In [79]:

```
import pandas as pd
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns
data = pd.read_csv("train_data1.csv", engine= "python")
data.head(10)
```

Out[79]:

	case_id	Hospital_code	Hospital_type_code	City_Code_Hospital	Hospital_region_code	Avai R Hos
0	1	8	c	3	Z	
1	2	2	c	5	Z	
2	3	10	e	1	X	
3	4	26	b	2	Y	
4	5	26	b	2	Y	
5	6	23	a	6	X	
6	7	32	f	9	Y	
7	8	23	a	6	X	
8	9	1	d	10	Y	
9	10	10	e	1	X	

In [80]:

```
age_lst = data["Age"].unique()

age_lst.sort()
age_dict = dict(zip(age_lst, range(len(age_lst))))
data["new_age"]=data["Age"].replace(age_dict)
print(age_dict)

stay_list = data["Stay"].unique()
stay_list.sort()
dept_Stay = dict(zip(stay_list, range(len(stay_list))))
data["new_stay"]= data["Stay"].replace(dept_Stay)
print(dept_Stay)

data.head()
```

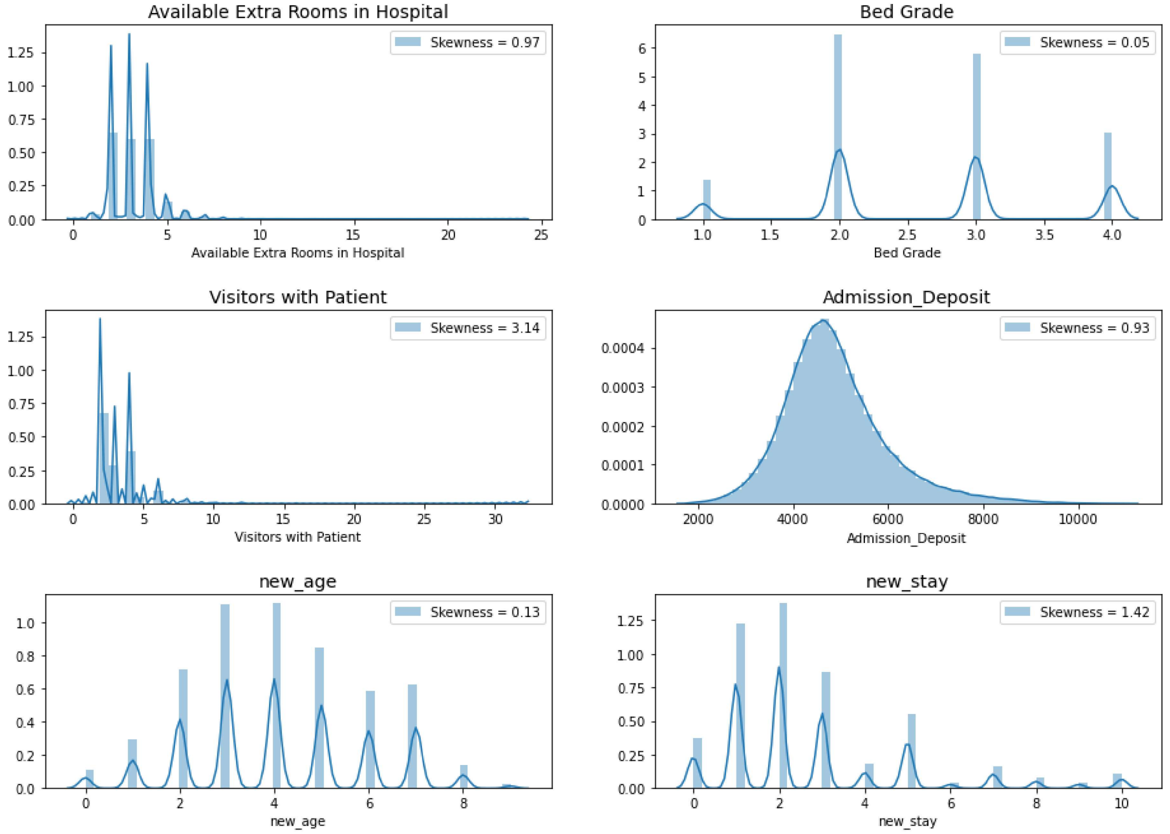
```
{'0-10': 0, '11-20': 1, '21-30': 2, '31-40': 3, '41-50': 4, '51-60': 5, '61
-70': 6, '71-80': 7, '81-90': 8, '91-100': 9}
{'0-10': 0, '11-20': 1, '21-30': 2, '31-40': 3, '41-50': 4, '51-60': 5, '61
-70': 6, '71-80': 7, '81-90': 8, '91-100': 9, 'More than 100 Days': 10}
```

Out[80]:

_code	Hospital_type_code	City_Code_Hospital	Hospital_region_code	Available Extra Rooms in Hospital	Department	Wa
8	c	3	Z	3	radiotherapy	
2	c	5	Z	2	radiotherapy	
10	e	1	X	2	anesthesia	
26	b	2	Y	2	radiotherapy	
26	b	2	Y	2	radiotherapy	

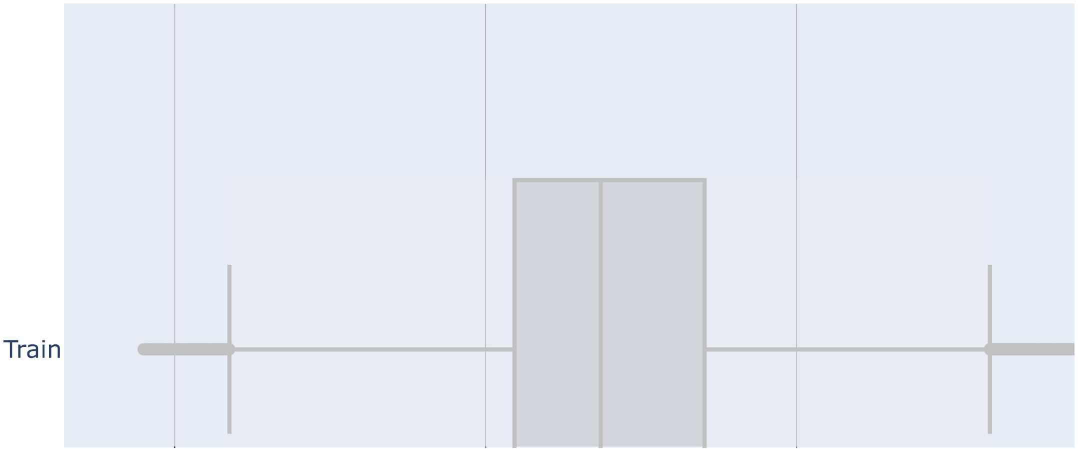
In [81]:

```
numerical_data = data[['Available Extra Rooms in Hospital', 'Bed Grade', 'Vis
                    , 'Admission_Deposit', 'new_age','new_stay']]
fig, new_plot =plt.subplots(3,2, figsize=(14,10))
fig.tight_layout(pad=5.0)

for new_plot, n in zip(new_plot.flatten(), numerical_data.columns.tolist()):
    sns.distplot(ax=new_plot, a=numerical_data[n], label="Skewness = %.2f"%(r
    new_plot.set_title(n, fontsize = 14)
    new_plot.legend(loc = 'best')
```



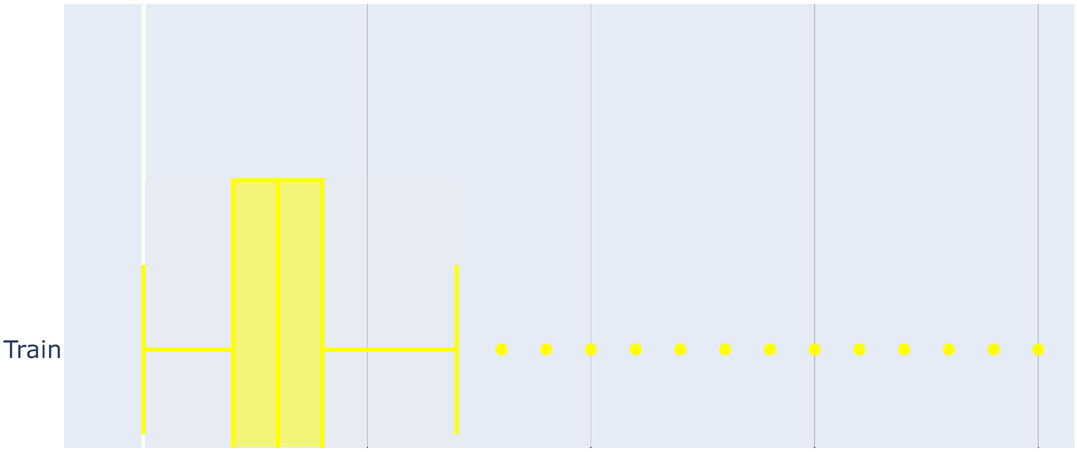
In [82]:

```
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go

from sklearn.preprocessing import LabelEncoder
fig = go.Figure()
fig.add_trace(go.Box(x=data['Admission_Deposit'],
                     marker_color="silver",
                     name="Train"))
```



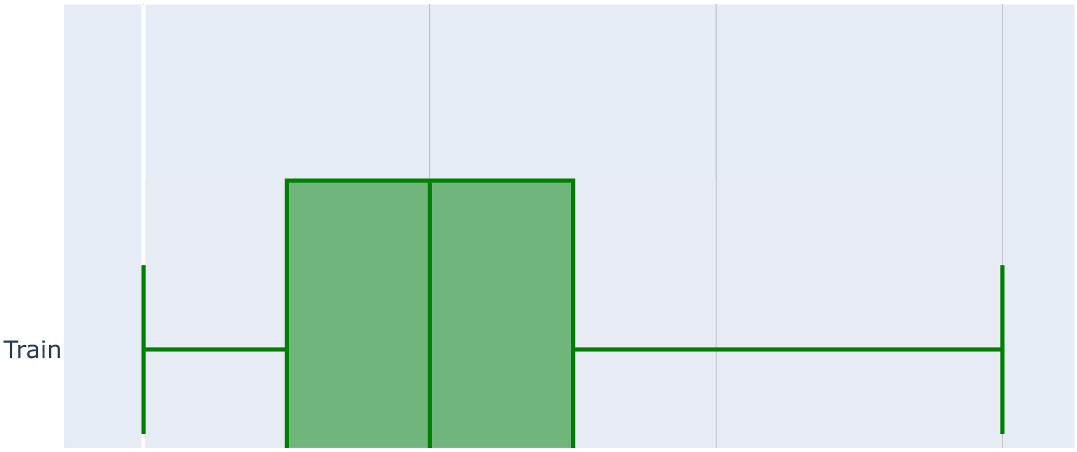
In [83]:

```
from sklearn.preprocessing import LabelEncoder
fig1 = go.Figure()
fig1.add_trace(go.Box(x=data['Visitors with Patient'],
                     marker_color="yellow",
                     name="Train"))
```



In [84]:

```
from sklearn.preprocessing import LabelEncoder
fig1 = go.Figure()
fig1.add_trace(go.Box(x=data['new_stay'],
                      marker_color="green",
                      name="Train"))
```



In [85]:

```
ilable Extra Rooms in Hospital'].quantile(0.25)
ilable Extra Rooms in Hospital'].quantile(0.75)

(data['Available Extra Rooms in Hospital'] < (q1 - 1.5 * iqr)) | (data['Availa
```

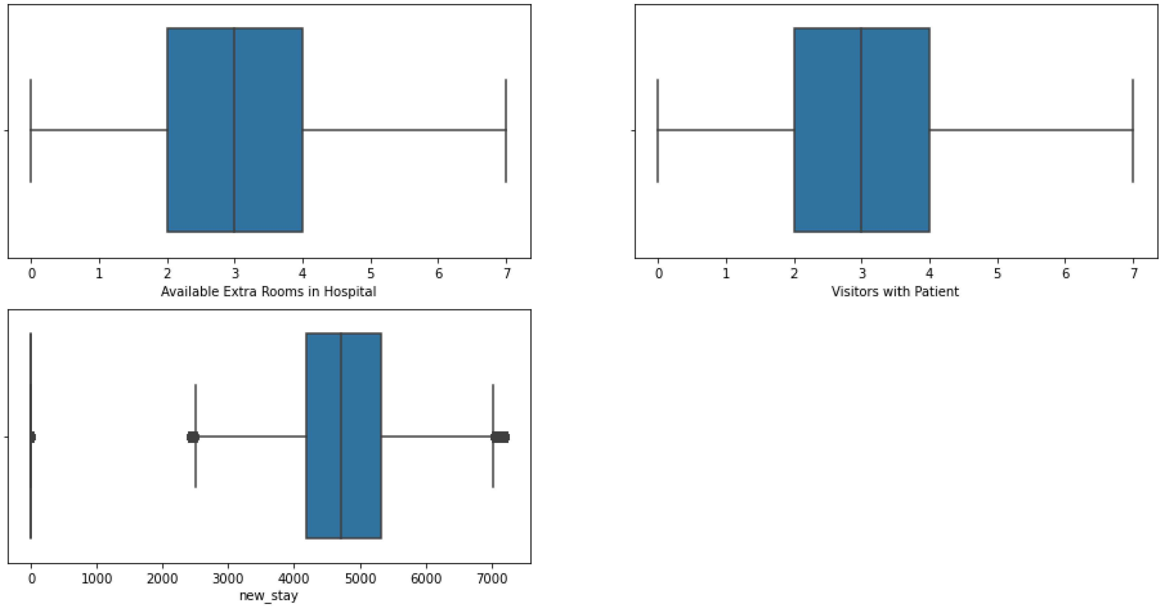
In [86]:

```
q1=data['Visitors with Patient'].quantile(0.25)
q3 = data['Visitors with Patient'].quantile(0.75)
iqr = q3-q1
data = data[~ ((data['Visitors with Patient'] < q1 - 1.5 * iqr) | (data['Visi

q1=data['Admission_Deposit'].quantile(0.25)
q3 = data['Admission_Deposit'].quantile(0.75)
iqr = q3-q1
data = data[~ ((train['Admission_Deposit'] < q1 - 1.5 * iqr) | (data['Admissi
```

```
<ipython-input-86-e483c9db69c1>:9: UserWarning:

Boolean Series key will be reindexed to match DataFrame index.
```

In [87]:

```
fig, ax = plt.subplots(2,2, figsize = (16,8))
sns.boxplot(ax = ax[0, 0], x = data['Available Extra Rooms in Hospital'])
sns.boxplot(ax = ax[0, 1], x = data['Visitors with Patient'])
sns.boxplot(ax = ax[1, 0], x = data['Admission_Deposit'])
sns.boxplot(ax = ax[1, 0], x = data['new_stay'])

fig.delaxes(ax[1,1])
plt.show()
```

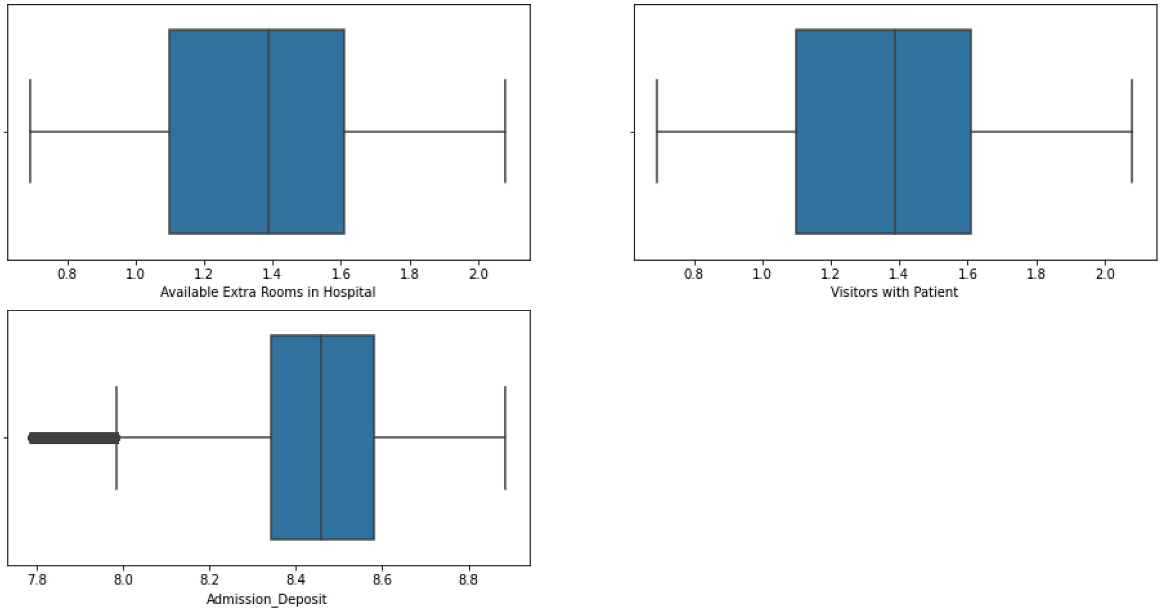


In [88]:

```
data['Available Extra Rooms in Hospital'] = np.log(train['Available Extra Roo
data['Visitors with Patient'] = np.log(train['Visitors with Patient'] + 1)
data['Admission_Deposit'] = np.log(train['Admission_Deposit'] + 1)


# Remove outliers after log transform on data train
data = data[data['Available Extra Rooms in Hospital'] > 0]
data = data[data['Visitors with Patient'] > 0]
data = data[data['Admission_Deposit'] > 0]
```

In [89]:

```
fig, ax = plt.subplots(2,2, figsize = (16,8))
sns.boxplot(ax = ax[0, 0], x = data['Available Extra Rooms in Hospital'])
sns.boxplot(ax = ax[0, 1], x = data['Visitors with Patient'])
sns.boxplot(ax = ax[1, 0], x = data['Admission_Deposit'])



fig.delaxes(ax[1,1])
plt.show()
```



In [90]:

```
new1 = data.drop(['case_id', 'Hospital_code','Age', 'City_Code_Hospital', 'Ci
         'Hospital_type_code' ,'Stay']
         , axis = 1)
new1.head()
```

Out[90]:

	Hospital_region_code	Available Extra Rooms in Hospital	Department	Ward_Type	Bed Grade	patientid	Type of Admission	Seve Illn
0	Z	1.386294	radiotherapy	R	2.0	31397	Emergency	Extr
1	Z	1.098612	radiotherapy	S	2.0	31397	Trauma	Extr
2	X	1.098612	anesthesia	S	2.0	31397	Trauma	Extr
4	Y	1.098612	radiotherapy	S	2.0	31397	Trauma	Extr
5	X	1.098612	anesthesia	S	2.0	31397	Trauma	Extr

In [91]:

```
import numpy as np

new_dept = new1["Department"].unique()
new_dept.sort()
new_dept = dict(zip(new_dept, range(len(new_dept))))
new1.Department.replace(new_dept, inplace= True)
print(new_dept)

new_hosp_code = new1["Hospital_region_code"].unique()
new_hosp_code.sort()
new_hosp_code= dict(zip(new_hosp_code, range(len(new_hosp_code))))
new1.Hospital_region_code.replace(new_hosp_code, inplace = True)
print(new_hosp_code)

new_ward_type = new1["Ward_Type"].unique()
new_ward_type.sort()
new_ward_type = dict(zip(new_ward_type, range(len(new_ward_type))))
new1.replace(new_ward_type, inplace=True)
print(new_ward_type)

new_type_admiss = new1["Type of Admission"].unique()
new_type_admiss.sort()
new_type_admiss = dict(zip(new_type_admiss, range(len(new_type_admiss))))
new1["Type of Admission"].replace(new_type_admiss, inplace=True)
print(new_type_admiss)

new_severity = new1["Severity of Illness"].unique()
new_severity .sort()
new_severity  = dict(zip(new_severity, range(len(new_severity ))))
new1["Severity of Illness"].replace(new_severity , inplace=True)
print(new_severity )
```

```
{'TB & Chest disease': 0, 'anesthesia': 1, 'gynecology': 2, 'radiotherapy':
3, 'surgery': 4}
{'X': 0, 'Y': 1, 'Z': 2}
{'P': 0, 'Q': 1, 'R': 2, 'S': 3, 'T': 4, 'U': 5}
{'Emergency': 0, 'Trauma': 1, 'Urgent': 2}
{'Extreme': 0, 'Minor': 1, 'Moderate': 2}
```

In [92]:

```
new1.head()
```

Out[92]:

	Hospital_region_code	Available Extra Rooms in Hospital	Department	Ward_Type	Bed Grade	patientid	Type of Admission	Seve Illr
0	2	1.386294	3	2	2.0	31397	0	
1	2	1.098612	3	3	2.0	31397	1	
2	0	1.098612	1	3	2.0	31397	1	
4	1	1.098612	3	3	2.0	31397	1	
5	0	1.098612	1	3	2.0	31397	1	

In [93]:

```
column_names = ['new_stay','new_age','Available Extra Rooms in Hospital','Bec
                'patientid','Type of Admission','Visitors with Patient','Admi
                'Department','Severity of Illness','Hospital_region_code']

new1 = new1.reindex(columns=column_names)
```

In [94]:

```
new2 = new1
```

Out[94]:

	new_stay	new_age	Available Extra Rooms in Hospital	Bed Grade	patientid	Type of Admission	Visitors with Patient	Admission_Deposit
0	0	5	1.386294	2.0	31397	0	1.098612	8.499436
1	4	5	1.098612	2.0	31397	1	1.098612	8.691986
2	3	5	1.098612	2.0	31397	1	1.098612	8.465057
4	4	5	1.098612	2.0	31397	1	1.098612	8.623174
5	1	5	1.098612	2.0	31397	1	1.098612	8.400659

In [111]:

```
new2 = new1.drop(['Visitors with Patient','Hospital_region_code','Visitors wi
           , axis = 1)
new3 = new2.dropna()

new3.isna().sum()
```

Out[111]:

```
new_stay                             0
new_age                              0
Available Extra Rooms in Hospital    0
Bed Grade                            0
Type of Admission                    0
Admission_Deposit                    0
Department                           0
Severity of Illness                  0
dtype: int64
```

In [105]:

```
x_train = new3.iloc[:, 1:].values
y_train = new3.iloc[:, 0].values
```

In [106]:

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score


x_train_split, x_val_split, y_train_split, y_val_split = train_test_split(x_t
clf = RandomForestClassifier(n_estimators=300, max_depth = 20, min_samples_le
clf.fit(x_train_split, y_train_split)
y_pred = clf.predict(x_val_split)
accuracy = accuracy_score(y_pred, y_val_split)
print('Accuracy :',accuracy)
```

```
Accuracy : 0.3195481934508361
```

In [115]:

```
x = new3.drop(["new_stay",'Bed Grade'], axis=1).to_numpy()
y = new3['new_stay'].values
```

In [117]:

```
X_train, X_val, Y_train, Y_val = train_test_split(x, y, test_size = 0.2, ranc
```

In [119]:

```
clf_rf = RandomForestClassifier(n_estimators=1000, max_depth=15)

clf_rf.fit(X_train, Y_train)

Y_pred_rf = clf_rf.predict(X_val)
# get the accuracy score
acc_rf = accuracy_score(Y_pred_rf, Y_val)
print(acc_rf)
```

```
0.31042128603104213
```

In [ ]: