Importing Important Libraries

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

Load the dataset

In [2]:
```
df=pd.read_csv('thyroid_disease.csv')
df
```

Out[2]:

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_meds	sick	pregnant	thyroid_surgery	I131_treatment	query_hypothyroid	...	TT4	T4U_mea
0	29	F	f	f	f	f	f	f	f	t	...	NaN	
1	29	F	f	f	f	f	f	f	f	f	...	128.0	
2	41	F	f	f	f	f	f	f	f	f	...	NaN	
3	36	F	f	f	f	f	f	f	f	f	...	NaN	
4	32	F	f	f	f	f	f	f	f	f	...	NaN	
...	...	...	...	...	...	...	...	...	...	...	...	...	
9167	56	M	f	f	f	f	f	f	f	f	...	64.0	
9168	22	M	f	f	f	f	f	f	f	f	...	91.0	
9169	69	M	f	f	f	f	f	f	f	f	...	113.0	
9170	47	F	f	f	f	f	f	f	f	f	...	75.0	
9171	31	M	f	f	f	f	f	f	f	t	...	66.0	

9172 rows × 31 columns

dropping redundant attributes from thyroidDF dataset

In [3]:
```
df.drop(['TSH_measured', 'T3_measured', 'TT4_measured', 'T4U_measured', 'FTI_measured', 'TBG_measured',
                'patient_id', 'referral_source'], axis=1, inplace=True)
```

In [4]:
```
df
```

Out[4]:

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_meds	sick	pregnant	thyroid_surgery	I131_treatment	query_hypothyroid	...	tumor	hypopitu
0	29	F	f	f	f	f	f	f	f	t	...	f	
1	29	F	f	f	f	f	f	f	f	f	...	f	
2	41	F	f	f	f	f	f	f	f	f	...	f	
3	36	F	f	f	f	f	f	f	f	f	...	f	
4	32	F	f	f	f	f	f	f	f	f	...	f	
...	...	...	...	...	...	...	...	...	...	...	...	...	
9167	56	M	f	f	f	f	f	f	f	f	...	f	
9168	22	M	f	f	f	f	f	f	f	f	...	f	
9169	69	M	f	f	f	f	f	f	f	f	...	f	
9170	47	F	f	f	f	f	f	f	f	f	...	f	
9171	31	M	f	f	f	f	f	f	f	t	...	f	

9172 rows × 23 columns

Cheking Null Values in Data set

In [5]:
```
df.isnull().sum()
```

Out[5]:
```
age                     0
sex                   307
on_thyroxine            0
query_on_thyroxine      0
on_antithyroid_meds     0
sick                    0
pregnant                0
thyroid_surgery         0
I131_treatment          0
query_hypothyroid       0
query_hyperthyroid      0
lithium                 0
goitre                  0
tumor                   0
hypopituitary           0
psych                   0
TSH                   842
T3                   2604
TT4                   442
T4U                   809
FTI                   802
TBG                  8823
target                  0
dtype: int64
```

In [6]:
```
df.shape
```

Out[6]:
```
(9172, 23)
```

In [7]:
```
df.columns
```

Out[7]:
```
Index(['age', 'sex', 'on_thyroxine', 'query_on_thyroxine',
       'on_antithyroid_meds', 'sick', 'pregnant', 'thyroid_surgery',
       'I131_treatment', 'query_hypothyroid', 'query_hyperthyroid', 'lithium',
       'goitre', 'tumor', 'hypopituitary', 'psych', 'TSH', 'T3', 'TT4', 'T4U',
       'FTI', 'TBG', 'target'],
      dtype='object')
```

In [8]:
```
df['target'].unique()
```

Out[8]:
```
array(['-', 'S', 'F', 'AK', 'R', 'I', 'M', 'N', 'G', 'K', 'A', 'KJ', 'L',
       'MK', 'Q', 'J', 'C|I', 'O', 'LJ', 'H|K', 'D', 'GK', 'MI', 'P',
       'FK', 'B', 'GI', 'C', 'GKJ', 'OI', 'D|R', 'E'], dtype=object)
```

re-mapping target vaues to diagnostic groups

In [9]:
```

diagnoses = {'-': 'negative',
             'A': 'hyperthyroid',
             'B': 'hyperthyroid',
             'C': 'hyperthyroid',
             'D': 'hyperthyroid',
             'E': 'hypothyroid',
             'F': 'hypothyroid',
             'G': 'hypothyroid',
             'H': 'hypothyroid'}
df['target'] = df['target'].map(diagnoses) # re-mapping
# dropping observations with 'target' null after re-mapping
df.dropna(subset=['target'], inplace=True)
```

dataset initial summary

In [10]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7546 entries, 0 to 9171
Data columns (total 23 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   age                  7546 non-null   int64  
 1   sex                  7296 non-null   object 
 2   on_thyroxine         7546 non-null   object 
 3   query_on_thyroxine   7546 non-null   object 
 4   on_antithyroid_meds  7546 non-null   object 
 5   sick                 7546 non-null   object 
 6   pregnant             7546 non-null   object 
 7   thyroid_surgery      7546 non-null   object 
 8   I131_treatment       7546 non-null   object 
 9   query_hypothyroid    7546 non-null   object 
 10  query_hyperthyroid   7546 non-null   object 
 11  lithium              7546 non-null   object 
 12  goitre               7546 non-null   object 
 13  tumor                7546 non-null   object 
 14  hypopituitary        7546 non-null   object 
 15  psych                7546 non-null   object 
 16  TSH                  6824 non-null   float64
 17  T3                   5337 non-null   float64
 18  TT4                  7192 non-null   float64
 19  T4U                  6870 non-null   float64
 20  FTI                  6877 non-null   float64
 21  TBG                  259 non-null    float64
 22  target               7546 non-null   object 
dtypes: float64(6), int64(1), object(16)
memory usage: 1.4+ MB
```

distributions of numeric variables

In [11]:

```
df.describe()
```

Out[11]:

	age	TSH	T3	TT4	T4U	FTI	TBG
count	7546.000000	6824.000000	5337.000000	7192.000000	6870.000000	6877.000000	259.000000
mean	78.013782	5.421753	2.020935	105.203373	0.967322	110.571745	22.955019
std	1305.258137	26.080471	0.809865	32.606462	0.162315	36.600867	6.088392
min	1.000000	0.005000	0.050000	2.000000	0.190000	1.400000	0.100000
25%	37.000000	0.570000	1.600000	87.000000	0.870000	93.000000	20.000000
50%	55.000000	1.400000	2.000000	103.000000	0.960000	108.000000	23.000000
75%	67.000000	2.700000	2.300000	121.000000	1.060000	125.000000	27.000000
max	65526.000000	530.000000	18.000000	430.000000	2.120000	839.000000	45.000000

Observations:

--> Max age value is 65,526 years old. Either that person is really really old or this is a mistake. There are likely more inconsistencies like this one throughout the data. --> Persons with age above 100 will be removed since they have target negative and we wont be losing to much information by omitting them

inspecting observations with age > 100

In [12]:

```
df[df.age > 100]
```

Out[12]:

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_meds	sick	pregnant	thyroid_surgery	I131_treatment	query_hypothyroid	...	tumor	hypop
2976	455	F	f	f	f	f	f	f	f	f	...	f	
5710	65511	M	f	f	f	f	f	f	f	f	...	f	
6392	65512	M	f	f	f	f	f	f	f	f	...	f	
8105	65526	F	f	f	f	f	f	f	f	f	...	f	

4 rows × 23 columns

changing age of observations with ('age' > 100) to null

In [13]:
```
df['age'] = np.where((df.age > 100), np.nan, df.age)
```

In [14]:
```
df['target'].value_counts()
```

Out[14]:
```
negative        6771
hypothyroid      593
hyperthyroid     182
Name: target, dtype: int64
```

Exploratory Data Analysis

In [15]:
```
# ---> We begin our EDA by looking at the distribution of Hormone levels in blood for each of our target classes.
# ---> This helps us get an idea for how good of a predictor each of these attributes can be.
```

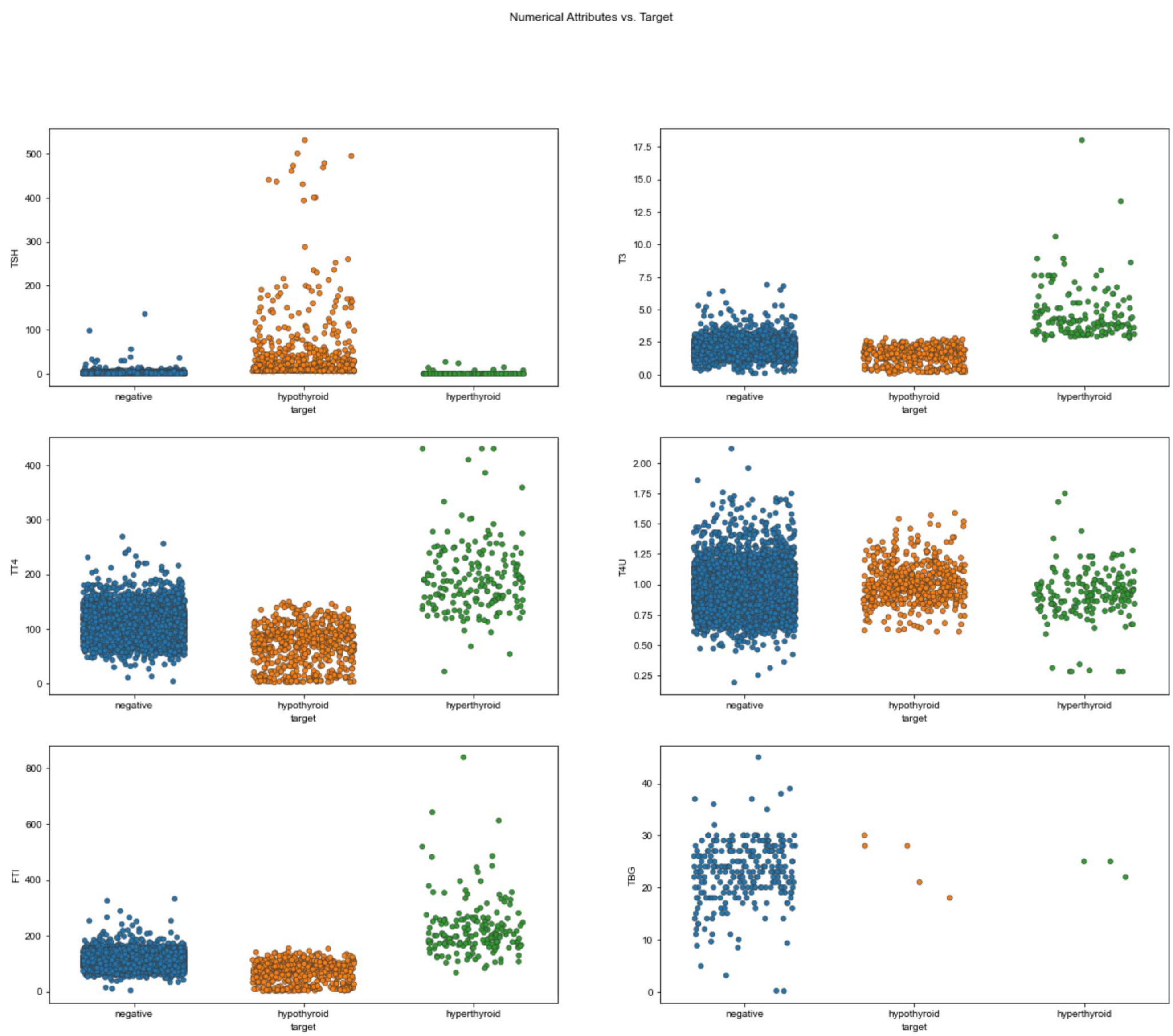
setting up grid for multiple seaborn plots

In [16]:

```
fig, axes = plt.subplots(3,2,figsize=(20,16))
fig.suptitle('Numerical Attributes vs. Target')
sns.set_style('whitegrid');

# TSH vs. 'target'
sns.stripplot(x=df.target, y=df.TSH, linewidth=0.6 , jitter= 0.3,ax=axes[0, 0])

# T3 vs. 'target'
sns.stripplot(x=df.target, y=df.T3, linewidth=0.6, jitter= 0.3, ax=axes[0, 1])

# TT4 vs. 'target'
sns.stripplot(x=df.target, y=df.TT4, linewidth=0.6, jitter= 0.3, ax=axes[1, 0])

# T4U vs. 'target'
sns.stripplot(x=df.target, y=df.T4U, linewidth=0.6, jitter= 0.3, ax=axes[1, 1])

# FTI vs. 'target'
sns.stripplot(x=df.target, y=df.FTI, linewidth=0.6, jitter= 0.3, ax=axes[2, 0])

# TBG vs. 'target'
sns.stripplot(x=df.target, y=df.TBG, linewidth=0.6, jitter= 0.3, ax=axes[2, 1])
```

Out[16]:

```
<AxesSubplot:xlabel='target', ylabel='TBG'>
```



Observations:

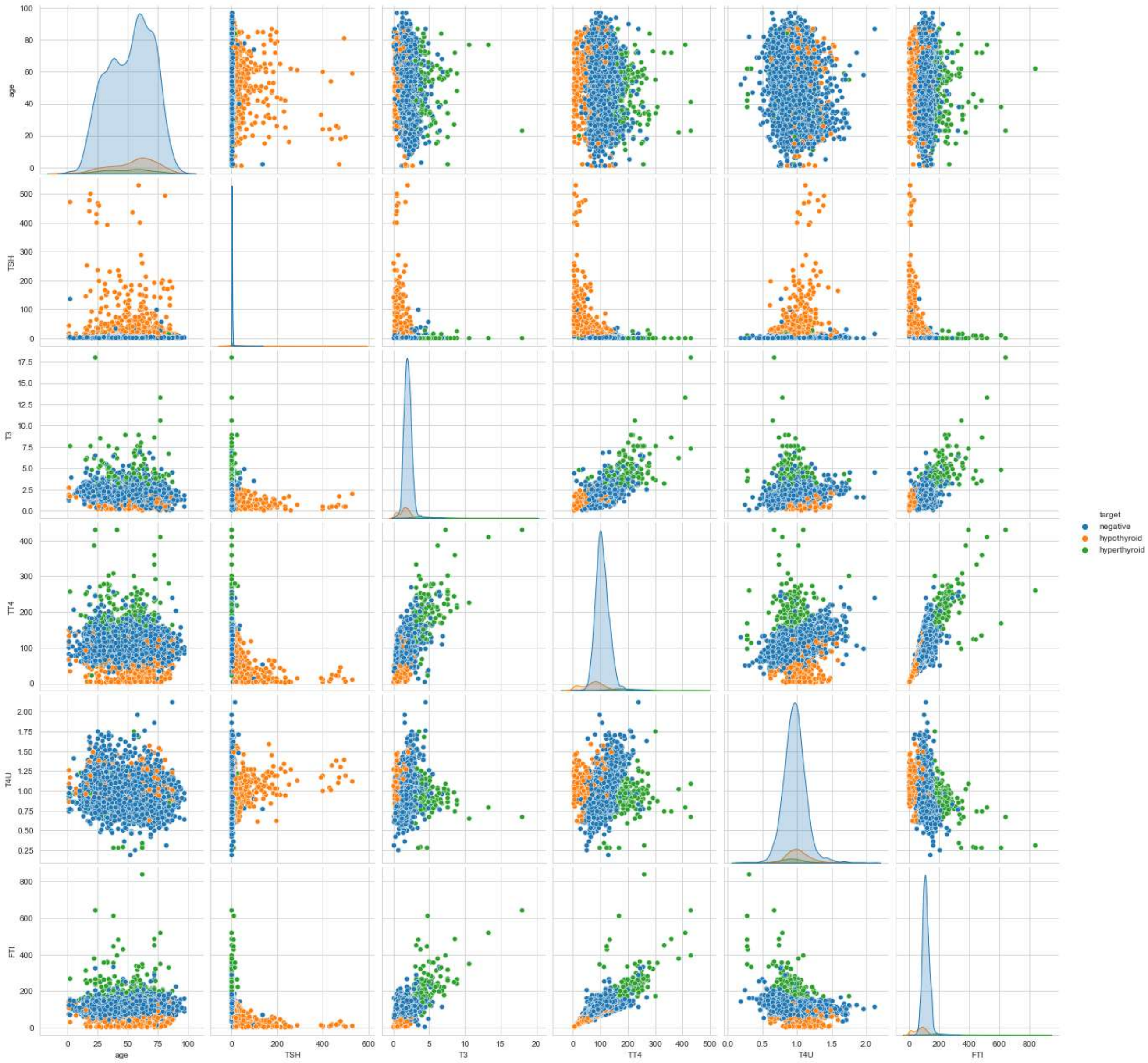
In [17]:

```
# ---> Immediately we can hypothesize that FTI, T3, and TT4 will be good feature additions to our models.
#      TSH looks like it might be good as well but we need to handle the outliers for 'target' hypo and analyze the attribute dis
# ---> This is all in-line with the knowledge discovered about Hormone level tests during our initial research
```

Let's continue by creating a pairplot of our numeric variables and seeing if we can spot any clusters forming between variables.

In [18]:

```
numericalDF = df[['age', 'TSH', 'T3', 'TT4', 'T4U', 'FTI', 'target']].copy()
sns.set_style('whitegrid');
sns.pairplot(numericalDF, hue='target', height=3);
plt.show()
```



In the diagonals of the pairplot we can see the distributions of each numeric variable with respect to one another. It is apparent how unbalanced the dataset is, with so many negative 'target' compared to hypothyroid or hyperthyroid.

Observations:

We can see that for some Hormone test vs others there are nice clusters that form. This is encouragin because it means that they do a good job at separating out each of our target classes. FTI vs T3 FTI vs T4U FTI vs age T4U vs TT4 TT4 vs age TT4 vs T3 There is severe target class imbalance... but we knew this from the start. It is normal for this type of data. We will have to treat with resampling protocol as well as using models that handle this well.

Investigating Feature Correlations

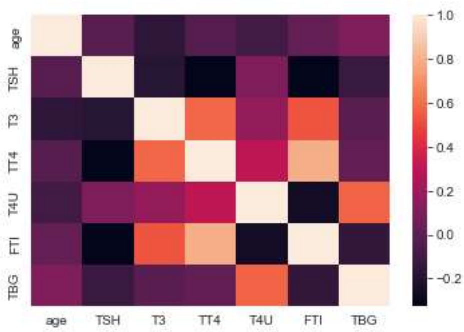
In [19]:
```
# Now lets give some  attributes and take a look at the correlation
# between  of our  numerical attributes between one another.
```

In [20]:
```
df_mat = df.corr()
round(df_mat,2)
```

Out[20]:

	age	TSH	T3	TT4	T4U	FTI	TBG
age	1.00	-0.02	-0.16	-0.03	-0.09	0.02	0.10
TSH	-0.02	1.00	-0.19	-0.32	0.10	-0.32	-0.12
T3	-0.16	-0.19	1.00	0.58	0.17	0.52	-0.01
TT4	-0.03	-0.32	0.58	1.00	0.30	0.79	0.01
T4U	-0.09	0.10	0.17	0.30	1.00	-0.26	0.57
FTI	0.02	-0.32	0.52	0.79	-0.26	1.00	-0.15
TBG	0.10	-0.12	-0.01	0.01	0.57	-0.15	1.00

In [21]:
```
sns.heatmap(df_mat);
```



Handling Inconsistencies

Investigating Outliers

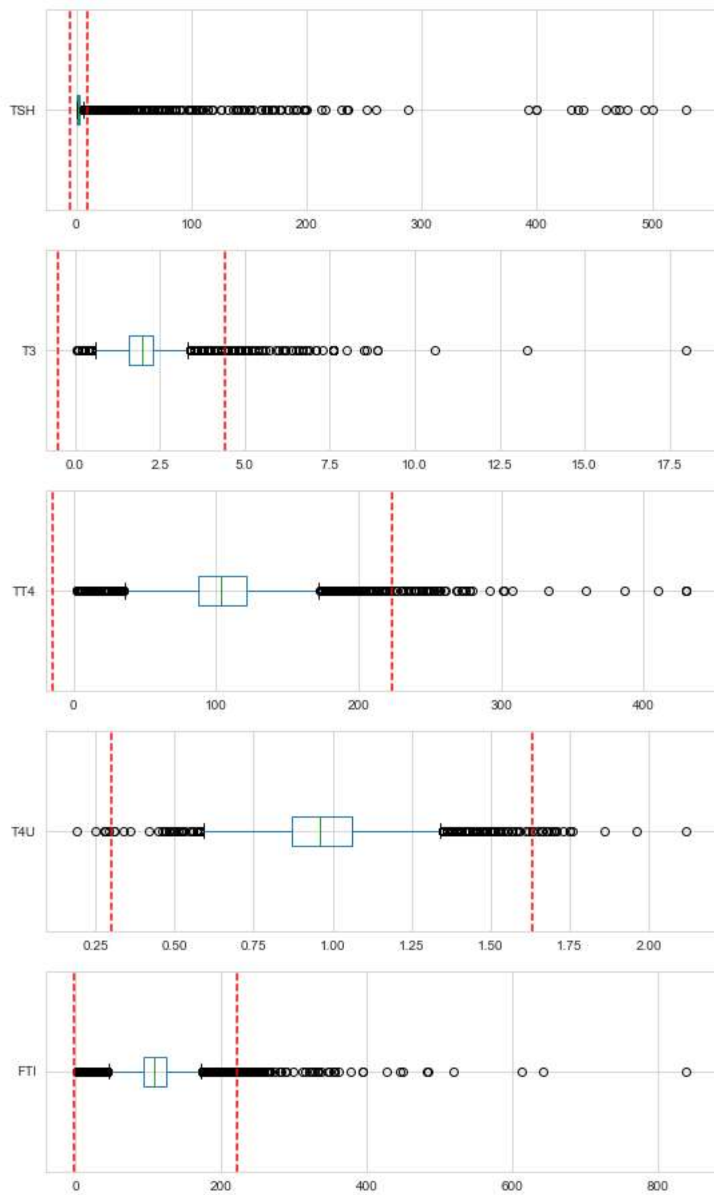
The code below calculates the Inter-quartile ranges for our Hormone test numeric attributes in order to determin mild and sever outliers. Depending on the severity of the issue we will decide how to handle them in the next section.

In [22]:
```
# TSH
Q1_TSH = df['TSH'].quantile(0.25)
Q3_TSH = df['TSH'].quantile(0.75)
IQR_TSH = Q3_TSH - Q1_TSH
lower_TSH = df['TSH'] < (Q1_TSH - 3 * IQR_TSH)
upper_TSH = df['TSH'] > (Q3_TSH + 3 * IQR_TSH)
print('TSH:', 'lower outliers -', sum(lower_TSH), ' | upper outliers -', sum(upper_TSH))

# T3
Q1_T3 = df['T3'].quantile(0.25)
Q3_T3 = df['T3'].quantile(0.75)
IQR_T3 = Q3_T3 - Q1_T3
lower_T3 = df['T3'] < (Q1_T3 - 3 * IQR_T3)
upper_T3 = df['T3'] > (Q3_T3 + 3 * IQR_T3)
print('T3:', 'lower outliers -', sum(lower_T3), ' | upper outliers -', sum(upper_T3))

# TT4
Q1_TT4 = df['TT4'].quantile(0.25)
Q3_TT4 = df['TT4'].quantile(0.75)
IQR_TT4 = Q3_TT4 - Q1_TT4
lower_TT4 = df['TT4'] < (Q1_TT4 - 3 * IQR_TT4)
upper_TT4 = df['TT4'] > (Q3_TT4 + 3 * IQR_TT4)
print('TT4:', 'lower outliers -', sum(lower_TT4), ' | upper outliers -', sum(upper_TT4))

# T4U
Q1_T4U = df['T4U'].quantile(0.25)
Q3_T4U = df['T4U'].quantile(0.75)
IQR_T4U = Q3_T4U - Q1_T4U
lower_T4U = df['T4U'] < (Q1_T4U - 3 * IQR_T4U)
upper_T4U = df['T4U'] > (Q3_T4U + 3 * IQR_T4U)
print('T4U:', 'lower outliers -', sum(lower_T4U), ' | upper outliers -', sum(upper_T4U))

# FTI
Q1_FTI = df['FTI'].quantile(0.25)
Q3_FTI = df['FTI'].quantile(0.75)
IQR_FTI = Q3_FTI - Q1_FTI
lower_FTI = df['FTI'] < (Q1_FTI - 3 * IQR_FTI)
upper_FTI = df['FTI'] > (Q3_FTI + 3 * IQR_FTI)
print('FTI:', 'lower outliers -', sum(lower_FTI), ' | upper outliers -', sum(upper_FTI))
```

```
TSH: lower outliers - 0  | upper outliers - 456
T3: lower outliers - 0  | upper outliers - 84
TT4: lower outliers - 0  | upper outliers - 55
T4U: lower outliers - 7  | upper outliers - 22
FTI: lower outliers - 0  | upper outliers - 83
```

In [23]:
```
# preparing boxplots
fig, axs= plt.subplots(nrows = 5, figsize=(9,16))

# TSH
df.boxplot(column='TSH', ax=axs[0], vert = False)
axs[0].axvline(x=(Q1_TSH - 3*IQR_TSH), color='r', linestyle='--')
axs[0].axvline(x=(Q3_TSH + 3*IQR_TSH), color='r', linestyle='--')
# T3
df.boxplot(column='T3', ax=axs[1], vert = False)
axs[1].axvline(x=(Q1_T3 - 3*IQR_T3), color='r', linestyle='--')
axs[1].axvline(x=(Q3_T3 + 3*IQR_T3), color='r', linestyle='--')
# TT4
df.boxplot(column='TT4', ax=axs[2], vert = False)
axs[2].axvline(x=(Q1_TT4 - 3*IQR_TT4), color='r', linestyle='--')
axs[2].axvline(x=(Q3_TT4 + 3*IQR_TT4), color='r', linestyle='--')
# T4U
df.boxplot(column='T4U', ax=axs[3], vert = False)
axs[3].axvline(x=(Q1_T4U - 3*IQR_T4U), color='r', linestyle='--')
axs[3].axvline(x=(Q3_T4U + 3*IQR_T4U), color='r', linestyle='--')
# FTI
df.boxplot(column='FTI', ax=axs[4], vert = False)
axs[4].axvline(x=(Q1_FTI - 3*IQR_FTI), color='r', linestyle='--')
axs[4].axvline(x=(Q3_FTI + 3*IQR_FTI), color='r', linestyle='--')
```

Out[23]: <matplotlib.lines.Line2D at 0x1e31d169fd0>



It seems that we have many severe outliers present. However, given our research about these values, this type of variance is normal within this context and is to be expected, especially when dealing with persons who will have alterations in these values given their medical conditions.

Observations:

In [24]:
```
#---> It doesnt seem like a coincidence that most missing values present are from blood tests
#---> We need to investigate this further in order to decide the best approach to handling them
```

calculating missingess of entire DF

In [25]:
```

missingness =df.isnull().sum().sum() / df.count().sum()
print('Overall Missingness of thyroidDF is: {:.2f}%'.format(missingness * 100))

# Create table for missing data analysis
def missing_table(df):
    total = df.isnull().sum().sort_values(ascending=False)
    percent = (df.isnull().sum()/df.isnull().count()).sort_values(ascending=False)
    missing_data = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
    return missing_data

# Analyze missing data
missing_table(df).head(10)
```

```
Overall Missingness of thyroidDF is: 7.54%
```

Out[25]:

	Total	Percent
TBG	7287	0.965677
T3	2209	0.292738
TSH	722	0.095680
T4U	676	0.089584
FTI	669	0.088656
TT4	354	0.046912
sex	250	0.033130
age	4	0.000530
goitre	0	0.000000
psych	0	0.000000

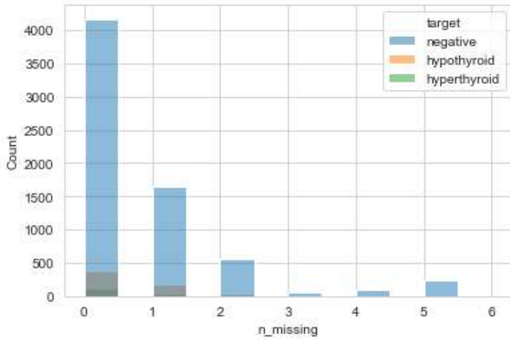
Missing Table Summary

In [26]:
```
# thyroidDF['TBG'] - 96.56% missing
# The 'TBG' attribute is almost entirely missing from the dataset. This column will be removed at once!

# thyroidDF['age'] - 0.045% missing
# We will also go ahead and drop these 4 observations from the dataset. All 4 of these observations belong to observations with

# thyroidDF['Sex'] - 3.37% missing
# There are a total of 300 observations where 'sex' is null. In an attempt to preserve some of these values,
#we check how many of these observations also have 'pregnant' == True.
#There are 3 such observations. Assuming the 'pregnancy' attribute is correct for these observations, we can confidently say thes
```

In [27]:
```
# dropping 'TBG' attribute from dataset
df.drop(['TBG'], axis=1, inplace=True)

# dropping 4 observations with abnormal 'age' from dataset
df.dropna(subset=['age'], inplace=True)

# changing sex of observations with ('pregnant' == True) & ('sex' == null) to Female
df['sex'] = np.where((df.sex.isnull()) & (df.pregnant == 't'), 'F', df.sex)
```

Now lets take a look at missing values per row. If we are moving forward with imputation, we dont want to keep rows that have too many missing values (especially since most missing values in the dataset are from the most important attributes).

In [28]:
```
# count missing values per row
df['n_missing'] = df.isnull().sum(axis=1)
sns.histplot(df, x='n_missing', binwidth=0.5, hue='target');
```



It looks like after removing TBG from the dataset, most of the observations in our dataset have zero missing values. A lot have one missing value and some have two. a minority have 3 or more. Before we do any imputation we will remove the observations that are missing data for more than 2 columns.

In [29]:
```
# calculating missingess of entire DF
missingness = df.isnull().sum().sum() /df.count().sum()
print('Overall Missingness of thyroidDF is: {:.2f}%'.format(missingness * 100))

# remove rows with 3 or more missing values
df.drop(df.index[df['n_missing'] > 2], inplace=True)
print
missing_table(df).head(10)
```

```
Overall Missingness of thyroidDF is: 2.89%
```

Out[29]:

	Total	Percent
T3	1910	0.267432
TSH	406	0.056847
T4U	290	0.040605
FTI	283	0.039625
sex	208	0.029123
TT4	6	0.000840
age	0	0.000000
goitre	0	0.000000
target	0	0.000000
psych	0	0.000000

In [30]:
```
# replacing boolean strings with binary 0 and 1
df.replace('f', 0, inplace=True)
df.replace('t', 1, inplace=True)

# replacing sex with binary 0 and 1
df.replace('M', 0, inplace=True) # male mapped to 0
df.replace('F', 1, inplace=True) # female mapped to 1

thyroid_df = df.replace(np.nan, 0)

# re-mapping target vaues to diagnostic groups
# *** I get different final results by ordering these classes differently ***
diagnoses = {'negative': 0,
             'hypothyroid': 1,
             'hyperthyroid': 2}
thyroid_df['target'] =thyroid_df['target'].map(diagnoses)
```

In [31]:
```
thyroid_df
```

Out[31]:

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_meds	sick	pregnant	thyroid_surgery	I131_treatment	query_hypothyroid	...	tumor	hypopit
1	29.0	1.0	0	0	0	0	0	0	0	0	...	0	
7	28.0	1.0	0	0	0	0	0	0	0	0	...	0	
8	28.0	1.0	0	0	0	0	0	0	0	0	...	0	
9	28.0	1.0	0	0	0	0	0	0	0	0	...	0	
10	54.0	1.0	0	0	0	0	0	0	0	0	...	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
9166	70.0	1.0	0	0	0	0	0	0	0	0	...	0	
9167	56.0	0.0	0	0	0	0	0	0	0	0	...	0	
9168	22.0	0.0	0	0	0	0	0	0	0	0	...	0	
9170	47.0	1.0	0	0	0	0	0	0	0	0	...	0	
9171	31.0	0.0	0	0	0	0	0	0	0	1	...	0	

7142 rows × 23 columns

Traning Data:

In [32]:
```
from sklearn.model_selection import train_test_split
X = thyroid_df.drop('target', axis=1).copy()
y = thyroid_df['target'].copy()

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

Model Creation:

In [33]:
```
# Import the model we are using
from sklearn.ensemble import RandomForestClassifier
# Instantiate model with 100 decision trees
rc_model = RandomForestClassifier(n_estimators=100, criterion='entropy')

rc_model.fit(X_train, y_train)

rc_pred = rc_model.predict(X_test)
```

classification_report:

In [34]:
```
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report

print('classification_report:')
print(classification_report(y_test,rc_pred))

acc=accuracy_score(y_test,rc_pred)
print('accuracy of the model :',acc)
```

```
classification_report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      1597
           1       0.95      1.00      0.97       145
           2       0.89      0.77      0.83        44

    accuracy                           0.99      1786
   macro avg       0.95      0.92      0.93      1786
weighted avg       0.99      0.99      0.99      1786

accuracy of the model : 0.9876819708846585
```

In [ ]: