	Italy	190
4	Sri Lanka	135
8	Australia	125
1	UK	100
7	Portugal	100
5	India	95
2	USA	81
3	France	70
6	China	55
B	Unknown	49
10	Germany	0
11	Belgium	0
12	Japan	0

Table 3: Table showing the country with the largest number of samples taken , the number of samples is labelled under the column 'counts'.

Analysing least amount of samples per center (7.3)

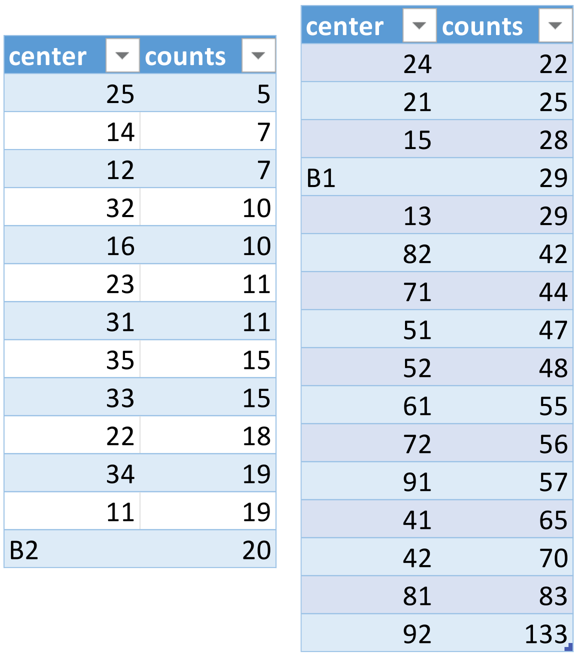
The number of samples per center where found using a simpler query due to only requiring information from one table.



```
SELECT center , count(sample_characteristics.sample_id) as counts
FROM sample_characteristics
GROUP BY center ORDER BY count(sample_characteristics.sample_id) ASC;
```

Figure 19: code used to form table 4. Count function used to count number of samples using the number of sample_ids.

This derived a table showing centre 25 collected the least amount of samples at 5 samples collected (table 4).



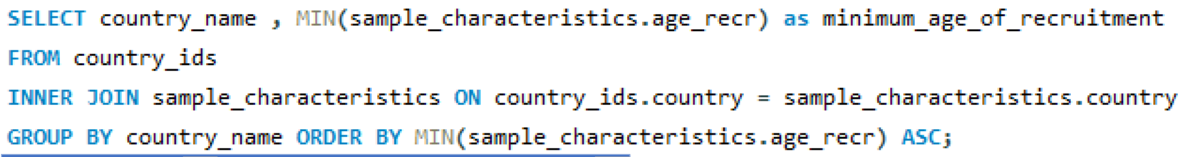
center	counts
25	5
14	7
12	7
32	10
16	10
23	11
31	11
35	15
33	15
22	18
34	19
11	19
B2	20
24	22
21	25
15	28
B1	29
13	29
82	42
71	44
51	47
52	48
61	55
72	56
91	57
41	65
42	70
81	83
92	133

table 4: Table split in two half showing the head and tail of the full table showing the amount of samples taken by center.

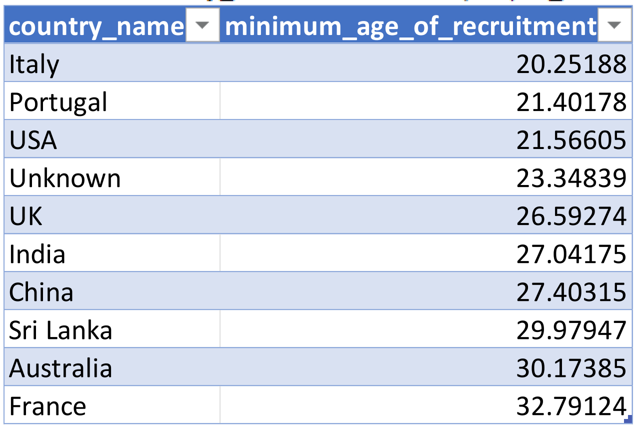
Deducing the country name with the lowest age of recruitment (7.4)

The query used to find the average age of recruitment was again used as a template however also consulted a SQL website to determine how to get the lowest value in a column leading to the discovery of the MIN function (*How to Find the Minimum Value of a Column in SQL.* 2016). This got the lowest age of recruitment per country, also used the CAST and AS DECIMAL function to represent the ages as whole numbers. Both code and table with all decimals and without are shown below:

Min age of recruitment with decimals



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

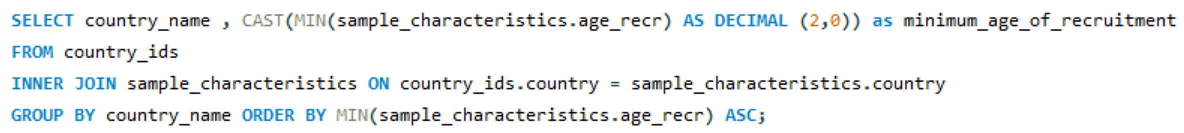


country_name	minimum_age_of_recruitment
Italy	20.25188
Portugal	21.40178
USA	21.56605
Unknown	23.34839
UK	26.59274
India	27.04175
China	27.40315
Sri Lanka	29.97947
Australia	30.17385
France	32.79124

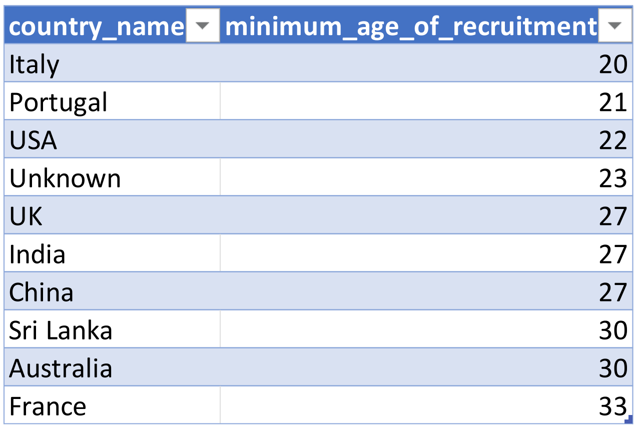
Figure 20: code used to create (table 5) in which country name and the minimum age of recruitment where selected. Minimum age was grouped by the country name.

Table 5:Table showing the country name and the minimum age of recruitment of these countries to maximum decimal points

Min age of recruitment without decimals



```
SELECT country_name , CAST(MIN(sample_characteristics.age_recr) AS DECIMAL (2,0)) as minimum_age_of_recruitment
FROM country_ids
INNER JOIN sample_characteristics ON country_ids.country = sample_characteristics.country
GROUP BY country_name ORDER BY MIN(sample_characteristics.age_recr) ASC;
```



country_name	minimum_age_of_recruitment
Italy	20
Portugal	21
USA	22
Unknown	23
UK	27
India	27
China	27
Sri Lanka	30
Australia	30
France	33

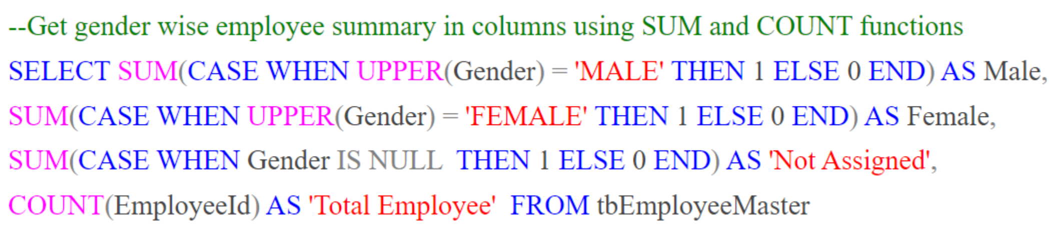
Figure 21: code is similar to above code (figure 19) however the CAST and AS DECIMAL functions used to get the minimum age of recruitment two no decimal places.

Table 5:Table showing the country name and the minimum age of recruitment of these countries to nearest whole number.

The largest number of male samples per country (7.5)

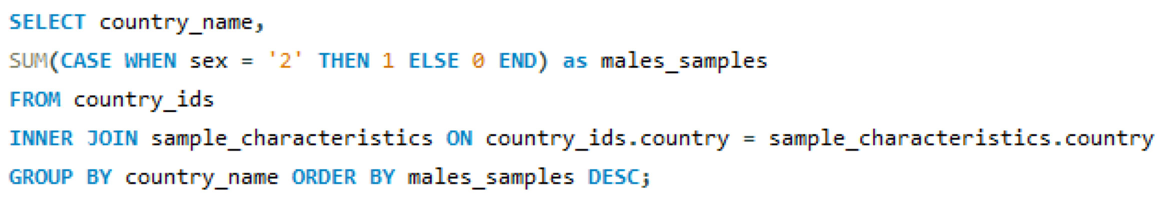
Query to look at the greatest number of male samples per country given sex 1 represents females and 2 represents males. Attempted to use function detailed in the finding null code however did not work(figure 15). Therefore, consulted webcodeexpert website to find a template to base code upon (Raghuvanshi, 2022) . Used template detailing how to find employee summary for males and females to form code.

A.



```
--Get gender wise employee summary in columns using SUM and COUNT functions
SELECT SUM(CASE WHEN UPPER(Gender) = 'MALE' THEN 1 ELSE 0 END) AS Male,
SUM(CASE WHEN UPPER(Gender) = 'FEMALE' THEN 1 ELSE 0 END) AS Female,
SUM(CASE WHEN Gender IS NULL  THEN 1 ELSE 0 END) AS 'Not Assigned',
COUNT(EmployeeId) AS 'Total Employee'  FROM tbEmployeeMaster
```

B.



```
SELECT country_name,
SUM(CASE WHEN sex = '2' THEN 1 ELSE 0 END) as males_samples
FROM country_ids
INNER JOIN sample_characteristics ON country_ids.country = sample_characteristics.country
GROUP BY country_name ORDER BY males_samples DESC;
```

Figure 22: A)code from webcodeexpert website that served as template for code used to identify the number of males and females using SELECT SUM(CASE WHEN UPPER(Gender) = 'MALE' THEN 1 ELSE 0 END) AS Male which was used to give any MALE value in the gender column the value of 1 get the sum of males there are given that one male =1.B) code used for London_Paris_db database collect the number of male samples per country. The integer 2 within these represented males in the sex column. The males (2) where given the value of 1 using THEN 1 and the females were given the value of 0 using ELSE 0 END. The query then uses the same logic from the code used in min age of recruitment section to find the greatest number of male samples collected per country in this case being Italy.

country_name	male_samples
Italy	106
Sri Lanka	76
Australia	69
Portugal	66
UK	60
India	48
France	43
USA	41
Unknown	31
China	27

Table 6: table showing the total number of male samples under the male_sample column per country represented within the country_name column.

Analysing the number of distinct centres supplying samples per country (7.6)

To look at the number of distinct centres within each country, the function do this this had to be established. Using w3resource discovered that using the DISTINCT function with the COUNT function will allow to count the distinct centres in each country (*SQL COUNT() with DISTINCT.* 2022). Using previous code as a template, was able to form a query able to identify amount of distinct centres using the code:

```
SELECT country_name , count( DISTINCT center) as numer_of_distinct_centers
FROM country_ids
INNER JOIN sample_characteristics ON country_ids.country = sample_characteristics.country
GROUP BY country_name ORDER BY numer_of_distinct_centers  DESC;
```

Which generated a table showing Italy had the most distinct centres supplying it samples.

country_name	numer_of_distinct_centers
Italy	29
Australia	27
Sri Lanka	27
UK	26
Portugal	25
India	24
France	23
USA	23
China	22
Unknown	22

Counting which centre country pair had the most samples (7.7)

The count the number of samples per centre country pair, the distinct values of centre and country for the samples is required. Used the SELECT DISTINCT function to select distinct values of each column (*SQL SELECT DISTINCT Statement.* 2022). By using group by with two variables enables separation of each value by two variables being center and country_name allowing the analysis of number of samples per pair of country and center

```
SELECT DISTINCT country_name,center,count(sample_characteristics.sample_id) as number_of_samples
FROM country_ids
INNER JOIN sample_characteristics ON country_ids.country = sample_characteristics.country
GROUP BY center,country_name ORDER BY count(sample_characteristics.sample_id)  DESC;
```

Figure 23: code used to form table giving number of samples per country_name and center pair. Sample_id was counted to determine number of samples per center and GROUB BY used two variables (century and countryname) to group number of samples per country name and center pairs.

This produced a table showing that centre 92 has produced the most Italian samples shown here:

country_name	center	number_of_samples
Italy	92	28
Australia	92	20
Sri Lanka	92	18
Italy	41	15
Sri Lanka	91	14
India	92	14
Italy	72	14
Portugal	81	13
Sri Lanka	42	12
Italy	81	12
Italy	91	12
UK	92	11
France	92	11
Italy	82	11

Table 7:truncated table showing the top 14 number of samples collected per country_name and center pair. The greatest amount of samples have been taken from Italy in center 92 at 28 samples .

Creating new data from old data

Creating new table from old tables (8.1)

To create the table with columns (sample_id,dob, sex,cc_status, age_recr,country,country_name,SNP1, SNP2,SNP3,SNP4, SNP5) from old data ,research was required to identify how to make a table from previously used data in other tables. Referred to the MySQL cookbook to suggest template code to use (*How to Create One Table From Another Table in SQL.* 2016). Then used the inner join function to only include the samples that where prevalent in both table sample_charactertics and sample_genotypes and to link the tables through their corresponding primary and foreign keys. This resulted in writing the correct query(figure 23b)

a.

```
SELECT
   id,
   name,
   price
INTO florist
FROM product
WHERE category='flower';
```

b.

```
CREATE TABLE sample_char_genotypes
AS SELECT
sample_characteristics.sample_id,dob,sex,cc_status,age_recr,
sample_characteristics.country,
country_ids.country_name,
SNP1,SNP2,SNP3,SNP4,SNP5
FROM country_ids
INNER JOIN sample_characteristics ON country_ids.country = sample_characteristics.country
INNER JOIN sample genotypes ON sample characteristics.sample id=sample genotypes.sample id
```

Figure 24: code used to create table sample_char_genotypes form data from sample_charcteristics,sample_genotypes and country_ids table. a) Template code from MySQL cookbook used to create code to from sample_char_genotype table . b)CREATE TABLE used to create a table names sample_char_genotypes , the AS SELECT function is used to select the right columns from all three previous tables. INNER JOIN used to link three previous tables together and to ensure values corresponding to each sample_id and country are the same across the tables.

This gave a table of 500 records shown here:

sample_id	dob	sex	cc_status	age_recr	country	country_name	SNP1	SNP2	SNP3	SNP4	SNP5
A00001_0016	09-Feb-25	2	1	48.96646	1	UK	0	2	1	0	1
A00001_0065	12-Nov-56	2	1	63.5948	1	UK	2	1	0	0	2
A00001_0095	24-Mar-43	2	1	56.61328	1	UK	0	0	1	0	0
A00001_0096	03-Jul-33	2	0	66.69678	1	UK	0	2	2	0	0
A00001_0115	11-May-46	1	1	26.59274	1	UK	1	2	0	0	2

Table 8 : Truncated table showing the top 5 results of the table of 500 records created from the code in (figure 23b)

Where any keys carrier along (8.2)

By using the mySQL workbench functionalities the database tab contains a reverse engineering function that will reverse engineer and EER diagram from the table/tables selected (MySQL,2022). Looking at the EER diagram no foreign or primary keys seem to have been transferred in the process.

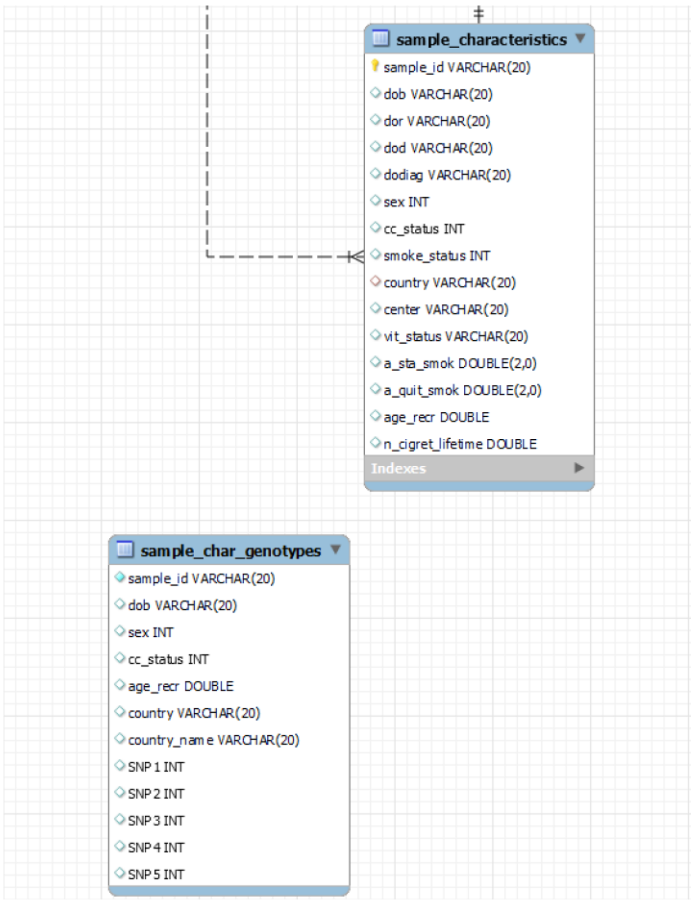
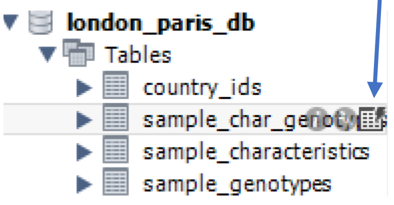


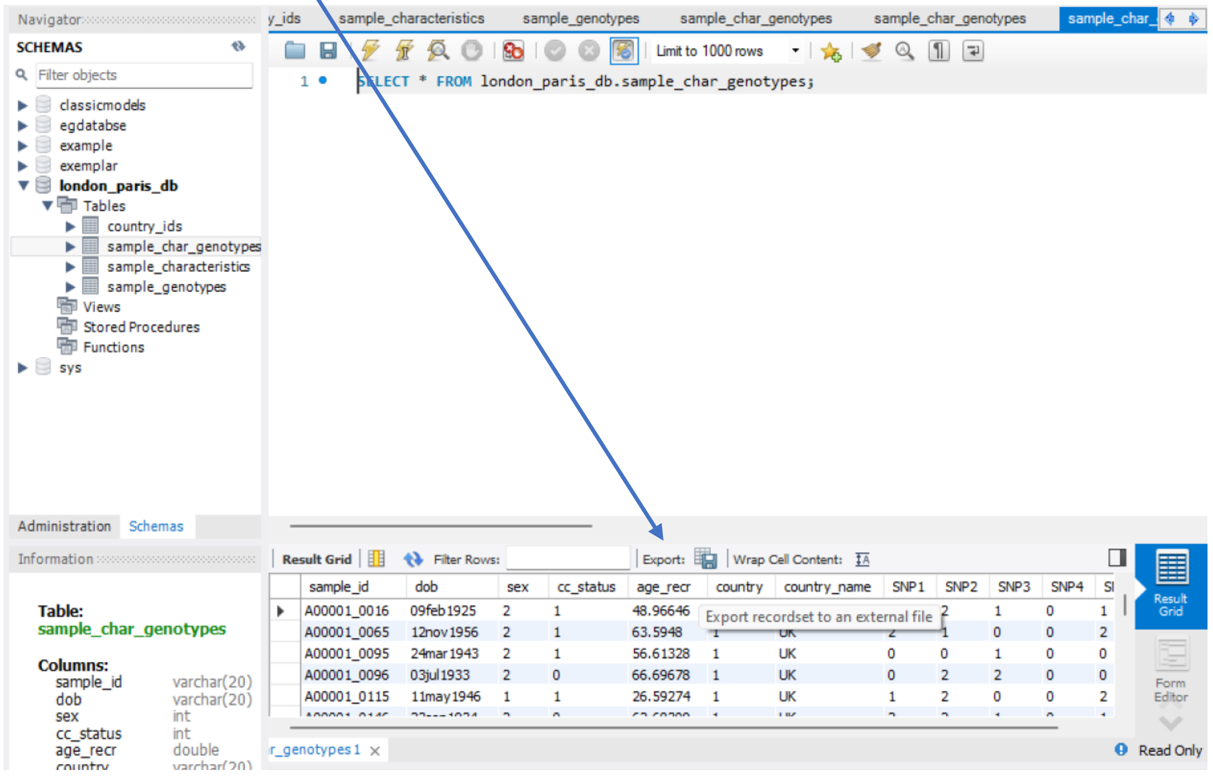
Figure 25: Figure showing section of EER diagram of new London_paris_db database including the new sample_char_genotypes table. The lack of arrow shows there is no link between the three databases and the sample_char_genotypes table seen by the lack of a link between sample_characteristics and sample_char_genotypes table. The lack of yellow or red bulb suggest the primary and foreign key have not been transferred and need to be added again.

exporting sample_char_genotypes as csv (8.3)

a)first clicked on the select table function in mySQL workbench.



b)Then clicked on the export button on the page for table



```
SELECT * FROM london_paris_db.sample_char_genotypes;
```

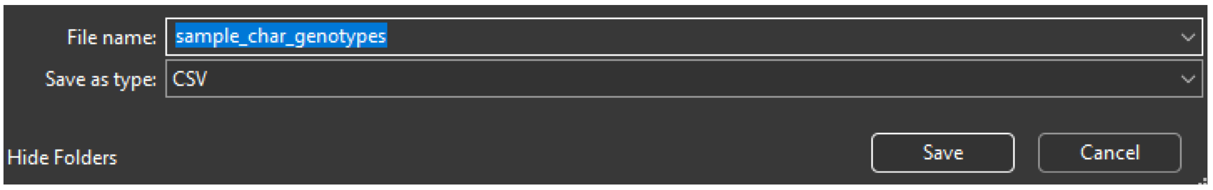
Table:
sample_char_genotypes

Columns:

sample_id	varchar(20)
dob	varchar(20)
sex	int
cc_status	int
age_recr	double
country	varchar(20)

sample_id	dob	sex	cc_status	age_recr	country	country_name	SNP1	SNP2	SNP3	SNP4	SI
A00001_0016	09feb1925	2	1	48.96646	[illegible]	[illegible]	[illegible]	2	1	0	1
A00001_0065	12nov1956	2	1	63.5948	1	UK	2	1	0	0	2
A00001_0095	24mar1943	2	1	56.61328	1	UK	0	0	1	0	0
A00001_0096	03jul1933	2	0	66.69678	1	UK	0	2	2	0	0
A00001_0115	11may1946	1	1	26.59274	1	UK	1	2	0	0	2

c)File can then finally be saved as a csv

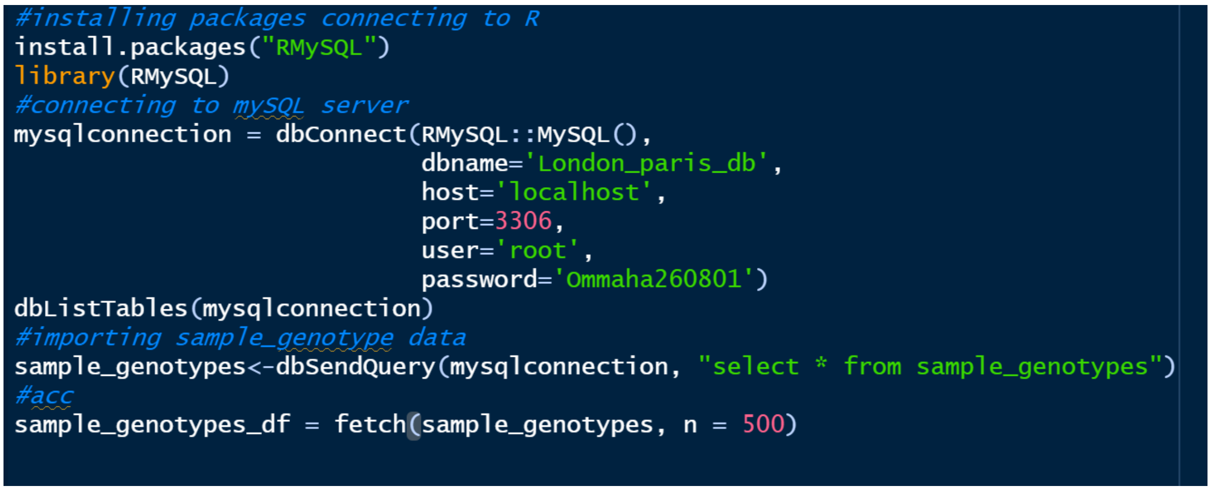


File name: sample_char_genotypes

Save as type: CSV

Connecting r to mySQL (9)

Referred to porjectpro website to garner solution to connecting r to mySQL (*How to connect to mysql using R -.* 2021).In which a small script was created to go with instructions as to how to run sql in R(figure26).

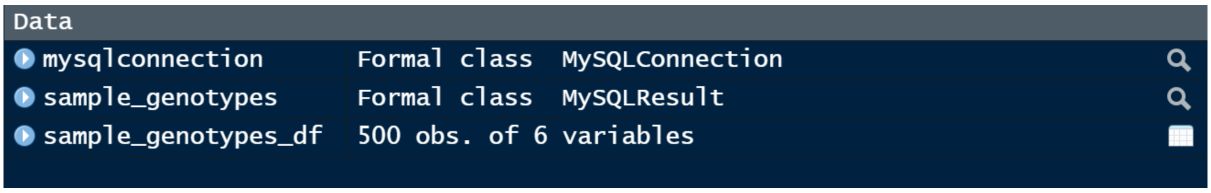


```r
#installing packages connecting to R
install.packages("RMySQL")
library(RMySQL)
#connecting to mySQL server
mysqlconnection = dbConnect(RMySQL::MySQL(),
                            dbname='London_paris_db',
                            host='localhost',
                            port=3306,
                            user='root',
                            password='Ommaha260801')
dbListTables(mysqlconnection)
#importing sample_genotype data
sample_genotypes<-dbSendQuery(mysqlconnection, "select * from sample_genotypes")
#acc
sample_genotypes_df = fetch(sample_genotypes, n = 500)
```

Figure 26 screenshot of R code detailing the connection process to MySQL

1. First installed the package 'RMySQL' so that SQL can interface with R.
2. Then loaded the package up an connected to my server using the db connect function
3. used 'dblisttables' to see the name of my tables
4. then sent a query to collect the data from the database being connected to
5. To load in the values within the table, used the function 'fetch' to bring in the values.

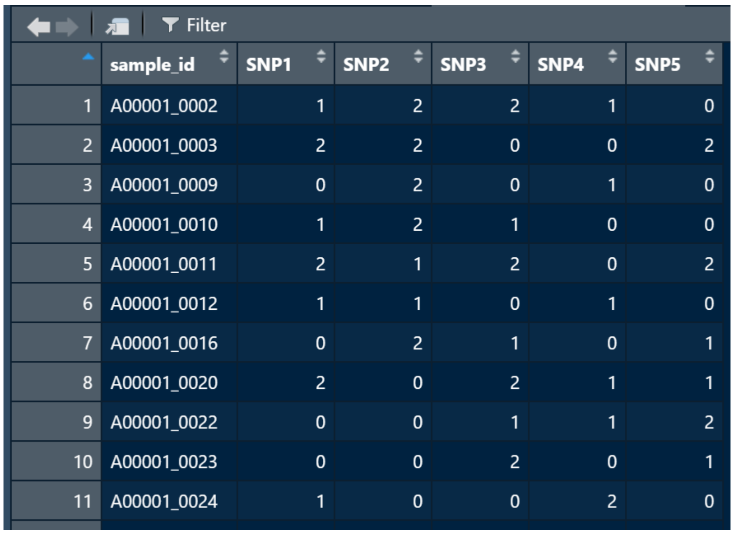
The data will be loaded in and can now be accessed through R and in the data section within Rstudios:



```
Data
mysqlconnection        Formal class  MySQLConnection
sample_genotypes       Formal class  MySQLResult
sample_genotypes_df    500 obs. of 6 variables
```

Figure 27: screenshot of data section in R containing the table sample_genotypes_df pulled from the MySQL localhost database

Accessing the data will provide the table as seen within SQL:



Filter

	sample_id	SNP1	SNP2	SNP3	SNP4	SNP5
1	A00001_0002	1	2	2	1	0
2	A00001_0003	2	2	0	0	2
3	A00001_0009	0	2	0	1	0
4	A00001_0010	1	2	1	0	0
5	A00001_0011	2	1	2	0	2
6	A00001_0012	1	1	0	1	0
7	A00001_0016	0	2	1	0	1
8	A00001_0020	2	0	2	1	1
9	A00001_0022	0	0	1	1	2
10	A00001_0023	0	0	2	0	1
11	A00001_0024	1	0	0	2	0

Figure 28: screenshot of SNP table showing the that the table is structured exactly the same as the sample_genotypes table in the MySQL localhost database

References

References

Error 3948 (42000) At Line 1: Loading Local Data Is Disabled; This Must Be Enabled On Both The Client And Server Sides With Code Examples. (2022) -09-25T07:54:15+00:00. https://www.folkstalk.com/2022/09/error-3948-42000-at-line-1-loading-local-data-is-disabled-this-must-be-enabled-on-both-the-client-and-server-sides-with-code-examples.html [Accessed Nov 12, 2022].

SQL AS. (2022) https://www.w3schools.com/sql/sql_ref_as.asp [Accessed Nov 16, 2022].

SQL COUNT() with DISTINCT. (2022) https://www.w3resource.com/sql/aggregate-functions/count-with-distinct.php [Accessed Nov 13, 2022].

SQL SELECT DISTINCT Statement. (2022) https://www.w3schools.com/sql/sql_distinct.asp [Accessed Nov 13, 2022].

How to connect to mysql using R -. (2021) https://www.projectpro.io/recipes/connect-mysql-r [Accessed Nov 13, 2022].

How to Create One Table From Another Table in SQL. (2016) https://learnsql.com/cookbook/how-to-create-one-table-from-another-table-in-sql/ [Accessed Nov 13, 2022].

How to Find the Minimum Value of a Column in SQL. (2016a) https://learnsql.com/cookbook/how-to-find-the-minimum-value-of-a-column-in-sql/ [Accessed Nov 13, 2022].

How to Find the Minimum Value of a Column in SQL. (2016b) https://learnsql.com/cookbook/how-to-find-the-minimum-value-of-a-column-in-sql/ [Accessed Nov 13, 2022].

SQL AVG() and CAST() inside Avg() for decimal (2022) value. https://www.w3resource.com/sql/aggregate-functions/avg-decimal-places-using-cast-within-avg.php [Accessed Nov 13, 2022].

SQL COUNT function. https://www.w3resource.com/sql/aggregate-functions/count-function.php [Accessed Nov 13, 2022].

SQL COUNT() with DISTINCT. https://www.w3resource.com/sql/aggregate-functions/count-with-distinct.php [Accessed 2022].

SQL FOREIGN KEY Constraint. https://www.w3schools.com/sql/sql_foreignkey.asp [Accessed Nov 13, 2022].

Baxter, M. (2020) MySQL Bugs: #91872: MySQL Workbench 8.0 restricts usage of LOAD DATA LOCAL INFILE. https://bugs.mysql.com/bug.php?id=91872). [Accessed Nov 13, 2022].

Erkec, E. (2020) SQL AVG() function introduction and examples. *SQL Shack - articles about database auditing, server performance, data recovery, and more*. -03-02T16:10:51+00:00. https://www.sqlshack.com/sql-avg-function-introduction-and-examples/ [Accessed Nov 13, 2022].

MySQL. (2022a) *MySQL :: MySQL 8.0 Reference Manual :: 13.2.7 LOAD DATA Statement*. https://dev.mysql.com/doc/refman/8.0/en/load-data.html . [Accessed Nov 13, 2022].

MySQL. (2022b) *MySQL :: MySQL 8.0 Reference Manual :: 3.3.1 Creating and Selecting a Database*. https://dev.mysql.com/doc/refman/8.0/en/creating-database.html [Accessed 13/11/2022].

Raghuvanshi, L. (2022) Sql Server query to count male,female and total employees | Get gender wise employee summary. *Asp.Net,C#.Net,VB.Net,MVC,jQuery,JavaScipt,AJAX,WCF,Sql Server example*. https://www.webcodeexpert.com/2015/05/sql-server-query-to-count-malefemale.html [Accessed Nov 13, 2022].