Reference Laboratory Project - Section 2: Data Creation

Created by Ryan Breen

Typically, data is usually produced by devices or sometimes even entered by human beings and derived by uploading, migrating, or connecting to a data source. Due to the nature of healthcare industry, the data used in the **Reference Laboratory Project** is created using `Python`. The data is converted from dataframes to .csv files that will be then uploaded into a `PostgreSQL` database where at that point the data is imagined to be sitauted as it were in the actual business.

The main premise of data creation listed in this section is to create tables with columns as attributes seen in the entity relation diagram of the **Reference Laboratory Project**. The data tables will be further normalized and 'cleaned' in the 'Logical Schema' section (next section). For now pertinent information is obtained as it were directly from the memory locations it were obtained. To make this information more human-readable' normalization, business logic, and the use of a visual/information dashboard is created to bem read by employees and customers alike corresponding to day-to-day functions encounter between the interaction of the reference laboratory and individuals.

All columns used in the tables below are created using random array creation from `Pandas`, `NumPy`, and `Random` Python modules. They have **all been created fictitiously** ,but have been *synthesized* to emulate real world data found in a reference laboratory project based on my prior experience as a technologist working in such a laboratory.

Further tables are created by the *Database Definition Language* using `SQL` by linking *foreign keys*, autoincrementing primary keys for these newly created tables, and using random functions in `PostgreSQL` that emulate devices such as analyzers in the lab to fill in data in these newly created tables.

In [77]:
```
#pip install varname
```

Import Libraries:

In [1]:
```
import numpy as np
from numpy import random
import pandas as pd
import scipy
import random
from varname import nameof
import os
```

In [2]:
```
path = os.getcwd()
csv_=path+'\\csv\\'
data_=path+'\\data\\'
sql_=path+'\\sql\\'
```

The following three `Python` functions below create `SQL` scripts by utilizing the use of file creation and string formatting. All three functions are `home-brewed` functions that are used as time savers and all scripts should be checked manually for accruacy.

The following function creates the SQL script that creates the insert statements:

In [3]:
```
def INSERT(csv, sql=sql_+"Lab_Project_DDL_INSERT.txt"):
    ''' 1. imports csv dataframe as .csv
        2. converts each dataframe row to formatted string
        3. inserts formatted string as one line of a .txt (uft-8) file
        4. .txt(utf-8) file used as DDL INSERT statement '''

    import pandas as pd
    df = pd.read_csv(csv_+ csv + ".csv", header=None)  # create df
    file = open(sql, "a")
    temp1 = []  # temp list to store row values
    temp2 = []  # temp list to store entire formatted row
    for cols in range(len(df)):
        for rows in df:
            if rows < (len(df.columns) - 1):  # all instances except last column are formatted as such
                temp1.append(f"'{df.iat[cols, rows]}',")
            if rows == (len(df.columns) - 1):  # when last column is reached, finish string formatting
                start = f"INSERT INTO {csv}\nVALUES("
                total = start + " ".join(temp1) + f"'{str(df.iat[cols, rows])}');\n"
                temp1.clear()
                temp2.append(total)
                del start
                del total
    for t in temp2:  # place formatted string = df row as one 'line' in the file
        print(t, file=file)
```

The following function creates the SQL script that creates all tables:

In [4]:
```
def TABLES (name, sql=sql_+"Lab_Project_DDL_TABLES.txt"):
    import pandas as pd
    title = str(name)
    df = eval(name)
    from varname import nameof
    name = nameof(df)
    table = f"CREATE TABLE {title}("
    cols = df.columns
    cols = cols.tolist()
    dts = [dt for dt in df.dtypes.map(lambda x : x.name)]
    typ = []
    for dt in dts:
        if dt=='int32':
            typ.append('integer')
```

```
        if dt=='float64':
            typ.append('numeric')
        if dt == 'object':
            typ.append('varchar(50)')
    typ[0] = 'serial'
    last=') ;'
    file = open(sql, "a")
    print(table, file=file)
    for c,t in zip(cols,typ):
        print(f'\t{c}\t\t\t{t},', file=file)
    print(f'PRIMARY KEY({cols[0]})', file=file)
    print(last, file=file)
    file.close()
```

The following function creates the SQL script that creates all foreign keys:

In [5]:

```
def FK (name, sql=sql_+'\Lab_Project_DDL_FK.txt'):
    import pandas as pd
    child_table = 'child_'
    parent_table =  str(name)
    constraint_name = child_table+parent_table+'_fk'
    df = eval(name)
    cols = df.columns
    parent_column =  cols.tolist()[0]
    fk_column = parent_column
    l1=f'ALTER TABLE {child_table}\n'
    l2=f'ADD CONSTRAINT {constraint_name}\n'
    l3=f'FOREIGN KEY ({fk_column})\n'
    l4=f'REFERENCES {parent_table} ({parent_column});'
    l5=''
    file = open(sql, 'a')
    output=l1+l2+l3+l4+l5
    print(output, file=file)
    file.close
```

Below in each section are each *raw* Table with associated Columns:

Below the following dataframe is used in multiple tables. The apostrophes in each column must be remove not to conflict with the SQL script.

In [3]:

```
address = pd.read_csv(data_+'cities.csv')
address.City = address.City.str.replace("'","^")
```

Address (n=50100)

50000 customers and 100 employees

In [9]:

```
# Let one column contain the city, state, and gps coordinate

address_id = np.arange(10000000,10050100,1)

nums = [i for i in np.random.randint(1,9999,50100)] ### EXCLUDE ###
plants = pd.read_csv(data_+'plants.csv', header=None).loc[:,0].to_list() ### EXCLUDE ###
plants = random.choices(plants, k = 50100)
suffices = ['Street', 'Road', 'Way', 'Avenue', 'Lane'] * 10020 ### EXCLUDE ###
street = [f'{i} {j.title()} {k}' for i,j,k in zip(nums, plants, suffices)]

# NOTE - # address of Employees for 5 lab locations = first five cities

city1 = address.City.to_list() * 50
city2 = address.City.head().to_list() * 20 # Employees
city =  city1 + city2


state1 = address.State.to_list() * 50
state2 = address.State.head().to_list() * 20 # Employees
state = state1 + state2

lat1 = address.lat.to_list() * 50
lat2 = address.lat.head().to_list() * 20 # Employees
lat =  lat1 + lat2

lon1 = address.lon.to_list() * 50
lon2 = address.lon.head().to_list() * 20 # Employees
lon =  lon1 + lon2

addresses = list(zip(address_id, street, city, state, lat, lon))
```

In [11]:

```
### CREATE DATAFRAME
Addresses = pd.DataFrame(addresses, columns=['address_id','street', 'city', 'state', 'lat',
                                             'lon'])
Addresses.to_csv(csv_+'Addresses.csv', header = None, index = False)
Addresses.tail()
```

Out[11]:

	address_id	street	city	state	lat	lon
50095	10050095	25 Poison Flower Street	Marysville	Washington	48.051764	-122.177082
50096	10050096	586 Strawberry Road	Perris	California	33.782519	-117.228648
50097	10050097	1069 Silver Leaf Maple Way	Cleveland	Ohio	41.499320	-81.694361
50098	10050098	1356 Scotch Cap Avenue	Worcester	Massachusetts	42.262593	-71.802293

	address_id	street	city	state	lat	lon
50099	10050099	3805 Yellow Rocket Lane	Columbia	South Carolina	34.000710	-81.034814

In [86]:
```
INSERT('Addresses')
TABLES('Addresses')
FK('Addresses')
```

In [87]:
```
len(Addresses)
```

Out[87]:
```
50100
```

Customers (n=50000)

In [4]:
```
#People - 50,000 people throughout United States
customer_id = np.arange(10000000,10050100,1)[:50000]

address_id = np.arange(10000000,10050100,1)

# Import tree names to represent street names
# DOB
month = [i for i in np.random.randint(1,12,50000)] ### EXCLUDE ###
day = [i for i in np.random.randint(1,28,50000)] ### EXCLUDE ###
year = [i for i in np.random.randint(1920,2020, 50000)] ### EXCLUDE ###

dob = [f'{i}/{j}/{k}' for i,j,k in zip (month,day, year)]
gender = random.choices(['Female','Male'], k = 50000)
race = random.choices(['African American','Caucasian', 'Hispanic', 'Asian'],
                      weights=[8000,30000,8000,4000],
                      k = 50000)

spanish = random.choices(['yes','no'], weights=[5000,45000], k = 50000)

english  = random.choices(['yes','no'], weights=[500,45500], k = 50000)

married = random.choices(['yes','no'], weights=[45000, 5000],k = 50000)

insurance = random.choices(['private','public','none'],
                           weights=[30000,10000,10000], k = 50000)

customers = list(zip(customer_id, address_id, dob, gender, race,
                     spanish, english, married, insurance))
```

In [5]:

```
### CREATE DATAFRAME
Customers = pd.DataFrame(customers, columns=['customer_id','address_id',
                                             'dob', 'gender', 'race',
                                            'spanish_lang','english_only',
                                            'marital_status', 'insurance'])
Customers.to_csv(csv_+'Customers.csv', header = None, index = False)
Customers.head()
```

Out[5]:

	customer_id	address_id	dob	gender	race	spanish_lang	english_only	marital_status	insurance
0	10000000	10000000	1/19/1974	Female	Caucasian	no	no	yes	public
1	10000001	10000001	4/16/1980	Female	Caucasian	no	no	yes	private
2	10000002	10000002	5/3/1962	Male	Caucasian	no	yes	yes	none
3	10000003	10000003	2/22/1931	Female	Caucasian	no	no	yes	private
4	10000004	10000004	4/15/2008	Male	Hispanic	no	no	yes	private

In [90]:
```
INSERT('Customers')
TABLES('Customers')
FK('Customers')
```

In [91]:
```
len(Customers)
```

Out[91]:
```
50000
```

In [24]:
```
weights=[1000,6000,5000,8000,9000,
         8000,7000,4000,1000,1000]
sum(weights)
```

Out[24]:
```
50000
```

Customer_Surveys (n=25000)

In [25]:
```
#Customer Survey - Monthly survey of customers scale 1-10
survey_id = np.arange(10000000,10025000,1)

customer_id = np.arange(10000000,10050000,1)

date = pd.date_range('2021-01-01','2021-12-31').strftime("%m/%d/%Y").tolist()
dates = random.choices(date, k = 25000)
```

```
courteous = random.choices(list(range(1,11)),
                           weights=[1000,2000,3000,8000,
                                    8000,8000,6000,8000,3000,3000],
                           k= 50000)

schedule = random.choices(list(range(1,11)),
                           weights=[1000,7000,4000,8000,5000,
                                    8000,3000,8000,5000,1000]
                          ,k= 50000)

costs = random.choices(list(range(1,11)),
                           weights=[1000,1000,2000,5000,8000,9000,
                                    8000,5000,6000,5000]
                         ,k= 50000)

delivery = random.choices(list(range(1,11)),
                           weights=[1000,2000,5000,8000,9000,
                                    8000,5000,6000,5000,1000]
                          ,k= 50000)

overall = random.choices(list(range(1,11)),
                           weights=[1000,6000,5000,8000,9000,
                                    8000,7000,4000,1000,1000]
                         ,k= 50000)

customer_surveys = list(zip(survey_id, customer_id, dates, courteous, schedule,
                           costs, delivery, overall))
```

In [26]:

```
### CREATE DATAFRAME
Customer_Surveys = pd.DataFrame(customer_surveys, columns=['survey_id', 'customer_id',
                                                           'dates', 'courteous', 'schedule',
                                                  'costs', 'delivery', 'overall'])
Customer_Surveys.to_csv(csv_+'Customer_Surveys.csv', header = None, index = False)
Customer_Surveys.head()
```

Out[26]:

	survey_id	customer_id	dates	courteous	schedule	costs	delivery	overall
0	10000000	10000000	10/25/2021	7	4	7	6	6
1	10000001	10000001	05/20/2021	8	3	9	5	7
2	10000002	10000002	03/04/2021	10	8	5	7	2
3	10000003	10000003	10/20/2021	1	5	8	9	7
4	10000004	10000004	08/16/2021	6	8	9	3	8

In [94]:

```
INSERT('Customer_Surveys')
```

```
TABLES('Customer_Surveys')
FK('Customer_Surveys')
```

In [95]:
```
len(Customer_Surveys)
```

Out[95]:
```
25000
```

Laboratories (n=5)

In [96]:
```
#Labs -  5 labs locations throughout United States
lab_id= np.arange(100,105,1)

# Lab names
names = ['Main Lab', 'Central Lab', 'North Lab', 'Downtown Lab', 'East Lab']

# Let one column contain the city, state, and gps coordinate
nums = [i for i in np.random.randint(1,9999,50000)][:5] ### EXCLUDE ###
plants = pd.read_csv(data_+'plants.csv', header=None).loc[:5,0].to_list() ### EXCLUDE ###
suffices = ['Street', 'Road', 'Way', 'Avenue', 'Lane'] ### EXCLUDE ###
street = [f'{i} {j.title()} {k}' for i,j,k in zip(nums, plants, suffices)]

address = pd.read_csv(data_+'cities.csv')### EXCLUDE ###
city = address.City.to_list()[:5]

state = address.State.to_list()[:5]

lat = address.lat.to_list()[:5]

lon = address.lon.to_list()[:5]

laboratories = list(zip(lab_id, names, street, city, state, lat, lon))
```

In [97]:
```
### CREATE DATAFRAME
Laboratories = pd.DataFrame(laboratories, columns=['lab_id', 'names', 'street', 'city',' state',
                                                   'lat', 'lon'])
Laboratories.to_csv(csv_+'Laboratories.csv', header = None, index = False)
Laboratories.head()
```

Out[97]:

	lab_id	names	street	city	state	lat	lon
0	100	Main Lab	5181 Alder Street	Marysville	Washington	48.051764	-122.177082
1	101	Central Lab	8286 African Rice Road	Perris	California	33.782519	-117.228648
2	102	North Lab	4288 African Violet Way	Cleveland	Ohio	41.499320	-81.694361

	lab_id	names	street	city	state	lat	lon
3	103	Downtown Lab	9734 Algerian Oak Quercus Avenue	Worcester	Massachusetts	42.262593	-71.802293
4	104	East Lab	4818 Almond Lane	Columbia	South Carolina	34.000710	-81.034814

In [98]:
```
INSERT('Laboratories')
TABLES('Laboratories')
FK('Laboratories')
```

In [99]:
```
len(Laboratories)
```

Out[99]: 5

Expenses (n=60)

12 monthly reports for 5 different labs

In [100…
```
#Lab Operation - 12 bills for each month
group_id = np.arange(10000,10060,1)
lab_id= [100]*12+[101]*12+[102]*12+[103]*12+[104]*12
month = pd.date_range('2021-01-01','2021-12-31',
             freq='MS').strftime("%b").tolist()*6
electric = np.random.normal(500,10,60).round(2)
water = np.random.normal(300,10,60).round(2)
waste = np.random.normal(600,10,60).round(2)

expenses = list(zip(group_id, lab_id, month,
                    electric, water, waste))
```

In [101…
```
### CREATE DATAFRAME
Expenses = pd.DataFrame(expenses, columns=['group_id', 'lab_id', 'month', 'electric', 'water', 'waste'])
Expenses.to_csv(csv_+'Expenses.csv', header = None, index = False)
Expenses.head()
```

Out[101…

	group_id	lab_id	month	electric	water	waste
0	10000	100	Jan	511.25	296.55	617.14
1	10001	100	Feb	493.58	301.71	605.55
2	10002	100	Mar	494.20	289.40	604.74
3	10003	100	Apr	497.57	299.09	604.79

	group_id	lab_id	month	electric	water	waste
4	10004	100	May	486.51	304.73	604.39

In [102…

```
INSERT('Expenses')
TABLES('Expenses')
FK('Expenses')
```

In [103…

```
len(Expenses)
```

Out[103…

```
60
```

Employees (n=100)

In [3]:

```
#Employment Info - 100 employees representing admin, techs, assistants, security, custodians, hr
employee_id = np.arange(10000000,10000100,1)

address_id = np.arange(10050000,10050100,1)

lab_id = random.choices([100,101,102,103,104],k=100)

first = [f'{chr(i)}***' for i in np.random.randint(65,90,50000)]

last = [f'{chr(i)}***' for i in np.random.randint(65,90,50000)]

position = ['Manager']*5+['Techs']*30+['Plebs']*30+['Inventory']*10+['Security']*25

manager_salary = np.random.normal(100000,10,5).round(2).tolist()
tech_salary = np.random.normal(60000,10,30).round(2).tolist()
phleb_salary = np.random.normal(40000,10,30).round(2).tolist()
inventory_salary = np.random.normal(65000,10,10).round(2).tolist()
security_salary = np.random.normal(45000,10,25).round(2).tolist()
salary =  manager_salary + tech_salary + phleb_salary + inventory_salary + security_salary

certified =  ['Yes']*5+['Yes']*20+['No']*10+['Yes']*25+['No']*5+['Yes']*35

month1 = [i for i in np.random.randint(1,12,100)]
day1 = [i for i in np.random.randint(1,28,100)]
year1 = [i for i in np.random.randint(1960,2000,100)]
dob = [f'{i}/{j}/{k}' for i,j,k in zip (month1,day1, year1)]

# hire dates
hired = [pd.to_datetime(i) + pd.Timedelta(weeks=1092) for i in dob]
hired = [i.strftime("%m/%d/%Y") for i in hired]
```

```
gender = random.choices(['Female','Male'], weights=[80,20],
                        k = 100)
race = random.choices(['African American','Caucasian', 'Hispanic', 'Asian'],
                      weights = [20, 50, 10, 20],k = 100)
employees = list(zip(employee_id, address_id, lab_id, first, last, position,
                     salary,certified, dob,hired,
                     gender, race))
```

In [34]:

```
### CREATE DATAFRAME
Employees = pd.DataFrame(employees, columns=['employee_id', 'address_id', 'lab_id','first',
                                             'last', 'position', 'salary',
                                                'certified','dob','hired','gender','race'])
Employees.to_csv(csv_+'Employees.csv', header = None, index = False)
Employees.head()
```

Out[34]:

	employee_id	address_id	lab_id	first	last	position	salary	certified	dob	hired	gender	race
0	10000000	10050000	104	F***	O***	Manager	99980.49	Yes	11/6/1960	10/11/1981	Female	African American
1	10000001	10050001	102	V***	T***	Manager	99987.88	Yes	2/12/1975	01/17/1996	Male	African American
2	10000002	10050002	100	W***	J***	Manager	99996.57	Yes	2/23/1998	01/28/2019	Female	Asian
3	10000003	10050003	101	M***	G***	Manager	100007.25	Yes	8/13/1965	07/18/1986	Female	Caucasian
4	10000004	10050004	102	G***	H***	Manager	100027.29	Yes	2/21/1991	01/26/2012	Male	Hispanic

In [106…

```
INSERT('Employees')
TABLES('Employees')
FK('Employees')
```

In [107…

```
len(Employees)
```

Out[107…

```
100
```

Employee_Survey (n=1200)

In [108…

```
#Employee Survey - monthly survey scale 1-10 =
employee_id = np.arange(10000000,10001200,1)[:100].tolist()*12

survey_id= np.arange(10000000,10001201,1)

pay = np.random.normal(7,1,1200).round(0)
```

```
promotion=np.random.normal(5,1,1200).round(0)

manager=np.random.normal(8,1,1200).round(0)

volume=np.random.normal(7,1,1200).round(0)

tools=np.random.normal(7,1,1200).round(0)

overall=np.random.normal(7,1,1200).round(0)

employee_surveys = list(zip(survey_id, employee_id, pay, promotion,
                            manager, volume, tools, overall))
```

In [109…

```
### CREATE DATAFRAME
Employee_Surveys = pd.DataFrame(employee_surveys , columns=['survey_id', 'employee_id','pay', 'promotion', 'manager',
                                                            'work_volume', 'available_tools', 'overall'])
Employee_Surveys.to_csv(csv_+'Employee_Surveys.csv', header = None, index = False)
Employee_Surveys.head()
```

Out[109…

	survey_id	employee_id	pay	promotion	manager	work_volume	available_tools	overall
0	10000000	10000000	6.0	4.0	6.0	10.0	8.0	8.0
1	10000001	10000001	8.0	5.0	8.0	6.0	8.0	7.0
2	10000002	10000002	7.0	6.0	8.0	8.0	8.0	8.0
3	10000003	10000003	6.0	6.0	9.0	6.0	7.0	6.0
4	10000004	10000004	8.0	6.0	8.0	7.0	8.0	7.0

In [110…

```
INSERT('Employee_Surveys')
TABLES('Employee_Surveys')
FK('Employee_Surveys')
```

In [111…

```
len(Employee_Surveys)
```

Out[111…

```
1200
```

Shipments (n=60)

In [112…

```
#Supply Chain -  Per periodic shipment of various items in the company

inventory_id = np.arange(10000000,10000600,1)
```

```
# lab
lab_id = [100]*12+[101]*12+[102]*12+[103]*12+[104]*12

# shipping dates
ship_date1 = [f'{i+1}/01/2021' for i in range(12)]
ship_date2 = [f'{i+1}/01/2021' for i in range(12)]
ship_date3 = [f'{i+1}/03/2021' for i in range(12)]
ship_date4 = [f'{i+1}/02/2021' for i in range(12)]
ship_date5 = [f'{i+1}/01/2021' for i in range(12)]
shipped = ship_date1 + ship_date2 + ship_date3 + ship_date4 + ship_date5

# add some variance to use for KPI reports
arrival_date1 = [f'{i+1}/{12%(i+1)+5}/2021' for i in range(12)]
arrival_date2 = [f'{i+1}/{12%(i+1)+7}/2021' for i in range(12)]
arrival_date3 = [f'{i+1}/{12%(i+1)+8}/2021' for i in range(12)]
arrival_date4 = [f'{i+1}/{12%(i+1)+3}/2021' for i in range(12)]
arrival_date5 = [f'{i+1}/{12%(i+1)+5}/2021' for i in range(12)]
arrival = arrival_date1 + arrival_date2 + arrival_date3 + arrival_date4 + arrival_date5

# cost for each shipment
cost1 = np.random.normal(600,10,12).round(2).tolist()
cost2 = np.random.normal(600,10,12).round(2).tolist()
cost3 = np.random.normal(610,10,12).round(2).tolist()
cost4 = np.random.normal(620,10,12).round(2).tolist()
cost5 = np.random.normal(600,10,12).round(2).tolist()
costs = cost1 + cost2 + cost3 + cost4 + cost5

shipments = list(zip(inventory_id,lab_id,shipped,arrival,costs))
```

In [113…

```
### CREATE DATAFRAME
Shipments = pd.DataFrame(shipments , columns=['inventory_id','lab_id',
                                              'ship_date', 'arrival_date',
                                              'cost'])
Shipments.to_csv(csv_+'Shipments.csv', header = None, index = False)
Shipments.head()
```

Out[113…

	inventory_id	lab_id	ship_date	arrival_date	cost
0	10000000	100	1/01/2021	1/5/2021	605.62
1	10000001	100	2/01/2021	2/5/2021	602.48
2	10000002	100	3/01/2021	3/5/2021	604.67
3	10000003	100	4/01/2021	4/5/2021	595.85
4	10000004	100	5/01/2021	5/7/2021	624.90

In [114…
```
INSERT('Shipments')
TABLES('Shipments')
FK('Shipments')
```

In [115…
```
len(Shipments)
```

Out[115… 60

Analyzers (n=25)

In [270…
```
# Equiptment - catologe of all equiptment
# 4 analyzers per a site = 20 total
sn = np.arange(10000000,10000025,1)

lab_id = ['100']*5+['101']*5+['102']*5+['103']*5+['104']*5

panel_id = ['10000000','10000001','10000002','10000003','10000004'] * 5
panel_id = [int(panel) for panel in panel_id]

device= ['Sysmex','Siemens_1','Siemens_2','Roche','Abbott']
device = device * 5

analyzers = list(zip(sn,lab_id,panel_id,device))
len('10000003')
```

Out[270… 8

In [271…
```
### CREATE DATAFRAME
Analyzers = pd.DataFrame(analyzers ,columns=['serial_number', 'lab_id', 'panel_id','device'])
Analyzers.to_csv(csv_+'Analyzers.csv', header = None, index = False)
Analyzers.head()
```

Out[271…

	serial_number	lab_id	panel_id	device
0	10000000	100	10000000	Sysmex
1	10000001	100	10000001	Siemens_1
2	10000002	100	10000002	Siemens_2
3	10000003	100	10000003	Roche
4	10000004	100	10000004	Abbott

In [118…
```
INSERT('Analyzers')
TABLES('Analyzers')
FK('Analyzers')
```

In [119…
```
len(Analyzers)
```

Out[119… 25

QC_Definitions (n=390)

26 tests each with 3 different QC 'levels'(concentration amount) for each of the 5 Laboratories

In [266…
```
# Quality Control Test Definitions
qc_definition_id = np.arange(10000000,10000390,1)

level= ['1']*26 + ['2']*26 + ['3']*26
level = level * 5

lab_id = ['100']*78+['101']*78+['102']*78+['103']*78+['104']*78

# analyte names as seen on reports
analytes = ['SODIUM', 'POTASSIUM', 'BICARBONATE', 'CALCIUM', 'CHLORIDE', 'GLUCOSE',
                'PROTEIN','CREATININE', 'GFR', 'UREA NITROGEN',
                'AST','ALT', 'GGT', 'ALBUMIN' ,
                'CHOLESTEROL', 'TRIGLYCERIDE', 'LDL', 'HDL',
                'WBC', 'RBC', 'HEMOGLOBIN', 'HEMATOCRIT', 'PLATELET',
                'PROTIME', 'APTT', 'INR'] * 15

# average analyte value as tracked by laboratory
mid_mean = [140,4,25,9,105,130,
                7,1,60,15,
                22,24,80,4.2,
                200,60,80,40,
                8,4.5,14,42,250,
                12,16,1.1]
low_mean = np.multiply(mid_mean,0.9).tolist()
high_mean = np.multiply(mid_mean,1.1).tolist()
mean1 = np.r_[low_mean,mid_mean,high_mean] # lab 1
mean2 = np.r_[np.multiply(low_mean,1.1),np.multiply(mid_mean,1.1),np.multiply(high_mean,1.005)] # lab 2
mean3 = np.r_[np.multiply(low_mean,1.005),np.multiply(mid_mean,0.999),np.multiply(high_mean,0.999)] # lab 3
mean4 = np.r_[np.multiply(low_mean,1.0001),np.multiply(mid_mean,1.0001),np.multiply(high_mean,1.0001)] # lab 4
mean5 = np.r_[np.multiply(low_mean,1.02),np.multiply(mid_mean,1.0001),np.multiply(high_mean,0.99)] # lab 5
mean = np.r_[mean1, mean2, mean3, mean4, mean5]

# analyte_sd = 1% of the mean
sd = []
```

```
for m in mean:
    sd.append(round(m*0.03,2))
sd = sd

# unit of measure is measurement units relative to the ordered test - i.e. mg/dL
units = ['mmol/L', 'mmol/L', 'mmol/L', 'mg/dL', 'mmol/L', 'mg/dL',
                    'g/dL','mg/dL', 'mmol/L','mg/dL',
                    'U/L', 'U/L', 'U/L', 'g/dL',
                    'mmol/L', 'mg/dL', 'g/dL', 'U/L',
                    'U/L','/mL','g/dL', '%', '/mL',
                    'sec', 'sec', 'sec'] * 15
mult = [10,4,4,5,3]
mult = mult * 5 * 3
sn = np.arange(10000000,10000025,1)
sn = [str(s) for s in sn] * 3
serial_number=[]
for m,s in zip(mult,sn):
    temp = [s]
    temp = temp * m
    serial_number.extend(temp)
    del temp

qc_defintions = list(zip(qc_definition_id, level,analytes,mean,sd,units,serial_number, lab_id))
```

In [267…

```
### CREATE DATAFRAME
QC_Definitions = pd.DataFrame(qc_defintions ,columns=['qc_definition_id', 'level', 'analytes',
                                                      'mean','sd', 'units','serial_number', 'lab_id'])
QC_Definitions.to_csv(csv_+'QC_Definitions.csv', header = None, index = False)
QC_Definitions.head()
```

Out[267…

	qc_definition_id	level	analytes	mean	sd	units	serial_number	lab_id
0	10000000	1	SODIUM	126.0	3.78	mmol/L	10000000	100
1	10000001	1	POTASSIUM	3.6	0.11	mmol/L	10000000	100
2	10000002	1	BICARBONATE	22.5	0.68	mmol/L	10000000	100
3	10000003	1	CALCIUM	8.1	0.24	mg/dL	10000000	100
4	10000004	1	CHLORIDE	94.5	2.84	mmol/L	10000000	100

In [122…

```
INSERT('QC_Definitions')
TABLES('QC_Definitions')
FK('QC_Definitions')
```

In [123…

```
len(QC_Definitions)
```

Out[123… 75

Quality Control Results (n=142350)

- 5 Labs run 3 QC Levels each consisting of 26 Tests run daily
- a total of 142 350 individual test results
- the actual test result = nan value that will be determined using `postgreSQL` `pl/pgsql` language

In [9]:
```
qc_result_id = np.arange(10000000,10142350,1)

qc_definition_id = np.arange(10000000,10000078,1).tolist() * 5 * 365

datetime = pd.date_range(start ='1-1-2021',
         end ='12-31-2021', freq ='1D').tolist() * 5 * 3 * 26

results = [np.nan]*142350

qc_results = list(zip(qc_result_id,qc_definition_id,
                      datetime, results))
```

In [10]:
```
QC_Results = pd.DataFrame(qc_results, columns=['qc_result_id','qc_definition_id',
                                               'datetime','results'])
QC_Results.to_csv(csv_+'QC_Results.csv', header = None, index = False)
QC_Results.head()
```

Out[10]:

	qc_result_id	qc_definition_id	datetime	results
0	10000000	10000000	2021-01-01	NaN
1	10000001	10000001	2021-01-02	NaN
2	10000002	10000002	2021-01-03	NaN
3	10000003	10000003	2021-01-04	NaN
4	10000004	10000004	2021-01-05	NaN

In [126…
```
INSERT('QC_Results')
TABLES('QC_Results')
FK('QC_Results')
```

In [127…

```
len(QC_Results)
```

Out[127… 142350

Panels (n=5)

5 Panels with corresponding costs

In [9]:
```
panel_id = np.arange(10000000,10000005,1)

panels = ['CMP','HFT','LIP','CBC','COAG']

costs = [39.99,34.99,28.99,33.00,49.99]

panels_ = list(zip(panel_id, panels, costs))
```

In [10]:
```
### CREATE DATAFRAME
Panels = pd.DataFrame(panels_, columns=['panel_id','panel','cost'])
Panels.to_csv(csv_+'Panels.csv', header = None, index = False)
Panels.head()
```

Out[10]:

	panel_id	panel	cost
0	10000000	CMP	39.99
1	10000001	HFT	34.99
2	10000002	LIP	28.99
3	10000003	CBC	33.00
4	10000004	COAG	49.99

In [130…
```
INSERT('Panels')
TABLES('Panels')
FK('QC_Results')
```

In [131…
```
len(Panels)
```

Out[131… 5

Containers

A dictionary containing a container_id and a container type

In [251…
```
container_id = np.arange(10000000,11000001,1)
panels = ['10000000','10000001','10000002','10000003','10000004']
container_type = ['Red','Red','Green','Lavendar','Blue']
containers = list(zip(container_id, panels, container_type))
```

In [253…
```
Containers = pd.DataFrame(containers, columns=['container_id','panel_id','container_type'])
Containers.to_csv(csv_+'Containers.csv', header = None, index = False)
Containers.head()
```

Out[253…

	container_id	panel_id	container_type
0	10000000	10000000	Red
1	10000001	10000001	Red
2	10000002	10000002	Green
3	10000003	10000003	Lavendar
4	10000004	10000004	Blue

In [134…
```
INSERT('Containers')
TABLES('Containers')
FK('Containers')
```

In [135…
```
len(Containers)
```

Out[135…
```
5
```

Orders (n=1000000)

1 000 000 Orders for 50000 Customers within 1 year period for all 5 Labs

In [11]:
```
lab_id = ['100','101','102','103','104']
lab_id = random.choices(lab_id, k=1000000)

datetime = pd.date_range(start ='1-1-2021',
          end ='12-31-2021',periods=1000000).strftime("%m/%d/%Y %H:%M:%S")


customer_id = customer_id = np.arange(10000000,10050000,1)
```

```
customer_id = random.choices(customer_id,k=1000000)

panels = ['CMP','HFT','LIP','CBC','COAG']
panel = random.choices(panels, weights=[250000,250000,200000,200000,100000],k = 1000000)

panel_id=[]
for p in panel:
    panel_id.append(Panels[Panels['panel'].str.contains(str(p))]['panel_id'].values[0])

orders = list(zip(lab_id, datetime, customer_id, panel, panel_id))
Orders = pd.DataFrame(orders, columns=['lab_id','datetime',
                                       'customer_id','panel','panel_id'])

Orders = Orders.reindex(columns = ['lab_id','datetime',
                                   'customer_id','panel','panel_id'])

Orders = Orders.sort_values(['datetime']).reset_index(drop=True)
Orders['order_id'] = np.arange(10000000,11000000,1)

# split datetime into date and time
Orders[['date','time']] = Orders['datetime'].str.split(expand=True)
Orders = Orders.drop(['datetime'], axis=1)
Orders = Orders.reindex(columns = ['order_id','lab_id','customer_id',
                                   'panel','panel_id','date','time'])
```

In [7]:

```
Orders.panel_id = Orders.panel_id.map(int)
Orders.lab_id = Orders.lab_id.map(int)
Orders.to_csv(csv_+'Orders.csv', header = None, index = False)
Orders.head()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
C:\WINDOWS\TEMP/ipykernel_11948/1413432619.py in <module>
----> 1 Orders.panel_id = Orders.panel_id.map(int)
      2 Orders.lab_id = Orders.lab_id.map(int)
      3 Orders.to_csv(csv_+'Orders.csv', header = None, index = False)
      4 Orders.head()

NameError: name 'Orders' is not defined
```

In [147…

```
Orders.dtypes
```

Out[147…

```
order_id        int32
lab_id          int64
customer_id     int32
panel          object
panel_id        int64
```

```
date           object
time           object
dtype: object
```

In [142…
```
INSERT('Orders')
TABLES('Orders')
FK('Orders')
```

In [143…
```
len(Orders)
```

Out[143…
```
1000000
```

In [6]:
```
# convert time to a datetime object to allow for time delta
df['DataFrame Column'] = pd.to_datetime(df['DataFrame Column'], format=specify your format)

# create a list of random hours to subtract from the hours portion of time (i.e. 1,2,3,)
timestamp_list = [base + datetime.timedelta(days=1000000) for x in range(n_days)]

# perform pandas apply function to vector subtract
# np.subtract(orders.time as date time, random time delta list
for x in timestamp_list:
        print(x)
```

```
2022-09-22 16:11:02.782733
2022-09-23 16:11:02.782733
2022-09-24 16:11:02.782733
2022-09-25 16:11:02.782733
2022-09-26 16:11:02.782733
2022-09-27 16:11:02.782733
2022-09-28 16:11:02.782733
2022-09-29 16:11:02.782733
2022-09-30 16:11:02.782733
2022-10-01 16:11:02.782733
```

Test Definitions (n=130)

The patient reference mean and sd for all 26 Tests for 5 Labs for a total of 130 individual instances.

In [16]:
```
# Quality Control Test Definitions
test_definition_id = np.arange(10000000,10000130,1)

panel_id = ['10000000']*10+['10000001']*4+['10000002']*4+['10000003']*5+['10000004']*3
panel_id_ = panel_id * 5

lab_id = ['100']*26+['101']*26+['102']*26+['103']*26+['104']*26
```

```
tests = ['SODIUM', 'POTASSIUM', 'BICARBONATE', 'CALCIUM', 'CHLORIDE', 'GLUCOSE',
                'PROTEIN','CREATININE', 'GFR', 'UREA NITROGEN',
                'AST','ALT', 'GGT', 'ALBUMIN' ,
                'CHOLESTEROL', 'TRIGLYCERIDE', 'LDL', 'HDL',
                'WBC', 'RBC', 'HEMOGLOBIN', 'HEMATOCRIT', 'PLATELET',
                'PROTIME', 'APTT', 'INR'] * 5

### average analyte value as tracked by laboratory 1
mean1 = [140,4,25,9,105,130, #CMP
                7,1,60,15,   #CMP cont...
                22,24,80,4.2,   #HFT
                200,60,80,40,   #LIP
                8,4.5,14,42,250,  #CBC
                12,16,1.1]   #COAG
# analyte_sd = 1% of the mean
sd1 = [round(m*0.01,2) for m in mean1]

### average analyte value as tracked by laboratory 2
mean2 = np.multiply(mean1,0.99)
# analyte_sd = 1% of the mean
sd2 = [round(m*0.01,2) for m in mean2]

### average analyte value as tracked by laboratory 3
mean3 = np.multiply(mean1,0.97)
# analyte_sd = 1% of the mean
sd3 = [round(m*0.01,2) for m in mean3]

### average analyte value as tracked by laboratory 4
mean4 = np.multiply(mean1,0.95)
# analyte_sd = 1% of the mean
sd4 = [round(m*0.01,2) for m in mean4]

### average analyte value as tracked by laboratory 5
mean5 = np.multiply(mean1,1.02)
# analyte_sd = 1% of the mean
sd5 = [round(m*0.01,2) for m in mean5]
 # collaborate
means = np.r_[mean1,mean2,mean3,mean4,mean5]

sds = sd1+sd2+sd3+sd4+sd5

# unit of measure is measurement units relative to the ordered test - i.e. mg/dL
units = ['mmol/L', 'mmol/L', 'mmol/L', 'mg/dL', 'mmol/L', 'mg/dL',
                    'g/dL','mg/dL', 'mmol/L','mg/dL',
                    'U/L', 'U/L', 'U/L', 'g/dL',
                    'mmol/L', 'mg/dL', 'g/dL', 'U/L',
                    'U/L','/mL','g/dL', '%', '/mL',
                    'sec', 'sec', 'sec'] * 5
```

```
mult = [10,4,4,5,3]
mult = mult * 5
sn = np.arange(10000000,10000025,1)
sn = [str(s) for s in sn]
serial_number=[]
for m,s in zip(mult,sn):
    temp = [s]
    temp = temp * m
    serial_number.extend(temp)
    del temp

test_defintions = list(zip(test_definition_id,panel_id_,lab_id,
                           tests,means,sds,units,serial_number))
```

In [17]:

```
### CREATE DATAFRAME
Test_Definitions = pd.DataFrame(test_defintions ,columns=['test_definition_id', 'panel_id', 'lab_id','tests',
                                                          'mean','sd', 'units', 'serial_number'])
Test_Definitions.to_csv(csv_+'Test_Definitions.csv', header = None, index = False)
Test_Definitions.panel_id = Test_Definitions.panel_id.map(int)
Test_Definitions.lab_id = Test_Definitions.lab_id.map(int)
Test_Definitions.head()
```

Out[17]:

	test_definition_id	panel_id	lab_id	tests	mean	sd	units	serial_number
0	10000000	10000000	100	SODIUM	140.0	1.40	mmol/L	10000000
1	10000001	10000000	100	POTASSIUM	4.0	0.04	mmol/L	10000000
2	10000002	10000000	100	BICARBONATE	25.0	0.25	mmol/L	10000000
3	10000003	10000000	100	CALCIUM	9.0	0.09	mg/dL	10000000
4	10000004	10000000	100	CHLORIDE	105.0	1.05	mmol/L	10000000

In [138…

```
INSERT('Test_Definitions')
TABLES('Test_Definitions')
FK('Test_Definitions')
```

In [139…

```
len(Test_Definitions)
```

Out[139…

26

Samples (n=1000000)

- samples.contianer_id is dependent on ordered panel and type of container specific to the order
- Create corresponding container_id with SQL left join on orders and containers table and use containers.container_id
- Create corresponding container_id with SQL left join on orders and use orders.customer_id

In [144…
```
sample_id = np.arange(10000000,11000000,1)
order_id = np.arange(10000000,11000000,1)
container_id = [np.nan] * 1000000
employee_id = np.arange(10000000,10000100,1)
employee_id = random.choices(employee_id, k=1000000)
samples = list(zip(sample_id, order_id, container_id, customer_id, employee_id))
```

In [145…
```
Samples = pd.DataFrame(samples, columns=['sample_id', 'order_id', 'container_id',
                                         'customer_id', 'employee_id'])
Samples.to_csv(csv_+'Samples.csv', header = None, index = False)
Samples.head()
```

Out[145…

	sample_id	order_id	container_id	customer_id	employee_id
0	10000000	10000000	NaN	10006688	10000005
1	10000001	10000001	NaN	10012117	10000054
2	10000002	10000002	NaN	10010032	10000017
3	10000003	10000003	NaN	10019115	10000087
4	10000004	10000004	NaN	10009420	10000096

In [146…
```
INSERT('Samples')
TABLES('Samples')
FK('Samples')
```

In [147…
```
len(Samples)
```

Out[147…
```
1000000
```

Patient Results

- 1 000 000 Patient Results that depend on what test are ordered
- The orders table is left joined with panels table to derive corresponding tests
- Similarly, orders table is left joined with analyzers table to derive corresponding analyzer
- The resultant two aforementioned joins then allow to correspond a specific mean value for a particular analyzer for a particular test

- The latter mentioned mean value is then used within a `SQL Random()` function to create a mock analyzer result for a particular test.

In [20]:
```
# Create result for each test within each ordered panel:
df = pd.merge(Orders,Test_Definitions, left_on=['panel_id','lab_id'], right_on = ['panel_id','lab_id'])
result = [np.random.normal(x,np.multiply(x,0.05)) for x in df['mean'].values]
df['result'] = result
df['result'] = df['result'].round(2)
df['test_result_id'] = test_result_id = np.add(df.index,10000000)
Patient_Results = df.loc[:,['test_result_id', 'order_id','test_definition_id','date','time','customer_id',
                            'lab_id','panel','tests','result','units']]
```

In [21]:
```
Patient_Results.to_csv(csv_+'Patient_Results.csv', header = None, index = False)
Patient_Results.head()
```

Out[21]:

	test_result_id	order_id	test_definition_id	date	time	customer_id	lab_id	panel	tests	result	units
0	10000000	10000000	10000088	01/01/2021	00:00:00	10024734	103	HFT	AST	23.40	U/L
1	10000001	10000000	10000089	01/01/2021	00:00:00	10024734	103	HFT	ALT	22.57	U/L
2	10000002	10000000	10000090	01/01/2021	00:00:00	10024734	103	HFT	GGT	76.20	U/L
3	10000003	10000000	10000091	01/01/2021	00:00:00	10024734	103	HFT	ALBUMIN	3.84	g/dL
4	10000004	10000007	10000088	01/01/2021	00:03:40	10004719	103	HFT	AST	20.78	U/L

In [150…
```
INSERT('Patient_Results')
TABLES('Patient_Results')
FK('Patient_Results')
```

In [151…
```
len(Patient_Results)
```

Out[151…
```
1000000
```

Verifying the total number of testing with all ordered panels:

In [244…
```
#'CMP','HFT','LIP','CBC','COAG'
total = []
for panel in Orders.panel.values:
    if panel == 'CMP':
        total.append(10)
    if panel == 'HFT':
        total.append(4)
    if panel == 'LIP':
```

```
        total.append(4)
    if panel == 'CBC':
        total.append(5)
    if panel == 'COAG':
        total.append(3)
sum(total)
```

Out[244…

```
5598358
```

End of Section 2

In [152…

```
print("The End")
```

```
The End
```