In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from IPython import get_ipython
import warnings
warnings.filterwarnings("ignore")
```

In [2]:

```
data = pd.read_csv('Fertile_Man_2020.csv')
```

In [3]:

```
data.head()
```

Out[3]:

	PI	Semen Volume (ml)	Sperm Concentration (106/ml)	Total Number (106)	Total Motility (%)	Progressive Motility (%)	Non-progressive Motility (%)	Immot Spermatoz (
0	Aboutorabi	3.2	27.0	86.4	35	20	15	
1	Aboutorabi	0.8	136.0	108.8	47	35	12	
2	Aboutorabi	2.0	71.0	142.0	49	42	7	
3	Aboutorabi	1.0	35.0	35.0	50	28	22	
4	Aboutorabi	2.0	46.0	92.0	51	28	33	

In [4]:

```
data.tail()
```

Out[4]:

	PI	Semen Volume (ml)	Sperm Concentration (106/ml)	Total Number (106)	Total Motility (%)	Progressive Motility (%)	Non-progressive Motility (%)	Immotile Spermatozoa (%)
3584	Tang	1.7	23.0	39.1	53	52	1	NO RESULT
3585	Tang	2.5	110.0	275.0	66	66	0	NO RESULT
3586	Tang	2.0	109.0	218.0	64	44	20	36
3587	Tang	6.2	96.0	595.2	39	29	10	61
3588	Tang	3.0	36.0	108.0	54	38	16	46

In [5]:

```
data.shape
```

Out[5]:

```
(3589, 10)
```

In [6]:

```
data.columns
```

Out[6]:

```
Index(['PI', 'Semen Volume (ml)', 'Sperm Concentration (106/ml)',
       'Total Number (106)', 'Total Motility (%)', 'Progressive Motility
(%)',
       'Non-progressive Motility (%)', 'Immotile Spermatozoa (%)',
       'Vitality (%)', 'Normal Forms (%)'],
      dtype='object')
```

In [7]:

```
data.duplicated().sum()
```

Out[7]:

```
220
```

In [8]:

```
data = data.drop_duplicates()
```

In [9]:

```
data.isnull().sum()
```

Out[9]:

```
PI                              0
Semen Volume (ml)               0
Sperm Concentration (106/ml)    0
Total Number (106)              0
Total Motility (%)              0
Progressive Motility (%)        0
Non-progressive Motility (%)    0
Immotile Spermatozoa (%)        0
Vitality (%)                    0
Normal Forms (%)                0
dtype: int64
```

In [10]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3369 entries, 0 to 3588
Data columns (total 10 columns):
 #   Column                          Non-Null Count  Dtype 
---  ------                          --------------  ----- 
 0   PI                              3369 non-null   object
 1   Semen Volume (ml)               3369 non-null   object
 2   Sperm Concentration (106/ml)    3369 non-null   object
 3   Total Number (106)              3369 non-null   object
 4   Total Motility (%)              3369 non-null   object
 5   Progressive Motility (%)        3369 non-null   object
 6   Non-progressive Motility (%)    3369 non-null   object
 7   Immotile Spermatozoa (%)        3369 non-null   object
 8   Vitality (%)                    3369 non-null   object
 9   Normal Forms (%)                3369 non-null   object
dtypes: object(10)
memory usage: 289.5+ KB
```

In [11]:

```
data.nunique()
```

Out[11]:

```
PI                               10
Semen Volume (ml)                97
Sperm Concentration (106/ml)    644
Total Number (106)             1652
Total Motility (%)               88
Progressive Motility (%)         91
Non-progressive Motility (%)     52
Immotile Spermatozoa (%)         85
Vitality (%)                     66
Normal Forms (%)                 67
dtype: int64
```

In [12]:

```
data['PI'].unique()
```

Out[12]:

```
array(['Aboutorabi', 'Auger', 'Baker', 'Evgeni', 'Haugen', 'Jensen',
       'Lotti', 'Swan', 'Zedan', 'Tang'], dtype=object)
```

In [13]:

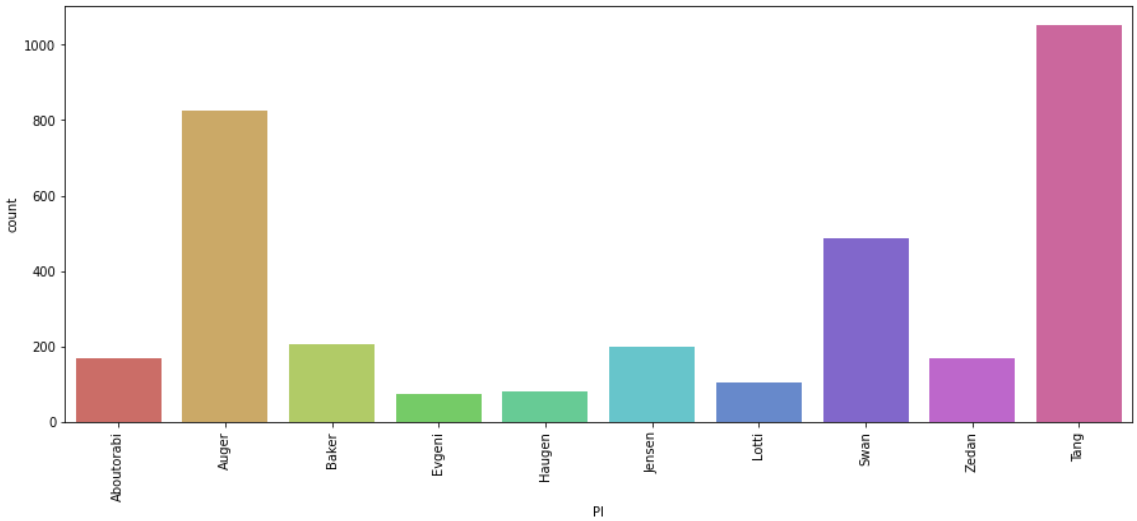
```
data['PI'].value_counts()
```

Out[13]:

```
Tang          1050
Auger          826
Swan           487
Baker          206
Jensen         199
Zedan          170
Aboutorabi     168
Lotti          105
Haugen          82
Evgeni          76
Name: PI, dtype: int64
```

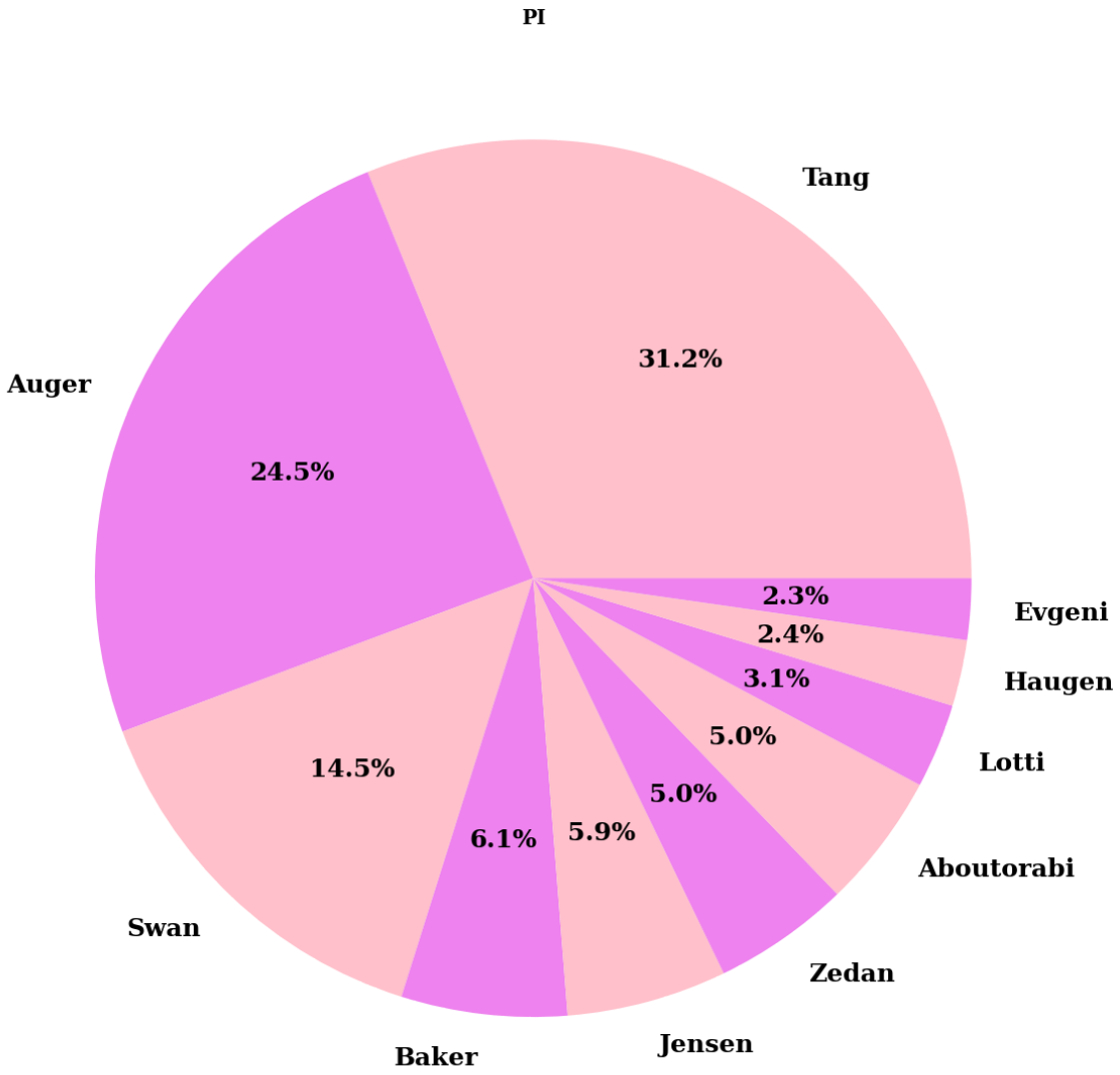
In [14]:

```
plt.figure(figsize=(15,6))
sns.countplot('PI', data = data, palette = 'hls')
plt.xticks(rotation = 90)
plt.show()
```



In [15]:

```
plt.figure(figsize=(30,20))
plt.pie(data['PI'].value_counts(), labels=data['PI'].value_counts().index,
        colors = ['pink', 'violet'], autopct='%1.1f%%', textprops={ 'fontsize': 25,
                                            'color': 'black',
                                            'weight': 'bold',
                                            'family': 'serif' })
hfont = {'fontname':'serif', 'weight': 'bold'}
plt.title('PI', size=20, **hfont)
plt.show()
```



In [16]:

```
data_new = data[['Semen Volume (ml)', 'Sperm Concentration (106/ml)',
       'Total Number (106)', 'Total Motility (%)', 'Progressive Motility (%)',
       'Non-progressive Motility (%)', 'Immotile Spermatozoa (%)',
       'Vitality (%)', 'Normal Forms (%)']]
```

In [17]:

```
data_new = data_new.replace({'Semen Volume (ml)': {'NO RESULT': 0}})
data_new = data_new.replace({'Sperm Concentration (106/ml)': {'NO RESULT': 0}})
data_new = data_new.replace({'Total Number (106)': {'NO RESULT': 0}})
data_new = data_new.replace({'Total Motility (%)': {'NO RESULT': 0}})
data_new = data_new.replace({'Progressive Motility (%)': {'NO RESULT': 0}})
data_new = data_new.replace({'Non-progressive Motility (%)': {'NO RESULT': 0}})
data_new = data_new.replace({'Immotile Spermatozoa (%)': {'NO RESULT': 0}})
data_new = data_new.replace({'Normal Forms (%)': {'NO RESULT': 0}})
data_new = data_new.replace({'Vitality (%)': {'NO RESULT': 0}})
```

In [18]:

```
data_new
```

Out[18]:

	Semen Volume (ml)	Sperm Concentration (106/ml)	Total Number (106)	Total Motility (%)	Progressive Motility (%)	Non-progressive Motility (%)	Immotile Spermatozoa (%)	Vital (
0	3.2	27.0	86.4	35	20	15	65	
1	0.8	136.0	108.8	47	35	12	53	
2	2.0	71.0	142.0	49	42	7	51	
3	1.0	35.0	35.0	50	28	22	50	
4	2.0	46.0	92.0	51	28	33	49	
...	...	...	...	...	...	...	...	
3579	2.0	115.0	230.0	79	77	2	0	
3581	4.0	22.0	88.0	35	32	3	0	
3586	2.0	109.0	218.0	64	44	20	36	
3587	6.2	96.0	595.2	39	29	10	61	
3588	3.0	36.0	108.0	54	38	16	46	

3369 rows × 9 columns

In [19]:

```
for i in data_new.columns:
    data_new[i] = data_new[i].astype(float)
```

In [20]:

```
data_new.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3369 entries, 0 to 3588
Data columns (total 9 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   Semen Volume (ml)             3369 non-null   float64
 1   Sperm Concentration (106/ml)  3369 non-null   float64
 2   Total Number (106)            3369 non-null   float64
 3   Total Motility (%)            3369 non-null   float64
 4   Progressive Motility (%)      3369 non-null   float64
 5   Non-progressive Motility (%)  3369 non-null   float64
 6   Immotile Spermatozoa (%)      3369 non-null   float64
 7   Vitality (%)                  3369 non-null   float64
 8   Normal Forms (%)              3369 non-null   float64
dtypes: float64(9)
memory usage: 263.2 KB
```

In [21]:

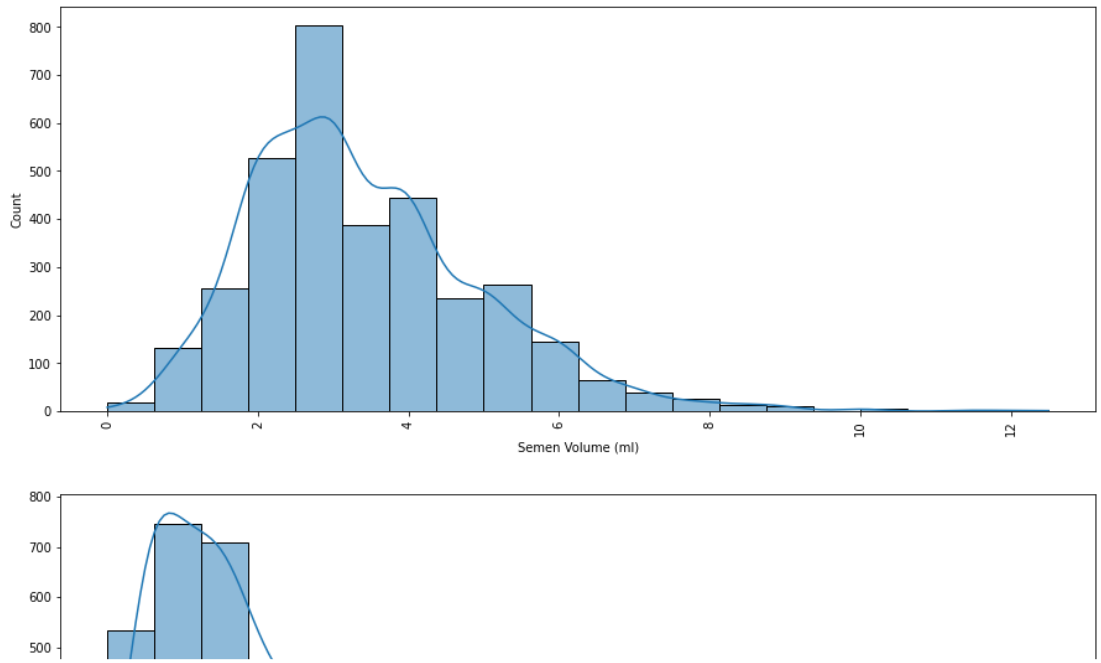
```
data_new.describe()
```

Out[21]:

	Semen Volume (ml)	Sperm Concentration (106/ml)	Total Number (106)	Total Motility (%)	Progressive Motility (%)	Non-progressive Motility (%)	Spe
count	3369.000000	3369.000000	3369.000000	3369.000000	3369.000000	3369.000000	336
mean	3.431849	83.989849	274.623390	62.426239	50.285841	10.602553	3
std	1.555602	65.363518	235.739172	17.721234	18.633336	9.861524	1
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	2.300000	37.000000	111.300000	55.000000	43.000000	3.000000	2
50%	3.100000	68.000000	214.000000	64.000000	53.000000	8.000000	3
75%	4.200000	112.000000	370.000000	73.000000	62.000000	15.000000	4
max	12.500000	532.000000	3115.200000	100.000000	94.000000	56.000000	10

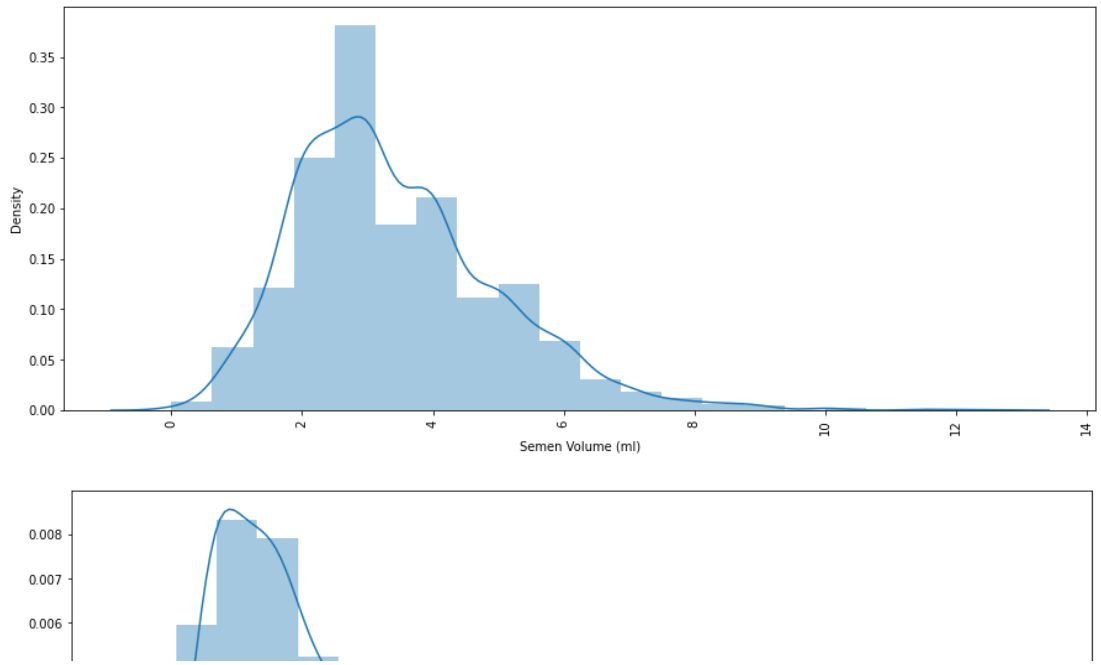
In [22]:

```
for i in data_new.columns:
    plt.figure(figsize=(15,6))
    sns.histplot(data_new[i], bins = 20, kde = True, palette = 'hls')
    plt.xticks(rotation = 90)
    plt.show()
```



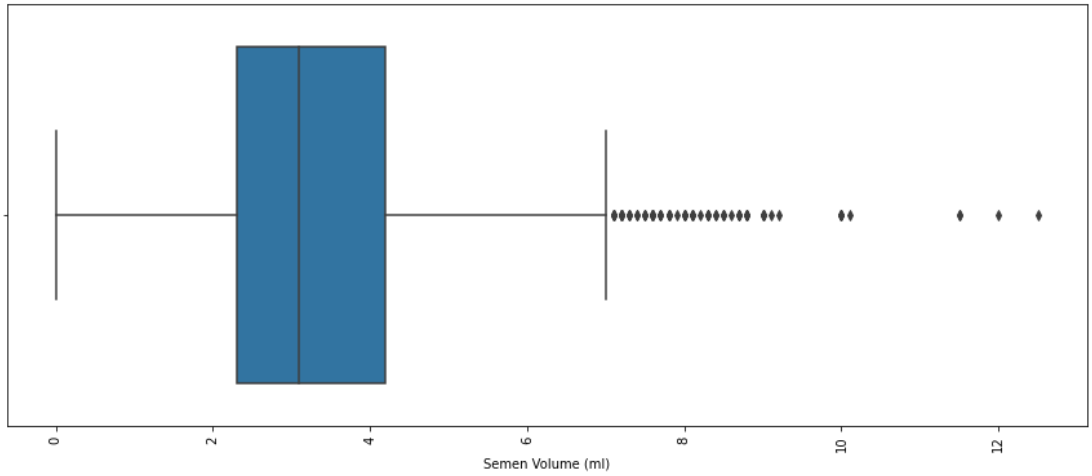
In [23]:

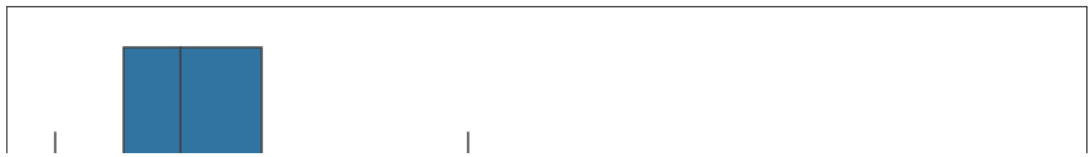
```
for i in data_new.columns:
    plt.figure(figsize=(15,6))
    sns.distplot(data_new[i], bins = 20, kde = True)
    plt.xticks(rotation = 90)
    plt.show()
```



In [24]:

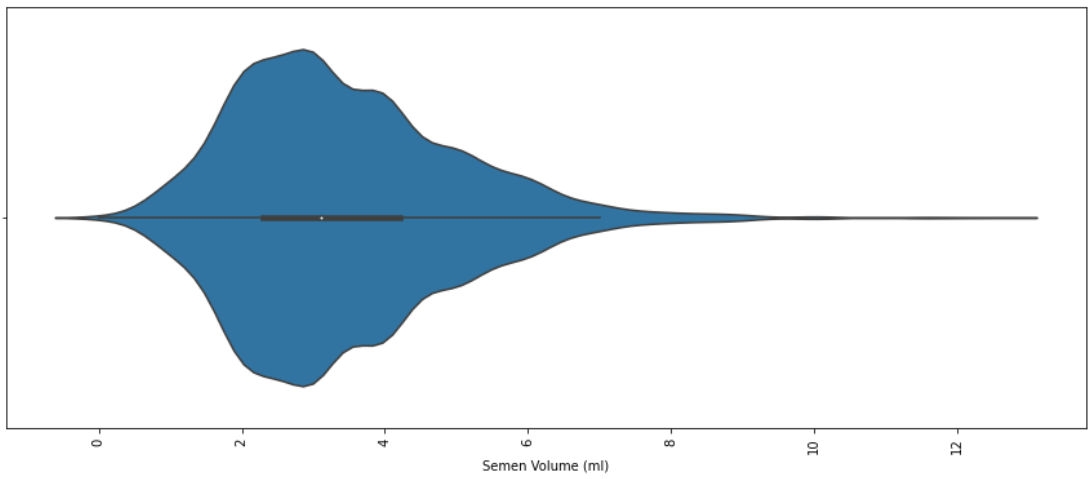
```
for i in data_new.columns:
    plt.figure(figsize=(15,6))
    sns.boxplot(data_new[i])
    plt.xticks(rotation = 90)
    plt.show()
```





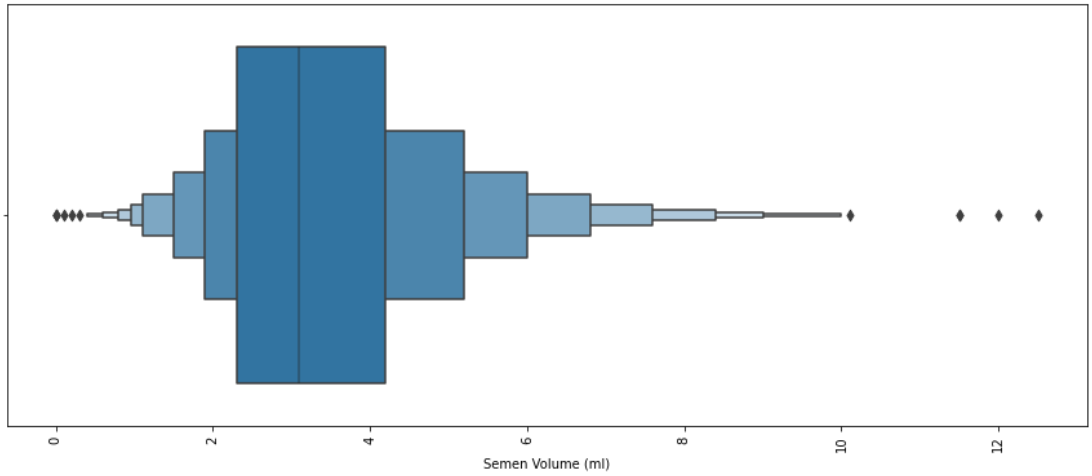
In [25]:

```
for i in data_new.columns:
    plt.figure(figsize=(15,6))
    sns.violinplot(data_new[i])
    plt.xticks(rotation = 90)
    plt.show()
```



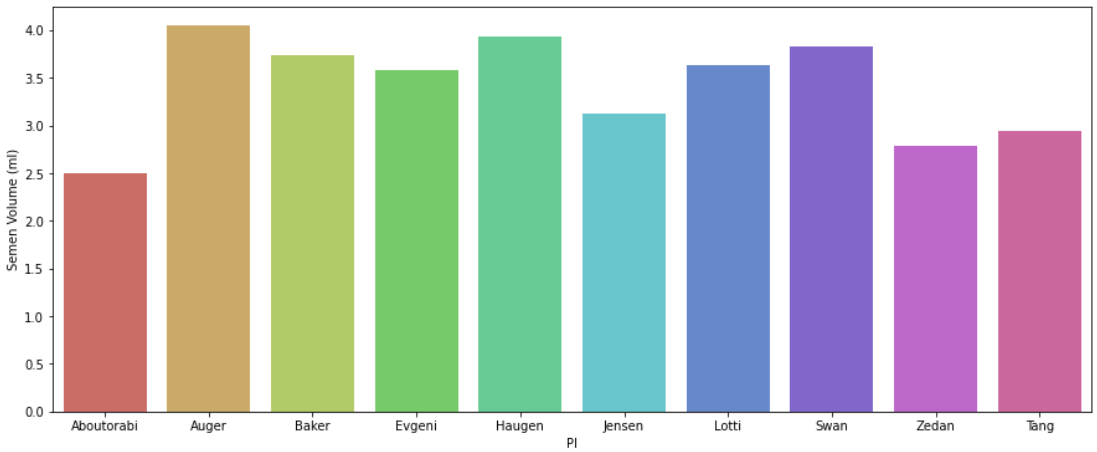
In [26]:

```
for i in data_new.columns:
    plt.figure(figsize=(15,6))
    sns.boxenplot(data_new[i])
    plt.xticks(rotation = 90)
    plt.show()
```



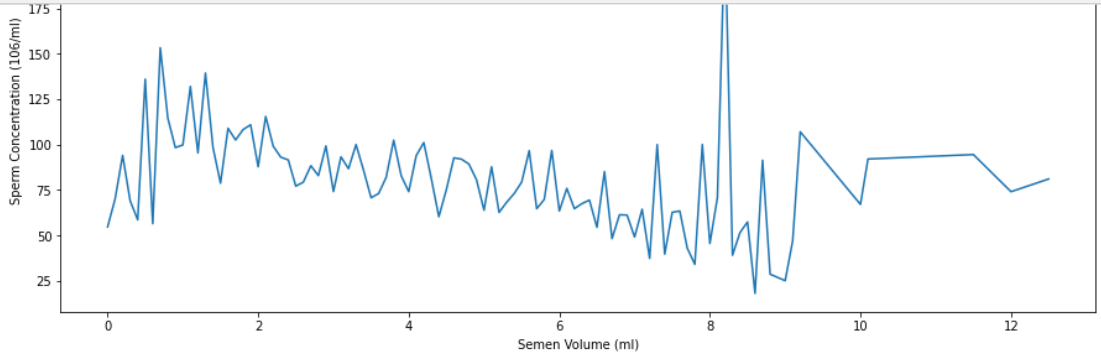
In [27]:

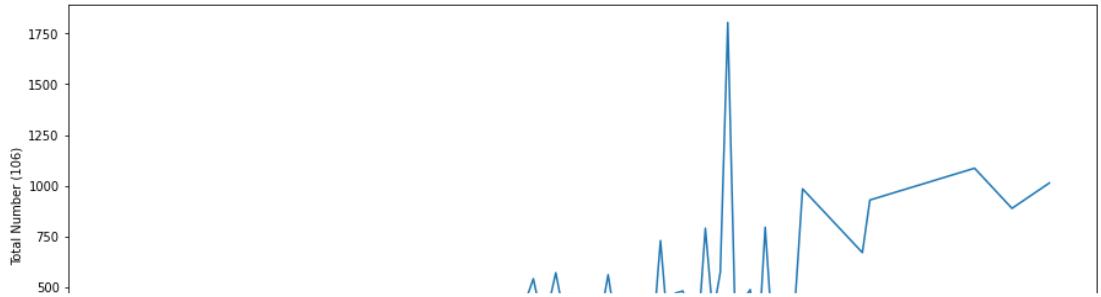
```
for i in data_new.columns:
    plt.figure(figsize=(15,6))
    sns.barplot(x = data['PI'], y = data_new[i], ci = None, palette = 'hls')
    plt.show()
```



In [28]:

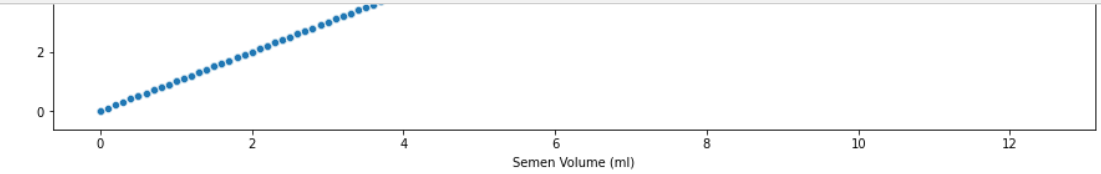
```
for j in data_new.columns:
    for i in data_new.columns:
        plt.figure(figsize=(15,6))
        sns.lineplot(x = data_new[j], y = data_new[i], ci = None, palette = 'hls')
        plt.show()
```

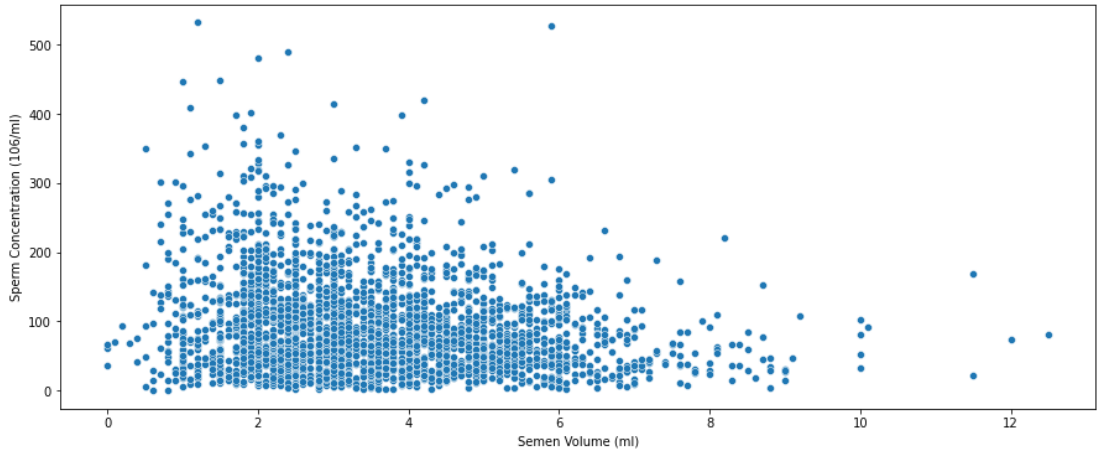




In [29]:

```
for j in data_new.columns:
    for i in data_new.columns:
        plt.figure(figsize=(15,6))
        sns.scatterplot(x = data_new[j], y = data_new[i], ci = None, palette = 'hls')
        plt.show()
```





In [30]:

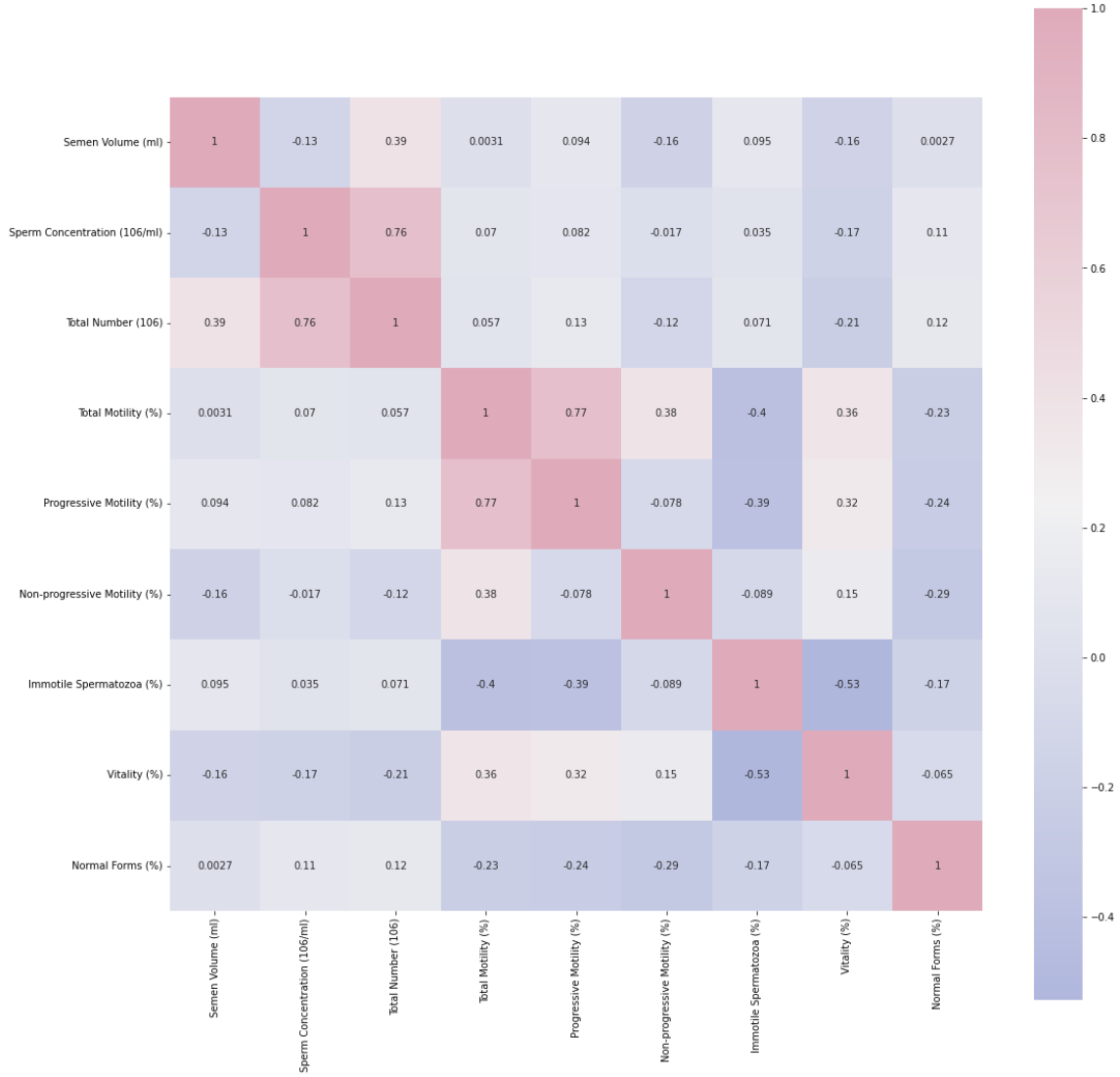
```
corrmat = data_new.corr()
corrmat
```

Out[30]:

	Semen Volume (ml)	Sperm Concentration (106/ml)	Total Number (106)	Total Motility (%)	Progressive Motility (%)	Non-progressive Motility (%)	Sper
Semen Volume (ml)	1.000000	-0.132747	0.394834	0.003056	0.093729	-0.163921	
Sperm Concentration (106/ml)	-0.132747	1.000000	0.762657	0.070431	0.082049	-0.016940	
Total Number (106)	0.394834	0.762657	1.000000	0.056822	0.127535	-0.116299	
Total Motility (%)	0.003056	0.070431	0.056822	1.000000	0.772534	0.377100	-
Progressive Motility (%)	0.093729	0.082049	0.127535	0.772534	1.000000	-0.077893	-
Non-progressive Motility (%)	-0.163921	-0.016940	-0.116299	0.377100	-0.077893	1.000000	-
Immotile Spermatozoa (%)	0.095168	0.034547	0.070889	-0.400565	-0.388460	-0.089031	
Vitality (%)	-0.156211	-0.170896	-0.210887	0.364078	0.324613	0.148656	-
Normal Forms (%)	0.002695	0.109208	0.117101	-0.226283	-0.237217	-0.290616	-

In [31]:

```
cmap = sns.diverging_palette(260,-10,s=50, l=75, n=6,
                             as_cmap=True)
plt.subplots(figsize=(18,18))
sns.heatmap(corrmat,cmap= cmap,annot=True, square=True)
plt.show()
```



In [32]:

```
Q1 = data_new.quantile(0.25)
Q3 = data_new.quantile(0.75)
IQR = Q3 - Q1
print(IQR)
```

```
Semen Volume (ml)                 1.9
Sperm Concentration (106/ml)     75.0
Total Number (106)              258.7
Total Motility (%)               18.0
Progressive Motility (%)         19.0
Non-progressive Motility (%)     12.0
Immotile Spermatozoa (%)         22.0
Vitality (%)                     69.0
Normal Forms (%)                 14.0
dtype: float64
```

In [33]:

```
X = data_new.drop('Total Motility (%)', axis=1)
y = data_new['Total Motility (%)']
```

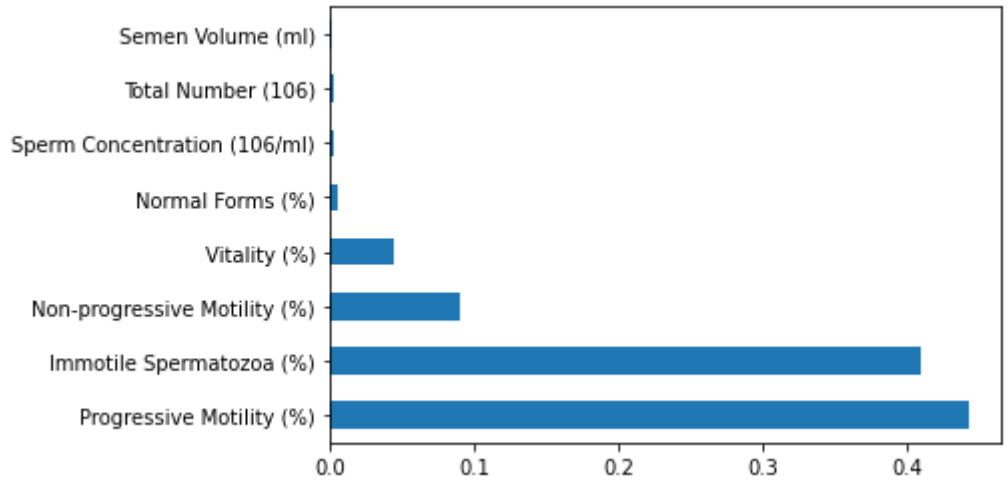
In [34]:

```
from sklearn.ensemble import ExtraTreesRegressor
model = ExtraTreesRegressor()
model.fit(X,y)
print(model.feature_importances_)
```

```
[0.00205569 0.00254137 0.0024164  0.44256015 0.09026303 0.40957871
 0.04526748 0.00531717]
```

In [35]:

```
feat_importances = pd.Series(model.feature_importances_, index=X.columns)
feat_importances.nlargest(8).plot(kind='barh')
plt.show()
```



In [36]:

```
X.columns
```

Out[36]:

```
Index(['Semen Volume (ml)', 'Sperm Concentration (106/ml)',
       'Total Number (106)', 'Progressive Motility (%)',
       'Non-progressive Motility (%)', 'Immotile Spermatozoa (%)',
       'Vitality (%)', 'Normal Forms (%)'],
      dtype='object')
```

In [37]:

```
top_8 = pd.DataFrame({'Feature Importance': feat_importances.nlargest(8)})
```

In [38]:

```
top_8
```

Out[38]:

	Feature Importance
Progressive Motility (%)	0.442560
Immotile Spermatozoa (%)	0.409579
Non-progressive Motility (%)	0.090263
Vitality (%)	0.045267
Normal Forms (%)	0.005317
Sperm Concentration (106/ml)	0.002541
Total Number (106)	0.002416
Semen Volume (ml)	0.002056

In [39]:

```
X = X[['Vitality (%)', 'Non-progressive Motility (%)','Immotile Spermatozoa (%)',
       'Progressive Motility (%)']]
```

In [40]:

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
```

In [41]:

```
X = scaler.fit_transform(X)
```

In [42]:

```
from sklearn.model_selection import train_test_split
```

In [43]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size= 0.25,
                                                    random_state= 42)
```

In [44]:

```
from sklearn.metrics import mean_squared_error, r2_score
```

In [45]:

```
from sklearn.linear_model import LinearRegression
```

In [46]:

```
reg = LinearRegression()
reg.fit(X_train, y_train)
```

Out[46]:

```
LinearRegression()
```

In [47]:

```
y_pred = reg.predict(X_test)
```

In [48]:

```
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
print(f"R-squared score: {r2}")
```

```
Mean Squared Error: 72.60078169753363
R-squared score: 0.7618978062608279
```

In [49]:

```
from sklearn.tree import DecisionTreeRegressor
```

In [50]:

```
tree_reg = DecisionTreeRegressor()
```

In [51]:

```
tree_reg.fit(X_train, y_train)
```

Out[51]:

```
DecisionTreeRegressor()
```

In [52]:

```
y_pred = tree_reg.predict(X_test)
```

In [53]:

```
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"R-squared score: {r2}")
```

```
Mean Squared Error: 10.876631079478054
R-squared score: 0.9643289003236261
```

In [54]:

```
from sklearn.ensemble import RandomForestRegressor
```

In [55]:

```
rf_reg = RandomForestRegressor()
```

In [56]:

```
rf_reg.fit(X_train, y_train)
```

Out[56]:

RandomForestRegressor

RandomForestRegressor()

In [57]:

```
y_pred = rf_reg.predict(X_test)
```

In [58]:

```
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"R-squared score: {r2}")
```

```
Mean Squared Error: 8.656957324684004
R-squared score: 0.971608562856787
```