```
import pandas as pd
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

Question 1:

- the types of variables: 'UVA', 'US', 'PVR', 'PMU', 'CYM', 'PTR', 'MUC', 'SS',

```
'UVJ', 'SSY', 'CLU', 'DV', 'USY', 'AGE'
```

UVA - binary - statistics: mode

US - Low 0–6, high 7–20 Categorical --> statistics: median and means

PVR - Categorical - statistics: median and mean

PMU - Categorical - statistics: median and mean

CYM - Binary - statistics:mode

PTR - Categorical - statistics: median and mean

MUCP - Negative ≤0, positive >0 = Categorical - statistics: median and mean

SS - binary - statistics: mode

UVJ - binary - statistics: mode

SSY binary - statistics: mode

CLU - binary - statistics: mode

DV binary - statistics: mode

USY- binary - statistics: mode

Age - Nominal - statistics: mode

How many different diagnoses the data contains? [0 1 2 3 4]

5 diagnoses total

The average age is 52.327586206896555 of all patients

```
df = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/inco13par.txt", sep="\t")
```

```
df
```

	NO	DIAGNOSI	UVA	US	PVR	PMU	CYM	PTR	MUC	SS	UVJ	SSY	CLU	DV	USY	AGE
0	2	0	0	8	1	0.05	0	68	0	0	1	1	0	0	1	62
1	3	0	0	4	30	0.2	0	72	-9	0		1	0	0	0	46
2	4	0	0	4	40	0.1	0	72	-4	1	1	1	0	0	0	84
3	5	0	0	11	60	0.15	0	71	-1	0		1	0	0	1	53
4	6	0	0	8	5		0		-14	1		1	0	0	0	46
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
524	421	4	0	2	3							0	0	0	0	41
525	455	4	0	1	15							0	0	0	0	64
526	466	4	0	3	12							0	0	0	0	63
527	472	4	0	1								0	0	0	0	57
528	473	4	0	1	2							0	0	0	0	54

529 rows × 16 columns

```
print(df.columns)
```

```
Index(['NO', 'DIAGNOSI', 'UVA', 'US', 'PVR', 'PMU', 'CYM', 'PTR', 'MUC', 'SS',
       'UVJ', 'SSY', 'CLU', 'DV', 'USY', 'AGE'],
      dtype='object')
```

```
datatypes = df.dtypes
```

```
datatypes
```

```
NO           int64
DIAGNOSI     int64
UVA         object
US          object
PVR         object
PMU         object
CYM         object
PTR         object
MUC         object
SS          object
UVJ         object
SSY         object
CLU          int64
DV           int64
USY         object
AGE         object
dtype: object
```

```
for variable in df.columns:
  print(f"Column {variable}")
  print(df[variable].unique())
```

```
Column NO
[  2   3   4   5   6   8  10  11  13  14  15  16  17  19  21  24  25  26
  27  29  33  34  37  38  46  48  49  50  51  53  55  57  59  60  62  66
  72  73  77  78  79  80  81  84  87  88  91  92  95  96  97  98  99 100
 101 102 103 104 106 108 109 111 113 114 116 118 123 124 130 132 133 135
 139 140 142 143 144 145 146 149 151 153 154 155 156 157 159 160 161 162
 163 164 165 171 173 175 176 177 178 179 180 181 182 184 187 189 190 191
 192 193 194 195 196 197 198 199 200 202 204 205 206 207 208 209 210 212
 215 216 217 218 219 220 222 223 224 225 230 232 233 235 236 238 241 243
 244 245 246 247 248 254 255 256 257 258 259 260 261 262 263 265 266 267
 270 274 275 276 277 279 280 281 283 284 285 286 287 290 291 292 293 296
 301 304 305 306 308 309 310 314 315 318 321 326 328 329 331 332 334 336
 337 340 341 342 343 345 348 350 355 356 357 361 362 363 364 366 367 369
 372 373 374 376 378 380 382 383 386 387 389 390 391 395 396 397 398 400
 402 406 409 413 414 415 416 417 418 420 422 423 425 426 428 429 432 433
 434 436 437 438 439 440 442 443 444 446 447 448 451 452 454 457 458 459
 460 461 462 464 465 468 470 471 474 475 476 477 478 482 495 506 510 523
 527 549 572 573 589 614 625 650 652 665 669 670 683 685 687 688 692 697
 702 703 704 705 707 708 709 710 711 712 716 717 718 719 720 721 723   1
   7  20  23  30  36  39  41  43  54  56  70 107 115 119 120 125 128 131
 134 136 137 138 147 152 158 166 167 169 170 185 188 211 214 226 228 231
 234 237 239 250 272 273 289 294 300 303 307 311 316 344 360 379 392 401
 410 411 412 424 427 430 431 441 445 449 453 456 463 467 469 483 490 492
 494 496 498 501 512 513 520 522 528 531 532 534 535 536 537 539 542 544
 551 552 553 554 555 561 564 567 574 583 585 586 595 599 601 603 604 605
 608 609 615 618 620 622 623 626 627 634 640 644 647 648 651 653 656 664
 668 671 673 676 677 680 691 694 696 701 713 714 722 105 186 229 299 327
 381 408 486 491 505 525 548 556 562 563 565 579 580 587 610 611 616 619
 628 629 632 637 642 655 672 674 678 699  22  90 110 365 487 489 516 533
 593 621 639 646 659 695 727 269 338 339 347 354 358 370 375 384 388 393
 404 419 421 455 466 472 473]
Column DIAGNOSI
[0 1 2 3 4]
Column UVA
['0' ' ' '1']
Column US
['8' '4' '11' ' ' '5' '6' '3' '14' '7' '10' '9' '18' '2' '1' '13' '0' '15'
 '12' '16']
Column PVR
['1' '30' '40' '60' '5' '80' '20' ' ' '10' '2' '100' '160' '75' '150' '50'
 '200' '45' '8' '25' '120' '31' '4' '16' '7' '44' '35' '110' '6' '3' '250'
 '130' '15' '43' '23' '105' '55' '70' '90' '18' '65' '180' '19' '140' '95'
 '9' '12']
Column PMU
['0.05' '0.2' '0.1' '0.15' ' ' '0.95' '0.4' '0.8' '0.3' '0.6' '0.5' '0.7'
 '0.9']
Column CYM
['0' ' ' '1']
Column PTR
```

```
['68' '72' '71' ' ' '57' '27' '102' '82' '48' '86' '62' '80' '64' '81'
 '52' '21' '98' '63' '79' '50' '94' '89' '56' '69' '93' '78' '60' '70'
 '119' '75' '91' '73' '61' '95' '88' '83' '90' '84' '40' '85' '76' '109'
 '67' '87' '55' '65' '74' '66' '58' '142' '47' '51' '45' '17' '53' '46'
 '92' '49' '36' '104' '103' '32' '124' '31' '100' '54' '139' '59' '16'
 '105' '77' '20' '35' '28' '38' '34' '33' '99' '96' '128']
Column MUC
['0' '-9' '-4' '-1' '-14' '-11' ' ' '-17' '-8' '-19' '-24' '-12' '16' '-2'
 '-26' '-27' '-22' '14' '-3' '-23' '4' '6' '10' '-20' '-5' '-31' '-25'
```

```
df['AGE'].isna().any()
```

```
False
```

```
AGE_COLUMN = pd.to_numeric(df['AGE'], errors='coerce')
```

```
print(AGE_COLUMN.mean())
```

```
52.327586206896555
```

Question 2: We find that only AGE has missing values so we group by the data by DIAGNOSI now missing values are replaced according to groups of DIAGNOSI

```
for columns in df.columns:
  print(f'Column {columns}, has missing values: {df[columns].isna().any()}')
```

```
Column NO, has missing values: False
Column DIAGNOSI, has missing values: False
Column UVA, has missing values: False
Column US, has missing values: False
Column PVR, has missing values: False
Column PMU, has missing values: False
Column CYM, has missing values: False
Column PTR, has missing values: False
Column MUC, has missing values: False
Column SS, has missing values: False
Column UVJ, has missing values: False
Column SSY, has missing values: False
Column CLU, has missing values: False
Column DV, has missing values: False
Column USY, has missing values: False
Column AGE, has missing values: False
```

```
df['AGE'].unique()
```

```
array(['62', '46', '84', '53', '73', '63', '60', '40', '55', '67', '59',
       '48', '44', '27', '51', '45', '41', '50', '33', ' ', '57', '47',
       '37', '54', '65', '42', '64', '43', '36', '58', '32', '34', '49',
       '39', '72', '66', '38', '31', '35', '56', '30', '70', '69', '61',
       '52', '74', '79', '68', '86', '75', '71', '76', '78', '81', '26',
       '89', '77', '29', '28'], dtype=object)
```

```
df['AGE'] = pd.to_numeric(df['AGE'], errors='coerce')
```

```
df['AGE'].isna().any()
```

```
True
```

```
df.groupby(['DIAGNOSI'])['AGE'].mean()
```

```
DIAGNOSI
0    50.444795
1    55.014388
2    55.545455
3    58.200000
4    53.944444
Name: AGE, dtype: float64
```

```
df["AGE"] = df.groupby("DIAGNOSI")['AGE'].transform(lambda x: x.fillna(x.mean()))
```

```
df['AGE'].isna().any()
```

```
False
```

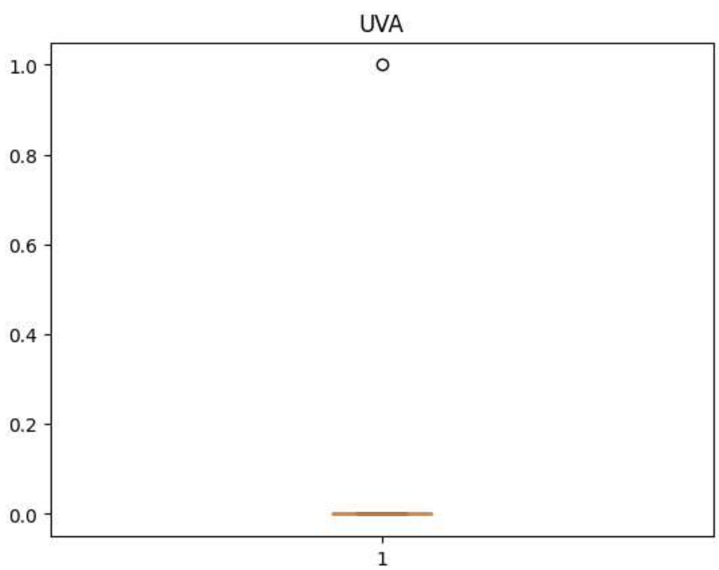
Question 3: Box plot for UVA, US, CYM and PTR Histogram for Age

What can you say about age distribution over all cases in the data?

0 and 1 looks Normal 2, 3, and 4 looks left skew

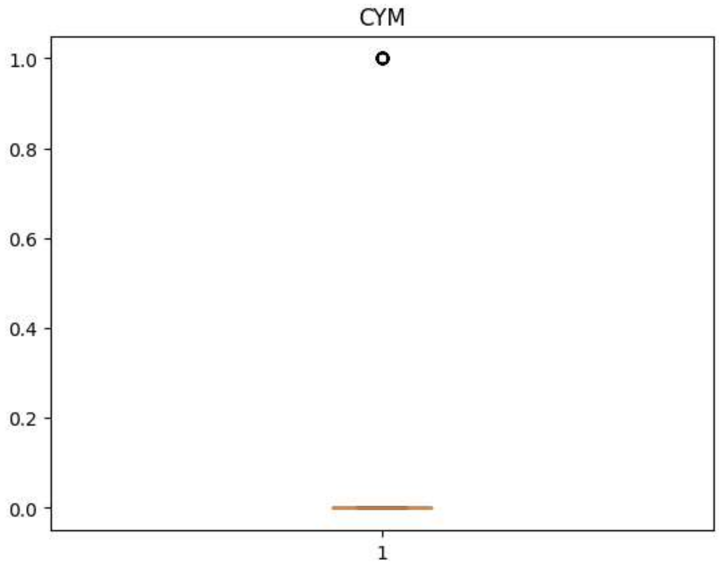
```
import matplotlib.pyplot as plt
import numpy as np


# Creating plot
df['UVA'] = pd.to_numeric(df['UVA'], errors='coerce')
plt.boxplot(df['UVA'].dropna())
# show plot
plt.title('UVA')
plt.show()
```

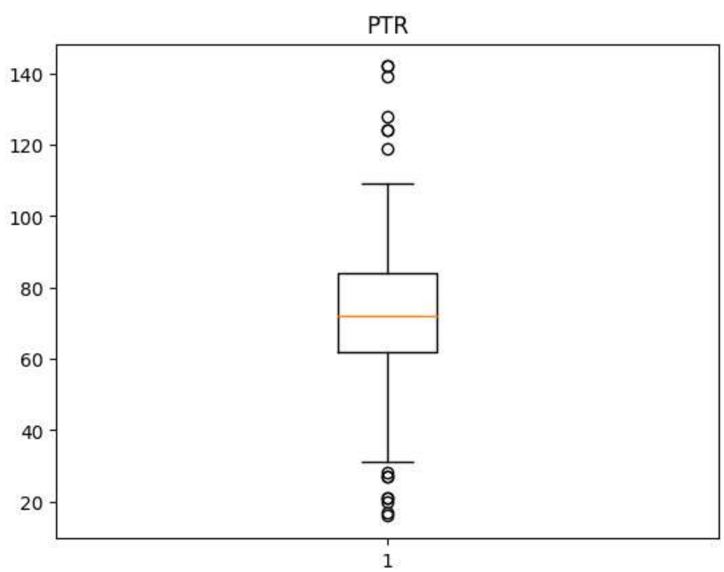



```
# Creating plot
df['US'] = pd.to_numeric(df['US'], errors='coerce')
plt.boxplot(df['US'].dropna())
# show plot
plt.title('US')
plt.show()
```


```
# Creating plot
df['CYM'] = pd.to_numeric(df['CYM'], errors='coerce')
plt.boxplot(df['CYM'].dropna())
# show plot
plt.title('CYM')
plt.show()
```

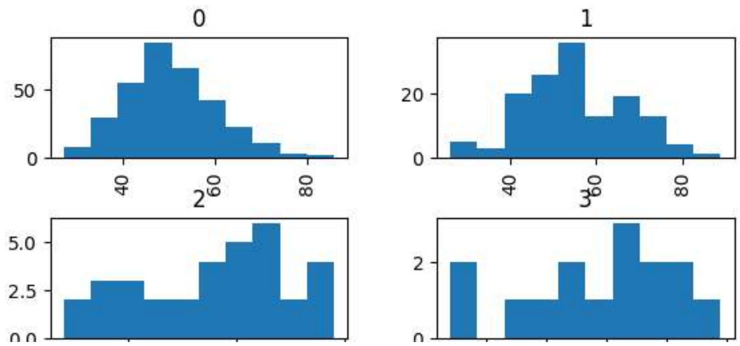



```
# Creating plot
df['PTR'] = pd.to_numeric(df['PTR'], errors='coerce')
plt.boxplot(df['PTR'].dropna())
# show plot
plt.title('PTR')
plt.show()
```




```
df.hist(by="DIAGNOSI", column='AGE')
```

```
array([[<Axes: title={'center': '0'}>, <Axes: title={'center': '1'}>],
       [<Axes: title={'center': '2'}>, <Axes: title={'center': '3'}>],
       [<Axes: title={'center': '4'}>, <Axes: >]], dtype=object)
```



Question 4:

row 2 and 269: 8.0, row 2 and 393: 2.0, and row 269-393: 6.0

So it seems that row 2 and row 393 are the closest in terms of distance

What kind of problems you may encounter when you use Euclidean distance measure? if the data is not metric then it might not perform well

In addition to Euclidean distance, there are plenty of others. What other distance measures you have heard of?

Manhattan or Block city (or Hamming) distance, Tshebyschev distance, Minkowski distances, Mahalanobis distance

```
row_2_age = df.iloc[1]['AGE']
row_269_age = df.iloc[268]['AGE']
row_393_age = df.iloc[392]['AGE']

import math

distance_1 = math.sqrt((row_2_age-row_269_age)**2)
distance_2 = math.sqrt((row_2_age-row_393_age)**2)
distance_3 = math.sqrt((row_269_age-row_393_age)**2)

print(f'row 2 and 269: {distance_1}, row 2 and 393: {distance_2}, and row 269-393: {distance_3}')
```

```
row 2 and 269: 8.0, row 2 and 393: 2.0, and row 269-393: 6.0
```

Question 5: