In [3]:
```
import pandas as pd
import matplotlib.pyplot as plt
```

In [4]:
```
train = pd.read_csv("train.csv", index_col="PassengerId")

print(train.shape)
train.head()
```

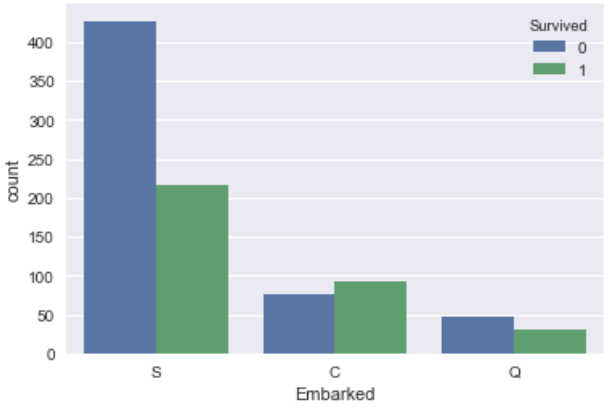
```
(891, 11)
```

Out[4]:

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

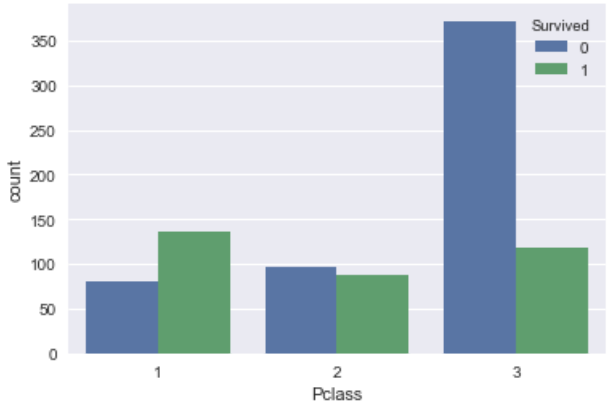
In [5]:
```
# matplotlib inline 노트북을 실행한 브라우저에서 바로 그림을 보여주기
%matplotlib inline

# seaborn
import seaborn as sns

#  hue : Survived 기준으로 쪼개어서 이를 시각화 해 준다.
# survived : 0,1을 기준으로 시각화를 나누어준다.
sns.countplot(data=train, x="Embarked", hue="Survived")
#sns.countplot(data=train, x="Embarked")
```

Out[5]: <matplotlib.axes._subplots.AxesSubplot at 0x1dd9cc924a8>



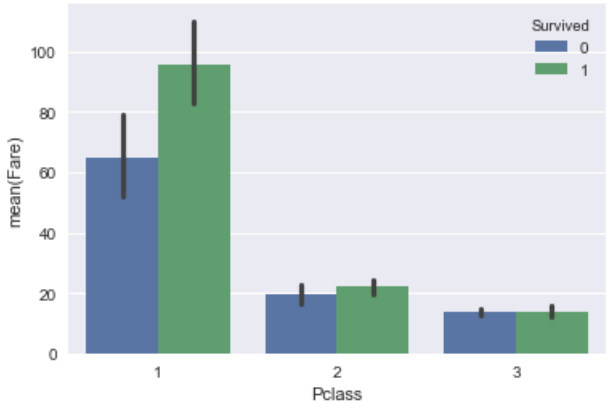
In [6]:
```
# seaborn의 버전마다 보이는 것이 약간 다를 수 있다.
# Survived 를 기준으로 클래스별 데이터 확인
sns.countplot(data=train, x="Pclass", hue="Survived")
```

Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x1dd9ccf9ac8>



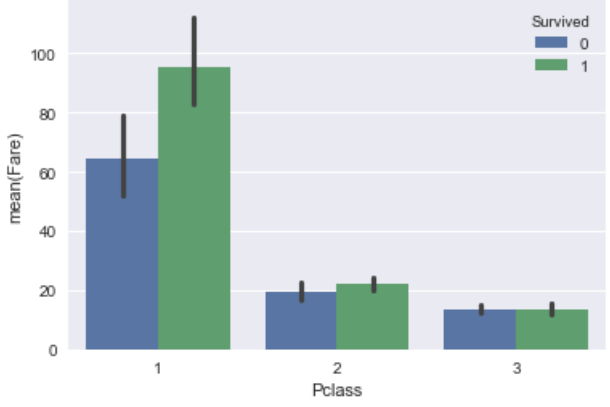
In [7]:
```
# barplot를 보겠다.
# barplot 는 y를 지정하면 정수이어야 하고,
# y의 평균값을 구하게 된다.
# Fare 요금의 평균값이 나오게 된다.
# 그리고 중앙의 선은 표준편차를 나타낸다.
sns.barplot(data=train, x="Pclass", y="Fare", hue="Survived")
```

Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x1dd9bbe2f98>



In [8]:
```
# hue를 이용하여 구하게 된다.
# 가운데는 표준편차를 나타나게 된다.
# bar가 길면 여러가지 데이터가 차이가 크다.
# 내가 풀려는 문제는 분류는 countplot를 주로 시각화하되 이를 나누는 것이 좋다.
# 내가 풀려는 문제는 Regression 만약 우리가 정수형을 맞출 목적은 barplot를 쓰고 y에 count를 쓰면 좋다.
# 가능한 y는 정수형이어야 한다.
sns.barplot(data=train, x="Pclass", y="Fare", hue="Survived")
```

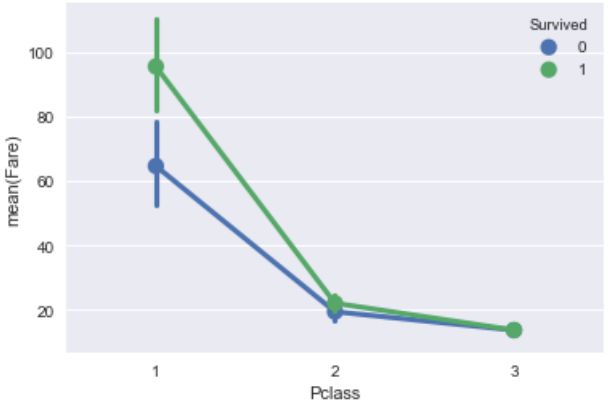
Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x1dd9cdeb9e8>



PointPlot

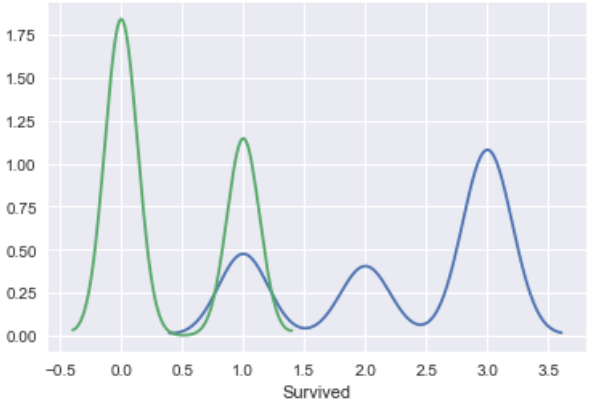
In [9]:
```
# pointplot은 barplot의 나머지 부분을
# 그대로 나머지를 붙여넣으면 된다.
# 그대로인데 모양만 다르다.

sns.pointplot(data=train, x="Pclass", y="Fare", hue="Survived")
```

Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x1dd9d0a4358>



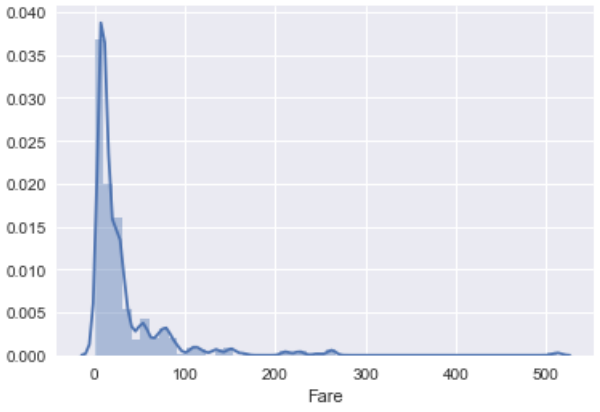
In [10]:
```
# pointplot은 barplot의 나머지 부분을
#sns.pointplot(data=train, x="Pclass", y="Fare", hue="Survived")

sns.distplot(train["Pclass"], hist=False)
sns.distplot(train["Survived"], hist=False)
```

Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x1dd9d12f9b0>



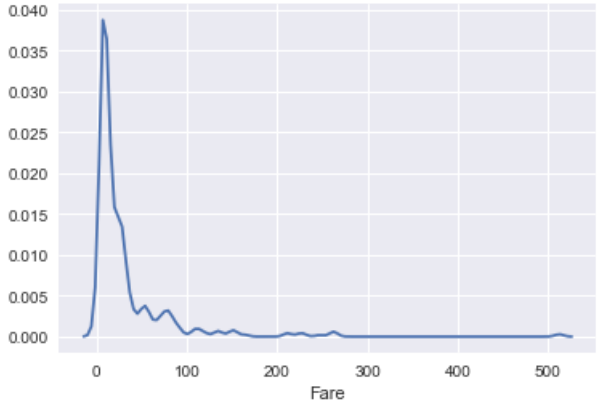
In [11]:
```
# distplot는 두개의 그래프를 합쳐놓은 것이다.
# barplot을 쓰고 pointplot를 쓰고 이런 순서로 확인해 본다.
# distplot은 넣을 때, 아예 컬럼을 넣게 된다.
sns.distplot(train["Fare"])
```

Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x1dd9d201cc0>



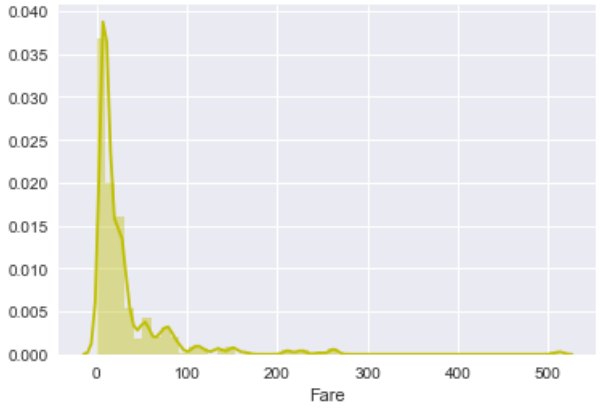
In [12]:
```
# barplot을 쓰고 pointplot를 쓰고 이런 순서로 확인해 본다.
# distplot은 넣을 때, 아예 컬럼을 넣게 된다.
# hist과 선이 보이게 되는데 hist를 없애 보겠다.
# 시각화의 중요한 것은 직관적이어야 한다.
# 직관적이 아니면 시각화가 아니다. 초등학생이 보더라도 직관적이어야 한다.#
# 바로 볼때, Action plan이 나오도록 시각화를 하는 것이 좋다.
# 만약 데이터가 좋지 않을 때도 있다. 이때 조금 시각화가 방해될 경우가 있다.
# 아웃라이어가 발생할 수 있다.
# 아웃라이어가 있어, 이를 빼고 한번 그려보자.
sns.distplot(train["Fare"], hist=False)
```

Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x1dd9d379a90>



In [13]:
```
# 기타 옵션 - vertical 수평
#sns.distplot(train["Fare"], vertical=True)

# 색 변경하기
sns.distplot(train["Fare"], color="y")
```

Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x1dd9d4d4898>



In [14]:
```
# 아웃라이어가 있어, 이를 빼고 한번 그려보자.
train[train["Fare"]<100 ]
```

Out[14]:

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q

In [15]:
```
# 진도를 빠르게 뽑고 싶다면
# 안보고 따라할 수 있다면 가장 Best이다.
# 새로운 것을 배울 때, 처음에는 안보고 칠 수 있도록 한다.
# 사람이 스마트해서 빨리 배운다.
# 판다스 문법 보고 하루 잡고 2,3번 치면 빨리 배운다.
# 데이터를 빨리 정리하고 중요하다.
low_fare = train[train["Fare"]<100]
print(low_fare.shape)
low_fare.head()
```

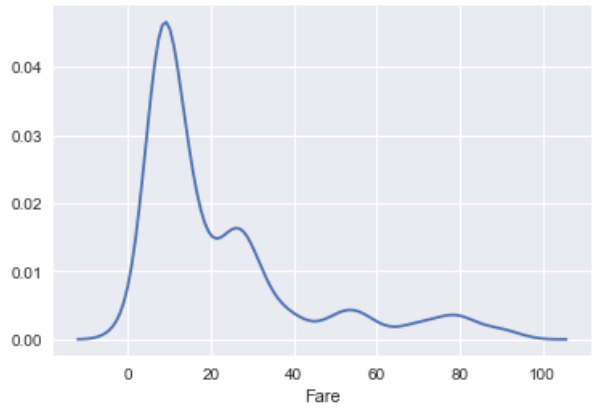
```
(838, 11)
```

Out[15]:

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

In [16]:
```
# Tool이 능숙하고 하면 더 좋아진다.
# 데이터로 전향하고 싶다면 Pandas에 익숙해 지는 것을 추천한다.
# 아웃라이어를 제거한 이후에 했을 때, 그래프가 훨씬 좋아졌다.
# 확실히 평균이 18정도인 것 같고, ...

sns.distplot(low_fare["Fare"], hist=False)
```

Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x1dd9d55d400>



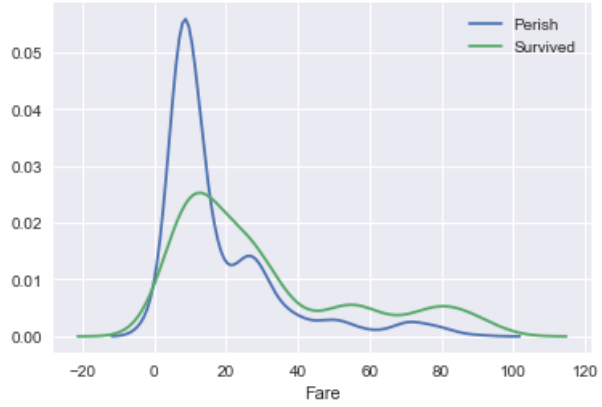
In [17]:
```
sum1 = low_fare["Fare"].sum()
print(low_fare["Fare"].describe())
print(low_fare["Fare_P"].describe())
print(sum1)
low_fare["Fare_P"] = low_fare["Fare"]/18781.2
# sns.countplot(data=low_fare, x="Fare")
# print(low_fare[["Fare","Fare_P" ]])
sns.distplot(low_fare["Fare"], hist=False)
```

```
count    838.000000
mean      22.411942
std       20.827218
min        0.000000
25%        7.895800
50%       13.000000
75%       27.720800
max       93.500000
Name: Fare, dtype: float64

---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
C:WAnaconda3WlibWsite-packagesWpandasWcoreWindexesWbase.py in get_loc(self, key, method, tolerance)
   2392             try:
-> 2393                 return self._engine.get_loc(key)
   2394             except KeyError:

pandasW_libsWindex.pyx in pandas._libs.index.IndexEngine.get_loc (pandasW_libsWindex.c:5239)()

pandasW_libsWindex.pyx in pandas._libs.index.IndexEngine.get_loc (pandasW_libsWindex.c:5085)()

pandasW_libsWhashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item (pandasW_libsWh
ashtable.c:20405)()

pandasW_libsWhashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item (pandasW_libsWh
ashtable.c:20359)()

KeyError: 'Fare_P'

During handling of the above exception, another exception occurred:

KeyError                                  Traceback (most recent call last)
<ipython-input-17-a4e298823cab> in <module>()
      2 sum1 = low_fare["Fare"].sum()
      3 print(low_fare["Fare"].describe())
----> 4 print(low_fare["Fare_P"].describe())
      5 print(sum1)
      6 low_fare["Fare_P"] = low_fare["Fare"]/18781.2

C:WAnaconda3WlibWsite-packagesWpandasWcoreWframe.py in __getitem__(self, key)
   2060             return self._getitem_multilevel(key)
   2061         else:
-> 2062             return self._getitem_column(key)
   2063
   2064     def _getitem_column(self, key):

C:WAnaconda3WlibWsite-packagesWpandasWcoreWframe.py in _getitem_column(self, key)
   2067         # get column
   2068         if self.columns.is_unique:
-> 2069             return self._get_item_cache(key)
   2070
   2071         # duplicate columns & possible reduce dimensionality

C:WAnaconda3WlibWsite-packagesWpandasWcoreWgeneric.py in _get_item_cache(self, item)
   1532         res = cache.get(item)
   1533         if res is None:
-> 1534             values = self._data.get(item)
   1535             res = self._box_item_values(item, values)
   1536             cache[item] = res

C:WAnaconda3WlibWsite-packagesWpandasWcoreWinternals.py in get(self, item, fastpath)
   3588
   3589             if not isnull(item):
-> 3590                 loc = self.items.get_loc(item)
   3591             else:
   3592                 indexer = np.arange(len(self.items))[isnull(self.items)]
```

```
C:₩Anaconda3₩lib₩site-packages₩pandas₩core₩indexes₩base.py in get_loc(self, key, method, tolerance)
   2393                 return self._engine.get_loc(key)
   2394             except KeyError:
-> 2395                 return self._engine.get_loc(self._maybe_cast_indexer(key))
   2396
   2397         indexer = self.get_indexer([key], method=method, tolerance=tolerance)

pandas₩_libs₩index.pyx in pandas._libs.index.IndexEngine.get_loc (pandas₩_libs₩index.c:5239)()

pandas₩_libs₩index.pyx in pandas._libs.index.IndexEngine.get_loc (pandas₩_libs₩index.c:5085)()

pandas₩_libs₩hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item (pandas₩_libs₩hashtable.c:20405)()

pandas₩_libs₩hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item (pandas₩_libs₩hashtable.c:20359)()

KeyError: 'Fare_P'
```

In [18]:
```
# 생존자와 사망자의 운임요금을 보고 싶다.
# distplot은 hue가 안된다.
# distplot은 넣을 때부터 컬럼을 하나 골라버린다. 그래서 hue를 넣을 수 없다.
# low_fare["Survived"]==0
perish = low_fare[low_fare["Survived"]==0]
survived = low_fare[low_fare["Survived"]==1]

# label를 통해 오른쪽 위쪽에 레이블을 넣을 수 있다.
sns.distplot(perish["Fare"], hist=False, label="Perish")
sns.distplot(survived["Fare"], hist=False, label="Survived")
```

Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x1dd9d722b38>

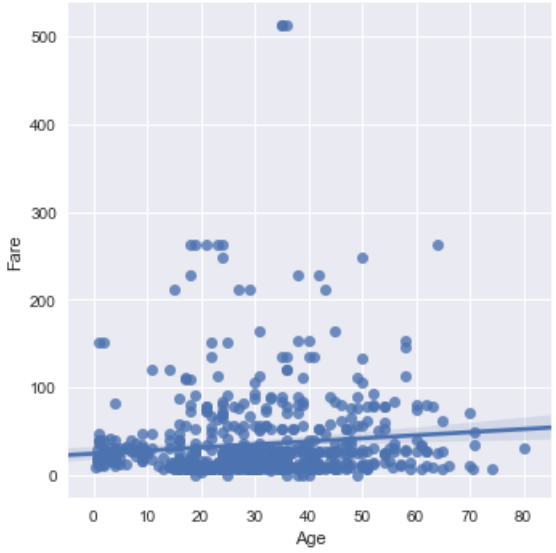


In [19]:
```
# countplot
# barplot
# distplot
```

Implot

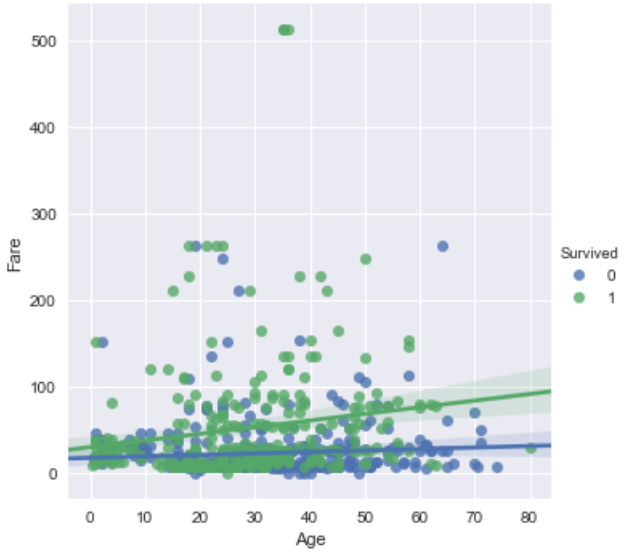
In [20]:
```
# 산점도 찍어보기 - ScatterPlot
# 나이와 요금에 대한 시각화
# x축, y축이 정수이어야 한다.
# x와 y의 값에 점을 하나 찍는다.
# 생존자와 사망자의 차이를 보는 것이 중요하다. hue이용가능
sns.lmplot(data=train, x="Age", y="Fare")
```

Out[20]: <seaborn.axisgrid.FacetGrid at 0x1dd9d6a5da0>



In [21]:
```
# 나이와 요금에 대한 시각화
# 생존자와 사망자의 차이를 보는 것이 중요하다. hue이용가능
# 파란색은 : 생존자,
# 선은 회귀선이다. 회귀선을 찍는데 데이터가 많아지면 조금 오래걸린다.
sns.lmplot(data=train, x="Age", y="Fare", hue="Survived")
```

Out[21]: <seaborn.axisgrid.FacetGrid at 0x1dd9d98fcc0>



In [22]:
```
# 나이와 요금에 대한 시각화
# 생존자와 사망자의 차이를 보는 것이 중요하다. hue이용가능
# 파란색은 : 생존자(1), 사망(0)
# 선은 회귀선이다. 회귀선을 지울려면 fit_reg를 이용한다.
sns.lmplot(data=train, x="Age", y="Fare", hue="Survived", fit_reg=False)
```

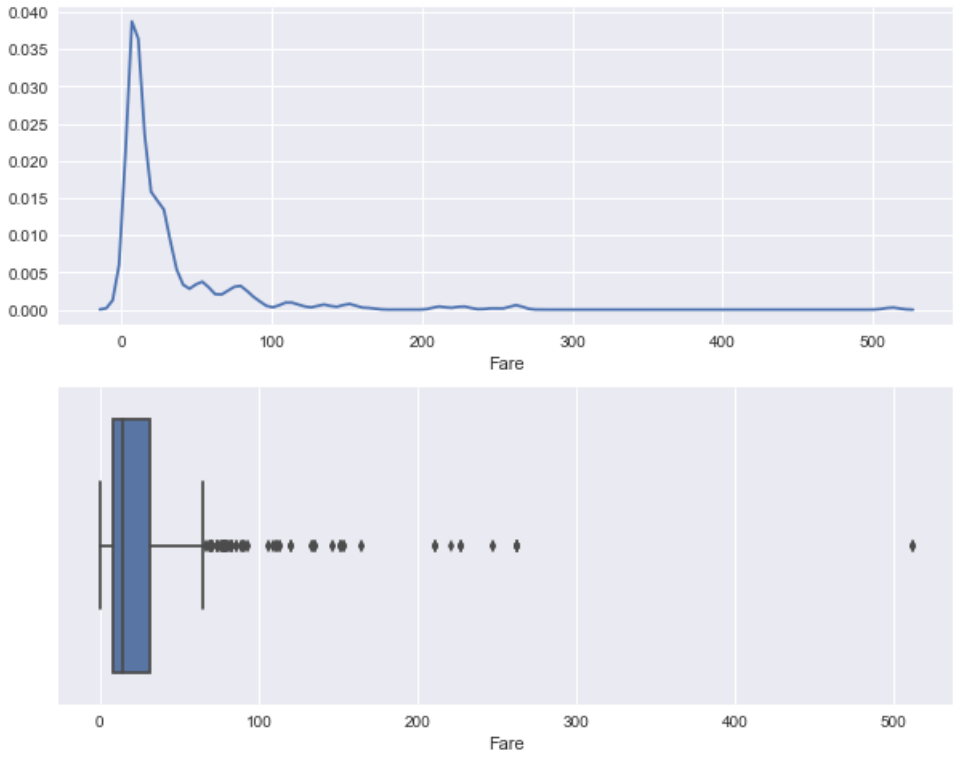
Out[22]: <seaborn.axisgrid.FacetGrid at 0x1dd9dcd3e48>

In [23]:
```
# 뭔가가 확 드러나지 않는다. scatter plot이다.
# 뭔가 500달러 이상이 뭔가 그렇다.
# 아웃라어이를 확인 후, 제거해 보자.
# python 아웃라이어 기준 :
# import matplotlib.pyplot as plt
print(train["Fare"].describe())
print(train["Fare"].quantile())
```

```
count    891.000000
mean      32.204208
std       49.693429
min        0.000000
25%        7.910400
50%       14.454200
75%       31.000000
max      512.329200
Name: Fare, dtype: float64
14.4542
```

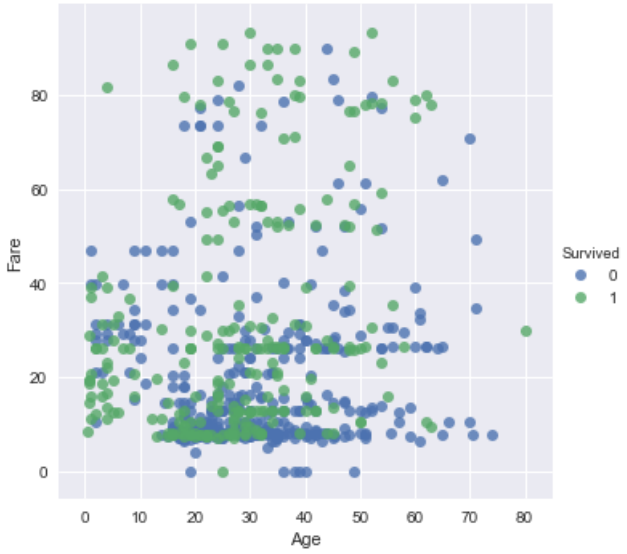
In [24]:
```
plt.figure(figsize=(10,8))
plt.subplot(211)
sns.distplot(train["Fare"], hist=False)

plt.subplot(212)
sns.boxplot(x=train["Fare"])
```

Out[24]: <matplotlib.axes._subplots.AxesSubplot at 0x1dd9de33a20>



In [25]:
```
# 뭔가가 확 드러나지 않는다. scatter plot이다.
# 뭔가 500달러 이상이 뭔가 그렇다.
# 아웃라어이를 제거해 보자.
low_fare = train[train["Fare"]<100]
sns.lmplot(data=low_fare, x="Age", y="Fare", hue="Survived", fit_reg=False)
```

Out[25]: <seaborn.axisgrid.FacetGrid at 0x1dd9dcf9f60>

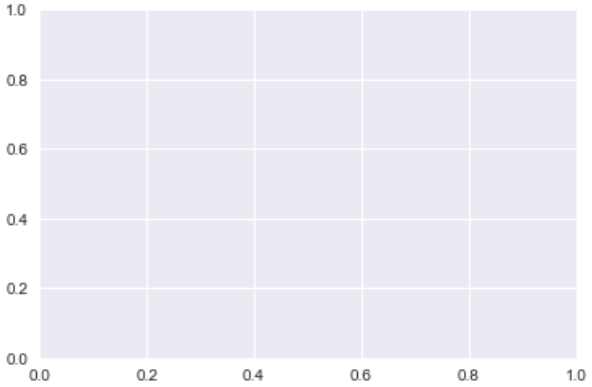


In [26]:
```
# 팁 : 시각화를 하기 전에 뭔가 내가 상상하고 그래프를 상상하고 그린다.
# 그런데 의외의 일들이 생긴다. 이 일에 대해서 뭔가 생겼을 때,
# 이에 대한 Hint를 찾을 수 있다.
# Age, Fare에 대해서 상상하고 하면 시각화를 하면 힌트가 발생한다.
# 타이타닉은 사망자가 생존자보다 많다.
# 5개의 plot를 보고 이를 더 전문적으로 하고 싶다면 Matplotlib를 하기를 추천
# 기본적으로 Seaborn으로 시각화를 하고 전문적으로 하고 싶다면 Matplotlib를 하기를 추천한다.
```

SubPlots

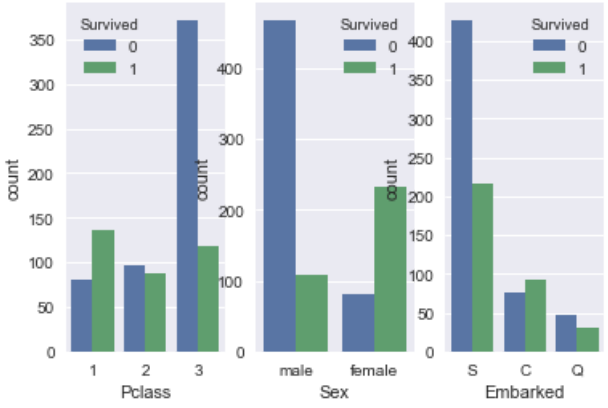
In [27]:
```
# subplot를 쓰면 하나의 컬럼에 여러개의 그래프를 한번에 그릴 수 있다.
import matplotlib.pyplot as plt
plt.subplots()
```

Out[27]:
```
(<matplotlib.figure.Figure at 0x1dd9dccbfd0>,
 <matplotlib.axes._subplots.AxesSubplot at 0x1dd9e079630>)
```



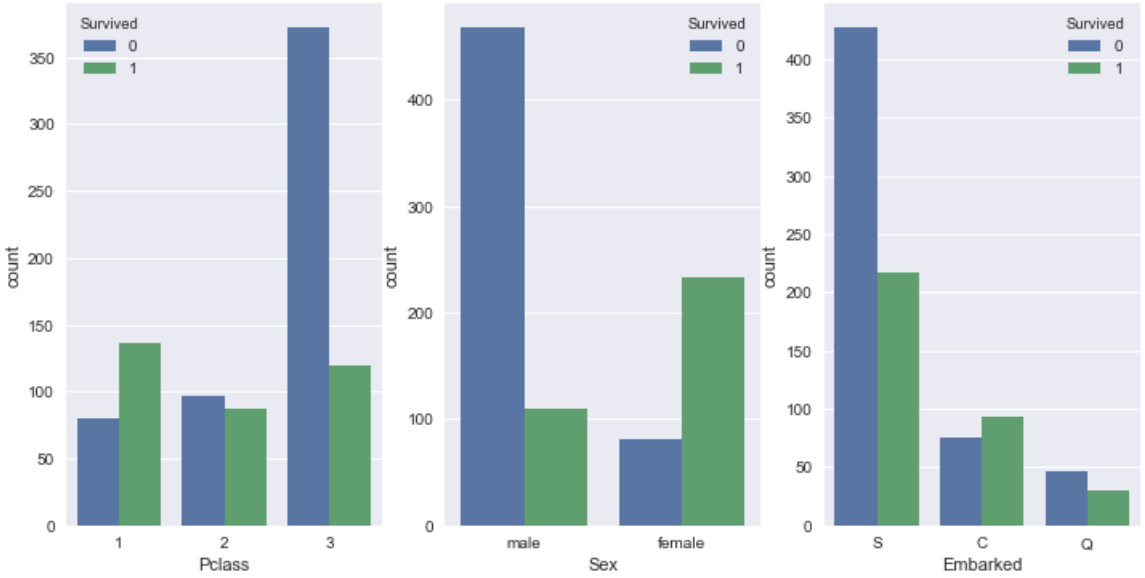
In [28]:
```
# 3개의 그래프를 한번에 볼 수 있다.
# 단점은 예쁘지 않다.
figure , (ax1, ax2, ax3) = plt.subplots(nrows=1, ncols=3)

sns.countplot(data=train, x="Pclass", hue="Survived", ax=ax1)
sns.countplot(data=train, x="Sex", hue="Survived", ax=ax2)
sns.countplot(data=train, x="Embarked", hue="Survived", ax=ax3)
```

Out[28]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1dd9dfdb550>
```



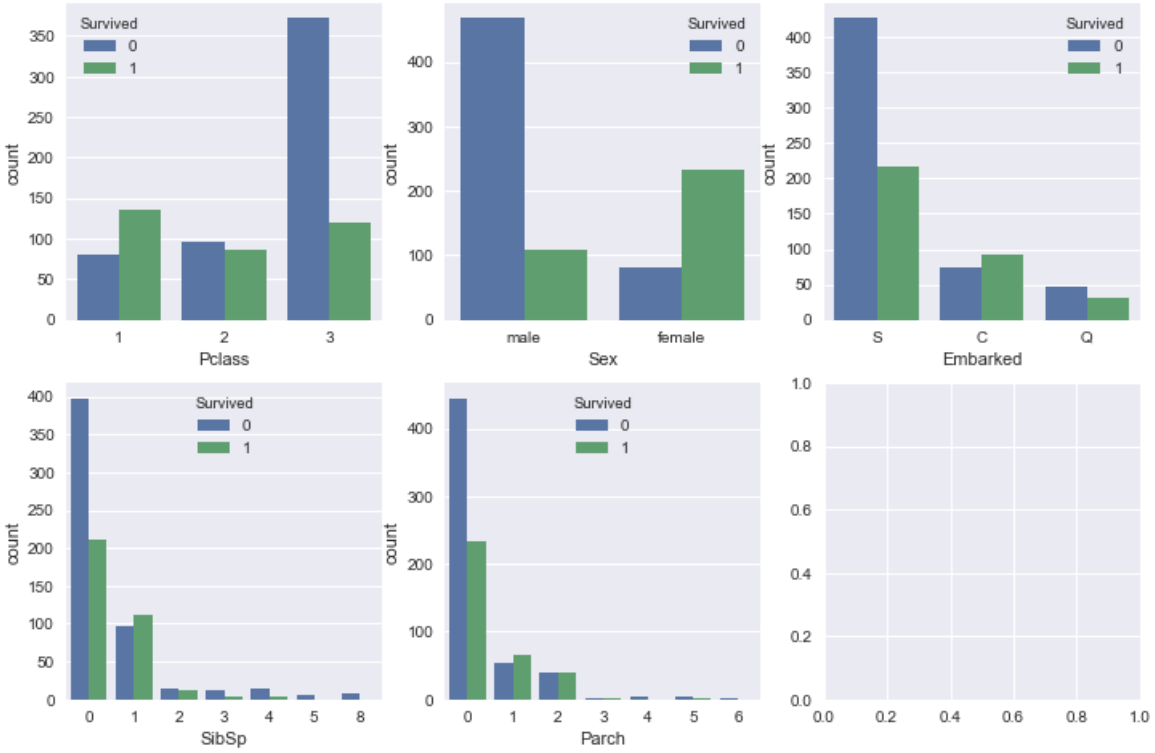
In [29]:
```
# 3개의 그래프를 한번에 볼 수 있다.
# 단점은 예쁘지 않다.
# 이 단점을 해결하기 위해 너비를 조절해 줄 필요가 있다.
# set_size_inches(x, y)
figure , (ax1, ax2, ax3) = plt.subplots(nrows=1, ncols=3)
figure.set_size_inches(12,6)

sns.countplot(data=train, x="Pclass", hue="Survived", ax=ax1)
sns.countplot(data=train, x="Sex", hue="Survived", ax=ax2)
sns.countplot(data=train, x="Embarked", hue="Survived", ax=ax3)
```

Out[29]: <matplotlib.axes._subplots.AxesSubplot at 0x1dd9f351a20>



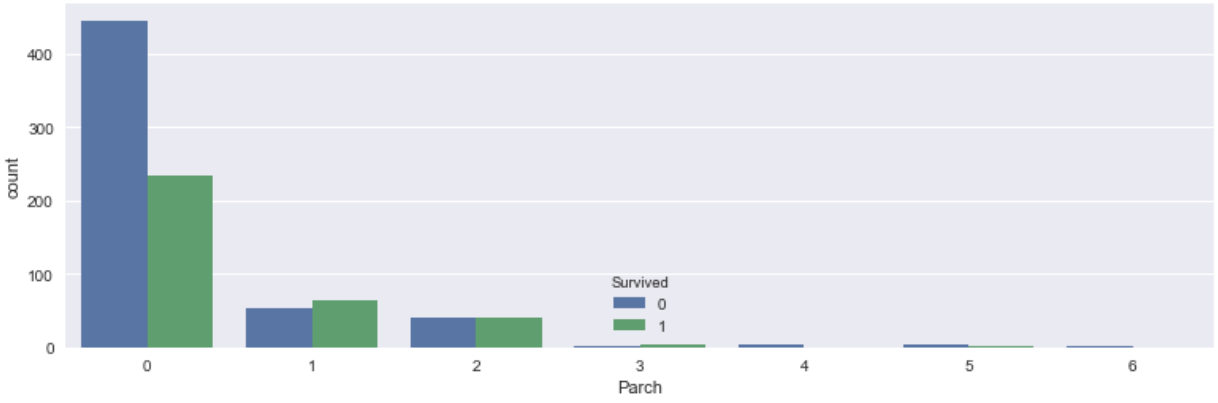
In [30]:
```
# 3개의 그래프를 한번에 볼 수 있다.
# 이제는 row를 두줄 로 해 보겠다.
# set_size_inches(x, y)
figure , ( (ax1, ax2, ax3) , (ax4, ax5, ax6) ) = plt.subplots(nrows=2, ncols=3)
figure.set_size_inches(12,8)

sns.countplot(data=train, x="Pclass", hue="Survived", ax=ax1)
sns.countplot(data=train, x="Sex", hue="Survived", ax=ax2)
sns.countplot(data=train, x="Embarked", hue="Survived", ax=ax3)
sns.countplot(data=train, x="SibSp", hue="Survived", ax=ax4)
sns.countplot(data=train, x="Parch", hue="Survived", ax=ax5)
```

Out[30]: <matplotlib.axes._subplots.AxesSubplot at 0x1dd9f673f60>



In [31]:
```
# col, row가 하나인 버전
figure, ax1 = plt.subplots(nrows=1, ncols=1)
figure.set_size_inches(13,4)
sns.countplot(data=train, x="Parch", hue="Survived", ax=ax1)
```

Out[31]: <matplotlib.axes._subplots.AxesSubplot at 0x1dd9f6bb198>



In [32]:
```
# 시각화를 하는데 불편함이 없다.
```