Basic SQL

File access required: In Colab this notebook requires first uploading files **Cities.csv**, **Countries.csv**, **Players.csv**, and **Teams.csv** using the *Files* feature in the left toolbar. If running the notebook on a local computer, simply ensure these files are in the same workspace as the notebook.

```
!pip install prettytable==0.7.2
!pip install ipython-sql
```

```
Collecting prettytable==0.7.2
  Downloading prettytable-0.7.2.zip (28 kB)
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: prettytable
  Building wheel for prettytable (setup.py) ... done
  Created wheel for prettytable: filename=prettytable-0.7.2-py3-none-any.whl size=13695 sha256=969fb822fbf30751f94b5060062bf0919
  Stored in directory: /root/.cache/pip/wheels/ca/f9/66/1ebeb8cdff2211eebb6fce02957f9e0a9ae3da4b7e65512d1b
Successfully built prettytable
Installing collected packages: prettytable
  Attempting uninstall: prettytable
    Found existing installation: prettytable 3.17.0
    Uninstalling prettytable-3.17.0:
      Successfully uninstalled prettytable-3.17.0
Successfully installed prettytable-0.7.2
Requirement already satisfied: ipython-sql in /usr/local/lib/python3.12/dist-packages (0.5.0)
Requirement already satisfied: prettytable in /usr/local/lib/python3.12/dist-packages (from ipython-sql) (0.7.2)
Requirement already satisfied: ipython in /usr/local/lib/python3.12/dist-packages (from ipython-sql) (7.34.0)
Requirement already satisfied: sqlalchemy>=2.0 in /usr/local/lib/python3.12/dist-packages (from ipython-sql) (2.0.45)
Requirement already satisfied: sqlparse in /usr/local/lib/python3.12/dist-packages (from ipython-sql) (0.5.4)
Requirement already satisfied: six in /usr/local/lib/python3.12/dist-packages (from ipython-sql) (1.17.0)
Requirement already satisfied: ipython-genutils in /usr/local/lib/python3.12/dist-packages (from ipython-sql) (0.2.0)
Requirement already satisfied: greenlet>=1 in /usr/local/lib/python3.12/dist-packages (from sqlalchemy>=2.0->ipython-sql) (3.3.0
Requirement already satisfied: typing-extensions>=4.6.0 in /usr/local/lib/python3.12/dist-packages (from sqlalchemy>=2.0->ipytho
Requirement already satisfied: setuptools>=18.5 in /usr/local/lib/python3.12/dist-packages (from ipython->ipython-sql) (75.2.0)
Collecting jedi>=0.16 (from ipython->ipython-sql)
  Downloading jedi-0.19.2-py2.py3-none-any.whl.metadata (22 kB)
Requirement already satisfied: decorator in /usr/local/lib/python3.12/dist-packages (from ipython->ipython-sql) (4.4.2)
Requirement already satisfied: pickleshare in /usr/local/lib/python3.12/dist-packages (from ipython->ipython-sql) (0.7.5)
Requirement already satisfied: traitlets>=4.2 in /usr/local/lib/python3.12/dist-packages (from ipython->ipython-sql) (5.7.1)
Requirement already satisfied: prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0 in /usr/local/lib/python3.12/dist-packages (from ipy
Requirement already satisfied: pygments in /usr/local/lib/python3.12/dist-packages (from ipython->ipython-sql) (2.19.2)
Requirement already satisfied: backcall in /usr/local/lib/python3.12/dist-packages (from ipython->ipython-sql) (0.2.0)
Requirement already satisfied: matplotlib-inline in /usr/local/lib/python3.12/dist-packages (from ipython->ipython-sql) (0.2.1)
Requirement already satisfied: pexpect>4.3 in /usr/local/lib/python3.12/dist-packages (from ipython->ipython-sql) (4.9.0)
Requirement already satisfied: parso<0.9.0,>=0.8.4 in /usr/local/lib/python3.12/dist-packages (from jedi>=0.16->ipython->ipython
Requirement already satisfied: ptyprocess>=0.5 in /usr/local/lib/python3.12/dist-packages (from pexpect>4.3->ipython->ipython-sq
Requirement already satisfied: wcwidth in /usr/local/lib/python3.12/dist-packages (from prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2
Downloading jedi-0.19.2-py2.py3-none-any.whl (1.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.6/1.6 MB 61.7 MB/s eta 0:00:00
Installing collected packages: jedi
Successfully installed jedi-0.19.2
```

```
# Set-up
%load_ext sql
%sql sqlite://
import pandas as pd
```

```
# Create database tables from CSV files
with open('Cities.csv') as f: Cities = pd.read_csv(f, index_col=0)
%sql drop table if exists Cities;
%sql --persist Cities


with open('Countries.csv') as f: Countries = pd.read_csv(f, index_col=0)
%sql drop table if exists Countries;
%sql --persist Countries
```

```
 * sqlite://
Done.
 * sqlite://
 * sqlite://
Done.
 * sqlite://
'Persisted countries'
```

Look at sample of Cities and Countries tables

```
%%sql
select * from Cities limit 5
```

```
 * sqlite://
Done.
```

city	country	latitude	longitude	temperature
Aalborg	Denmark	57.03	9.92	7.52
Aberdeen	United Kingdom	57.17	-2.08	8.1
Abisko	Sweden	63.35	18.83	0.2
Adana	Turkey	36.99	35.32	18.67
Albacete	Spain	39.0	-1.87	12.62

```
%%sql
select * from Countries limit 5
```

```
 * sqlite://
Done.
```

country	population	EU	coastline
Albania	2.9	no	yes
Andorra	0.07	no	no
Austria	8.57	yes	no
Belarus	9.48	no	no
Belgium	11.37	yes	yes

Basic Select statement

Select columns
From tables
Where condition

Find all countries not in the EU

```
%%sql
select country
from Countries
where EU = 'no'
```

```
 * sqlite://
Done.
```

country
Albania
Andorra
Belarus
Bosnia and Herzegovina
Iceland
Kosovo
Liechtenstein
Macedonia
Moldova
Montenegro
Norway
Serbia
Switzerland
Turkey
Ukraine

Find all cities with temperature between -5 and 5; return city, country, and temperature

```
%%sql
select city, country, temperature
from Cities
where temperature > -5 and temperature < 5
```

```
 * sqlite://
Done.
   city    country  temperature
Abisko     Sweden   0.2
Augsburg   Germany  4.54
Bergen     Norway   1.75
Bodo       Norway   4.5
Helsinki   Finland  4.19
Innsbruck  Austria  4.54
Kiruna     Sweden   -2.2
Orsha      Belarus  4.93
Oslo       Norway   2.32
Oulu       Finland  1.45
Salzburg   Austria  4.62
Tallinn    Estonia  4.82
Tampere    Finland  3.59
Tartu      Estonia  4.36
Trondheim  Norway   4.53
Turku      Finland  4.72
Uppsala    Sweden   4.17
```

Ordering

Modify previous query to sort by temperature

```
%%sql
select city, country, temperature
from Cities
where temperature > -5 and temperature < 5
order by temperature
```

```
 * sqlite://
Done.
   city    country  temperature
Kiruna     Sweden   -2.2
Abisko     Sweden   0.2
Oulu       Finland  1.45
Bergen     Norway   1.75
Oslo       Norway   2.32
Tampere    Finland  3.59
Uppsala    Sweden   4.17
Helsinki   Finland  4.19
Tartu      Estonia  4.36
Bodo       Norway   4.5
Trondheim  Norway   4.53
Augsburg   Germany  4.54
Innsbruck  Austria  4.54
Salzburg   Austria  4.62
Turku      Finland  4.72
Tallinn    Estonia  4.82
Orsha      Belarus  4.93
```

Modify previous query to sort by country, then temperature descending

```
%%sql
select city, country, temperature
from Cities
where temperature > -5 and temperature < 5
order by country ASC, temperature DESC
```

```
 * sqlite://
Done.
  city      country  temperature
Salzburg   Austria   4.62
Innsbruck  Austria   4.54
Orsha      Belarus   4.93
Tallinn    Estonia   4.82
Tartu      Estonia   4.36
Turku      Finland   4.72
Helsinki   Finland   4.19
Tampere    Finland   3.59
Oulu       Finland   1.45
Augsburg   Germany   4.54
Trondheim  Norway    4.53
Bodo       Norway    4.5
Oslo       Norway    2.32
Bergen     Norway    1.75
Uppsala    Sweden    4.17
Abisko     Sweden    0.2
Kiruna     Sweden    -2.2
```

Your Turn

Find all countries with no coastline and with population > 9. Return the country and population, in descending order of population.

```
%%sql
select country, population
from Countries
where coastline = 'no' and population > 9
order by population DESC
```

```
 * sqlite://
Done.
   country       population
Czech Republic  10.55
Hungary         9.82
Belarus         9.48
```

Multiple tables in From clause - Joins

Find all cities with longitude < 10 not in the EU, return city and longitude

```
Cities.head(2) # python command = dataframe
```

```
                 country   latitude  longitude  temperature
    city
Aalborg          Denmark     57.03       9.92         7.52
Aberdeen  United Kingdom     57.17      -2.08         8.10
```

Next steps: Generate code with Cities   New interactive sheet

```
Countries.head(2)
```

```
          population  EU  coastline
country
Albania         2.90  no        yes
Andorra         0.07  no         no
```

Next steps: Generate code with Countries   New interactive sheet

```
%%sql
select city, longitude
from Cities, Countries -- 2 tables
where Cities.country = Countries.country -- get data from the two tables.
and longitude < 10 and EU = 'no' -- this are their conditions

-- SQL: comment "--"
```

```
 * sqlite://
Done.
   city     longitude
Andorra     1.52
Basel       7.59
Bergen      5.32
Geneva      6.14
Stavanger   5.68
Zurich      8.56
```

Modify previous query to also return country (error then fix)

```
%%sql
select city, longitude, Cities.country
from Cities, Countries
where Cities.country = Countries.country -- if they have the same field name. put the table name in the select if you need to c
and longitude < 10 and EU = 'no'
```

```
 * sqlite://
Done.
   city     longitude   country
Andorra     1.52        Andorra
Basel       7.59        Switzerland
Bergen      5.32        Norway
Geneva      6.14        Switzerland
Stavanger   5.68        Norway
Zurich      8.56        Switzerland
```

Find all cities with latitude < 50 in a country with population < 5; return city, country, and population, sorted by country

```
%%sql
select city, Cities.country, population
from Cities, Countries
where Cities.country = Countries.country
and latitude < 50 and population < 5
order by Cities.country
```

```
 * sqlite://
Done.
   city            country           population
Elbasan     Albania                  2.9
Andorra     Andorra                  0.07
Sarajevo    Bosnia and Herzegovina   3.8
Rijeka      Croatia                  4.23
Split       Croatia                  4.23
Skopje      Macedonia                2.08
Balti       Moldova                  4.06
Chisinau    Moldova                  4.06
Podgorica   Montenegro               0.63
Ljubljana   Slovenia                 2.07
```

Inner Join -- just FYI

Same query as above

```
%%sql
select city, Cities.country, population
from Cities inner join Countries
     on Cities.country = Countries.country -- condition of the INNER JOIN.
where latitude < 50 and population < 5
order by Cities.country
```

```
 * sqlite://
Done.
```

city	country	population
Elbasan	Albania	2.9
Andorra	Andorra	0.07
Sarajevo	Bosnia and Herzegovina	3.8
Rijeka	Croatia	4.23
Split	Croatia	4.23
Skopje	Macedonia	2.08
Balti	Moldova	4.06
Chisinau	Moldova	4.06
Podgorica	Montenegro	0.63
Ljubljana	Slovenia	2.07

Select *

Modify previous queries to return all columns

Your Turn

Find all cities with latitude > 45 in a country with no coastline and with population > 9. Return the city, country, latitude, and whether it's in the EU.

```
%%sql
select city, Cities.country, latitude, EU
from Cities, Countries
where Cities.country = Countries.country
and latitude > 45 and coastline = 'no' and population > 9
```

```
 * sqlite://
Done.
```

city	country	latitude	EU
Brest	Belarus	52.1	no
Brno	Czech Republic	49.2	yes
Budapest	Hungary	47.5	yes
Debrecen	Hungary	47.53	yes
Gyor	Hungary	47.7	yes
Hrodna	Belarus	53.68	no
Mazyr	Belarus	52.05	no
Minsk	Belarus	53.9	no
Orsha	Belarus	54.52	no
Ostrava	Czech Republic	49.83	yes
Pinsk	Belarus	52.13	no
Prague	Czech Republic	50.08	yes
Szeged	Hungary	46.25	yes

Aggregation and Grouping

Find the average temperature for all cities

```
%%sql
select avg(temperature) as avgTemp
from Cities
```

```
 * sqlite://
Done.
```

avgTemp
9.497840375586858

Modify previous query to find average temperature of cities with latitude > 55

```
%%sql
select avg(temperature)
from Cities
where latitude > 55
```

```
 * sqlite://
Done.
avg(temperature)
4.985185185185185
```

Modify previous query to also find minimum and maxiumum temperature of cities with latitude > 55

```
%%sql
select min(temperature) as Min_val, max(temperature) as Max_val
from Cities
where latitude > 55
```

```
 * sqlite://
Done.
Min_val Max_val
-2.2    8.6
```

Modify previous query to return number of cities with latitude > 55

Rename result column as northerns

```
Cities.head(1)
```

	country	latitude	longitude	temperature
city				
Aalborg	Denmark	57.03	9.92	7.52

Next steps: Generate code with Cities   New interactive sheet

```
Countries.head(1)
```

	population	EU	coastline
country			
Albania	2.9	no	yes

Next steps: Generate code with Countries   New interactive sheet

Find the minimum and maximum temperature of cities in the EU (then not in the EU)

```
%%sql
select min(temperature), max(temperature)
from Cities, Countries
where Cities.country = Countries.Country
and EU = 'no'
```

```
 * sqlite://
Done.
min(temperature) max(temperature)
1.75             18.67
```

Your Turn

Find the number of cities with latitude > 45 in countries with no coastline and with population > 9; also return the minimum and maximum latitude among those cities

```
%%sql
select count(city) as num_cities, min(latitude) as min_lat, max(latitude) as max_lat
from Cities
join countries on Cities.country = Countries.country
where latitude > 45 and coastline = 'no' and population > 9
```

```
 * sqlite://
Done.
num_cities min_lat max_lat
13         46.25   54.52
```

Find the average temperature for each country

```
%%sql
select country, avg(temperature)
from Cities
group by country
```

```
 * sqlite://
Done.
        country              avg(temperature)
Albania                  15.18
Andorra                  9.6
Austria                  6.144
Belarus                  5.946666666666666
Belgium                  9.65
Bosnia and Herzegovina   9.6
Bulgaria                 10.44
Croatia                  10.865
Czech Republic           7.8566666666666665
Denmark                  7.625
Estonia                  4.59
Finland                  3.4875
France                   10.151111111111112
Germany                  7.869285714285714
Greece                   16.9025
Hungary                  9.6025
Ireland                  9.299999999999999
Italy                    13.474666666666668
Latvia                   5.27
Lithuania                6.1433333333333335
Macedonia                9.36
Moldova                  8.415
Montenegro               9.99
Netherlands              8.756666666666668
Norway                   3.7260000000000004
Poland                   7.250000000000002
Portugal                 14.469999999999999
Romania                  9.224444444444444
Serbia                   9.85
Slovakia                 8.48
Slovenia                 9.27
Spain                    14.238333333333332
Sweden                   3.5866666666666673
Switzerland              7.253333333333333
Turkey                   11.726666666666665
Ukraine                  7.420000000000002
United Kingdom           8.649999999999999
```

Modify previous query to sort by descending average temperature

Modify previous query to show countries only

Find the average temperature for cities in countries with and without coastline

```
%%sql
select coastline, avg(temperature)
```

```
from Cities, Countries
where Cities.country = Countries.country
group by coastline
```

```
 * sqlite://
Done.
coastline  avg(temperature)
no         7.748000000000001
yes        9.784699453551914
```

Modify previous query to find the average temperature for cities in the EU and not in the EU, then all combinations of coastline and EU

Modify previous query to only include cities with latitude < 50, then latitude < 40

Your Turn

For each country in the EU, find the latitude of the northernmost city in the country, i.e., the maximum latitude. Return the country and its maximum latitude, in descending order of maximum latitude.

```
%%sql
select coastline, avg(temperature)
from Cities, Countries
where Cities.country = Countries.country
group by coastline
```

```
 * sqlite://
Done.
coastline  avg(temperature)
no         7.748000000000001
yes        9.784699453551914
```

A Bug in SQLite - just FYI

```
%%sql
select country, avg(temperature)
from Cities
group by country
```

```
 * sqlite://
Done.
        country            avg(temperature)
Albania                 15.18
Andorra                 9.6
Austria                 6.144
Belarus                 5.946666666666666
Belgium                 9.65
Bosnia and Herzegovina  9.6
Bulgaria                10.44
Croatia                 10.865
Czech Republic          7.8566666666666665
Denmark                 7.625
Estonia                 4.59
Finland                 3.4875
France                  10.151111111111112
Germany                 7.869285714285714
Greece                  16.9025
Hungary                 9.6025
Ireland                 9.299999999999999
Italy                   13.474666666666668
Latvia                  5.27
Lithuania               6.1433333333333335
Macedonia               9.36
Moldova                 8.415
Montenegro              9.99
Netherlands             8.756666666666668
Norway                  3.7260000000000004
Poland                  7.250000000000002
Portugal                14.469999999999999
Romania                 9.224444444444444
Serbia                  9.85
Slovakia                8.48
Slovenia                9.27
Spain                   14.238333333333332
Sweden                  3.5866666666666673
Switzerland             7.253333333333333
Turkey                  11.726666666666665
Ukraine                 7.420000000000002
United Kingdom          8.649999999999999
```

Modify previous query - add city to Select clause

Now focus on Austria and Sweden

```
%%sql
select *
from Cities
where country = 'Austria' or country = 'Sweden'
order by country
```

```
 * sqlite://
Done.
   city    country latitude longitude temperature
Graz      Austria  47.08    15.41     6.91
Innsbruck Austria  47.28    11.41     4.54
Linz      Austria  48.32    14.29     6.79
Salzburg  Austria  47.81    13.04     4.62
Vienna    Austria  48.2     16.37     7.86
Abisko    Sweden   63.35    18.83     0.2
Göteborg  Sweden   57.75    12.0      5.76
Kiruna    Sweden   67.85    20.22     -2.2
Malmö     Sweden   55.58    13.03     7.33
Stockholm Sweden   59.35    18.1      6.26
Uppsala   Sweden   59.86    17.64     4.17
```

```
%%sql
select country, city, avg(temperature)
from Cities
where country = 'Austria' or country = 'Sweden'
group by country
```

```
 * sqlite://
Done.
country  city    avg(temperature)
Austria  Graz    6.144
Sweden   Abisko  3.5866666666666673
```

Modify previous query to min(temperature), max(temperature), then together in both orders

The Limit clause

Return any three countries with population > 20

```
%%sql
select country
from Countries
where population > 20
limit 3
```

```
 * sqlite://
Done.
country
France
Germany
Italy
```

Find the ten coldest cities

```
%%sql
select city, temperature
from Cities
order by temperature
limit 10
```

```
 * sqlite://
Done.
  city    temperature
Kiruna    -2.2
Abisko    0.2
Oulu      1.45
Bergen    1.75
Oslo      2.32
Tampere   3.59
Uppsala   4.17
Helsinki  4.19
Tartu     4.36
Bodo      4.5
```

Your Turn

Find the five easternmost (greatest longitude) cities in countries with no coastline. Return the city and country names.

```
%%sql
select city,
       Cities.country
from Cities
join Countries
  on Cities.country = Countries.country
where coastline = 'no'
order by longitude desc
limit 5;
```

```
 * sqlite://
Done.
   city    country
Orsha      Belarus
Mazyr      Belarus
Chisinau   Moldova
Balti      Moldova
Minsk      Belarus
```

Your Turn - Basic SQL on World Cup Data

```
# Create database tables from CSV files
with open('Players.csv') as f: Players = pd.read_csv(f, index_col=0)
%sql drop table if exists Players;
%sql --persist Players
with open('Teams.csv') as f: Teams = pd.read_csv(f, index_col=0)
%sql drop table if exists Teams;
%sql --persist Teams
```

```
 * sqlite://
Done.
 * sqlite://
 * sqlite://
Done.
 * sqlite://
'Persisted teams'
```

Look at sample of Players and Teams tables

```
%%sql
select * from Players limit 5
```

```
 * sqlite://
Done.
 surname    team    position   minutes shots passes tackles saves
Abdoun     Algeria midfielder  16       0     6      0       0
Belhadj    Algeria defender    270      1     146    8       0
Boudebouz  Algeria midfielder  74       3     28     1       0
Bougherra  Algeria defender    270      1     89     11      0
Chaouchi   Algeria goalkeeper  90       0     17     0       2
```

```
%%sql
select * from Teams limit 5
```

```
 * sqlite://
Done.
    team      ranking games wins draws losses goalsFor goalsAgainst yellowCards redCards
Brazil        1       5     3    1     1      9        4            7           2
Spain         2       6     5    0     1      7        2            3           0
Portugal      3       4     1    2     1      7        1            8           1
Netherlands   4       6     6    0     0      12       5            15          0
Italy         5       3     0    2     1      4        5            5           0
```

1) What player on a team with "ia" in the team name played less than 200 minutes and made more than 100 passes? Return the player surname. Note: To check if attribute A contains string S use "A like '%S%'"

```
%%sql
select surname
from Players
join Teams
  on Players.team = Teams.team
where Teams.team like '%ia%'
  and minutes < 200
  and passes > 100;
```

```
 * sqlite://
Done.
 surname
Kuzmanovic
```

2) Find all players who took more than 20 shots. Return all player information in descending order of shots taken.

```
%%sql
select *
from Players
where shots > 20
order by shots DESC
```

```
 * sqlite://
Done.
surname  team      position minutes shots passes tackles saves
Gyan     Ghana     forward  501     27    151    1       0
Villa    Spain     forward  529     22    169    2       0
Messi    Argentina forward  450     21    321    10      0
```

3) Find the goalkeepers of teams that played more than four games. List the surname of the goalkeeper, the team, and the number of minutes the goalkeeper played.

```
%%sql
select surname,
       Players.team,
       minutes
from Players
join Teams
  on Players.team = Teams.team
where position = 'Goalkeeper'
  and games > 4;
```

```
 * sqlite://
Done.
surname team minutes
```

4) How many players who play on a team with ranking <10 played more than 350 minutes? Return one number in a column named 'superstar'.

```
%%sql
select count(*) as superstar
from Players
join Teams on Players.team = Teams.team
where ranking < 10 and minutes > 350
```

```
 * sqlite://
Done.
superstar
54
```

5) What is the average number of passes made by forwards? By midfielders? Write one query that gives both values with the corresponding position.

```
%%sql
select position, avg(passes)
from Players
group by position
```

```
 * sqlite://
Done.
 position     avg(passes)
defender   102.6436170212766
forward    50.82517482517483
goalkeeper 55.638888888888886
midfielder 95.2719298245614
```

6) Which team has the highest ratio of goalsFor to goalsAgainst? Return the team and the ratio.

```
%%sql
select team, goalsFor/goalsAgainst
from Teams
order by goalsFor/goalsAgainst DESC
limit 1
```

```
 * sqlite://
Done.
 team   goalsFor/goalsAgainst
Portugal 7
```

Your Turn Extra - Basic SQL on Titanic Data

File access required: In Colab these extra problems require first uploading **Titanic.csv** using the *Files* feature in the left toolbar. If running the notebook on a local computer, simply ensure this file is in the same workspace as the notebook.

```
# Create database table from CSV file
with open('Titanic.csv') as f: Titanic = pd.read_csv(f, index_col=0)
%sql drop table if exists Titanic;
%sql --persist Titanic
```

```
 * sqlite://
Done.
 * sqlite://
'Persisted titanic'
```

Look at sample of Titanic table

```
%%sql
select * from Titanic limit 5
```

```
 * sqlite://
Done.
 last              first               gender age  class  fare   embarked   survived
Abbing   Mr. Anthony                    M      42.0 3     7.55  Southampton no
Abbott   Mrs. Stanton (Rosa Hunt)       F      35.0 3     20.25 Southampton yes
Abbott   Mr. Rossmore Edward            M      16.0 3     20.25 Southampton no
Abelson Mr. Samuel                      M      30.0 2     24.0  Cherbourg   no
Abelson Mrs. Samuel (Hannah Wizosky)   F      28.0 2     24.0  Cherbourg   yes
```

1) How many passengers sailed for free (i.e, fare is zero)?

```
%%sql
select count(*) as free_passengers
from Titanic
where fare = 0;
```

```
 * sqlite://
Done.
free_passengers
15
```

2) How many married women over age 50 embarked in Cherbourg? (Married women's first names begin with "Mrs."). Note: To check if attribute A begins with string S use "A like 'S%'"

```
%%sql
select count(*) as married_over_50_cherbourg
from Titanic
where gender = 'F'
  and age > 50
  and embarked = 'Cherbourg'
  and first like 'Mrs.%';
```

```
 * sqlite://
Done.
married_over_50_cherbourg
```

3) Write three queries to find: (i) the total number of passengers; (ii) the number of passengers under 18; (iii) the number of passengers 18 or older. Notice that the second and third numbers don't add up to the first.

```
%%sql
select count(*) as total_passengers
from Titanic;
```

```
 * sqlite://
Done.
total_passengers
891
```

```
%%sql
select count(*) as total_passengers
from Titanic
where age <18;
```

```
 * sqlite://
Done.
total_passengers
113
```

```
%%sql
select count(*) as total_passengers
from Titanic
where age >= 18;
```

```
 * sqlite://
Done.
total_passengers
601
```

Missing values in SQL tables are given a special value called 'null', and conditions 'A is null' and 'A is not null' can be use in Where clauses to check whether attribute A has the 'null' value. Write a query to find the number of passengers whose age is missing -- now your passenger numbers should add up. Modify the query to also return the average fare paid by those passengers.

```
%%sql
select count(*) as missing_age,
avg(fare) as avg_missing_age
from Titanic
where age is null;
```

```
 * sqlite://
Done.
missing_age  avg_missing_age
177          22.159491525423757
```

4) Find all passengers whose age is not an integer; return last name, first name, and age, from youngest to oldest. Note: Consider using the round() function

```
%%sql
select last, first, age
from Titanic
where age is not round(age)
order by age
```

```
 * sqlite://
Done.
```

last	first	age
Thomas	Master Assad Alexander	0.42
Hamalainen	Master Viljo	0.67
Baclini	Miss Helene Barbara	0.75
Baclini	Miss Eugenie	0.75
Caldwell	Master Alden Gates	0.83
Richards	Master George Sibley	0.83
Allison	Master Hudson Trevor	0.92
Zabour	Miss Hileni	14.5
Lovell	Mr. John Hall ("Henry")	20.5
Hanna	Mr. Mansour	23.5
Sawyer	Mr. Frederick Charles	24.5
Novel	Mr. Mansouer	28.5
Williams	Mr. Leslie	28.5
Mangan	Miss Mary	30.5
Tomlin	Mr. Ernest Portage	30.5
Nasser	Mr. Nicholas	32.5
Webber	Miss Susan	32.5
Lemberopolous	Mr. Peter L	34.5
Navratil	Mr. Michel ("Louis M Hoffman")	36.5

5) What is the most common last name among passengers, and how many passengers have that last name?

```
%%sql
select last, count(*) as count
from Titanic
group by last
order by count DESC
limit 1
```