In [ ]:
```
#       ASSIGNMENT

# MINI-PROJECT 1: ABSENTEEISM AT WORK

# NAME:  OLUWASEGUN AJAYI
# CLASS: DATA ANALYTICS COHORT 12
```

In [4]:
```
#   IMPORTATION OF LIBRARIES THAT WILL BENEEDED


import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
```

In [8]:
```
# Import the data into jupyter notebook.

fileName = "Absenteeism_at_work.CSV"

filePath ="C:/Users/OLUWASEGUN/Documents/Myworkspace/"
df = pd.read_csv(filePath + fileName)
```

In [ ]:
```
# QUESTION 1.
# Describe the behavior of the dataset. Focus on Missing values, Duplicates, Shape, and features behavior.
```

In [ ]:
```
# ANSWERS TO QUESTION ONE(1)

# NOTE: The imported data labels was edited and
# spaces were removed on excel before being convert to csv
```

In [9]:
```
#        INTERPRETATION OF RESULT
# Loading of the first-five samples (rows)
# This tells me the nature and structure of my data and confirms that there 21 features in the data


df.head()
```

Out[9]:

	ID	Reasonforabsenc	Monthofabsenc	Dayofthewk	Seasons	Transportexpense	DistfrmResidencetoWork	Servicetime	Age	WorkloadAvgday
0	11	26	7	3	1	289	36	13	33	239,554
1	36	0	7	3	1	118	13	18	50	239,554
2	3	23	7	4	1	179	51	18	38	239,554
3	7	7	7	5	1	279	5	14	39	239,554
4	11	23	7	5	1	289	36	13	33	239,554

5 rows × 21 columns

In [117…

```
#        INTERPRETATION OF RESULT
# This will the return the last 5 rows/ samples
# This shows/confirms that I have about 740 (0-739) samples


df.tail()
```

Out[117…

	ID	Reasonforabsenc	Monthofabsenc	Dayofthewk	Seasons	Transportexpense	DistfrmResidencetoWork	Servicetime	Age	WorkloadAvgc
735	11	14	7	3	1	289	36	13	33	264,6
736	1	11	7	3	1	235	11	14	37	264,6
737	4	0	0	3	1	118	14	13	40	271,2
738	8	0	0	4	2	231	35	14	39	271,2
739	35	0	0	6	3	179	45	14	53	271,2

5 rows × 21 columns

In [10]:

```
#        INTERPRETATION OF RESULT
# THIS SHOWS THE SHAPE AND STRUCTURE (HOW MANY FEATURES 21 AND SAMPLES 740), WITH LABELS OF THE DATA,
# AND REVEALS THAT THERE ARE NO MISSING DATA IN ANY OF THE FEATURES OR SAMPLES
# This reveals the data type (integers 20 and object/string 1),
# also the memory usage (size of the data) (121.5 kb).
# No missing values
```

```
# Therefore, the data is essentially numeric.



df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 740 entries, 0 to 739
Data columns (total 21 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   ID                      740 non-null    int64 
 1   Reasonforabsenc         740 non-null    int64 
 2   Monthofabsenc           740 non-null    int64 
 3   Dayofthewk              740 non-null    int64 
 4   Seasons                 740 non-null    int64 
 5   Transportexpense        740 non-null    int64 
 6   DistfrmResidencetoWork  740 non-null    int64 
 7   Servicetime             740 non-null    int64 
 8   Age                     740 non-null    int64 
 9   WorkloadAvgday          740 non-null    object
 10  Hittarget               740 non-null    int64 
 11  Disciplinaryfailure     740 non-null    int64 
 12  Education               740 non-null    int64 
 13  Son                     740 non-null    int64 
 14  Socialdrinker           740 non-null    int64 
 15  Socialsmoker            740 non-null    int64 
 16  Pet                     740 non-null    int64 
 17  Weight                  740 non-null    int64 
 18  Height                  740 non-null    int64 
 19  BMI                     740 non-null    int64 
 20  Absenteeismtimeinhrs    740 non-null    int64 
dtypes: int64(20), object(1)
memory usage: 121.5+ KB
```

In [24]:

```
#        INTERPRETATION OF RESULT
# This specifically reveals the total samples and features of the data.
# 740 samples and 21 features respectively.


df.shape
```

Out[24]:

```
(740, 21)
```

In [12]:
```
#         INTERPRETATION OF RESULT
### THIS SHOWS THE DATA DISTRIBUTION AND SKEWNESS

# This will only return features that are only numeric.
# This gives me the summary statistics of the imported data
# The data is poorly behaved after careful inspection of the statistical features
# Hence, the cancer data requires a RENORMALISATION of the data before usage,
# having considered the features: MEAN, 50TH PERCENTILES, STANDARD DEVIATION, MINIMUM and MAXIMUM values.


df.describe()
```

Out[12]:

	ID	Reasonforabsenc	Monthofabsenc	Dayofthewk	Seasons	Transportexpense	DistfrmResidencetoWork	Servicetime	A
count	740.000000	740.000000	740.000000	740.000000	740.000000	740.000000	740.000000	740.000000	740.000
mean	18.017568	19.216216	6.324324	3.914865	2.544595	221.329730	29.631081	12.554054	36.450
std	11.021247	8.433406	3.436287	1.421675	1.111831	66.952223	14.836788	4.384873	6.478
min	1.000000	0.000000	0.000000	2.000000	1.000000	118.000000	5.000000	1.000000	27.000
25%	9.000000	13.000000	3.000000	3.000000	2.000000	179.000000	16.000000	9.000000	31.000
50%	18.000000	23.000000	6.000000	4.000000	3.000000	225.000000	26.000000	13.000000	37.000
75%	28.000000	26.000000	9.000000	5.000000	4.000000	260.000000	50.000000	16.000000	40.000
max	36.000000	28.000000	12.000000	6.000000	4.000000	388.000000	52.000000	29.000000	58.000

In [13]:
```
# HAVING CONSIDERED AND ANALYSE EACH FEATURES WITH:
# MINIMUM VALUES, MAXIMUM VALUES AND 50TH PERCENTILE THE DATA NEEDS RESCALING DUE TO DISPARITY
# USING THE MEAN VALUE AND 50TH PERCENTILE, THERE SEEMS TO BE NOT SO WIDELY APART
# THOUGH, THAT OF "ID" LOOKS VERY GOOD ACROSS THE FEATURE BUT NOT REALLY SO WITH OTHER FEATURES,
# THERE IS DISPARITY BETWEEEN OTHER FEATURES IN THEIR: MEAN AND 50TH.
# BALANCE IN SCALE OF MIN AND MAX VALUES AND THEIR MID POINTS AROSS
# EACH FEATURES REVEALS THAT IT IS NOT WELL BEHAVED AND REQUIRES RESCALING TO NORMALIZE THE DATA.


df.describe().T
```

Out[13]:

	count	mean	std	min	25%	50%	75%	max

	count	mean	std	min	25%	50%	75%	max
ID	740.0	18.017568	11.021247	1.0	9.0	18.0	28.0	36.0
Reasonforabsenc	740.0	19.216216	8.433406	0.0	13.0	23.0	26.0	28.0
Monthofabsenc	740.0	6.324324	3.436287	0.0	3.0	6.0	9.0	12.0
Dayofthewk	740.0	3.914865	1.421675	2.0	3.0	4.0	5.0	6.0
Seasons	740.0	2.544595	1.111831	1.0	2.0	3.0	4.0	4.0
Transportexpense	740.0	221.329730	66.952223	118.0	179.0	225.0	260.0	388.0
DistfrmResidencetoWork	740.0	29.631081	14.836788	5.0	16.0	26.0	50.0	52.0
Servicetime	740.0	12.554054	4.384873	1.0	9.0	13.0	16.0	29.0
Age	740.0	36.450000	6.478772	27.0	31.0	37.0	40.0	58.0
Hittarget	740.0	94.587838	3.779313	81.0	93.0	95.0	97.0	100.0
Disciplinaryfailure	740.0	0.054054	0.226277	0.0	0.0	0.0	0.0	1.0
Education	740.0	1.291892	0.673238	1.0	1.0	1.0	1.0	4.0
Son	740.0	1.018919	1.098489	0.0	0.0	1.0	2.0	4.0
Socialdrinker	740.0	0.567568	0.495749	0.0	0.0	1.0	1.0	1.0
Socialsmoker	740.0	0.072973	0.260268	0.0	0.0	0.0	0.0	1.0
Pet	740.0	0.745946	1.318258	0.0	0.0	0.0	1.0	8.0
Weight	740.0	79.035135	12.883211	56.0	69.0	83.0	89.0	108.0
Height	740.0	172.114865	6.034995	163.0	169.0	170.0	172.0	196.0
BMI	740.0	26.677027	4.285452	19.0	24.0	25.0	31.0	38.0
Absenteeismtimeinhrs	740.0	6.924324	13.330998	0.0	2.0	3.0	8.0	120.0

In [ ]:

```
```

In [22]:

```
#                INTERPRETATION
# CHECKING TO SEE IF THERE ARE MISSING VALUES ACROSS ALL FEATURES IN THE DATA
# HENCE, THE RESULT SHOWS, DATA IS TOTALLY FREE OF MISSING VALUES
```

```
df.isna().sum()
```

Out[22]:

```
ID                       0
Reasonforabsenc          0
Monthofabsenc            0
Dayofthewk               0
Seasons                  0
Transportexpense         0
DistfrmResidencetoWork   0
Servicetime              0
Age                      0
WorkloadAvgday           0
Hittarget                0
Disciplinaryfailure      0
Education                0
Son                      0
Socialdrinker            0
Socialsmoker             0
Pet                      0
Weight                   0
Height                   0
BMI                      0
Absenteeismtimeinhrs     0
dtype: int64
```

In [23]:

```
#         INTERPRETATION OF RESULT
# This confirms the total number of missing values from the entire dataset which is zero(0)


df.isna().sum().sum()
```

Out[23]:

```
0
```

In [ ]:

```
#               INTERPRETATION
# THE ABOVE CODE REVEALS THAT THERE ARE NO MISSING VALUE AS CONFIRMED WITH df.info()
```

In [21]:

```
#         INTERPRETATION
# TO CHECK FOR DUPLICATES IN  THE IMPORTED DATA "df"

df.duplicated()
```

Out[21]:
```
0      False
1      False
2      False
3      False
4      False
       ...
735    False
736    False
737    False
738    False
739    False
Length: 740, dtype: bool
```

In [15]:
```
#        INTERPRETATION OF RESULT
# TO CHECK THE SUM TOTAL OF ALL DUPLICATED SAMPLE (ROW) VALUES

df.duplicated().sum()
```

Out[15]:
```
34
```

In [ ]:
```
# THE RESULT ABOVE SHOWS, THERE ARE 34 DUPLICATED VALUES IN THE IMPORTED DATA "df"
```

In [ ]:
```
#        INTERPRETATION OF RESULT
# Therefore, to REMOVE DUPLICATES in the samples of label "ID"

# Knowing fully well that ID is a unique Identification which is specific and
# perculiar to individual employees
# df.drop_duplicates(subset ="ID", keep = False, inplace = True)
# The above code will delete all duplicated samples (rows) with same ID number and
# It will be left with 2 rows/samples with no duplication which will lead to data loss

# Hence, the code below will be appropriate to use to remove duplicates
# df2 = df.drop_duplicates(keep='first')
# Hence, the new data free of duplicates samples will be named as: "df2"
```

In [28]:
```
#        INTERPRETATION OF RESULT
### THIS WILL HELP REMOVE THE 34 DUPLICATED VALUES AND GIVE US A cleaned data df2
```

```
df2 = df.drop_duplicates(keep='first')
```

In [27]:

```
#         INTERPRETATION OF RESULT
# TO CONFIRM FOR DUPLICATES IN df2 HAVING DELETED THE DUPLICATES
# THIS CAN BE SEEN IN THE ENTRIES LENGTH FROM 740 BUT 706.


df2.duplicated()
```

Out[27]:

```
0      False
1      False
2      False
3      False
4      False
       ...
735    False
736    False
737    False
738    False
739    False
Length: 706, dtype: bool
```

In [29]:

```
#         INTERPRETATION OF RESULT
# CONFIRMS THAT THE 34 DUPLICATES IN THE ROWS "SAMPLES" HAS BEEN DROPPED


df2.duplicated().sum()
```

Out[29]:

```
0
```

In [36]:

```
#         INTERPRETATION OF RESULT
# THE 34 DUPLICATED SAMPLES (ROWS) HAS BEEN DROPPED
# HENCE, THE df2 BECOMES THE CLEAN DATA TO WORK WITH
```

In [37]:

```
#         INTERPRETATION OF RESULT
# RE-CONFIRMS THAT THERE ARE NO MISSING VALUES IN THE df2


df2.isna().sum()
```

Out[37]:
```
ID                       0
Reasonforabsenc          0
Monthofabsenc            0
Dayofthewk               0
Seasons                  0
Transportexpense         0
DistfrmResidencetoWork   0
Servicetime              0
Age                      0
WorkloadAvgday           0
Hittarget                0
Disciplinaryfailure      0
Education                0
Son                      0
Socialdrinker            0
Socialsmoker             0
Pet                      0
Weight                   0
Height                   0
BMI                      0
Absenteeismtimeinhrs     0
dtype: int64
```

In [38]:
```
#         INTERPRETATION OF RESULT
# This confirms zero (0) total number of missing values from the entire cleaned dataset df2


df2.isna().sum().sum()
```

Out[38]:
```
0
```

In [ ]:
```
#         INTERPRETATION OF RESULT
# THE df2 DATASET HAS NO MISSING VALUES
```

In [39]:
```
# Hence, df2 becomes the cleaned dataset to work with

#         INTERPRETATION OF RESULT
# Loading of the first-five samples (rows) of df2
# This tells me the nature and structure of my data and confirms that there 21 features in the data
```

```
df2.head()
```

Out[39]:

	ID	Reasonforabsenc	Monthofabsenc	Dayofthewk	Seasons	Transportexpense	DistfrmResidencetoWork	Servicetime	Age	WorkloadAvgday
0	11	26	7	3	1	289	36	13	33	239,554
1	36	0	7	3	1	118	13	18	50	239,554
2	3	23	7	4	1	179	51	18	38	239,554
3	7	7	7	5	1	279	5	14	39	239,554
4	11	23	7	5	1	289	36	13	33	239,554

5 rows × 21 columns

In [40]:

```
#        INTERPRETATION OF RESULT
# THIS SHOWS THE SHAPE AND STRUCTURE (HOW MANY FEATURES 21 AND SAMPLES REDUCED TO 706) WITH LABELS OF THE DATA,
# AND REVEALS THAT THERE ARE NO MISSING OR DUPLICATED VALUE(S) IN ANY OF THE FEATURES OR SAMPLES
# This reveals the data type (integers 20 and object/string 1),
# also the memory usage reduced (size of the data) (121.3 kb).
# This confirms that duplicated values has been dropped and reduced data size affirms it.
# Therefore, the data is essentially numeric.


df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 706 entries, 0 to 739
Data columns (total 21 columns):
 #   Column                 Non-Null Count  Dtype 
---  ------                 --------------  ----- 
 0   ID                     706 non-null    int64 
 1   Reasonforabsenc        706 non-null    int64 
 2   Monthofabsenc          706 non-null    int64 
 3   Dayofthewk             706 non-null    int64 
 4   Seasons                706 non-null    int64 
 5   Transportexpense       706 non-null    int64 
 6   DistfrmResidencetoWork 706 non-null    int64 
 7   Servicetime            706 non-null    int64 
 8   Age                    706 non-null    int64 
 9   WorkloadAvgday         706 non-null    object
```

```
 10  Hittarget              706 non-null    int64
 11  Disciplinaryfailure    706 non-null    int64
 12  Education              706 non-null    int64
 13  Son                    706 non-null    int64
 14  Socialdrinker          706 non-null    int64
 15  Socialsmoker           706 non-null    int64
 16  Pet                    706 non-null    int64
 17  Weight                 706 non-null    int64
 18  Height                 706 non-null    int64
 19  BMI                    706 non-null    int64
 20  Absenteeismtimeinhrs   706 non-null    int64
dtypes: int64(20), object(1)
memory usage: 121.3+ KB
```

In [41]:

```
#         INTERPRETATION OF RESULT
# We now have 706 samples(rows) and 21 features(columns) in df2 (as against
# 740 samples and 21 features in df) which depicts 34 duplicated samples were droopped.


df2.shape
```

Out[41]:

```
(706, 21)
```

In [43]:

```
#         INTERPRETATION OF RESULT
### THIS SHOWS THE DATA DISTRIBUTION AND SKEWNESS

# This will only return features that are only numeric.
# This gives me the summary statistics of the imported data
# The data is poorly behaved after careful inspection of the statistical features
# Hence, the data requires a RENORMALISATION of the data before usage,
# having considered the features: MEAN, 50TH PERCENTILES, STANDARD DEVIATION, MINIMUM and MAXIMUM values.


df2.describe().T
```

Out[43]:

	count	mean	std	min	25%	50%	75%	max
ID	706.0	18.192635	10.927472	1.0	10.00	18.0	28.00	36.0
Reasonforabsenc	706.0	18.882436	8.482877	0.0	13.00	23.0	26.00	28.0
Monthofabsenc	706.0	6.410765	3.404811	0.0	3.00	6.0	9.75	12.0

	count	mean	std	min	25%	50%	75%	max
Dayofthewk	706.0	3.890935	1.425503	2.0	3.00	4.0	5.00	6.0
Seasons	706.0	2.549575	1.121527	1.0	2.00	3.0	4.00	4.0
Transportexpense	706.0	222.977337	67.293426	118.0	179.00	225.0	260.00	388.0
DistfrmResidencetoWork	706.0	29.297450	14.706661	5.0	16.00	26.0	49.00	52.0
Servicetime	706.0	12.495751	4.370190	1.0	9.00	13.0	16.00	29.0
Age	706.0	36.478754	6.563404	27.0	31.00	37.0	40.00	58.0
Hittarget	706.0	94.548159	3.803854	81.0	92.25	95.0	97.00	100.0
Disciplinaryfailure	706.0	0.056657	0.231350	0.0	0.00	0.0	0.00	1.0
Education	706.0	1.291785	0.671499	1.0	1.00	1.0	1.00	4.0
Son	706.0	1.060907	1.104717	0.0	0.00	1.0	2.00	4.0
Socialdrinker	706.0	0.565156	0.496088	0.0	0.00	1.0	1.00	1.0
Socialsmoker	706.0	0.076487	0.265965	0.0	0.00	0.0	0.00	1.0
Pet	706.0	0.769122	1.333351	0.0	0.00	0.0	1.00	8.0
Weight	706.0	79.005666	12.862501	56.0	69.00	80.0	89.00	108.0
Height	706.0	172.202550	6.159814	163.0	169.00	171.0	172.00	196.0
BMI	706.0	26.635977	4.254901	19.0	24.00	25.0	31.00	38.0
Absenteeismtimeinhrs	706.0	7.143059	13.608120	0.0	2.00	3.0	8.00	120.0

In [ ]:

```
#         INTERPRETATION OF RESULT
# HAVING CONSIDERED AND ANALYSE EACH FEATURES WITH:
# MINIMUM VALUES, MAXIMUM VALUES AND 50TH PERCENTILE THE DATA NEEDS RESCALING DUE TO DISPARITY
# USING THE MEAN VALUE AND 50TH PERCENTILE, THERE SEEMS TO BE NOT SO WIDELY APART
# THOUGH, THAT OF "ID" LOOKS VERY GOOD ACROSS THE FEATURE BUT NOT REALLY SO WITH OTHER FEATURES,
# THERE IS DISPARITY BETWEEEN OTHER FEATURES IN THEIR: MEAN AND 50TH.
# BALANCE IN SCALE OF MIN AND MAX VALUES AND THEIR MID POINTS AROSS
# EACH FEATURES REVEALS THAT IT IS NOT WELL BEHAVED AND THE STANDARD DEVIATIONS HAS VARYING RANGE
# HENCE, THE DATA df2 REQUIRES RESCALING TO NORMALIZE THE DATA (df2)
```

In [ ]:

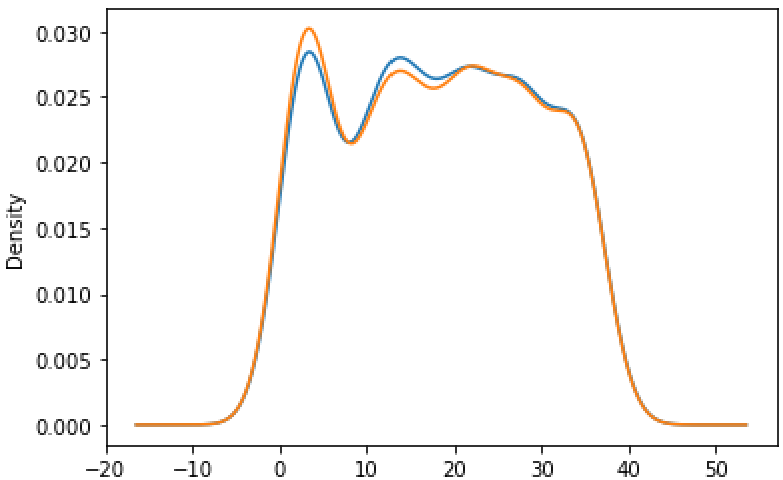
```
### HAVING VIRTUALLY CHECKED THE DISTRIBUTION... A DENSITY PLOT IS TO BE RUNNED
```

In [48]:

```
#         INTERPRETATION OF RESULT
# To check the distribution using density plot
# Also, juxtaposed the cleaned data df2 with the uncleaned data which shows slight changes


df2.ID.plot(kind = "density")     # Blue
df.ID.plot(kind = "density")      # Red
```

Out[48]:

```
<AxesSubplot:ylabel='Density'>
```

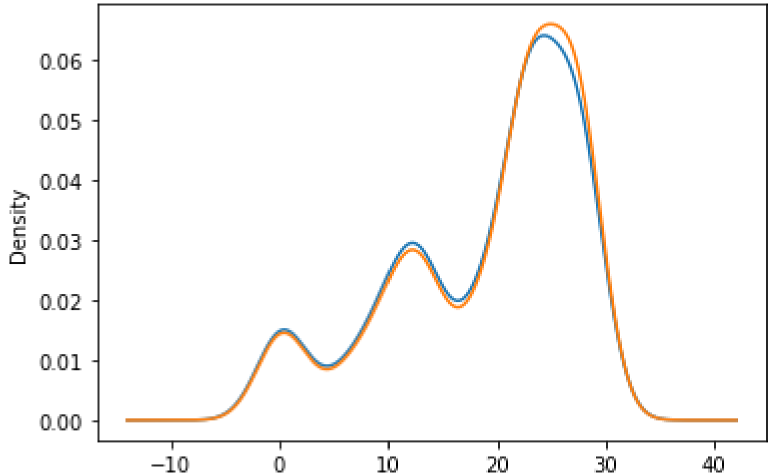


In [50]:

```
#         INTERPRETATION OF RESULT
# Also, juxtaposed the cleaned data df2 with the uncleaned data which shows slight changes


df2.Reasonforabsenc.plot(kind ="density")   # Blue
df.Reasonforabsenc.plot(kind ="density")    # Red
```

Out[50]:

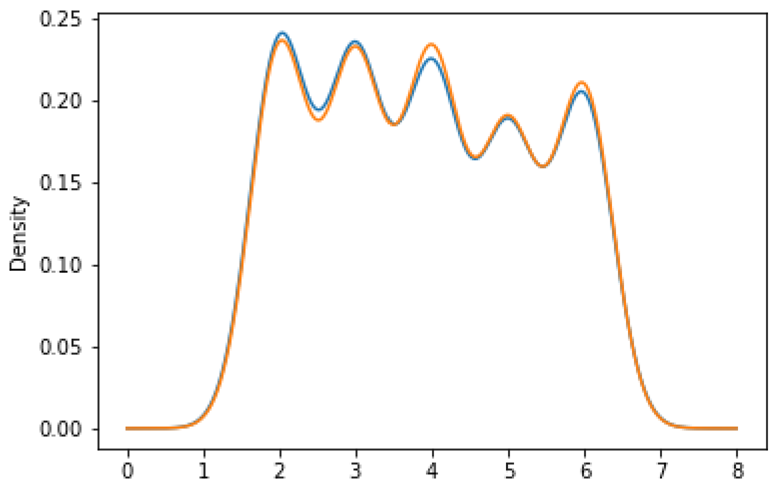
```
<AxesSubplot:ylabel='Density'>
```



In [52]:

```
#         INTERPRETATION OF RESULT
# Also, juxtaposed the cleaned data df2 with the uncleaned data which shows slight changes
# The 5 edges shows each workday of the week (Monday through Friday).


df2.Dayofthewk.plot(kind = "density")       # Blue
df.Dayofthewk.plot(kind = "density")        # Red
```

Out[52]: <AxesSubplot:ylabel='Density'>

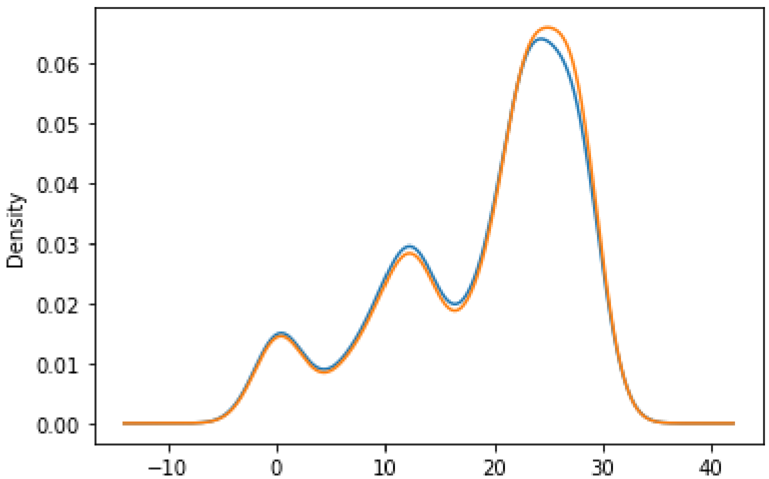


In [56]:

```
#         INTERPRETATION OF RESULT
```

```
# Also, juxtaposed the cleaned data df2 with the uncleaned data which shows slight changes


df2.Reasonforabsenc.plot(kind ="density")    # Blue
df.Reasonforabsenc.plot(kind ="density")     # Red
```

Out[56]: <AxesSubplot:ylabel='Density'>



In [57]:

```
df2.columns
```

Out[57]:

```
Index(['ID', 'Reasonforabsenc', 'Monthofabsenc', 'Dayofthewk', 'Seasons',
       'Transportexpense', 'DistfrmResidencetoWork', 'Servicetime', 'Age',
       'WorkloadAvgday ', 'Hittarget', 'Disciplinaryfailure', 'Education',
       'Son', 'Socialdrinker', 'Socialsmoker', 'Pet', 'Weight', 'Height',
       'BMI', 'Absenteeismtimeinhrs'],
      dtype='object')
```

In [61]:

```
newDf = df2[['ID', 'Reasonforabsenc', 'Monthofabsenc', 'Dayofthewk', 'Seasons',
       'Transportexpense', 'DistfrmResidencetoWork', 'Servicetime', 'Age',
       'WorkloadAvgday ', 'Hittarget', 'Disciplinaryfailure', 'Education',
       'Son', 'Socialdrinker', 'Socialsmoker', 'Pet', 'Weight', 'Height',
       'BMI', 'Absenteeismtimeinhrs']].copy()
```

In [64]:

```
lastCol = newDf.pop('WorkloadAvgday ')
```

In [66]:
```
newDf.columns
```

Out[66]:
```
Index(['ID', 'Reasonforabsenc', 'Monthofabsenc', 'Dayofthewk', 'Seasons',
       'Transportexpense', 'DistfrmResidencetoWork', 'Servicetime', 'Age',
       'Hittarget', 'Disciplinaryfailure', 'Education', 'Son', 'Socialdrinker',
       'Socialsmoker', 'Pet', 'Weight', 'Height', 'BMI',
       'Absenteeismtimeinhrs'],
      dtype='object')
```

In [65]:
```
lastCol
```

Out[65]:
```
0      239,554
1      239,554
2      239,554
3      239,554
4      239,554
        ...
735    264,604
736    264,604
737    271,219
738    271,219
739    271,219
Name: WorkloadAvgday , Length: 706, dtype: object
```

In [67]:
```
# Normalization of the data

from sklearn.preprocessing import Normalizer

from numpy import set_printoptions

arrays = newDf.values       # CONVERTING TO ARRAYS , REMOVING HEADERS AND TURNS IT INTO LIST

X = arrays[:,0:19]          # SEPARATING X AND Y      # TO PREDICT DAYS OF ABSENTEEISM
Y = arrays[:,19]

scaler =Normalizer().fit(X)
normalizeX = scaler.transform(X)
```

In [68]:
```
normalizeX
```

Out[68]:
```
array([[0.02994586, 0.07078113, 0.01905646, ..., 0.2450116 , 0.46824438,
        0.08167053],
```

```
       [0.1361314 , 0.        , 0.02646999, ..., 0.37057993, 0.67309415,
        0.11722426],
       [0.01033825, 0.07925994, 0.02412259, ..., 0.30670151, 0.58583434,
        0.10682861],
       ...,
       [0.01572963, 0.        , 0.        , ..., 0.38537605, 0.66850947,
        0.13370189],
       [0.02462448, 0.        , 0.        , ..., 0.30780598, 0.52327017,
        0.10773209],
       [0.12036948, 0.        , 0.        , ..., 0.26481285, 0.60184739,
        0.0859782 ]])
```

In [69]:

```
normDf = pd.DataFrame(normalizeX, columns = ['ID', 'Reasonforabsenc', 'Monthofabsenc', 'Dayofthewk', 'Seasons',
       'Transportexpense', 'DistfrmResidencetoWork', 'Servicetime', 'Age',
       'Hittarget', 'Disciplinaryfailure', 'Education', 'Son', 'Socialdrinker',
       'Socialsmoker', 'Pet', 'Weight', 'Height', 'BMI'])
```

In [70]:

```
normDf
```

Out[70]:

	ID	Reasonforabsenc	Monthofabsenc	Dayofthewk	Seasons	Transportexpense	DistfrmResidencetoWork	Servicetime	Age	Hitt
0	0.029946	0.070781	0.019056	0.008167	0.002722	0.786759	0.098005	0.035391	0.089838	0.2
1	0.136131	0.000000	0.026470	0.011344	0.003781	0.446208	0.049159	0.068066	0.189071	0.3
2	0.010338	0.079260	0.024123	0.013784	0.003446	0.616849	0.175750	0.062030	0.130951	0.3
3	0.019992	0.019992	0.019992	0.014280	0.002856	0.796831	0.014280	0.039984	0.111385	0.2
4	0.029960	0.062644	0.019066	0.013618	0.002724	0.787142	0.098052	0.035408	0.089881	0.2
...	...	...	...	...	...	...	...	...	...	
701	0.030084	0.038289	0.019145	0.008205	0.002735	0.790400	0.098458	0.035554	0.090253	0.2
702	0.003102	0.034123	0.021714	0.009306	0.003102	0.728981	0.034123	0.043429	0.114776	0.2
703	0.015730	0.000000	0.000000	0.011797	0.003932	0.464024	0.055054	0.051121	0.157296	0.3
704	0.024624	0.000000	0.000000	0.012312	0.006156	0.711032	0.107732	0.043093	0.120044	0.2
705	0.120369	0.000000	0.000000	0.020635	0.010317	0.615604	0.154761	0.048148	0.182274	0.3

706 rows × 19 columns

In [72]:
```
normDf ['Absenteeismtimeinhrs'] = df ['Absenteeismtimeinhrs']
normDf ['Absenteeismtimeinhrs'] = df ['Absenteeismtimeinhrs']
```

In [73]:
```
normDf
```

Out[73]:

	ID	Reasonforabsenc	Monthofabsenc	Dayofthewk	Seasons	Transportexpense	DistfrmResidencetoWork	Servicetime	Age	Hit
0	0.029946	0.070781	0.019056	0.008167	0.002722	0.786759	0.098005	0.035391	0.089838	0.2
1	0.136131	0.000000	0.026470	0.011344	0.003781	0.446208	0.049159	0.068066	0.189071	0.3
2	0.010338	0.079260	0.024123	0.013784	0.003446	0.616849	0.175750	0.062030	0.130951	0.3
3	0.019992	0.019992	0.019992	0.014280	0.002856	0.796831	0.014280	0.039984	0.111385	0.2
4	0.029960	0.062644	0.019066	0.013618	0.002724	0.787142	0.098052	0.035408	0.089881	0.2
...	...	...	...	...	...	...	...	...	...	
701	0.030084	0.038289	0.019145	0.008205	0.002735	0.790400	0.098458	0.035554	0.090253	0.2
702	0.003102	0.034123	0.021714	0.009306	0.003102	0.728981	0.034123	0.043429	0.114776	0.2
703	0.015730	0.000000	0.000000	0.011797	0.003932	0.464024	0.055054	0.051121	0.157296	0.3
704	0.024624	0.000000	0.000000	0.012312	0.006156	0.711032	0.107732	0.043093	0.120044	0.2
705	0.120369	0.000000	0.000000	0.020635	0.010317	0.615604	0.154761	0.048148	0.182274	0.3

706 rows × 20 columns

In [75]:
```
normDf['WorkloadAvgday '] =lastCol
```

In [76]:
```
normDf
```

Out[76]:

	ID	Reasonforabsenc	Monthofabsenc	Dayofthewk	Seasons	Transportexpense	DistfrmResidencetoWork	Servicetime	Age	Hit
0	0.029946	0.070781	0.019056	0.008167	0.002722	0.786759	0.098005	0.035391	0.089838	0.2
1	0.136131	0.000000	0.026470	0.011344	0.003781	0.446208	0.049159	0.068066	0.189071	0.3

	ID	Reasonforabsenc	Monthofabsenc	Dayofthewk	Seasons	Transportexpense	DistfrmResidencetoWork	Servicetime	Age	Hit
2	0.010338	0.079260	0.024123	0.013784	0.003446	0.616849	0.175750	0.062030	0.130951	0.3
3	0.019992	0.019992	0.019992	0.014280	0.002856	0.796831	0.014280	0.039984	0.111385	0.2
4	0.029960	0.062644	0.019066	0.013618	0.002724	0.787142	0.098052	0.035408	0.089881	0.2
...	...	...	...	...	...	...	...	...	...	
701	0.030084	0.038289	0.019145	0.008205	0.002735	0.790400	0.098458	0.035554	0.090253	0.2
702	0.003102	0.034123	0.021714	0.009306	0.003102	0.728981	0.034123	0.043429	0.114776	0.2
703	0.015730	0.000000	0.000000	0.011797	0.003932	0.464024	0.055054	0.051121	0.157296	0.3
704	0.024624	0.000000	0.000000	0.012312	0.006156	0.711032	0.107732	0.043093	0.120044	0.2
705	0.120369	0.000000	0.000000	0.020635	0.010317	0.615604	0.154761	0.048148	0.182274	0.3

706 rows × 21 columns

In [77]:

```
#         INTERPRETATION OF DATA TRANSFORMATION
# THIS SHOWS THE NORMALIZED DATA STATISTICAL DISTRIBUTION
# having considered the features: MEAN, 50TH PERCENTILES, STANDARD DEVIATION, MINIMUM and MAXIMUM values.
# They all range between 0 and 1
# looking at the mean and 50% values they are all less than 1 and
# considering the Standard deviation none is above 1 compared to the previous one df2.describe()
# we only worked on normalizer and on mean


normDf.describe().T
```

Out[77]:

	count	mean	std	min	25%	50%	75%	max
ID	706.0	0.059550	0.040349	0.003080	0.023597	0.058916	0.089945	0.138123
Reasonforabsenc	706.0	0.060798	0.028919	0.000000	0.040056	0.066983	0.082581	0.112134
Monthofabsenc	706.0	0.020377	0.011218	0.000000	0.010340	0.019985	0.028836	0.047677
Dayofthewk	706.0	0.012432	0.004790	0.004447	0.008185	0.012018	0.016096	0.023879
Seasons	706.0	0.008154	0.003744	0.002352	0.005435	0.007919	0.011019	0.016088

	count	mean	std	min	25%	50%	75%	max
Transportexpense	706.0	0.685081	0.115377	0.442379	0.618298	0.723170	0.768688	0.869681
DistfrmResidencetoWork	706.0	0.092009	0.046073	0.014279	0.049181	0.083901	0.122375	0.179228
Servicetime	706.0	0.040555	0.015970	0.003087	0.032333	0.039449	0.048938	0.095388
Age	706.0	0.117310	0.028539	0.065807	0.090694	0.114467	0.131566	0.191837
Hittarget	706.0	0.302858	0.043637	0.205047	0.272443	0.302362	0.335418	0.394432
Disciplinaryfailure	706.0	0.000170	0.000704	0.000000	0.000000	0.000000	0.000000	0.003825
Education	706.0	0.004164	0.002338	0.002223	0.002960	0.003435	0.003908	0.012262
Son	706.0	0.003207	0.003287	0.000000	0.000000	0.003087	0.005465	0.011932
Socialdrinker	706.0	0.001760	0.001577	0.000000	0.000000	0.002720	0.003092	0.003974
Socialsmoker	706.0	0.000244	0.000861	0.000000	0.000000	0.000000	0.000000	0.003974
Pet	706.0	0.002245	0.003932	0.000000	0.000000	0.000000	0.003010	0.031459
Weight	706.0	0.253512	0.057409	0.148224	0.204334	0.232908	0.307441	0.385376
Height	706.0	0.551754	0.080597	0.395769	0.494851	0.544695	0.608776	0.695535
BMI	706.0	0.085386	0.018554	0.047888	0.068799	0.079145	0.106727	0.133702
Absenteeismtimeinhrs	706.0	6.745042	12.620714	0.000000	2.000000	3.000000	8.000000	120.000000

In [ ]:

```
#         INTERPRETATION OF RESULT
# HAVING CONSIDERED AND ANALYSE EACH FEATURES WITH:
# MINIMUM VALUES, MAXIMUM VALUES AND 50TH PERCENTILE THE DATA HAS UNDERGONE TRANSFORMATION (0-1)
# USING THE MEAN VALUE AND 50TH PERCENTILE, THERE SEEMS TO BE NOT SO FAR APART
# THERE IS CLOSENESS BETWEEEN OTHER FEATURES IN THEIR: MEAN AND 50TH.
# BALANCE IN SCALE OF MIN AND MAX VALUES AND THEIR MID POINTS AROSS
# EACH FEATURES REVEALS THAT IT IS WELL BEHAVED AND THE STANDARD DEVIATIONS HAS SIMILAR RANGE (0-1) VALUES
# HENCE, THE normDF DATA HAS BEEN RESCALED AND NORMALIZED THE DATA (df2) after careful inspection of the
# statistical features.
```

In [ ]:

```
```

In [ ]:

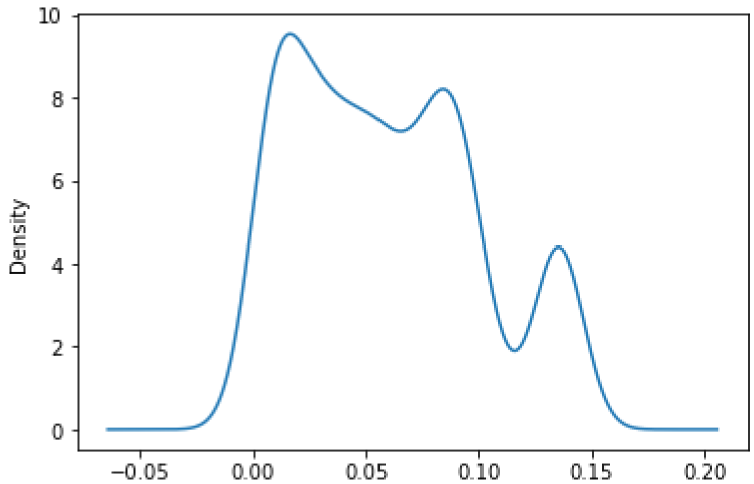
```
# ANSWER TO QUESTION TWO (2)
```

In [122…

```
normDf.ID.plot(kind = "density")
```

Out[122…

```
<AxesSubplot:ylabel='Density'>
```

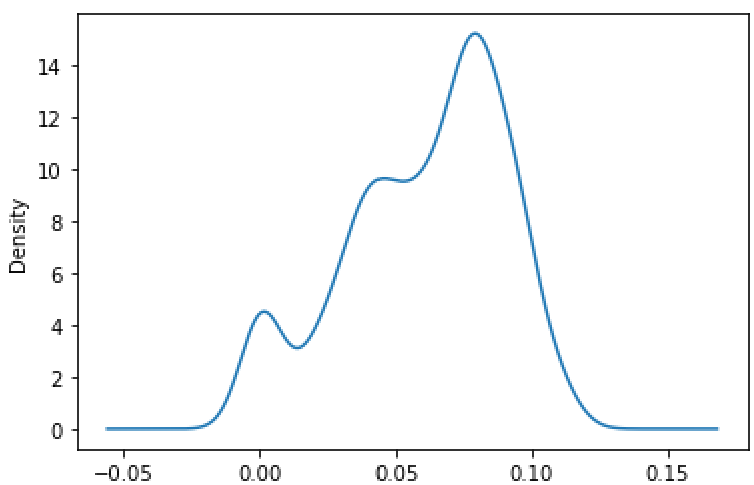


In [79]:

```
normDf.Reasonforabsenc.plot(kind = "density")
```

Out[79]:

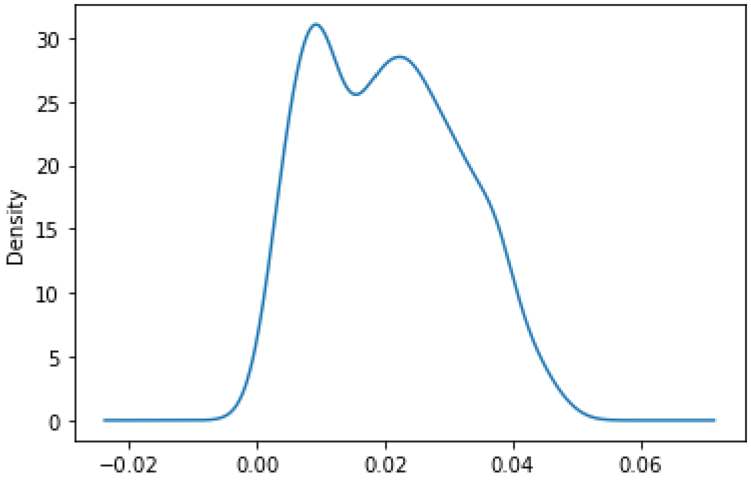
```
<AxesSubplot:ylabel='Density'>
```



In [80]:

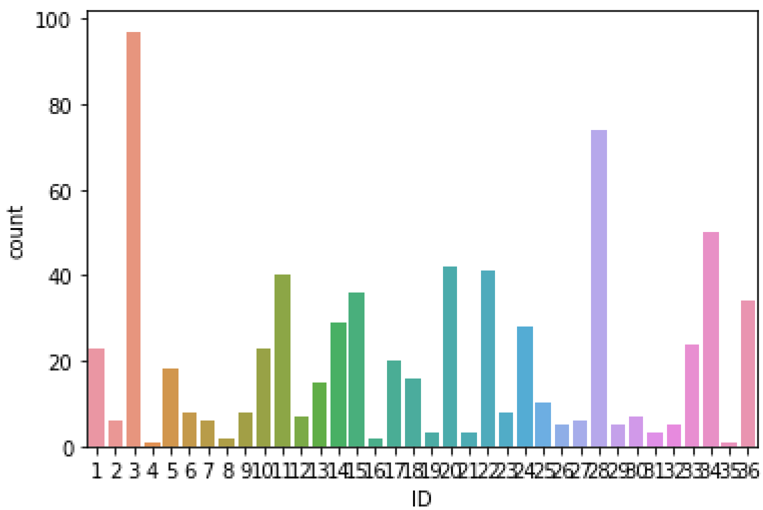
```
normDf.Monthofabsenc.plot(kind = "density")
```

Out[80]:
```
<AxesSubplot:ylabel='Density'>
```



In [82]:
```
# ID number 3 is the employee with the hightest absenteeism


sb.countplot(x = "ID", data = df2)
```

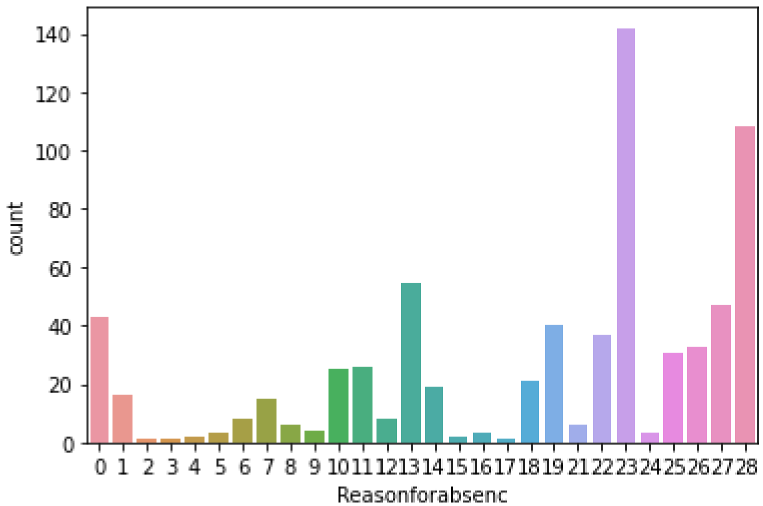
Out[82]:
```
<AxesSubplot:xlabel='ID', ylabel='count'>
```



In [83]:

```
# Number 23(blood donation) has the most reasons for absence


sb.countplot(x = "Reasonforabsenc", data = df2)
```

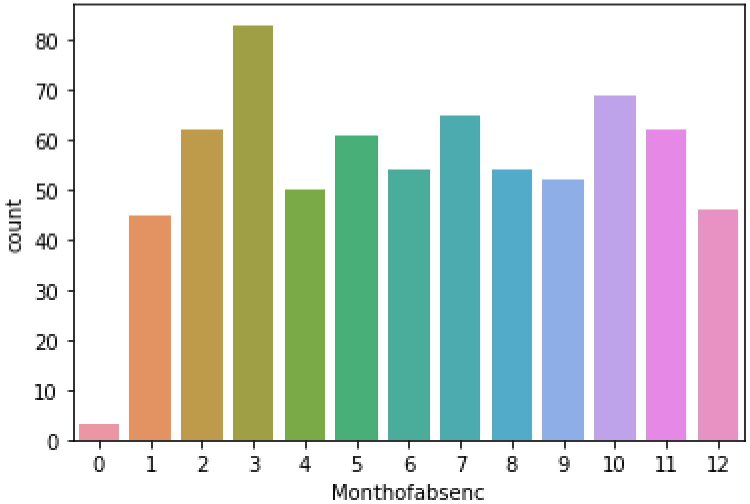
Out[83]: <AxesSubplot:xlabel='Reasonforabsenc', ylabel='count'>



In [84]:

```
# ...3rd month (MARCH) has the highest number absenteeism
#(probably, due to tax filing period and leave ends for the in 3rd month)
# This is the period of winter turning into spring.
# followed by the 7th, 2nd and 10th months follows
###  # summer vacation


sb.countplot(x = "Monthofabsenc", data = df2)
```

Out[84]: <AxesSubplot:xlabel='Monthofabsenc', ylabel='count'>

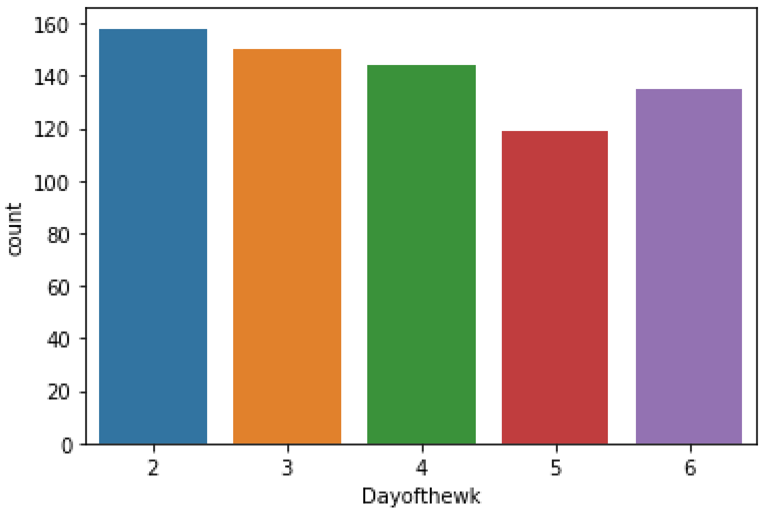


In [110…

```
# Monday being the first day of the work-week has the highest absence at
# work probably workers has travelled over the weekends,
# for partying with drinking and smoking and that could influence them resuming on monday


sb.countplot(x = "Dayofthewk", data = df2)
```

Out[110…

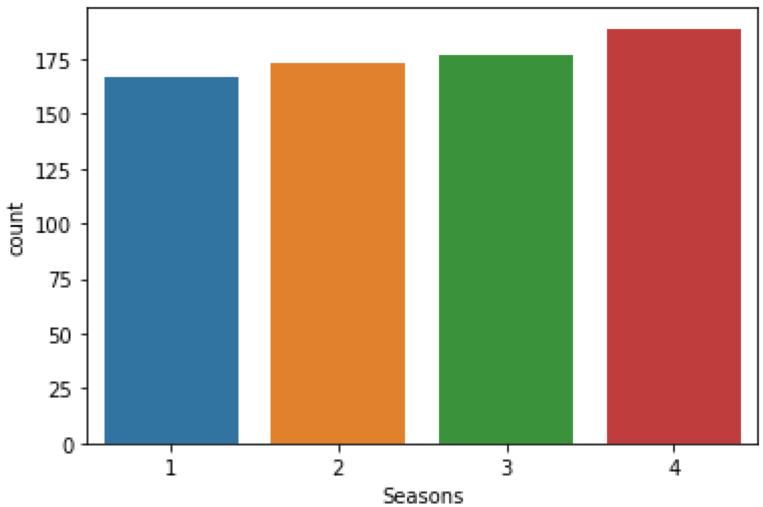
```
<AxesSubplot:xlabel='Dayofthewk', ylabel='count'>
```



In [95]:

```
# 4th seasons which Spring highest has influence on workers absence though, absenteeism cut across the 4 seasons.
# COMPARED WITH OTHER SEASONS OF THE YEAR.
# Absenteeism maybe as a result of heavy rain during Spring


# KEYS:
# Acoording data suppporting file:
# season 1: Summer
# season 2: Autumn
# season 3: Winter
# season 4: Spring



sb.countplot(x = "Seasons", data = df2)
```

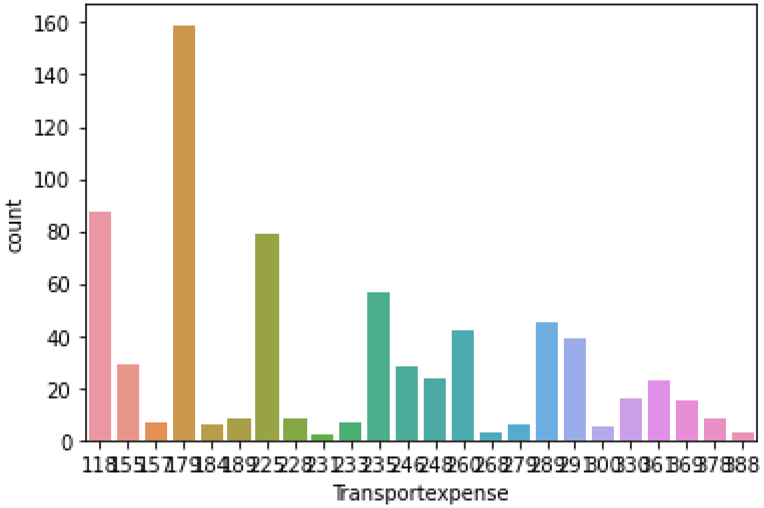
Out[95]: <AxesSubplot:xlabel='Seasons', ylabel='count'>



In [87]:

```
# Employees with transportation expenses of 179 tend to be more and this plays a role on the absence rate.
# Majority of the employees spend 179 on transportation.


sb.countplot(x = "Transportexpense", data = df2)
```

Out[87]: <AxesSubplot:xlabel='Transportexpense', ylabel='count'>

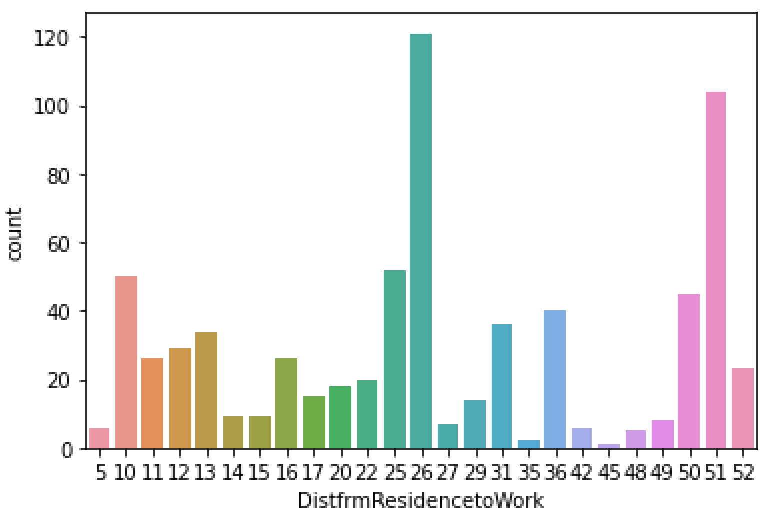


In [107…

```
# Higher number of the company employee reside at about 26Km distance to work location.
# The company has bulk of its employees coming from a distance of 26km and 51km.


sb.countplot(x = "DistfrmResidencetoWork", data = df2)
```

Out[107…

```
<AxesSubplot:xlabel='DistfrmResidencetoWork', ylabel='count'>
```

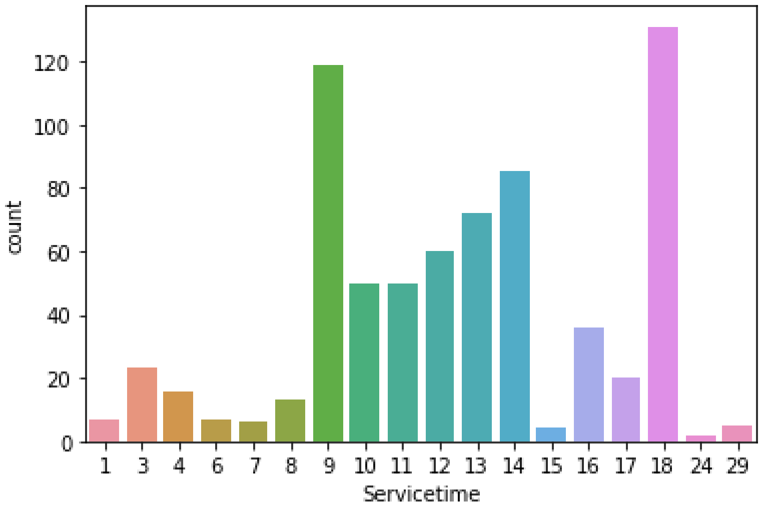


In [89]:

```
# Service time of 18 minutes is the highest
```

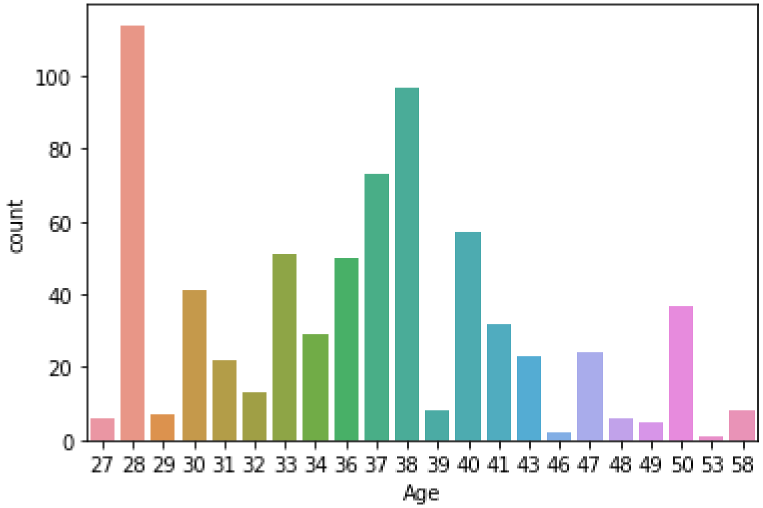
```
sb.countplot(x = "Servicetime", data = df2)
```

Out[89]: <AxesSubplot:xlabel='Servicetime', ylabel='count'>



In [90]:
```
# Company's employees with age 28 have the highest absenteeism rate
# This suggests that these are young guys in their early career and they seem
# not to really take work serious maybe due to not having dependent responsibilities.


sb.countplot(x = "Age", data = df2)
```

Out[90]: <AxesSubplot:xlabel='Age', ylabel='count'>



In [ ]:

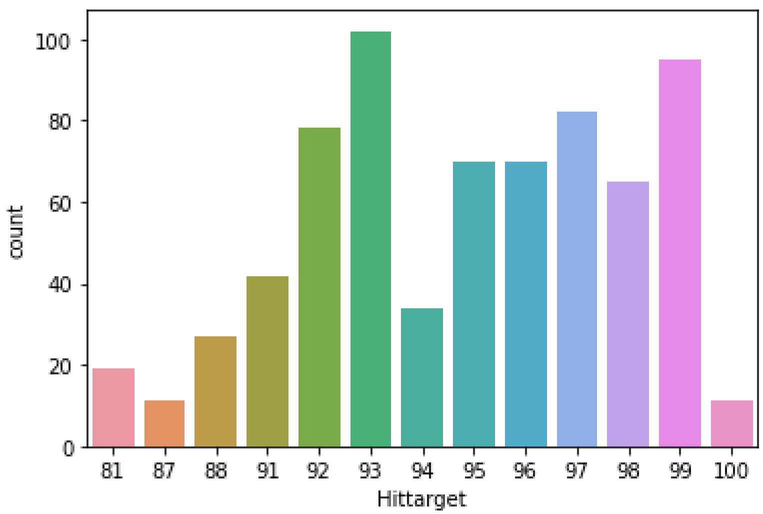
```
sb.countplot(x = "WorkloadAvgday", data = df2)
```

In [96]:

```
# Employees with hittarget 93 tend to be out of work more often than
# those with lesser hittarget


sb.countplot(x = "Hittarget", data = df2)
```

Out[96]:

```
<AxesSubplot:xlabel='Hittarget', ylabel='count'>
```

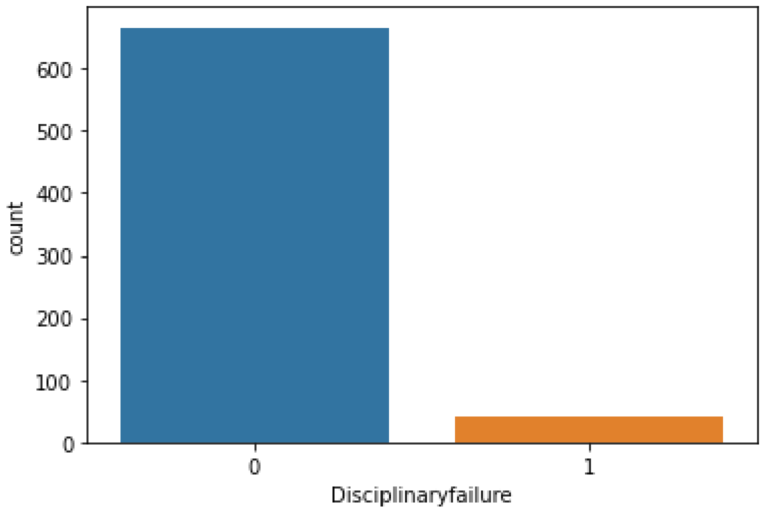


In [113…

```
# Disciplinary failure did not contributed to employee absenteeism
# COMPARE ... WITH EMPLOYEE OF AGE 28...
# YOUNGER AGE EMPLOYEE TENDS TO TAKE ADVANTAGE OF DISCIPLINARY FAILURE
# THIS IS NOT A SIGNIFICANT MAJOR CONTRIBUTOR TO ABSENTEEISM

# Acoording data suppporting file:
# YES= 1
# NO = 0

sb.countplot(x = "Disciplinaryfailure", data = df2)
```

Out[113…

```
<AxesSubplot:xlabel='Disciplinaryfailure', ylabel='count'>
```

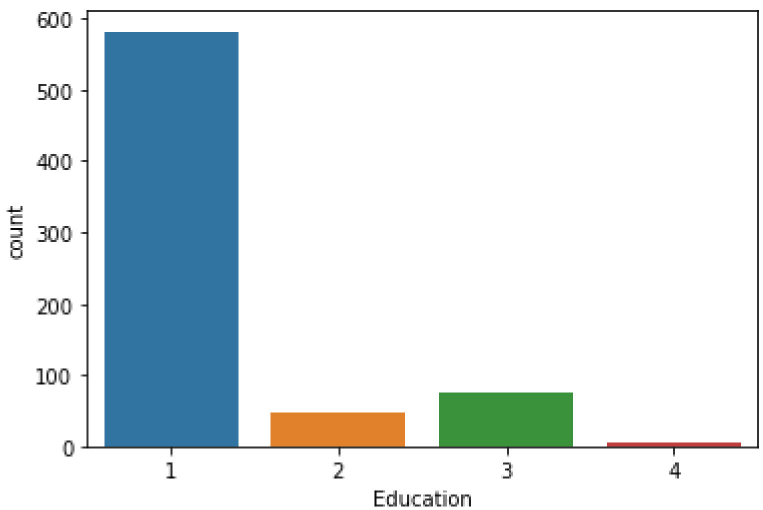


In [98]:

```
# Education level contributed greatly to employee absenteeism
# EMPLOYEES WITH HIGH SCHOOL EDUCATION TENDS TO HAVE THE HIGHEST ABSENTEEISM RATE WHILE
# EMPLOYEES WITH MASTERS AND DOCTOR EDUCATION HAVE THE LOWEST ABSENTEEISM RATE AND ARE
# LIKELY TO TAKE WORK MORE SERIOUSLY.
# HENCE, THE HIGHER EMPLOYEES EDUCATION, THE LESSER THE ABSENTEEISM RATE


# KEYS:
# Acoording data suppporting file:
# High school    = 1
# Graduate       = 2
# Post Graduate  = 3
# Masters& Doctor= 4
```

```
sb.countplot(x = "Education", data = df2)
```

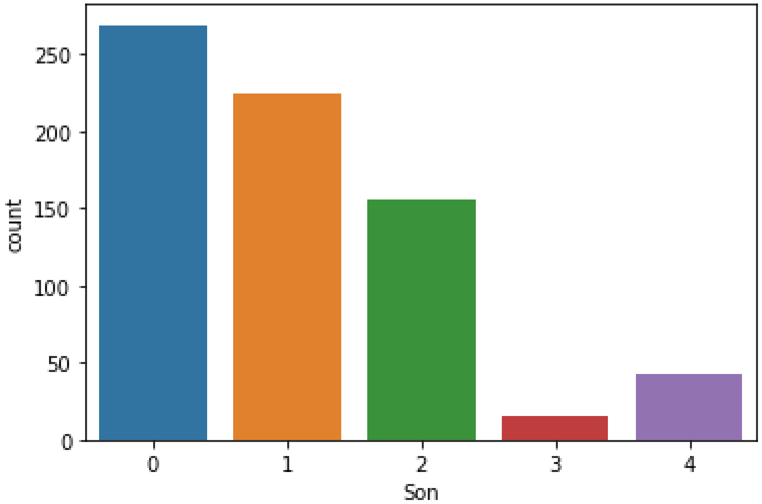
Out[98]: <AxesSubplot:xlabel='Education', ylabel='count'>



In [115…

```
# This shows that employees without son (children) seem to have the highest absenteeism
# probably due to not having deppendents like children that will saddle more responsibilities on them and
# Also, as their children increases the absenteeism dropped (influenced by more dependents)
# Therefore, employees with more son (children) have lesser absenteeism rate.
# THAT IS, ABSENTEEISM DROPPED/REDUCED AS EMPLOYEES DEPENDENT INCREASES

# HENCE, I CAN DEDUCE THAT EMPLOYEES WITH DEPENDENTS (CHILDREN) TENDS TO
# TAKE THEIR WORK MORE SERIOUSY AND ARE RESPONSIBLY

### COMPARE WITH EMPLOYEES OF YOUNGER AGE 28 (PROBABLY SINGLES WITH NO CHILDREN YET)


sb.countplot(x = "Son", data = df2)
```

Out[115…: <AxesSubplot:xlabel='Son', ylabel='count'>

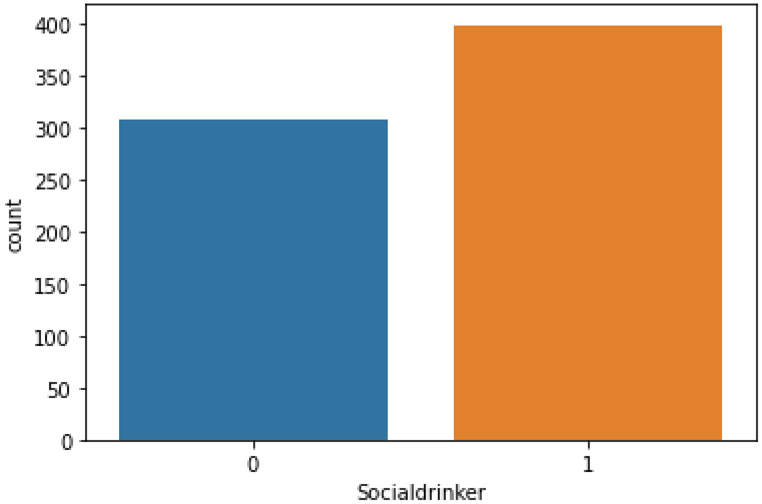


In [100…

```
# We have more employees that are social drinkers which affects absenteeism.
# Therefore, social drinker employees have higher absenteeism than the non-social drinker workers.

# HENCE, SOCIAL DRINKER EMPLOYEES WILL MOST-LIKELY HAVE HEALTH CHALLENGES AND
# THIS IS A RISK FACTOR TO HEALTH CHALLENGES THAT COULD WARRANT HIGH ABSENTEEISM



# KEYS:
# YES= 1
# NO=  0


sb.countplot(x = "Socialdrinker", data = df2)
```

Out[100…

```
<AxesSubplot:xlabel='Socialdrinker', ylabel='count'>
```

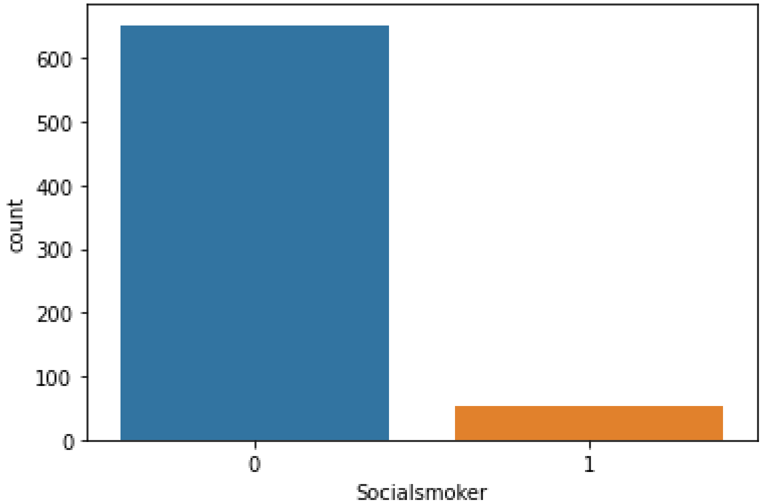


In [101…

```
# Non-social smoker employees have the highest contribution to absenteeism from
# work more often than social smokers

# HENCE, HIGHER PERCENTAGE OF THE COMPANY'S EMPLOYEES ARE NON-SOCIAL SMOKERS AND
# HENCE, MOST EMPLOYEES OF THE COMPANY ARE NON-SOCIAL SMOKER, BUT SOCIAL DRINKERS.

# KEYS:
# YES= 1
# NO=  0


sb.countplot(x = "Socialsmoker", data = df2)
```

Out[101…

```
<AxesSubplot:xlabel='Socialsmoker', ylabel='count'>
```

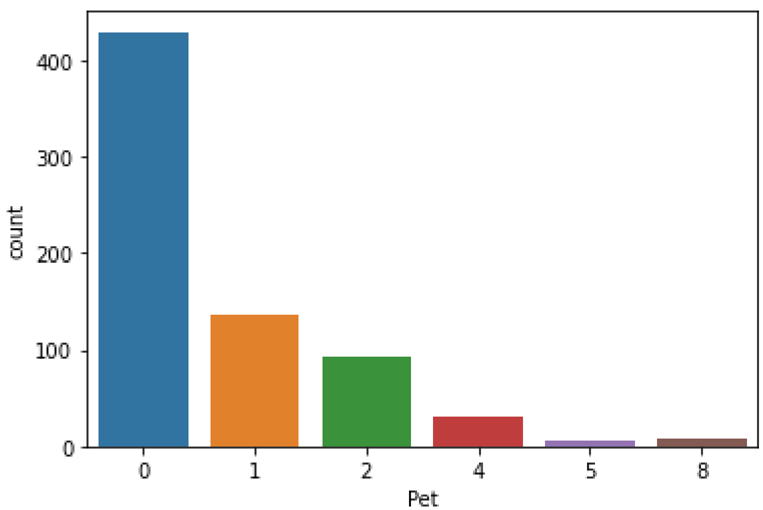


In [102…

```
# Employees without pets tends to have higher absenteeism rate
# This maybe due to lack of dependent(s) and responsibility
# THEREFORE, EMPLOYEES WITH MORE PETS HAS LOWER ABSENTEEISM RATE
# (INDIRECT RELATIONSHIP)


sb.countplot(x = "Pet", data = df2)
```

Out[102…

```
<AxesSubplot:xlabel='Pet', ylabel='count'>
```

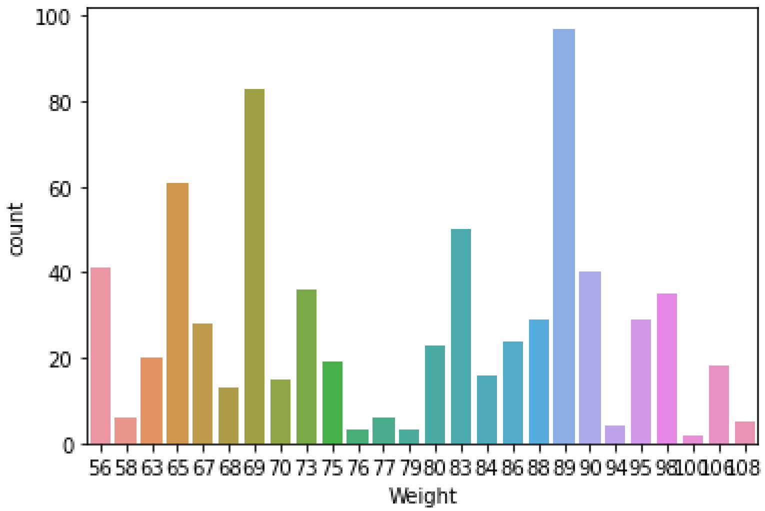


In [103…

```
# Employees with weight of 89kg have the highest absenteeism

sb.countplot(x = "Weight", data = df2)
```

Out[103…

```
<AxesSubplot:xlabel='Weight', ylabel='count'>
```

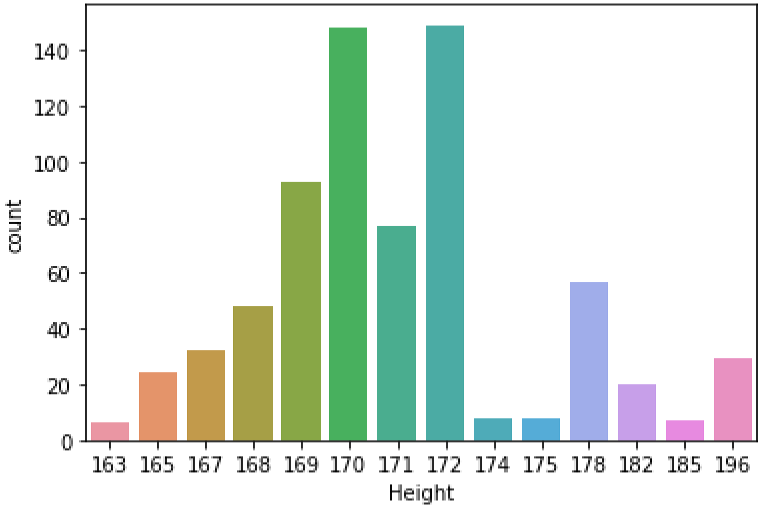


In [104…

```
# Employees with 172metres has the highest absenteeism


sb.countplot(x = "Height", data = df2)
```

Out[104…

```
<AxesSubplot:xlabel='Height', ylabel='count'>
```

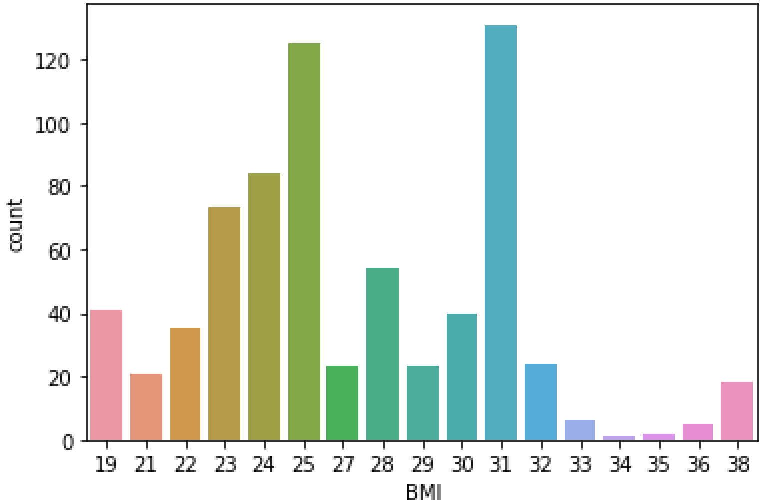


In [105…

```
# The Body Mass Index of 31 (OBESITY) has the highest rate of absenteeism and
# The higher your BMI, the higher your risk for certain diseases such as
# heart disease, high blood pressure, type 2 diabetes, gallstones,
# breathing problems, and certain cancers.

# HENCE, I CAN DEDUCE THAT EMPLOYEES WITH BMI OF 31 (OBESITY) HAS SOME HEALTH
# CHALLENGE(S) WHICH COULD HAVE INFLUENCED THEIR ABSENTEEISM RATE
# PROBABLY DUE TO MEDICAL ATTENTION OR APPOINTMRNTS.

# BMI CALCULATOR
# Underweight.... Below 18.5
# Normal..........18.5–24.9
# Overweight......25.0–29.9
# Obesity.........30.0 and Above

# CONCLUSION: HIGHER PERCENTAGE OF THE EMPLOYEES ARE OBESED AND OVER-WEIGHT AND
# THEY PRONE TO HAVE HEALTH CHALLENGES DUE TO HIGH BMI AND
# LIKELY BE MORE ABSENT  DUE TO HEALTH STATUS CAUSED BY THEIR SOCIAL DRINKING


sb.countplot(x = "BMI", data = df2)
```

Out[105…

```
<AxesSubplot:xlabel='BMI', ylabel='count'>
```

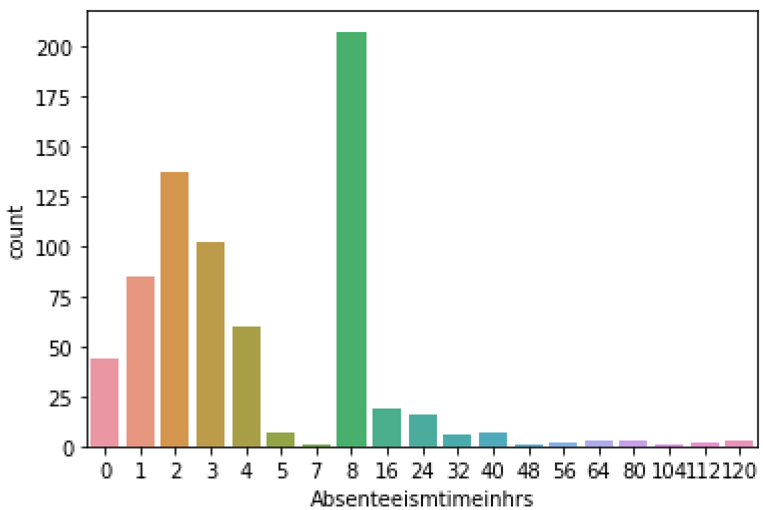


In [106…

```
# Employees with 8hours of Absenteeism time are the highest

# THIS 8HOURS COULD BE AS A RESULT OF DOCTORS APPOINTMENT(S)


sb.countplot(x = "Absenteeismtimeinhrs", data = df2)
```

Out[106…

```
<AxesSubplot:xlabel='Absenteeismtimeinhrs', ylabel='count'>
```



In [ ]:

```
# ANSWER TO QUESTION NUMBER THREE(3)
```

In [ ]:

```
# The key drivers of absenteeism in the data are:
# 1. Education
# 2. Pet number
# 3. Body Mass Index
# 4. Son (number of children)
# 5. Social drinkers
# 6. Age

# 1.EDUCATION (LOWER): Educational level contributed greatly to the employees absenteeism rate.
# Employees with High school (low) education tends to have the highest absenteeism rate while
# employees with Masters and Doctor (higher) education have the lowest/least absenteeism rate and
# this suggests that they take their work more seriously.
# Hence, the higher the employees education, the lesser/reduced the absenteeism rate.

# 2. SOCIAL DRINKER: Social drinker employees have higher absenteeism than the non-social drinker workers.
# Hence, social drinker employees will most-likely have health challenges and this is a risk
# factor to health challenges that could warrant high absenteeism.

# 3. SON (NUMBER OF CHILDREN): Employees without son (children) seem to have the highest absenteeism.
# Probably due to not having deppendents like children that will saddle more responsibilities on them and
# as their children increases the absenteeism dropped (influenced by more dependents).
# Therefore, employees with more son (children) tend to have lesser absenteeism rate.
# Absenteeism reduced as employees dependent increases.

# 4. PETS (NUMBER OF PETS): Employees without pets tends to have higher absenteeism rate, that is,
#    employees with more pets has lower absenteeism rate (indirect relationship).

# 5. BODY MASS INDEX (BMI):  The Body Mass Index of 31 (OBESITY) has the highest rate of absenteeism and
# The higher your BMI, the higher your risk for certain diseases.
# Hence, I can deduce that majority of the employees have BMI of 25 (over-weight) and 31 (obesity),
# may likely have some health challenge(s) which could have influenced their absenteeism rate.

# 6. AGE: Company's employees with age 28 have the highest absenteeism rate.
# This suggests that these are young guys in their early career and they seem
# not to really take work serious maybe due to not having dependent responsibilities.
```

In [ ]:

```
#                        CONCLUSION
# Most employees of the company are social drinker and about certain percentage are social smoker
# these habits tends to make them susceptible health challenges like:
# heart disease, high blood pressure, type 2 diabetes, gallstones, breathing problems, and certain cancers.
```

```
# this can be confirmed through the body mass index calculation as shown in
# the BMI chart analysis and the social drinker feature.

# KEYS
# BMI CALCULATOR
# Underweight.... Below 18.5
# Normal..........18.5-24.9
# Overweight......25.0-29.9
# Obesity.........30.0 and Above
# The higher the BMI, the more prone an employee to health challenges which can affect their absenteeism
# inorder to get medical attention and from the BMI chart analysis most of the employees have a high BMI.
```

In [ ]:

In [ ]:

In [ ]: