

In [1]:

```
import pandas as pd
```

In [2]:

```
df1 = pd.read_csv('week_approach_maskedID_timeseries.csv')
```

In [3]:

```
df1.head()
```

Out[3]:

	nr. sessions	nr. rest days	total kms	max km one day	total km Z3- Z4- Z5- T1- T2	nr. tough sessions (effort in Z5, T1 or T2)	nr. days with interval session	total km Z3-4	max km Z3- 4 one day	total km Z5- T1- T2	...	max training success.2	a'
0	5	2	22.2	16.4	11.8	1	2	10.0	10.0	0.6	...	0.0	0.
1	5	2	21.6	16.4	11.7	1	2	10.0	10.0	0.5	...	0.0	0.
2	5	2	21.6	16.4	11.7	1	2	10.0	10.0	0.5	...	0.0	0.
3	5	2	21.6	16.4	11.7	1	2	10.0	10.0	0.5	...	0.0	0.
4	6	1	39.2	17.6	18.9	1	3	17.2	10.0	0.5	...	0.0	0.

5 rows × 72 columns

In [4]:

```
df1.tail()
```

Out[4]:

	nr. sessions	nr. rest days	total kms	max km one day	total km Z3- Z4- Z5- T1- T2	nr. tough sessions (effort in Z5, T1 or T2)	nr. days with interval session	total km Z3-4	max km Z3- 4 one day	total km Z5- T1- T2	...	max training success.2	reci
42793	4	3	59.5	19.0	0.0	0	0	0.0	0.0	0.0	...	0.80	
42794	1	6	5.8	5.8	0.0	0	0	0.0	0.0	0.0	...	0.85	
42795	3	4	38.3	16.0	0.0	0	0	0.0	0.0	0.0	...	0.93	
42796	5	2	67.0	15.1	0.0	0	0	0.0	0.0	0.0	...	0.91	
42797	4	3	45.0	12.2	0.0	0	0	0.0	0.0	0.0	...	0.88	

5 rows × 72 columns

In [5]:

```
df1.shape
```

Out[5]:

```
(42798, 72)
```

In [6]:

```
df1.columns
```

Out[6]:

```
Index(['nr. sessions', 'nr. rest days', 'total kms', 'max km one day',  
      'total km Z3-Z4-Z5-T1-T2',  
      'nr. tough sessions (effort in Z5, T1 or T2)',  
      'nr. days with interval session', 'total km Z3-4',  
      'max km Z3-4 one day', 'total km Z5-T1-T2', 'max km Z5-T1-T2 one day',  
      'total hours alternative training', 'nr. strength trainings',  
      'avg exertion', 'min exertion', 'max exertion', 'avg training success',  
      'min training success', 'max training success', 'avg recovery',  
      'min recovery', 'max recovery', 'nr. sessions.1', 'nr. rest days.1',  
      'total kms.1', 'max km one day.1', 'total km Z3-Z4-Z5-T1-T2.1',  
      'nr. tough sessions (effort in Z5, T1 or T2).1',  
      'nr. days with interval session.1', 'total km Z3-4.1',  
      'max km Z3-4 one day.1', 'total km Z5-T1-T2.1',  
      'max km Z5-T1-T2 one day.1', 'total hours alternative training.1',  
      'nr. strength trainings.1', 'avg exertion.1', 'min exertion.1',  
      'max exertion.1', 'avg training success.1', 'min training success.1',  
      'max training success.1', 'avg recovery.1', 'min recovery.1',  
      'max recovery.1', 'nr. sessions.2', 'nr. rest days.2', 'total kms.2',  
      'max km one day.2', 'total km Z3-Z4-Z5-T1-T2.2',  
      'nr. tough sessions (effort in Z5, T1 or T2).2',  
      'nr. days with interval session.2', 'total km Z3-4.2',  
      'max km Z3-4 one day.2', 'total km Z5-T1-T2.2',  
      'max km Z5-T1-T2 one day.2', 'total hours alternative training.2',  
      'nr. strength trainings.2', 'avg exertion.2', 'min exertion.2',  
      'max exertion.2', 'avg training success.2', 'min training success.2',  
      'max training success.2', 'avg recovery.2', 'min recovery.2',  
      'max recovery.2', 'Athlete ID', 'injury', 'rel total kms week 0_1',  
      'rel total kms week 0_2', 'rel total kms week 1_2', 'Date'],  
      dtype='object')
```

In [7]:

```
df1.duplicated().sum()
```

Out[7]:

```
0
```

In [8]:

```
df1.isnull().sum()
```

Out[8]:

```
nr. sessions      0
nr. rest days     0
total kms         0
max km one day    0
total km Z3-Z4-Z5-T1-T2  0
..
injury            0
rel total kms week 0_1  0
rel total kms week 0_2  0
rel total kms week 1_2  0
Date              0
Length: 72, dtype: int64
```

In [9]:

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 42798 entries, 0 to 42797
```

```
Data columns (total 72 columns):
```

#	Column	Non-Null Count	Dtype
0	nr. sessions	42798 non-null	int64
1	nr. rest days	42798 non-null	int64
2	total kms	42798 non-null	float64
3	max km one day	42798 non-null	float64
4	total km Z3-Z4-Z5-T1-T2	42798 non-null	float64
5	nr. tough sessions (effort in Z5, T1 or T2)	42798 non-null	int64
6	nr. days with interval session	42798 non-null	int64
7	total km Z3-4	42798 non-null	float64
8	max km Z3-4 one day	42798 non-null	float64
9	total km Z5-T1-T2	42798 non-null	float64
10	max km Z5-T1-T2 one day	42798 non-null	float64
11	total hours alternative training	42798 non-null	float64
12	nr. strength trainings	42798 non-null	int64
13	avg exertion	42798 non-null	float64
14	min exertion	42798 non-null	float64
15	max exertion	42798 non-null	float64
16	avg training success	42798 non-null	float64
17	min training success	42798 non-null	float64
18	max training success	42798 non-null	float64
19	avg recovery	42798 non-null	float64
20	min recovery	42798 non-null	float64
21	max recovery	42798 non-null	float64
22	nr. sessions.1	42798 non-null	int64
23	nr. rest days.1	42798 non-null	int64
24	total kms.1	42798 non-null	float64
25	max km one day.1	42798 non-null	float64
26	total km Z3-Z4-Z5-T1-T2.1	42798 non-null	float64
27	nr. tough sessions (effort in Z5, T1 or T2).1	42798 non-null	int64
28	nr. days with interval session.1	42798 non-null	int64
29	total km Z3-4.1	42798 non-null	float64
30	max km Z3-4 one day.1	42798 non-null	float64
31	total km Z5-T1-T2.1	42798 non-null	float64
32	max km Z5-T1-T2 one day.1	42798 non-null	float64
33	total hours alternative training.1	42798 non-null	float64
34	nr. strength trainings.1	42798 non-null	int64
35	avg exertion.1	42798 non-null	float64
36	min exertion.1	42798 non-null	float64
37	max exertion.1	42798 non-null	float64
38	avg training success.1	42798 non-null	float64
39	min training success.1	42798 non-null	float64
40	max training success.1	42798 non-null	float64
41	avg recovery.1	42798 non-null	float64
42	min recovery.1	42798 non-null	float64
43	max recovery.1	42798 non-null	float64
44	nr. sessions.2	42798 non-null	int64
45	nr. rest days.2	42798 non-null	int64
46	total kms.2	42798 non-null	float64
47	max km one day.2	42798 non-null	float64
48	total km Z3-Z4-Z5-T1-T2.2	42798 non-null	float64
49	nr. tough sessions (effort in Z5, T1 or T2).2	42798 non-null	int64
50	nr. days with interval session.2	42798 non-null	int64
51	total km Z3-4.2	42798 non-null	float64
52	max km Z3-4 one day.2	42798 non-null	float64
53	total km Z5-T1-T2.2	42798 non-null	float64
54	max km Z5-T1-T2 one day.2	42798 non-null	float64
55	total hours alternative training.2	42798 non-null	float64
56	nr. strength trainings.2	42798 non-null	int64

```
57 avg exertion.2          42798 non-null float64
58 min exertion.2          42798 non-null float64
59 max exertion.2          42798 non-null float64
60 avg training success.2  42798 non-null float64
61 min training success.2  42798 non-null float64
62 max training success.2  42798 non-null float64
63 avg recovery.2          42798 non-null float64
64 min recovery.2          42798 non-null float64
65 max recovery.2          42798 non-null float64
66 Athlete ID              42798 non-null int64
67 injury                  42798 non-null int64
68 rel total kms week 0_1   42798 non-null float64
69 rel total kms week 0_2   42798 non-null float64
70 rel total kms week 1_2   42798 non-null float64
71 Date                    42798 non-null int64
```

dtypes: float64(54), int64(18)
memory usage: 23.5 MB

In [10]:

```
df1.describe()
```

Out[10]:

	nr. sessions	nr. rest days	total kms	max km one day	total km Z3-Z4-Z5-T1-T2	nr. tough sessions (effort in Z5, T1 or T2)	nr. i
count	42798.000000	42798.000000	42798.000000	42798.000000	42798.000000	42798.000000	42798.000000
mean	5.809337	1.874667	49.543911	14.009255	9.433621	0.930184	0.930184
std	2.484234	1.853287	36.715017	9.071678	8.887120	1.040631	1.040631
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	5.000000	1.000000	22.800000	9.000000	1.000000	0.000000	0.000000
50%	6.000000	1.000000	44.800000	13.400000	8.000000	1.000000	1.000000
75%	7.000000	3.000000	70.100000	18.300000	14.600000	2.000000	2.000000
max	14.000000	7.000000	242.000000	131.000000	100.000000	6.000000	6.000000

8 rows × 72 columns

In [11]:

```
df1 = df1.drop(['avg training success', 'min training success', 'max training success', 'avg  
              'avg training success.2', 'max training success.2', 'min training success.2',  
              'avg exertion.1', 'min exertion.1', 'max exertion.1', 'avg exertion.2', 'min e  
              'avg recovery', 'min recovery', 'max recovery', 'avg recovery.1', 'min recover  
              'rel total kms week 0_1', 'rel total kms week 0_2', 'rel total kms week 1_2'],
```

In [12]:

```
df1.nunique()
```

Out[12]:

nr. sessions	15
nr. rest days	8
total kms	1772
total km Z3-Z4-Z5-T1-T2	493
nr. tough sessions (effort in Z5, T1 or T2)	7
nr. days with interval session	8
total km Z3-4	378
max km Z3-4 one day	177
total km Z5-T1-T2	317
max km Z5-T1-T2 one day	136
total hours alternative training	679
nr. strength trainings	10
nr. sessions.1	15
nr. rest days.1	8
total kms.1	1769
max km one day.1	399
total km Z3-Z4-Z5-T1-T2.1	494
nr. tough sessions (effort in Z5, T1 or T2).1	7
nr. days with interval session.1	8
total km Z3-4.1	378
max km Z3-4 one day.1	177
total km Z5-T1-T2.1	315
max km Z5-T1-T2 one day.1	137
total hours alternative training.1	674
nr. strength trainings.1	10
nr. sessions.2	15
nr. rest days.2	8
total kms.2	1770
max km one day.2	396
total km Z3-Z4-Z5-T1-T2.2	494
nr. tough sessions (effort in Z5, T1 or T2).2	7
nr. days with interval session.2	8
total km Z3-4.2	380
max km Z3-4 one day.2	176
total km Z5-T1-T2.2	311
max km Z5-T1-T2 one day.2	138
total hours alternative training.2	691
nr. strength trainings.2	10
Athlete ID	74
injury	2
Date	2614
dtype: int64	

In [13]:

```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [14]:

```
df1['injury'].unique()
```

Out[14]:

```
array([0, 1], dtype=int64)
```

In [15]:

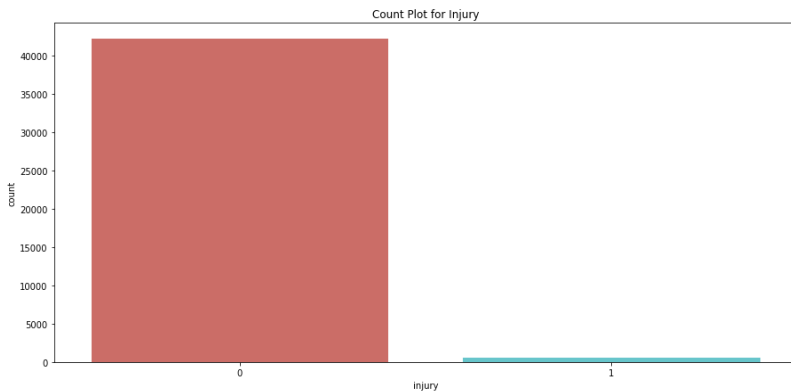
```
df1['injury'].value_counts()
```

Out[15]:

```
0    42223
1      575
Name: injury, dtype: int64
```

In [16]:

```
plt.figure(figsize=[15,7],)
plt.title('Count Plot for Injury')
sns.countplot(x = 'injury', data = df1, palette = 'hls')
plt.show()
```



In [17]:

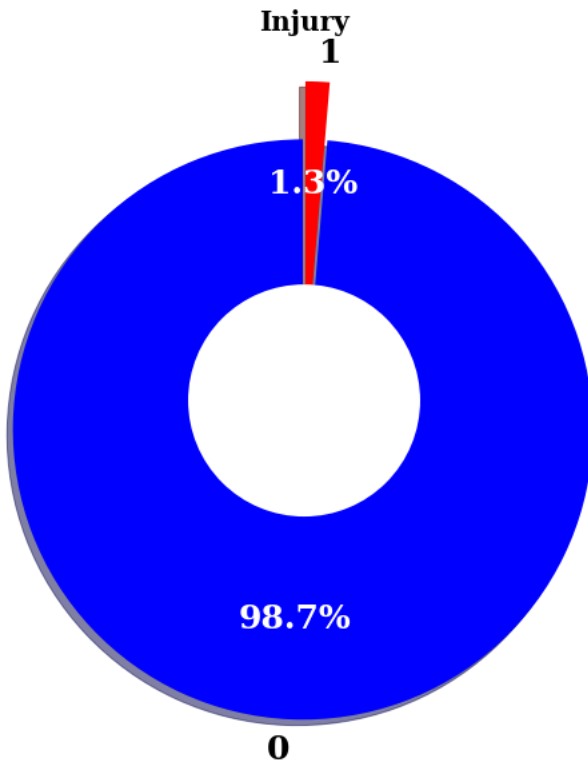
```
injury_data = df1['injury'].value_counts()

explode = (0.1, 0.1)
plt.figure(figsize=(14, 10))
patches, texts, pcts = plt.pie(injury_data,
                                labels = injury_data.index,
                                colors = ['blue', 'red'],
                                pctdistance = 0.65,
                                shadow = True,
                                startangle = 90,
                                explode = explode,
                                autopct = '%1.1f%%',
                                textprops={ 'fontsize': 25,
                                              'color': 'black',
                                              'weight': 'bold',
                                              'family': 'serif' })

plt.setp(pcts, color='white')

hfont = {'fontname':'serif', 'weight': 'bold'}
plt.title('Injury', size=20, **hfont)

centre_circle = plt.Circle((0,0),0.40,fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)
plt.show()
```



In [18]:

```
df1['Athlete ID'].unique()
```

Out[18]:

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
        34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
        51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67,
        68, 69, 70, 71, 72, 73], dtype=int64)
```

In [19]:

```
df1['Athlete ID'].value_counts()
```

Out[19]:

```
32    1791
20    1737
22    1478
43    1471
41    1393
...
66     148
39     129
15     126
55      48
60      43
Name: Athlete ID, Length: 74, dtype: int64
```

In [20]:

```
df2 = df1.copy()
```

In [21]:

```
df2 = df2.set_index('Athlete ID')
```

In [22]:

```
df2.head()
```

Out[22]:

Athlete ID	nr. sessions	nr. rest days	total kms	total km Z3-Z5-T1-T2	nr. tough sessions (effort in Z5, T1 or T2)	nr. days with interval session	total km Z3-4	max km Z3-4 one day	total km Z5-T1-T2	max km Z5-T1-T2 one day	nr. tough sessions (effort in Z5, T1 or T2).2	nr. ir ses
0	5	2	22.2	11.8	1	2	10.0	10.0	0.6	0.6	...	0
0	5	2	21.6	11.7	1	2	10.0	10.0	0.5	0.5	...	0
0	5	2	21.6	11.7	1	2	10.0	10.0	0.5	0.5	...	0
0	5	2	21.6	11.7	1	2	10.0	10.0	0.5	0.5	...	0
0	6	1	39.2	18.9	1	3	17.2	10.0	0.5	0.5	...	0

5 rows × 40 columns



In [23]:

```
df2.tail()
```

Out[23]:

Athlete ID	nr. sessions	nr. rest days	total kms	total km Z3-Z4-Z5-T1-T2	nr. tough sessions (effort in Z5, T1 or T2)	nr. days with interval session	total km Z3-4	max km Z3-4 one day	total km Z5-T1-T2	max km Z5-T1-T2 one day	...	nr. tough sessions (effort in Z5, T1 or T2).2	nr. sessions
71	4	3	59.5	0.0	0	0	0.0	0.0	0.0	0.0	...	0	
71	1	6	5.8	0.0	0	0	0.0	0.0	0.0	0.0	...	0	
71	3	4	38.3	0.0	0	0	0.0	0.0	0.0	0.0	...	0	
71	5	2	67.0	0.0	0	0	0.0	0.0	0.0	0.0	...	0	
71	4	3	45.0	0.0	0	0	0.0	0.0	0.0	0.0	...	0	

5 rows × 40 columns

In [24]:

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

Out[24]:

343

In [25]:

```
df2 = df2.drop_duplicates()
```

In [26]:

```
df2.isnull().sum()
```

Out[26]:

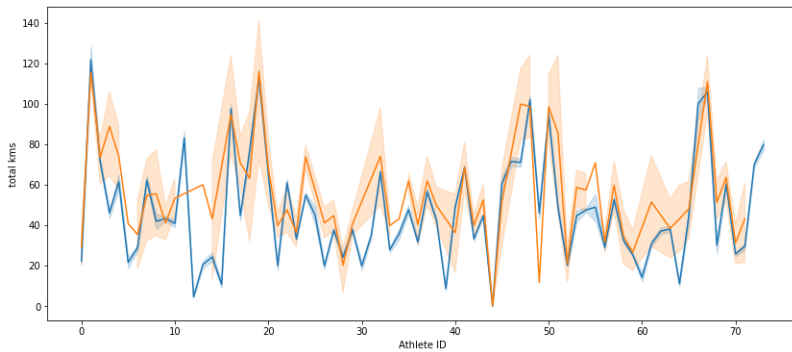
```
nr. sessions                                0
nr. rest days                              0
total kms                                  0
total km Z3-Z4-Z5-T1-T2                   0
nr. tough sessions (effort in Z5, T1 or T2) 0
nr. days with interval session              0
total km Z3-4                              0
max km Z3-4 one day                        0
total km Z5-T1-T2                         0
max km Z5-T1-T2 one day                   0
total hours alternative training            0
nr. strength trainings                     0
nr. sessions.1                            0
nr. rest days.1                           0
total kms.1                               0
max km one day.1                          0
total km Z3-Z4-Z5-T1-T2.1                 0
nr. tough sessions (effort in Z5, T1 or T2).1 0
nr. days with interval session.1           0
total km Z3-4.1                           0
max km Z3-4 one day.1                     0
total km Z5-T1-T2.1                       0
max km Z5-T1-T2 one day.1                 0
total hours alternative training.1          0
nr. strength trainings.1                   0
nr. sessions.2                            0
nr. rest days.2                           0
total kms.2                               0
max km one day.2                          0
total km Z3-Z4-Z5-T1-T2.2                 0
nr. tough sessions (effort in Z5, T1 or T2).2 0
nr. days with interval session.2           0
total km Z3-4.2                           0
max km Z3-4 one day.2                     0
total km Z5-T1-T2.2                       0
max km Z5-T1-T2 one day.2                 0
total hours alternative training.2          0
nr. strength trainings.2                   0
injury                                     0
Date                                       0
dtype: int64
```

In [27]:

```
df3 = df2[df2['injury'] == 0]
df4 = df2[df2['injury'] == 1]
```

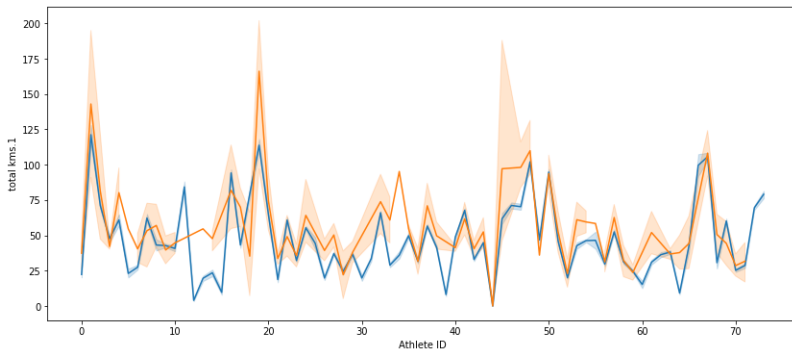
In [28]:

```
plt.figure(figsize = (14,6))
sns.lineplot(data=df3['total kms'])
sns.lineplot(data=df4['total kms'])
plt.show()
```



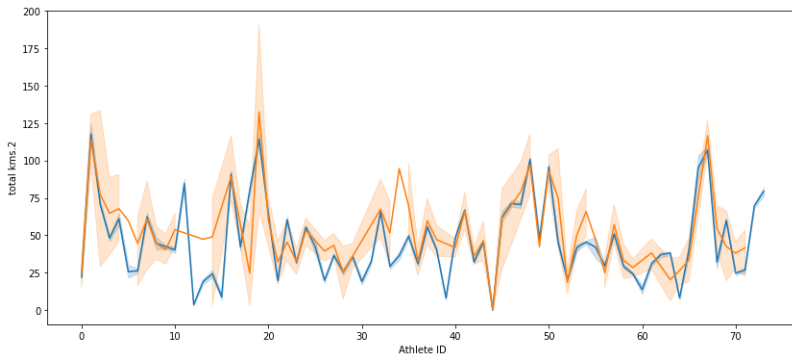
In [29]:

```
plt.figure(figsize = (14,6))
sns.lineplot(data=df3['total kms.1'])
sns.lineplot(data=df4['total kms.1'])
plt.show()
```



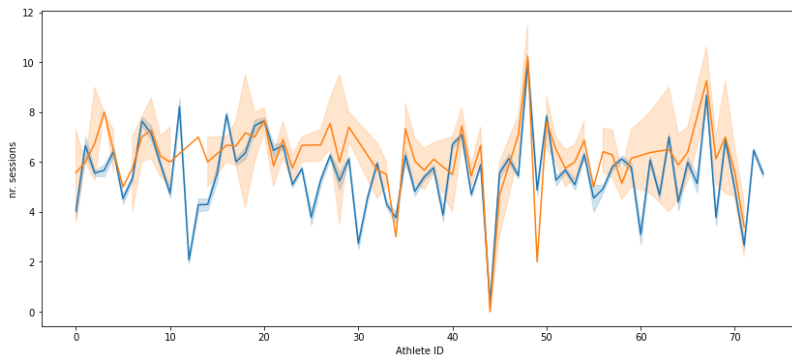
In [30]:

```
plt.figure(figsize = (14,6))
sns.lineplot(data=df3['total kms.2'])
sns.lineplot(data=df4['total kms.2'])
plt.show()
```



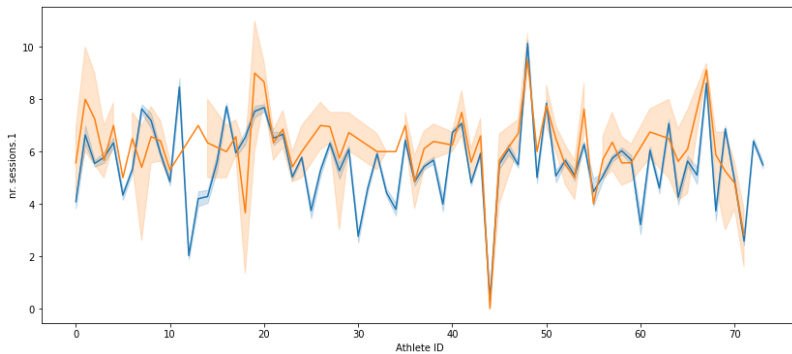
In [31]:

```
plt.figure(figsize = (14,6))
sns.lineplot(data=df3['nr. sessions'])
sns.lineplot(data=df4['nr. sessions'])
plt.show()
```



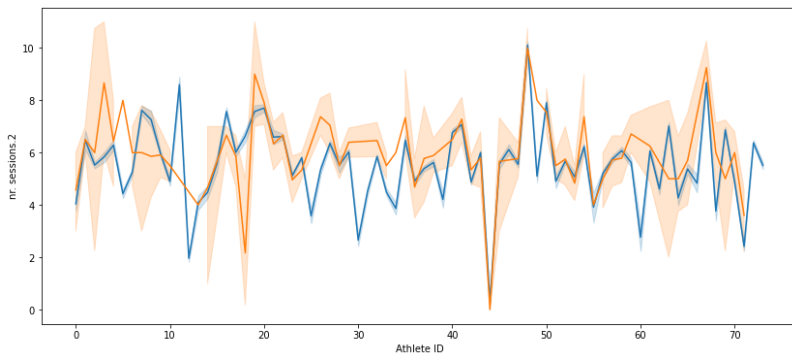
In [32]:

```
plt.figure(figsize = (14,6))
sns.lineplot(data=df3['nr. sessions.1'])
sns.lineplot(data=df4['nr. sessions.1'])
plt.show()
```



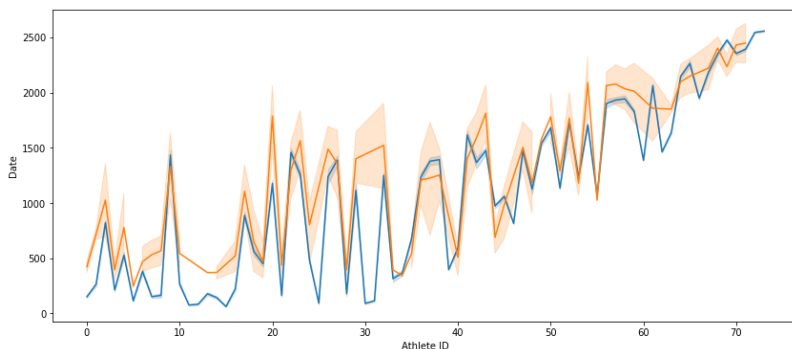
In [33]:

```
plt.figure(figsize = (14,6))
sns.lineplot(data=df3['nr. sessions.2'])
sns.lineplot(data=df4['nr. sessions.2'])
plt.show()
```



In [34]:

```
plt.figure(figsize = (14,6))
sns.lineplot(data=df3['Date'])
sns.lineplot(data=df4['Date'])
plt.show()
```



In [35]:

```
y = df2['injury']
X = df2.drop('injury', axis=1)
```

In [36]:

```
from imblearn.over_sampling import SMOTE
```

In [37]:

```
oversample = SMOTE()
X, y = oversample.fit_resample(X, y)
```

In [38]:

```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
```

In [39]:

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

In [40]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size = 0.3,
                                                    random_state = 0)
```

In [41]:

```
knn= KNeighborsClassifier(n_neighbors=5, metric='minkowski', p=2 )
knn.fit(X_train, y_train)
```

Out[41]:

```
▼ KNeighborsClassifier
KNeighborsClassifier()
```

In [42]:

```
y_pred_knn= knn.predict(X_test)
```

In [43]:

```
cm_knn = confusion_matrix(y_test, y_pred_knn)
```

In [44]:

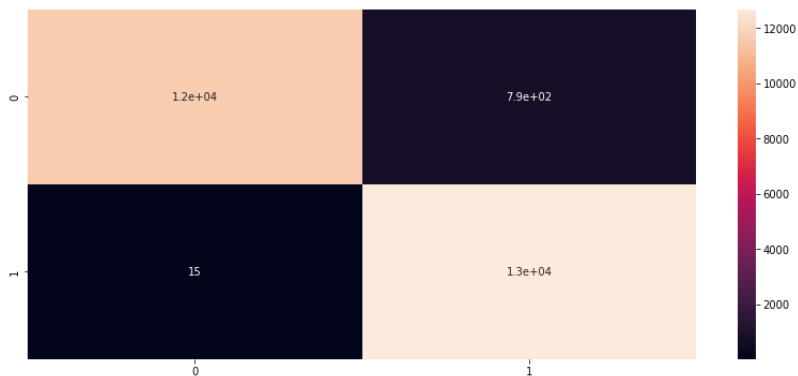
```
cm_knn
```

Out[44]:

```
array([[11635,  791],
       [  15, 12687]], dtype=int64)
```

In [45]:

```
plt.figure(figsize = (14,6))
sns.heatmap(cm_knn, annot = True)
plt.show()
```



In [46]:

```
print(classification_report(y_test, y_pred_knn))
```

	precision	recall	f1-score	support
0	1.00	0.94	0.97	12426
1	0.94	1.00	0.97	12702
accuracy			0.97	25128
macro avg	0.97	0.97	0.97	25128
weighted avg	0.97	0.97	0.97	25128

In [47]:

```
print(accuracy_score(y_test, y_pred_knn))
```

0.9679242279528812

In [48]:

```
lr = LogisticRegression(random_state=0)  
lr.fit(X_train, y_train)
```

Out[48]:

```
▼      LogisticRegression  
LogisticRegression(random_state=0)
```

In [49]:

```
y_pred_lr= lr.predict(X_test)
```

In [50]:

```
cm_lr = confusion_matrix(y_test, y_pred_lr)
```

In [51]:

```
cm_lr
```

Out[51]:

```
array([[ 9794, 2632],  
       [ 2339, 10363]], dtype=int64)
```

In [52]:

```
plt.figure(figsize = (14,6))
sns.heatmap(cm_lr, annot = True)
plt.show()
```



In [53]:

```
print(classification_report(y_test, y_pred_lr))
```

	precision	recall	f1-score	support
0	0.81	0.79	0.80	12426
1	0.80	0.82	0.81	12702
accuracy			0.80	25128
macro avg	0.80	0.80	0.80	25128
weighted avg	0.80	0.80	0.80	25128

In [54]:

```
print(accuracy_score(y_test, y_pred_lr))
```

0.8021728748806113

In [55]:

```
rf = RandomForestClassifier(n_estimators= 10, criterion="entropy")
rf.fit(X_train, y_train)
```

Out[55]:

```
RandomForestClassifier
RandomForestClassifier(criterion='entropy', n_estimators=10)
```

In [56]:

```
y_pred_rf= rf.predict(X_test)
```

In [57]:

```
cm_rf = confusion_matrix(y_test, y_pred_rf)
```

In [58]:

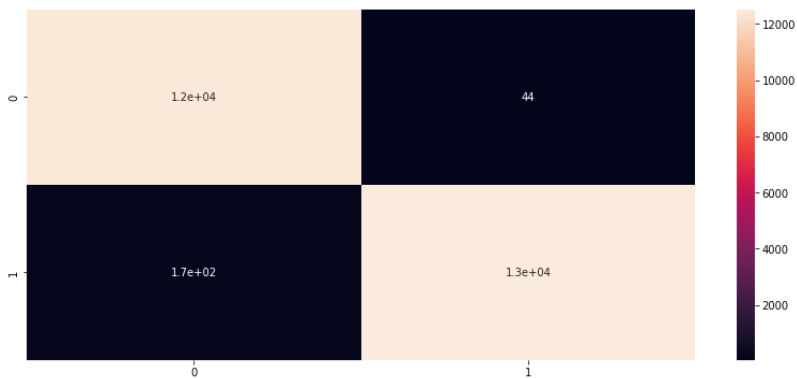
```
cm_rf
```

Out[58]:

```
array([[12382,    44],
       [ 174, 12528]], dtype=int64)
```

In [59]:

```
plt.figure(figsize = (14,6))
sns.heatmap(cm_rf, annot = True)
plt.show()
```



In [60]:

```
print(classification_report(y_test, y_pred_rf))
```

	precision	recall	f1-score	support
0	0.99	1.00	0.99	12426
1	1.00	0.99	0.99	12702
accuracy			0.99	25128
macro avg	0.99	0.99	0.99	25128
weighted avg	0.99	0.99	0.99	25128

In [61]:

```
print(accuracy_score(y_test, y_pred_rf))
```

0.9913244189748488