MedPracDB

April 2, 2023

1 Medical Practice Database

1.1 Database Course Final Project Part 2

1.2 Code version of 31March2023

1.3 Alex Bordanca & David Thiriot

1.4 Create SQLite database

[1]:
```
import sqlite3
import pandas as pd
```

[2]:
```
from pathlib import Path
Path('MedPracDB.db').touch()
# David Thiriot

#reference: https://mungingdata.com/sqlite/create-database-load-csv-python/
# This is a way to create an empty new .db file as a starting point
```

[3]:
```
# David Thiriot

conn = sqlite3.connect('MedPracDB.db')
c = conn.cursor()
```

[4]:
```
# David Thiriot

c.execute('''DROP TABLE IF EXISTS Doctors''')
c.execute('''CREATE TABLE Doctors (Doctor_ID int PRIMARY KEY ,
                                   Firstname text,
                                   Lastname text)''')
doctors_table = pd.read_csv('doctors table.csv', header=0)
doctors_table.to_sql('Doctors', conn, if_exists='append', index=False)
```

[4]:
```
10
```

[5]:
```
# David Thiriot

c.execute('''DROP TABLE IF EXISTS Patient_contact''')
```

```
c.execute('''CREATE TABLE Patient_contact (Patient_ID int PRIMARY KEY,
                                           Email text,
                                           Street_address text,
                                           City text,
                                           State text,
                                           Zip int)''')
patient_contact_table = pd.read_csv('patient contact table.csv', header=0)
patient_contact_table.to_sql('Patient_contact', conn, if_exists='append',␣
 ↪index=False)
```

```
[5]: 1000
```

```
[6]: # David Thiriot

c.execute('''DROP TABLE IF EXISTS Patient_finance''')
c.execute('''CREATE TABLE Patient_finance (Patient_ID int PRIMARY KEY,
                                           Amount_due real,
                                           Ins_co text)''')
patient_finance_table = pd.read_csv('patient finance table.csv', header=0)
patient_finance_table.to_sql('Patient_finance', conn, if_exists='append',␣
 ↪index=False)
```

```
[6]: 1000
```

```
[7]: # David Thiriot

c.execute('''DROP TABLE IF EXISTS Patient_doctors''')
c.execute('''CREATE TABLE Patient_doctors (Patient_ID int PRIMARY KEY,
                                           Doctor text NOT NULL)''')
patient_doctors_table = pd.read_csv('patient doctors table.csv', header=0)
patient_doctors_table.to_sql('Patient_doctors', conn, if_exists='append',␣
 ↪index=False)
```

```
[7]: 1000
```

```
[8]: # David Thiriot

c.execute('''DROP TABLE IF EXISTS Patient_health''')
c.execute('''CREATE TABLE Patient_health (Patient_ID int PRIMARY KEY,
                                          Current_smoker int,
                                          Condition_1 int,
                                          Condition_2 int,
                                          Condition_3 int,
                                          Condition_4 int,
                                          Condition_5 int,
                                          Condition_6 int,
                                          Condition_7 int,
                                                          Condition_8 int,
                                                          Condition_9 int,
                                                          Condition_10 int)''')
patient_health_table = pd.read_csv('patient health table.csv', header=0)
patient_health_table.to_sql('Patient_health', conn, if_exists='append',␣
  ↪index=False)
```

```
[8]: 1000
```

```
[9]: # David Thiriot

c.execute('''DROP TABLE IF EXISTS Patient_ID''')  # Require at least 1 name. ␣
  ↪That would go in the Firstname field.
c.execute('''CREATE TABLE Patient_ID (Patient_ID int PRIMARY KEY,
                                      Firstname text NOT NULL,
                                      Lastname text,
                                      DOB text,
                                      Age int,
                                      Biol_sex text,
                                      Ethnicity text)''')
patient_ID_table = pd.read_csv('patient ID table.csv', header=0)
patient_ID_table.to_sql('Patient_ID', conn, if_exists='append', index=False)
```

```
[9]: 1000
```

```
[10]: # David Thiriot

c.execute('''DROP TABLE IF EXISTS Patient_vitals''')
c.execute('''CREATE TABLE Patient_vitals (Patient_ID int PRIMARY KEY,
                                          Last_height real,
                                          Last_weight real,
                                          Last_heartrate real,
                                          Last_systolic_BP real,
                                          Last_diastolic_BP real)''')
patient_vitals_table = pd.read_csv('patient vitals table.csv', header=0)
patient_vitals_table.to_sql('Patient_vitals', conn, if_exists='append',␣
  ↪index=False)
```

```
[10]: 1000
```

```
[11]: # David Thiriot

c.execute('''SELECT * FROM Doctors''')

# Read the selection from the database into a pandas dataframe - looks nicer␣
  ↪and easier to work with
colnames = c.description   # gather column names from a new query
```

```
colnames_list = []
for row in colnames:
    colnames_list.append(row[0])

df = pd.DataFrame(c.fetchall(), columns=colnames_list)
df
```

[11]:
```
   Doctor_ID Firstname   Lastname
0          1   Michael      Smith
1          2     Sally   Williams
2          3    Dennis      Jones
3          4     Juana  Rodriguez
4          5        Li      Zhang
5          6   Ernesto      Perez
6          7  Veronica    Jackson
7          8  Patricia     Harris
8          9   Suleman    Tataryn
9         10     Alain      Petit
```

[12]:
```
# David Thiriot

c.execute('''SELECT * FROM Patient_doctors''')

# Read the selection from the database into a pandas dataframe - looks nicer␣
 ↪and easier to work with
colnames = c.description   # gather column names from a new query
colnames_list = []
for row in colnames:
    colnames_list.append(row[0])

df = pd.DataFrame(c.fetchall(), columns=colnames_list)
df.head(10)
```

[12]:
```
   Patient_ID    Doctor
0           1     Zhang
1           2   Tataryn
2           3    Harris
3           4     Jones
4           5     Petit
5           6  Williams
6           7     Smith
7           8     Smith
8           9     Smith
9          10     Smith
```

[13]:
```
# David Thiriot
```

```
c.execute('''SELECT * FROM Patient_contact''')

# Read the selection from the database into a pandas dataframe - looks nicer␣
↪and easier to work with
colnames = c.description   # gather column names from a new query
colnames_list = []
for row in colnames:
    colnames_list.append(row[0])

df = pd.DataFrame(c.fetchall(), columns=colnames_list)
df.head(10)
```

[13]:
```
   Patient_ID                Email               Street_address      City  \
0           1    YunPor5@gmail.com          872 Jakubowski Creek  New York
1           2    ColJal8@yahoo.com        347 Immanuel Mountains  New York
2           3    LigAus2@gmail.com         6040 Williamson Curve  New York
3           4      AliJor1@aol.com           93049 Audley Island  New York
4           5    DzeMer1@gmail.com     939 Nicolas Loaf Suite 330  New York
5           6  al-Bad5@verizon.net  4726 Warren Square Suite 033  New York
6           7      HriNat3@aol.com          7683 Connelly Knolls  New York
7           8      WatAle4@aol.com               684 Ivette Isle  New York
8           9    JohKia8@gmail.com      1373 Ancel Cape Suite 589  New York
9          10    RadTin7@gmail.com  124 Vicente Shores Suite 283  New York

      State    Zip
0  New York  10007
1  New York  10007
2  New York  10012
3  New York  10006
4  New York  10012
5  New York  10013
6  New York  10006
7  New York  10005
8  New York  10012
9  New York  10002
```

[14]:
```
# David Thiriot

c.execute('''SELECT * FROM Patient_finance''')

# Read the selection from the database into a pandas dataframe - looks nicer␣
↪and easier to work with
colnames = c.description   # gather column names from a new query
colnames_list = []
for row in colnames:
    colnames_list.append(row[0])
```

```
df = pd.DataFrame(c.fetchall(), columns=colnames_list)
df.head(10)
```

```
[14]:    Patient_ID  Amount_due                       Ins_co
      0           1       400.0               CityPlan Health
      1           2         0.0             MetroCare Basic
      2           3       300.0                      LifeWell
      3           4         0.0             New Day Medical
      4           5         0.0             Healthplan Plus
      5           6      5300.0  Health Partners of New York
      6           7      5700.0             Healthplan Plus
      7           8      2000.0                      LifeWell
      8           9       300.0                      LifeWell
      9          10      2400.0             CityPlan Health
```

```
[15]: # David Thiriot

      c.execute('''SELECT * FROM Patient_health''')

      # Read the selection from the database into a pandas dataframe - looks nicer␣
       ↪and easier to work with
      colnames = c.description   # gather column names from a new query
      colnames_list = []
      for row in colnames:
          colnames_list.append(row[0])

      df = pd.DataFrame(c.fetchall(), columns=colnames_list)
      df.head(10)
```

```
[15]:    Patient_ID  Current_smoker  Condition_1  Condition_2  Condition_3  \
      0           1               1            0            1            0
      1           2               0            0            0            0
      2           3               0            0            1            0
      3           4               0            0            0            0
      4           5               0            0            0            0
      5           6               0            0            1            0
      6           7               0            0            1            1
      7           8               0            0            0            0
      8           9               0            0            1            0
      9          10               0            0            0            1

         Condition_4  Condition_5  Condition_6  Condition_7  Condition_8  \
      0            0            0            0            0            0
      1            0            0            0            0            0
      2            0            0            0            0            0
      3            0            0            0            0            0
      4            0            0            0            0            0
```

```
5            0            0            0            0            1
6            0            0            0            0            1
7            0            0            1            0            0
8            0            0            0            0            0
9            0            0            1            0            0

   Condition_9  Condition_10
0            0             0
1            0             0
2            0             0
3            0             0
4            0             0
5            0             0
6            0             0
7            0             0
8            0             0
9            0             0
```

[16]:
```
# David Thiriot

c.execute('''SELECT * FROM Patient_ID''')

# Read the selection from the database into a pandas dataframe - looks nicer␣
 ↪and easier to work with
colnames = c.description   # gather column names from a new query
colnames_list = []
for row in colnames:
    colnames_list.append(row[0])

df = pd.DataFrame(c.fetchall(), columns=colnames_list)
df.head(10)
```

[16]:
```
   Patient_ID    Firstname           Lastname         DOB  Age Biol_sex  \
0           1  Porvangchee                Yun  1972-03-20   51     Male
1           2        Jalen            Collins  1948-06-15   75   Female
2           3       Austin          Lightfoot  1964-09-10   59     Male
3           4       Jordan                Ali  1964-02-24   59     Male
4           5        Meron           Dzerekey  1990-03-25   33   Female
5           6      Badraan            al-Safi  1969-09-18   54     Male
6           7      Natanya           Hritsick  1975-05-18   48   Female
7           8   Alexandra            Watkins  1968-08-06   55   Female
8           9        Kiana  Johnson-Dickerson  2000-05-23   23   Female
9          10          Tin              Rader  1980-07-04   43     Male

                    Ethnicity
0       Asian or Pacific Islander
1            Black (not Hispanic)
```

```
2  American Indian or Native Alaskan
3                 Black (not Hispanic)
4                 Black (not Hispanic)
5               Middle-Eastern, Arabic
6            Asian or Pacific Islander
7                 White (not Hispanic)
8                 Black (not Hispanic)
9            Asian or Pacific Islander
```

[17]:
```
# David Thiriot

c.execute('''SELECT * FROM Patient_vitals''')

# Read the selection from the database into a pandas dataframe - looks nicer␣
↪and easier to work with
colnames = c.description   # gather column names from a new query
colnames_list = []
for row in colnames:
    colnames_list.append(row[0])

df = pd.DataFrame(c.fetchall(), columns=colnames_list)
df.head(10)
```

[17]:
```
   Patient_ID  Last_height  Last_weight  Last_heartrate  Last_systolic_BP  \
0           1        175.0        200.0            58.0             123.0
1           2        155.0        197.0            73.0             127.0
2           3        172.0        216.0            76.0             121.0
3           4        174.0        215.0            74.0             130.0
4           5        171.0        167.0            77.0             112.0
5           6        176.0        218.0            77.0             123.0
6           7        162.0        159.0            77.0             124.0
7           8        158.0        175.0            81.0             127.0
8           9        165.0        172.0            97.0             126.0
9          10        179.0        229.0            76.0             126.0

   Last_diastolic_BP
0               69.0
1               73.0
2               75.0
3               74.0
4               67.0
5               70.0
6               76.0
7               66.0
8               72.0
9               71.0
```

```
[ ]:
```

1.5 Demonstrate different SQL operations

```
[18]: # List Table Patient_ID by alphabetical order of last name
      # David Thiriot

      c.execute('''SELECT * FROM Patient_ID
                   ORDER BY Lastname ASC''')

      # Read the selection from the database into a pandas dataframe - looks nicer␣
       ↪and easier to work with
      colnames = c.description    # gather column names from a new query
      colnames_list = []
      for row in colnames:
          colnames_list.append(row[0])

      df = pd.DataFrame(c.fetchall(), columns=colnames_list)
      df.head(10)
```

```
[18]:    Patient_ID Firstname  Lastname         DOB  Age Biol_sex  \
      0         920     Anali    Abeyta  1948-03-01   75   Female
      1          97   Shannon  Ackerman  1966-05-01   57   Female
      2         609  La-Deige     Adams  1988-10-31   35   Female
      3         635     Kayla     Adams  1989-07-07   34   Female
      4         670     Tasha   Afalava  1970-03-08   53   Female
      5         353     Lucia   Aguilar  1979-03-31   44   Female
      6         630      Jose   Aguilar  1980-08-17   43     Male
      7         972     Sarah    Alarid  1997-02-09   26   Female
      8         216     Piper      Alex  1963-09-29   60   Female
      9           4    Jordan       Ali  1964-02-24   59     Male

                               Ethnicity
      0                         Hispanic
      1             White (not Hispanic)
      2             Black (not Hispanic)
      3  American Indian or Native Alaskan
      4          Asian or Pacific Islander
      5                         Hispanic
      6                         Hispanic
      7             White (not Hispanic)
      8  American Indian or Native Alaskan
      9             Black (not Hispanic)
```

```
[19]: # Add a new column to the Patient_ID table, a "boolean" (actually int) for␣
       ↪Current_patient (1=Yes a current patient, 0=Not a current patient)
```

```
# David Thiriot

# Error handling idea for the case of when the column already exists is from
# Nick Dandoulakis on 01 March 2010, accessed on 28 March 2023 at
# https://stackoverflow.com/questions/2354696/
 ↪alter-table-sqlite-how-to-check-if-a-column-exists-before-alter-the-table/
 ↪2354829#2354829

try:
    c.execute('''ALTER TABLE Patient_ID ADD COLUMN Current_patient int''')
except:
    pass # handle the error in case the column Current_patient already exists

c.execute('''UPDATE Patient_ID SET Current_patient = 1''')
c.execute('''SELECT * FROM Patient_ID
             ORDER BY Lastname ASC''')

# Read the selection from the database into a pandas dataframe - looks nicer␣
 ↪and easier to work with
colnames = c.description   # gather column names from a new query
colnames_list = []
for row in colnames:
    colnames_list.append(row[0])

df = pd.DataFrame(c.fetchall(), columns=colnames_list)

patient_ID_table = df    #Doing this adds the column Current_patient, which is␣
 ↪used in the AddPatients function.
df.head(10)
```

[19]:
```
   Patient_ID Firstname  Lastname         DOB  Age Biol_sex  \
0         920     Anali    Abeyta  1948-03-01   75   Female
1          97   Shannon  Ackerman  1966-05-01   57   Female
2         609  La-Deige     Adams  1988-10-31   35   Female
3         635     Kayla     Adams  1989-07-07   34   Female
4         670     Tasha   Afalava  1970-03-08   53   Female
5         353     Lucia   Aguilar  1979-03-31   44   Female
6         630      Jose   Aguilar  1980-08-17   43     Male
7         972     Sarah    Alarid  1997-02-09   26   Female
8         216     Piper      Alex  1963-09-29   60   Female
9           4    Jordan       Ali  1964-02-24   59     Male

                           Ethnicity  Current_patient
0                           Hispanic                1
1               White (not Hispanic)                1
2               Black (not Hispanic)                1
3  American Indian or Native Alaskan                1
```

```
4         Asian or Pacific Islander               1
5                          Hispanic               1
6                          Hispanic               1
7                White (not Hispanic)             1
8  American Indian or Native Alaskan              1
9                Black (not Hispanic)             1
```

[20]:
```
# Add 2 new patients to the practice
# Recording patient data in each of the correct tables
# View the new records in each table
# David Thiriot

# Find the maximum number of Patient_ID

# List all the column names of all the tables
print("Tables and column names in MedPracDB\n")
print("Doctors: ",list(doctors_table.columns))
print("Patient_contact: ",list(patient_contact_table.columns))
print("Patient_finance: ",list(patient_finance_table.columns))
print("Patient_doctors: ",list(patient_doctors_table.columns))
print("Patient_health: ",list(patient_health_table.columns))
print("Patient_ID: ",list(patient_ID_table.columns), " + Current_patient")
print("Patient_vitals: ",list(patient_vitals_table.columns))

c.execute('''SELECT MAX(Patient_ID) FROM Patient_ID''')
Highest_Patient_ID = c.fetchall()[0][0]  #This is how to get an integer value

# Two new people
new_people = pd.DataFrame({
    'Patient_ID': [Highest_Patient_ID + 1, Highest_Patient_ID + 2],      # int
    'Email': ['newpatient1@email.com', 'newpatient2@zipmail.net'],       # text
    'Street_address': ['123 Anywhere Place', '456 Someplace Ave'],       # text
    'City': ['New York', 'New York'],                                    # text
    'State': ['New York', 'New York'],                                   # text
    'Zip': [12345, 54321],                                               # int
    'Amount_due': [1111.00, 2222.00],                                    # real
    'Ins_co': ['OK Insurance', 'Allright Insurance'],                    # text
    'Doctor': ['Zhang', 'Zhang'],                                        # text
    'Current_smoker': [1,0],                                             # int
    'Condition_1': [1,0],                                                 # int
    'Condition_2': [1,0],                                                 # int
    'Condition_3': [1,0],                                                 # int
    'Condition_4': [1,0],                                                 # int
    'Condition_5': [1,0],                                                 # int
    'Condition_6': [1,0],                                                 # int
    'Condition_7': [1,0],                                                 # int
    'Condition_8': [1,0],                                                 # int
    'Condition_9': [1,0],                                                # int
    'Condition_10': [1,0],                                               # int
    'Firstname': ['Mary', 'Robert'],                                     # text
    'Lastname': ['AAAAA', 'AAAAB'],                                      # text␣
↪
    'DOB': ['1980-01-01', '1981-02-02'],                                 # text
    'Age': [43, 42],                                                     # int
    'Biol_sex': ['Female', 'Male'],                                      # text
    'Ethnicity': ['Black', 'White'],                                     # text
    'Current_patient': [1,1],                                            # int
    'Last_height': [163.0, 175.0],                                       # real
    'Last_weight': [165.0, 210.0],                                       # real
    'Last_heartrate': [80, 71],                                          # real
    'Last_systolic_BP': [120, 125],                                      # real
    'Last_diastolic_BP': [70, 72]                                        # real
})

new_people
```

```
Tables and column names in MedPracDB

Doctors:  ['Doctor_ID', 'Firstname', 'Lastname']
Patient_contact:  ['Patient_ID', 'Email', 'Street_address', 'City', 'State',
'Zip']
Patient_finance:  ['Patient_ID', 'Amount_due', 'Ins_co']
Patient_doctors:  ['Patient_ID', 'Doctor']
Patient_health:  ['Patient_ID', 'Current_smoker', 'Condition_1', 'Condition_2',
'Condition_3', 'Condition_4', 'Condition_5', 'Condition_6', 'Condition_7',
'Condition_8', 'Condition_9', 'Condition_10']
Patient_ID:  ['Patient_ID', 'Firstname', 'Lastname', 'DOB', 'Age', 'Biol_sex',
'Ethnicity', 'Current_patient']  + Current_patient
Patient_vitals:  ['Patient_ID', 'Last_height', 'Last_weight', 'Last_heartrate',
'Last_systolic_BP', 'Last_diastolic_BP']
```

[20]:

```
   Patient_ID                    Email     Street_address      City  \
0        1001    newpatient1@email.com  123 Anywhere Place  New York
1        1002  newpatient2@zipmail.net   456 Someplace Ave  New York

      State    Zip  Amount_due              Ins_co Doctor  Current_smoker  \
0  New York  12345      1111.0        OK Insurance  Zhang               1
1  New York  54321      2222.0  Allright Insurance  Zhang               0

   …         DOB  Age  Biol_sex  Ethnicity  Current_patient  Last_height  \
0  …  1980-01-01   43    Female      Black                1        163.0
1  …  1981-02-02   42      Male      White                1        175.0

   Last_weight  Last_heartrate  Last_systolic_BP  Last_diastolic_BP
```

```
0        165.0              80               120                 70
1        210.0              71               125                 72

[2 rows x 32 columns]
```

[21]:
```
# Function to add patients to the MedPracDB
# David Thiriot

def AddPatients(NewPatients):   # Takes as argument a single dataframe with all␣
 ↪information for one or more new patients

    df_contact = NewPatients[list(patient_contact_table.columns)]
    df_finance = NewPatients[list(patient_finance_table.columns)]
    df_doctors = NewPatients[list(patient_doctors_table.columns)]
    df_health = NewPatients[list(patient_health_table.columns)]
    df_ID = NewPatients[list(patient_ID_table.columns)]
    df_vitals = NewPatients[list(patient_vitals_table.columns)]

    df_contact.to_sql('Patient_contact', conn, if_exists='append', index=False)
    df_finance.to_sql('Patient_finance', conn, if_exists='append', index=False)
    df_doctors.to_sql('Patient_doctors', conn, if_exists='append', index=False)
    df_health.to_sql('Patient_health', conn, if_exists='append', index=False)
    df_ID.to_sql('Patient_ID', conn, if_exists='append', index=False)
    df_vitals.to_sql('Patient_vitals', conn, if_exists='append', index=False)

AddPatients(new_people)
```

[22]:
```
# See the new patients in the database
# David Thiriot

c.execute('''SELECT * FROM Patient_ID
             WHERE Current_patient = 1
             ORDER BY Lastname ASC''')

# Read the selection from the database into a pandas dataframe - looks nicer␣
 ↪and easier to work with
colnames = c.description   # gather column names from a new query
colnames_list = []
for row in colnames:
    colnames_list.append(row[0])

df = pd.DataFrame(c.fetchall(), columns=colnames_list)
print("Table = Patient_ID")
df.head(5)
```

```
Table = Patient_ID
```

[22]:
```
   Patient_ID Firstname  Lastname         DOB  Age Biol_sex  \
0        1001      Mary     AAAAA  1980-01-01   43   Female
1        1002    Robert     AAAAB  1981-02-02   42     Male
2         920     Anali    Abeyta  1948-03-01   75   Female
3          97   Shannon  Ackerman  1966-05-01   57   Female
4         609  La-Deige     Adams  1988-10-31   35   Female

                Ethnicity  Current_patient
0                   Black                1
1                   White                1
2                Hispanic                1
3  White (not Hispanic)                1
4  Black (not Hispanic)                1
```

[23]:
```
# Attempt to add records in violation of the PRIMARY KEY (using a non-unique␣
↪Patient_ID)
# Try to add 2 new patients as Patient_ID 1001 and 1002
# With Firstnames UNNACCEPTABLE
# What happens?
# David Thiriot

# Two new people
unnacceptable_records = pd.DataFrame({
    'Patient_ID': [1001, 1002],      # int   # These Patient_ID are already␣
↪used! Testing on purpose.
    'Email': ['newpatient1@email.com', 'newpatient2@zipmail.net'],       # text
    'Street_address': ['123 Anywhere Place', '456 Someplace Ave'],       # text
    'City': ['New York', 'New York'],                                    # text
    'State': ['New York', 'New York'],                                   # text
    'Zip': [12345, 54321],                                               # int
    'Amount_due': [1111.00, 2222.00],                                    # real
    'Ins_co': ['OK Insurance', 'Allright Insurance'],                    # text
    'Doctor': ['Zhang', 'Zhang'],                                        # text
    'Current_smoker': [1,0],                                             # int
    'Condition_1': [1,0],                                                 # int
    'Condition_2': [1,0],                                                 # int
    'Condition_3': [1,0],                                                 # int
    'Condition_4': [1,0],                                                 # int
    'Condition_5': [1,0],                                                 # int
    'Condition_6': [1,0],                                                 # int
    'Condition_7': [1,0],                                                 # int
    'Condition_8': [1,0],                                                 # int
    'Condition_9': [1,0],                                                 # int
    'Condition_10': [1,0],                                                # int
    'Firstname': ['UNNACCEPTABLE', 'UNNACCEPTABLE'],                     # text
    'Lastname': ['AAAAA', 'AAAAB'],                                      # text␣
↪
```

```
    'DOB': ['1980-01-01', '1981-02-02'],                                    # text
    'Age': [43, 42],                                                        # int
    'Biol_sex': ['Female', 'Male'],                                         # text
    'Ethnicity': ['Black', 'White'],                                        # text
    'Current_patient': [1,1],                                               # int
    'Last_height': [163.0, 175.0],                                          # real
    'Last_weight': [165.0, 210.0],                                          # real
    'Last_heartrate': [80, 71],                                             # real
    'Last_systolic_BP': [120, 125],                                         # real
    'Last_diastolic_BP': [70, 72]                                           # real
})

unnacceptable_records[['Firstname', 'Lastname', 'Patient_ID']]
```

[23]:
```
      Firstname Lastname  Patient_ID
0  UNNACCEPTABLE    AAAAA        1001
1  UNNACCEPTABLE    AAAAB        1002
```

Note: The error from the code block below is left intentionally! It shows that the PRIMARY KEY constraint is working properly.

[24]:
```
# Try to enter the above unnacceptable Patient_ID into the database
# David Thiriot

AddPatients(unnacceptable_records)

# I'm leaving the IntegrityError: column Patient_ID is not unique
# below to demonstrate that the Primary Key constraint works
```

```
---------------------------------------------------------------------------
IntegrityError                            Traceback (most recent call last)
Cell In[24], line 4
      1 # Try to enter the above unnacceptable Patient_ID into the database
      2 # David Thiriot
----> 4 AddPatients(unnacceptable_records)
      6 # I'm leaving the IntegrityError: column Patient_ID is not unique
      7 # below to demonstrate that the Primary Key constraint works

Cell In[21], line 13, in AddPatients(NewPatients)
     10 df_ID = NewPatients[list(patient_ID_table.columns)]
     11 df_vitals = NewPatients[list(patient_vitals_table.columns)]
---> 13␣
  ↪df_contact.to_sql('Patient_contact', conn, if_exists='append', index=False)
     14 df_finance.to_sql('Patient_finance', conn, if_exists='append',␣
  ↪index=False)
     15 df_doctors.to_sql('Patient_doctors', conn, if_exists='append',␣
  ↪index=False)
```

```
File ~/anaconda3/envs/django/lib/python3.10/site-packages/pandas/core/generic.py:
 ↪2987, in NDFrame.to_sql(self, name, con, schema, if_exists, index,␣
 ↪index_label, chunksize, dtype, method)
   2830 """
   2831 Write records stored in a DataFrame to a SQL database.
   2832
   (…)
   2983 [(1,), (None,), (2,)]
   2984 """  # noqa:E501
   2985 from pandas.io import sql
-> 2987 return sql.to_sql(
   2988     self,
   2989     name,
   2990     con,
   2991     schema=schema,
   2992     if_exists=if_exists,
   2993     index=index,
   2994     index_label=index_label,
   2995     chunksize=chunksize,
   2996     dtype=dtype,
   2997     method=method,
   2998 )

File ~/anaconda3/envs/django/lib/python3.10/site-packages/pandas/io/sql.py:695,
 ↪in to_sql(frame, name, con, schema, if_exists, index, index_label, chunksize,
 ↪dtype, method, engine, **engine_kwargs)
    690 elif not isinstance(frame, DataFrame):
    691     raise NotImplementedError(
    692         "'frame' argument should be either a Series or a DataFrame"
    693     )
--> 695 return pandas_sql.to_sql(
    696     frame,
    697     name,
    698     if_exists=if_exists,
    699     index=index,
    700     index_label=index_label,
    701     schema=schema,
    702     chunksize=chunksize,
    703     dtype=dtype,
    704     method=method,
    705     engine=engine,
    706     **engine_kwargs,
    707 )

File ~/anaconda3/envs/django/lib/python3.10/site-packages/pandas/io/sql.py:2188 ␣
 ↪in SQLiteDatabase.to_sql(self, frame, name, if_exists, index, index_label,␣
 ↪schema, chunksize, dtype, method, **kwargs)
```

```
   2178 table = SQLiteTable(
   2179     name,
   2180     self,
   (…)
   2185     dtype=dtype,
   2186 )
   2187 table.create()
-> 2188 return table.insert(chunksize, method)

File ~/anaconda3/envs/django/lib/python3.10/site-packages/pandas/io/sql.py:946,␣
 ↪in SQLTable.insert(self, chunksize, method)
    943     break
    945 chunk_iter = zip(*(arr[start_i:end_i] for arr in data_list))
--> 946 num_inserted = exec_insert(conn, keys, chunk_iter)
    947 # GH 46891
    948 if is_integer(num_inserted):

File ~/anaconda3/envs/django/lib/python3.10/site-packages/pandas/io/sql.py:1894 ␣
 ↪in SQLiteTable._execute_insert(self, conn, keys, data_iter)
   1892 def _execute_insert(self, conn, keys, data_iter) -> int:
   1893     data_list = list(data_iter)
-> 1894     conn.executemany(self.insert_statement(num_rows=1), data_list)
   1895     return conn.rowcount

IntegrityError: UNIQUE constraint failed: Patient_contact.Patient_ID
```

[25]:
```
# Demonstrate a join that draws from several of the Tables in the database
# David Thiriot

# Dr. Zhang has dropped the insurance plan "New Day Medical" as they will no␣
 ↪longer cover females
# who have Condition_1
# Find all the females who need to be alerted, and retrieve their Email␣
 ↪addresses
# Criteria is Zhang + New Day Medical + Female + Condition_1

c.execute('''SELECT Patient_ID.Firstname AS Patient_first_name,
                    Patient_ID.Lastname AS Patient_last_name,
                    Patient_contact.Email AS Patient_Email,
                    Patient_ID.Patient_ID,
                    Patient_ID.Biol_sex AS 'Male/Female',
                    Patient_doctors.Doctor,
                    Patient_health.Condition_1 AS Has_condition1,
                    Patient_finance.Ins_co AS Insurance_company
             FROM Patient_ID
```

```
            JOIN Patient_contact on Patient_contact.Patient_ID = Patient_ID.
 ↪Patient_ID
            JOIN Patient_finance on Patient_finance.Patient_ID = Patient_ID.
 ↪Patient_ID
            JOIN Patient_doctors on Patient_doctors.Patient_ID = Patient_ID.
 ↪Patient_ID
            JOIN Patient_health on Patient_health.Patient_ID = Patient_ID.
 ↪Patient_ID
            WHERE Patient_doctors.Doctor = 'Zhang' AND
                  Patient_finance.Ins_co = 'New Day Medical' AND
                  Patient_ID.Biol_sex = "Female" AND
                  Patient_health.Condition_1 = 1 AND
                  Patient_ID.Current_patient = 1
            ORDER BY Patient_ID.Lastname ASC''')

# Read the selection from the database into a pandas dataframe - looks nicer␣
 ↪and easier to work with
colnames = c.description   # gather column names from a new query
colnames_list = []
for row in colnames:
    colnames_list.append(row[0])

df = pd.DataFrame(c.fetchall(), columns=colnames_list)
print("Inform these Female patients of Dr. Zhang that New Day Medical no longer␣
 ↪covers Condition 1.")
df
```

```
Inform these Female patients of Dr. Zhang that New Day Medical no longer covers
Condition 1.
```

[25]:
```
  Patient_first_name Patient_last_name         Patient_Email  Patient_ID  \
0              Maria          Gallegos     GalMar5@yahoo.com         543
1             Brenna          Guerrero   GueBre7@verizon.net         342
2             Sierra            Little     LitSie4@gmail.com         695
3           Brittany      Merriweather     MerBri2@yahoo.com         831
4           Michelle            Nguyen  NguMic1@fastmail.net         919
5             Hasana           el-Wali     el-Has6@yahoo.com         922

  Male/Female Doctor  Has_condition1 Insurance_company
0      Female  Zhang               1   New Day Medical
1      Female  Zhang               1   New Day Medical
2      Female  Zhang               1   New Day Medical
3      Female  Zhang               1   New Day Medical
4      Female  Zhang               1   New Day Medical
5      Female  Zhang               1   New Day Medical
```

```
[26]: # Demonstrate how to "Delete" a patient by changing the Patient_ID.
       ↪Current_patient value to 0 (zero).
      # David Thiriot
      # Patients don't really get deleted, they get flagged as not a current patient
      # Because the practice needs to keep their records for 30 years after they have␣
       ↪left, by policy

      # First, show all the patients who have the rare condition, condition 10

      c.execute('''SELECT Patient_ID.Firstname AS Patient_first_name,
                          Patient_ID.Lastname AS Patient_last_name,
                          Patient_contact.Email AS Patient_Email,
                          Patient_ID.Patient_ID,
                          Patient_doctors.Doctor,
                          Patient_health.Condition_10 AS Has_condition10
                   FROM Patient_ID
                   JOIN Patient_contact on Patient_contact.Patient_ID = Patient_ID.
       ↪Patient_ID
                   JOIN Patient_doctors on Patient_doctors.Patient_ID = Patient_ID.
       ↪Patient_ID
                   JOIN Patient_health on Patient_health.Patient_ID = Patient_ID.
       ↪Patient_ID
                   WHERE Patient_health.Condition_10 = 1 AND
                         Patient_ID.Current_patient = 1
                   ORDER BY Patient_ID.Lastname ASC''')

      # Read the selection from the database into a pandas dataframe - looks nicer␣
       ↪and easier to work with
      colnames = c.description   # gather column names from a new query
      colnames_list = []
      for row in colnames:
          colnames_list.append(row[0])

      df = pd.DataFrame(c.fetchall(), columns=colnames_list)
      print("Table = Patient_ID")
      df
```

```
Table = Patient_ID
```

```
[26]:   Patient_first_name Patient_last_name          Patient_Email  Patient_ID  \
      0               Mary             AAAAA  newpatient1@email.com        1001
      1              Lucia           Aguilar    AguLuc3@verizon.net         353
      2            Darrell            Currin       CurDar2@aol.com         197
      3            Matthew           Hopkins    HopMat6@verizon.net         819
      4              Jesse          Luttrell       LutJes2@aol.com         292
      5           Diondre          Mcclendon       MccDio4@aol.com         973
      6               Kyla             Platt      PlaKyl1@gmail.com         241
```

```
7          Dejaynay      Pritchard Jr      PriDej5@yahoo.com        205
8           Michael  Rasberry-Jenkins      RasMic5@gmail.com        125
9           Dehrien           Strauss      StrDeh2@gmail.com        910
10           Salena           Ventura      VenSal6@yahoo.com        163
11           Jeremy                Yi    YiJer7@fastmail.net        794
12         Abdullah           al-Akel   al-Abd9@fastmail.net        970
13         Shaddaad           al-Azer   al-Sha5@fastmail.net        648
14           Haibaa       al-Ebrahimi      al-Hai7@gmail.com        135
15    Abdur Rahmaan          el-Salaam      el-Abd8@gmail.com         74

       Doctor  Has_condition10
0       Zhang                1
1       Smith                1
2      Harris                1
3    Williams                1
4     Jackson                1
5     Tataryn                1
6      Harris                1
7   Rodriguez                1
8       Zhang                1
9       Petit                1
10  Rodriguez                1
11    Tataryn                1
12     Harris                1
13      Petit                1
14     Harris                1
15      Smith                1
```

[27]:
```
# Now we will update the Patient_health.Condition10 value of our new patient,␣
↪Mary AAAAA, to zero
# Effectively 'deleting' her without removing her from the database,
# and run the above query for Condition_10 patients again
# David Thiriot

c.execute('''UPDATE Patient_ID
             SET Current_patient = 0
             WHERE Lastname = 'AAAAA'
''')

c.execute('''SELECT Patient_ID.Firstname AS Patient_first_name,
                    Patient_ID.Lastname AS Patient_last_name,
                    Patient_contact.Email AS Patient_Email,
                    Patient_ID.Patient_ID,
                    Patient_doctors.Doctor,
                    Patient_health.Condition_10 AS Has_condition10
             FROM Patient_ID
```

```
            JOIN Patient_contact on Patient_contact.Patient_ID = Patient_ID.
↪Patient_ID
            JOIN Patient_doctors on Patient_doctors.Patient_ID = Patient_ID.
↪Patient_ID
            JOIN Patient_health on Patient_health.Patient_ID = Patient_ID.
↪Patient_ID
            WHERE Patient_health.Condition_10 = 1 AND
                  Patient_ID.Current_patient = 1
            ORDER BY Patient_ID.Lastname ASC''')

# Read the selection from the database into a pandas dataframe - looks nicer␣
↪and easier to work with
colnames = c.description   # gather column names from a new query
colnames_list = []
for row in colnames:
    colnames_list.append(row[0])

df = pd.DataFrame(c.fetchall(), columns=colnames_list)
print("Table = Patient_ID")
df

# Notice how patient Mary AAAAA is removed from the results below.
```

```
Table = Patient_ID
```

[27]:

```
   Patient_first_name Patient_last_name         Patient_Email  Patient_ID  \
0               Lucia           Aguilar   AguLuc3@verizon.net         353
1             Darrell            Currin       CurDar2@aol.com         197
2             Matthew           Hopkins   HopMat6@verizon.net         819
3               Jesse          Luttrell       LutJes2@aol.com         292
4             Diondre         Mcclendon       MccDio4@aol.com         973
5                Kyla             Platt     PlaKyl1@gmail.com         241
6            Dejaynay      Pritchard Jr     PriDej5@yahoo.com         205
7             Michael  Rasberry-Jenkins     RasMic5@gmail.com         125
8             Dehrien           Strauss     StrDeh2@gmail.com         910
9              Salena           Ventura     VenSal6@yahoo.com         163
10             Jeremy                Yi   YiJer7@fastmail.net         794
11           Abdullah           al-Akel  al-Abd9@fastmail.net         970
12           Shaddaad           al-Azer  al-Sha5@fastmail.net         648
13             Haibaa       al-Ebrahimi     al-Hai7@gmail.com         135
14      Abdur Rahmaan          el-Salaam     el-Abd8@gmail.com          74

      Doctor  Has_condition10
0      Smith                1
1     Harris                1
2   Williams                1
3    Jackson                1
```

```
4     Tataryn                 1
5      Harris                 1
6   Rodriguez                 1
7       Zhang                 1
8       Petit                 1
9   Rodriguez                 1
10    Tataryn                 1
11     Harris                 1
12      Petit                 1
13     Harris                 1
14      Smith                 1
```

[28]:
```
# A query example with graphs
# Where does Patient_ID = 100 fit in the distribution of height of all patients?
# David Thiriot

c.execute('''SELECT Patient_ID.Patient_ID AS Patient_ID,
                    Firstname AS Patient_first_name,
                    Lastname AS Patient_last_name,
                    Biol_sex AS 'Male/Female',
                    Last_height AS Height
             FROM Patient_ID
             JOIN Patient_vitals on Patient_ID.Patient_ID = Patient_vitals.
 ↪Patient_ID
             WHERE Patient_ID.Patient_ID = 100;
''')

colnames = c.description   # gather column names from a new query
colnames_list = []
for row in colnames:
    colnames_list.append(row[0])

df = pd.DataFrame(c.fetchall(), columns=colnames_list)
print("Table = Patient 100, height")
p100 = df
p100_height = p100.iloc[0,4]
#print(p100_height)
p100
```

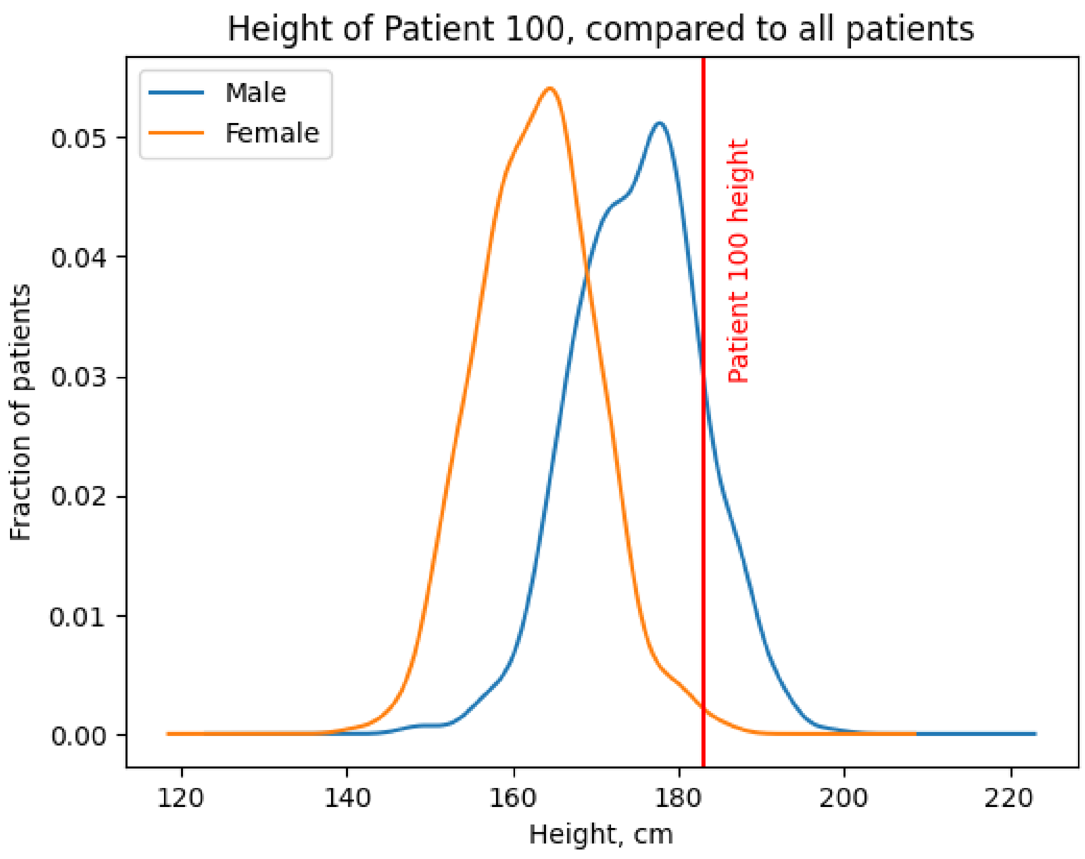
```
Table = Patient 100, height
```

[28]:
```
   Patient_ID Patient_first_name Patient_last_name Male/Female  Height
0         100              Wajdi           al-Qasim        Male   183.0
```

[29]:
```
# Continue -- Where does Patient_ID = 100 fit in the distribution of height of␣
 ↪all patients?
# David Thiriot
# Get the heights for Males into a dataframe

c.execute('''SELECT Last_height AS Height, Biol_sex AS 'Male/Female'
             FROM Patient_vitals
             JOIN Patient_ID on Patient_vitals.Patient_ID = Patient_ID.
 ↪Patient_ID
             WHERE Biol_sex = 'Male'
''')

# Read the selection from the database into a pandas dataframe - looks nicer␣
 ↪and easier to work with
colnames = c.description   # gather column names from a new query
colnames_list = []
for row in colnames:
    colnames_list.append(row[0])

df = pd.DataFrame(c.fetchall(), columns=colnames_list)
#print("Table = Heights for male patients")
male_heights = df

# Get the heights for Females into a dataframe

c.execute('''SELECT Last_height AS Height, Biol_sex AS 'Male/Female'
             FROM Patient_vitals
             JOIN Patient_ID on Patient_vitals.Patient_ID = Patient_ID.
 ↪Patient_ID
             WHERE Biol_sex = 'Female'
''')

# Read the selection from the database into a pandas dataframe - looks nicer␣
 ↪and easier to work with
colnames = c.description   # gather column names from a new query
colnames_list = []
for row in colnames:
    colnames_list.append(row[0])

df = pd.DataFrame(c.fetchall(), columns=colnames_list)
#print("Table = Heights for female patients")
female_heights = df
```

[30]:
```
# Continue -- Where does Patient_ID = 100 fit in the distribution of height of␣
 ↪all patients?
# David Thiriot

import matplotlib.pyplot as plt
```

```
male_heights['Height'].plot(kind='kde', label='Male')
female_heights['Height'].plot(kind='kde', label='Female')
plt.title("Height of Patient 100, compared to all patients")
plt.axvline(x=p100_height, color='r')
plt.text(186, 0.03, "Patient 100 height", rotation=90, color='r')
plt.legend(loc="upper left")
plt.xlabel("Height, cm")
plt.ylabel("Fraction of patients")
plt.show()
```



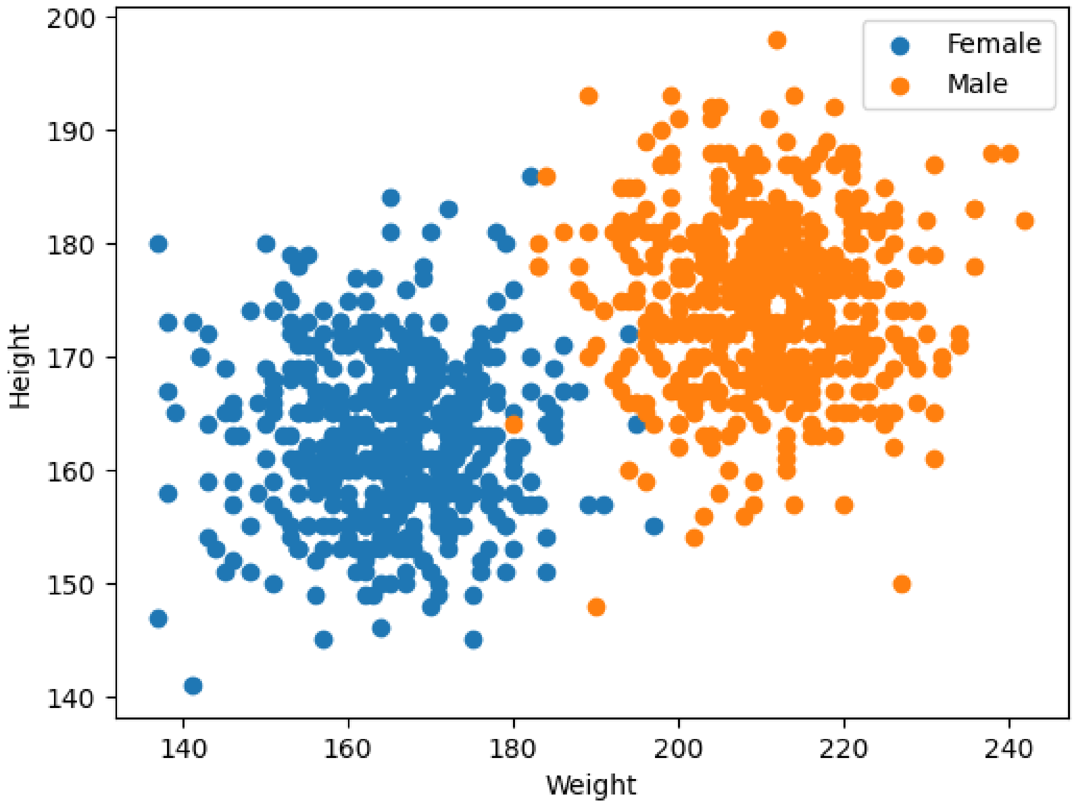
[31]:
```
# David Thiriot

c.execute('''SELECT * FROM sqlite_master;
''')

colnames = c.description    # gather column names from a new query
colnames_list = []
for row in colnames:
    colnames_list.append(row[0])
```

```
df = pd.DataFrame(c.fetchall(), columns=colnames_list)
#df
```

[32]:
```
# Alex Bordanca
# Calculate BMI plot for Male/female
import matplotlib.pyplot as plt

c.execute('''
select Last_height, Last_weight, pid.Biol_sex
from Patient_vitals
join Patient_ID as pid on Patient_vitals.Patient_ID = pid.Patient_ID;
''')

colnames = c.description   # gather column names from a new query
colnames_list = []
for row in colnames:
    colnames_list.append(row[0])

df = pd.DataFrame(c.fetchall(), columns=colnames_list)
df
```

[32]:
```
      Last_height  Last_weight Biol_sex
0           175.0        200.0     Male
1           155.0        197.0   Female
2           172.0        216.0     Male
3           174.0        215.0     Male
4           171.0        167.0   Female
…             …          …      …
997         172.0        206.0     Male
998         170.0        205.0     Male
999         160.0        194.0     Male
1000        163.0        165.0   Female
1001        175.0        210.0     Male

[1002 rows x 3 columns]
```

[33]:
```
#continued
groups = df.groupby('Biol_sex')

# plot the data for each group
fig, ax = plt.subplots()
for name, group in groups:
    ax.scatter(group.Last_weight, group.Last_height, label=name)

# add labels and legend
ax.set_xlabel('Weight')
ax.set_ylabel('Height')
ax.legend()

# show the plot
plt.show()
```



[34]:
```
#Alex Bordanca
#Create a list of correlations between other vitals and a dummy BMI variable
c.execute('''
select * from Patient_vitals;
''')

colnames = c.description    # gather column names from a new query
colnames_list = []
for row in colnames:
    colnames_list.append(row[0])

df = pd.DataFrame(c.fetchall(), columns=colnames_list)
df
```

[34]:
```
   Patient_ID  Last_height  Last_weight  Last_heartrate  Last_systolic_BP  \
0           1        175.0        200.0            58.0             123.0
```

```
1               2       155.0       197.0           73.0            127.0
2               3       172.0       216.0           76.0            121.0
3               4       174.0       215.0           74.0            130.0
4               5       171.0       167.0           77.0            112.0
…          …          …          …             …               …
997           998       172.0       206.0           64.0            127.0
998           999       170.0       205.0           69.0            128.0
999          1000       160.0       194.0           71.0            125.0
1000         1001       163.0       165.0           80.0            120.0
1001         1002       175.0       210.0           71.0            125.0

      Last_diastolic_BP
0                  69.0
1                  73.0
2                  75.0
3                  74.0
4                  67.0
…                 …
997                76.0
998                79.0
999                67.0
1000               70.0
1001               72.0

[1002 rows x 6 columns]
```

[35]:
```
#Continued
df['BMI_dummy'] = df['Last_weight']/(df['Last_height'])
```

[36]:
```
#continued
corr_matrix = df[df.columns[1:]].corr()['BMI_dummy'][:-1]
corr_matrix
```

[36]:
```
Last_height          0.180933
Last_weight          0.906541
Last_heartrate      -0.399995
Last_systolic_BP     0.225544
Last_diastolic_BP    0.149535
Name: BMI_dummy, dtype: float64
```

[37]:
```
c.execute('''
select Current_smoker, pf.Amount_due
from Patient_health
join Patient_finance as pf
on Patient_health.Patient_ID = pf.Patient_ID;
''')
```

```
colnames = c.description    # gather column names from a new query
colnames_list = []
for row in colnames:
    colnames_list.append(row[0])

df = pd.DataFrame(c.fetchall(), columns=colnames_list)
df
corr_matrix = df.corr()['Current_smoker'][1:]
corr_matrix
```

[37]:
```
Amount_due   -0.061749
Name: Current_smoker, dtype: float64
```

[38]:
```
# At the end of the work, close the database connection.
conn.close()
```

[ ]:
```
```

[ ]:
```
```