In [1]:
```
#라이브러리 싹 불러오기
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
data = pd.read_csv('data/trip.csv')
```

In [3]:
```
data.head()# 데이터 확인, 한번 전체적으로 확인하는 과정이 필요, 어떤 의미를 가지고 있는지 어떻
```

Out[3]:

	passenger_name	tpep_pickup_datetime	tpep_dropoff_datetime	payment_method	passe
0	Pamela Duffy	03/25/2017 8:55:43 AM	03/25/2017 9:09:47 AM	Debit Card	
1	Michelle Foster	04/11/2017 2:53:28 PM	04/11/2017 3:19:58 PM	Debit Card	
2	Tina Combs	12/15/2017 7:26:56 AM	12/15/2017 7:34:08 AM	Debit Card	
3	Anthony Ray	05/07/2017 1:17:59 PM	05/07/2017 1:48:14 PM	Cash	
4	Brianna Johnson	04/15/2017 11:32:20 PM	04/15/2017 11:49:03 PM	Debit Card	

In [4]:
```
data.info()#데이터 타입 확인, 시간 혹은 날짜 파악을 위해서 변환이 필요함을 파악, missing va
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22701 entries, 0 to 22700
Data columns (total 9 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   passenger_name         22701 non-null  object
 1   tpep_pickup_datetime   22701 non-null  object
 2   tpep_dropoff_datetime  22701 non-null  object
 3   payment_method         22701 non-null  object
 4   passenger_count        22701 non-null  int64
 5   trip_distance          22701 non-null  float64
 6   fare_amount            22698 non-null  float64
 7   tip_amount             22701 non-null  float64
 8   tolls_amount           22701 non-null  float64
dtypes: float64(4), int64(1), object(4)
memory usage: 1.6+ MB
```

In [5]:
```
data.describe() #통계적 정보 확인하기, outliner 확인할 것(e.g., 0?? max 36??, 요금이
```

Out[5]:

	passenger_count	trip_distance	fare_amount	tip_amount	tolls_amount
count	22701.000000	22701.000000	22698.000000	22701.000000	22701.000000
mean	1.643584	2.913400	13.024009	1.835745	0.312514
std	1.304942	3.653023	13.240074	2.800537	1.399153
min	0.000000	0.000000	-120.000000	0.000000	0.000000
25%	1.000000	0.990000	6.500000	0.000000	0.000000
50%	1.000000	1.610000	9.500000	1.350000	0.000000
75%	2.000000	3.060000	14.500000	2.450000	0.000000
max	36.000000	33.960000	999.990000	200.000000	19.100000

In [6]:
```
#데이터 복사본 남겨두기
import shutil

shutil.copy('data/trip.csv', 'data/trip_copy.csv')
```

```
---------------------------------------------------------------------------
OSError                                   Traceback (most recent call last)
/tmp/ipykernel_57/38989612.py in <module>
      2 import shutil
      3
----> 4 shutil.copy('data/trip.csv', 'data/trip_copy.csv')

/opt/conda/lib/python3.9/shutil.py in copy(src, dst, follow_symlinks)
    424     if os.path.isdir(dst):
    425         dst = os.path.join(dst, os.path.basename(src))
--> 426     copyfile(src, dst, follow_symlinks=follow_symlinks)
    427     copymode(src, dst, follow_symlinks=follow_symlinks)
    428     return dst

/opt/conda/lib/python3.9/shutil.py in copyfile(src, dst, follow_symlinks)
    263     else:
    264         try:
--> 265             with open(src, 'rb') as fsrc, open(dst, 'wb') as fdst:
    266                 # macOS
    267                 if _HAS_FCOPYFILE:

OSError: [Errno 30] Read-only file system: 'data/trip_copy.csv'
```

In [7]:
```
data.duplicated()
```

Out[7]:
```
0        False
1        False
2        False
3        False
4        False
         ...
22696    False
22697    False
22698    False
22699    False
22700    False
Length: 22701, dtype: bool
```

In [8]:
```
data[data.duplicated()]v #해당 부분 상세적으로 확인하기
```

Out[8]:

	passenger_name	tpep_pickup_datetime	tpep_dropoff_datetime	payment_method	pas
17	Sarah Gross	08/15/2017 7:48:08 PM	08/15/2017 8:00:37 PM	Cash	
204	Lisa Bullock	02/13/2017 4:25:41 PM	02/13/2017 4:55:35 PM	Cash	

In [9]:
```
data[data['passenger_name'] == 'Sarah Gross'] #중복 데이터 확인
```

Out[9]:

	passenger_name	tpep_pickup_datetime	tpep_dropoff_datetime	payment_method	pass
16	Sarah Gross	08/15/2017 7:48:08 PM	08/15/2017 8:00:37 PM	Cash	
17	Sarah Gross	08/15/2017 7:48:08 PM	08/15/2017 8:00:37 PM	Cash	

In [10]:
```
data = data.drop_duplicates() #중복 데이터 제거하기
```

In [11]:
```
data #중복된 데이터가 삭제된 것을 있는 것을 확인할 수 있음
```

Out[11]:

	passenger_name	tpep_pickup_datetime	tpep_dropoff_datetime	payment_method	p
0	Pamela Duffy	03/25/2017 8:55:43 AM	03/25/2017 9:09:47 AM	Debit Card	
1	Michelle Foster	04/11/2017 2:53:28 PM	04/11/2017 3:19:58 PM	Debit Card	
2	Tina Combs	12/15/2017 7:26:56 AM	12/15/2017 7:34:08 AM	Debit Card	
3	Anthony Ray	05/07/2017 1:17:59 PM	05/07/2017 1:48:14 PM	Cash	
4	Brianna Johnson	04/15/2017 11:32:20 PM	04/15/2017 11:49:03 PM	Debit Card	
...	...	...	...	...	
22696	Austin Johnson	02/24/2017 5:37:23 PM	02/24/2017 5:40:39 PM	Cash	
22697	Monique Williams	08/06/2017 4:43:59 PM	08/06/2017 5:24:47 PM	Cash	
22698	Drew Graves	09/04/2017 2:54:14 PM	09/04/2017 2:58:22 PM	Debit Card	
22699	Jonathan Copeland	07/15/2017 12:56:30 PM	07/15/2017 1:08:26 PM	Debit Card	
22700	Benjamin Miller	03/02/2017 1:02:49 PM	03/02/2017 1:16:09 PM	Cash	

22699 rows × 9 columns

In [12]:
```
data.isna().sum() #결측치 확인, 몇 건 있는지 확인
```

Out[12]:
```
passenger_name           0
tpep_pickup_datetime     0
tpep_dropoff_datetime    0
payment_method           0
passenger_count          0
trip_distance            0
fare_amount              3
tip_amount               0
tolls_amount             0
dtype: int64
```

In [13]:
```
data.isna().mean() #mean이 굉장히 미비한 편, 데이터가 충분하고 결측치가 많지 않다면 그냥 3
```

Out[13]:
```
passenger_name           0.000000
tpep_pickup_datetime     0.000000
tpep_dropoff_datetime    0.000000
payment_method           0.000000
passenger_count          0.000000
trip_distance            0.000000
fare_amount              0.000132
tip_amount               0.000000
tolls_amount             0.000000
dtype: float64
```

In [14]:
```
data.dropna() #결측치 3건 빼기
```

Out[14]:

	passenger_name	tpep_pickup_datetime	tpep_dropoff_datetime	payment_method	p
0	Pamela Duffy	03/25/2017 8:55:43 AM	03/25/2017 9:09:47 AM	Debit Card	
1	Michelle Foster	04/11/2017 2:53:28 PM	04/11/2017 3:19:58 PM	Debit Card	
2	Tina Combs	12/15/2017 7:26:56 AM	12/15/2017 7:34:08 AM	Debit Card	
3	Anthony Ray	05/07/2017 1:17:59 PM	05/07/2017 1:48:14 PM	Cash	
4	Brianna Johnson	04/15/2017 11:32:20 PM	04/15/2017 11:49:03 PM	Debit Card	
...	...	...	...	...	...
22696	Austin Johnson	02/24/2017 5:37:23 PM	02/24/2017 5:40:39 PM	Cash	
22697	Monique Williams	08/06/2017 4:43:59 PM	08/06/2017 5:24:47 PM	Cash	
22698	Drew Graves	09/04/2017 2:54:14 PM	09/04/2017 2:58:22 PM	Debit Card	
22699	Jonathan Copeland	07/15/2017 12:56:30 PM	07/15/2017 1:08:26 PM	Debit Card	
22700	Benjamin Miller	03/02/2017 1:02:49 PM	03/02/2017 1:16:09 PM	Cash	

22696 rows × 9 columns

In [15]:
```
data = data.dropna() #따로 저장하지 않으면 빠진 값 저장되지 않음
```

In [16]:
```
data.isna().mean() #결측치 제대로 처리 되었는지 다시 한 번 확인
```

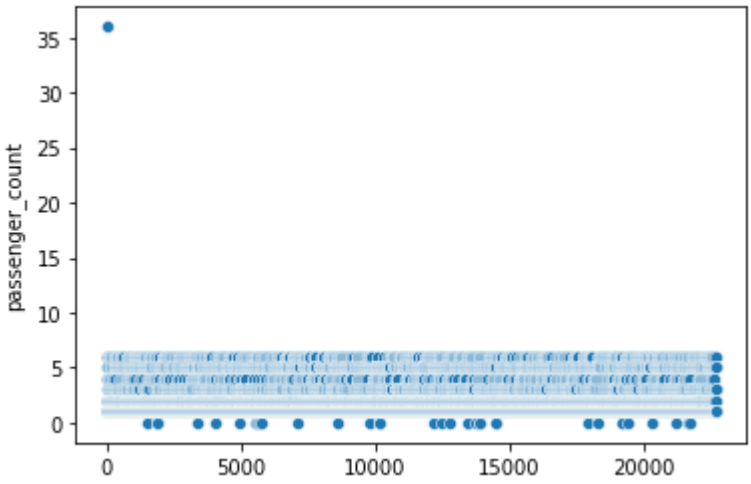
Out[16]:
```
passenger_name           0.0
tpep_pickup_datetime     0.0
tpep_dropoff_datetime    0.0
payment_method           0.0
passenger_count          0.0
trip_distance            0.0
fare_amount              0.0
tip_amount               0.0
tolls_amount             0.0
dtype: float64
```

In [17]:
```
#데이터 직접 확인해보았을 때 의구심이 들었던 컬럼을 중심으로
data['passenger_count'].sort_values()
#36이 갑자기 튀어나온 것 확인할 수 있음, visualisation으로 한번 더 확인해보기
```

Out[17]:
```
12804     0
19458     0
5565      0
5670      0
13718     0
         ..
416       6
4322      6
14500     6
0         6
64       36
Name: passenger_count, Length: 22696, dtype: int64
```

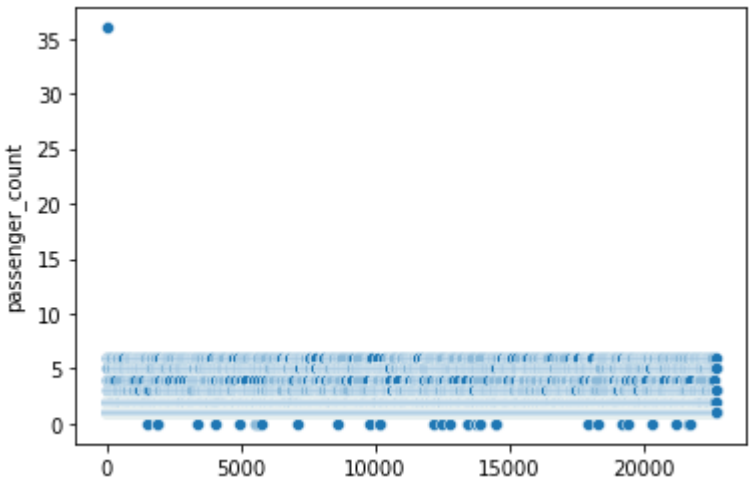
In [18]:
```
sns.scatterplot(x = data.index, y = data['passenger_count'])
#36에 outliner가 있는거 확인 가능함
```

Out[18]:
```
<AxesSubplot:ylabel='passenger_count'>
```



In [19]:
```
sns.scatterplot(x = data.index, y = data['passenger_count'])
```

Out[19]:
```
<AxesSubplot:ylabel='passenger_count'>
```



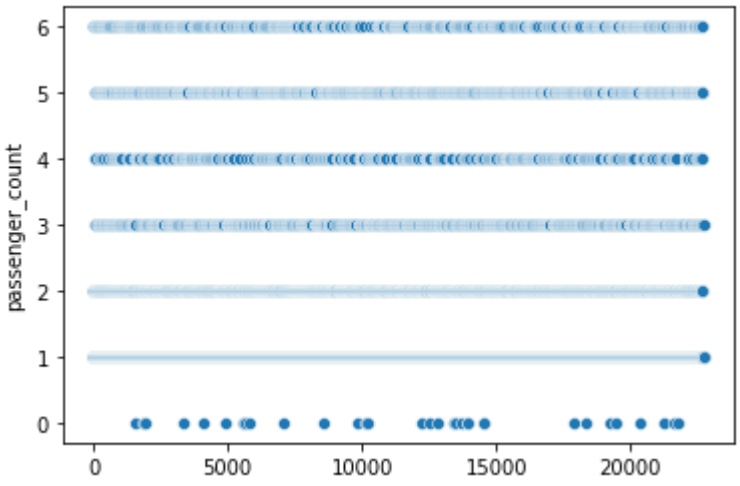
In [20]:
```
# passenger_count 컬럼의 이상치를 제거합니다.
# (passenger_count가 6을 초과하는 경우)
data = data[data['passenger_count'] <= 6]
```

In [21]:
```
len(data[data['passenger_count'] == 0])
```

Out[21]:
```
33
```

In [22]:
```
sns.scatterplot(x = data.index, y = data['passenger_count'])
```

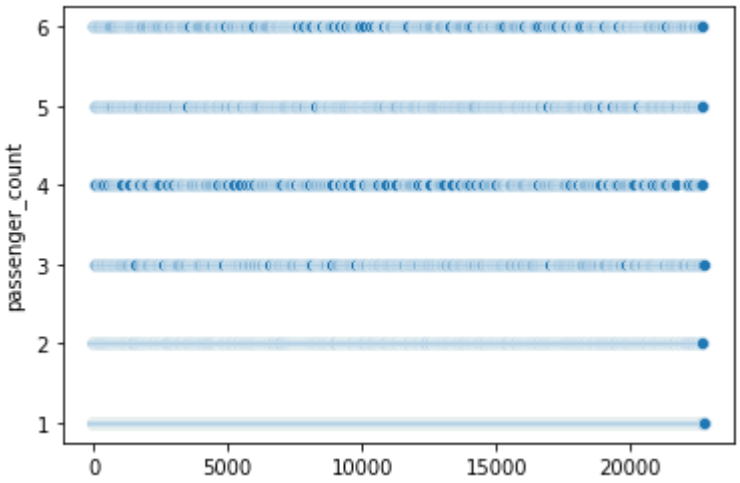
Out[22]:
```
<AxesSubplot:ylabel='passenger_count'>
```



In [23]:
```
data = data[data['passenger_count'] != 0] # 0 부근에 있는 이상치를 제거해줌
```

In [24]:
```
sns.scatterplot(x = data.index, y = data['passenger_count']) #데이터가 조금 정리
```

Out[24]:
```
<AxesSubplot:ylabel='passenger_count'>
```



In [25]:
```
data[data['trip_distance'] == 0] #147건 있는 것을 확인할 수 있음
```

Out[25]:

	passenger_name	tpep_pickup_datetime	tpep_dropoff_datetime	payment_method	p
129	Linda Kaufman	06/22/2017 8:05:33 AM	06/22/2017 8:05:40 AM	Debit Card	
248	Erik Perez	09/18/2017 8:50:53 PM	09/18/2017 8:51:03 PM	Cash	
293	Deborah Sanford	10/04/2017 7:46:24 PM	10/04/2017 7:46:50 PM	Cash	
321	Ryan Hughes	02/22/2017 4:01:44 AM	02/22/2017 4:01:53 AM	Cash	
426	David Parker	01/14/2017 7:00:26 AM	01/14/2017 7:00:53 AM	Cash	
...	...	...	...	...	
22192	Angela French	10/16/2017 8:34:07 AM	10/16/2017 8:34:10 AM	Credit Card	
22327	Kelsey Rogers	07/21/2017 11:30:29 PM	07/21/2017 11:31:12 PM	Debit Card	
22385	Joseph Castillo	01/07/2017 4:48:42 AM	01/07/2017 4:51:03 AM	Cash	
22568	Christine Edwards	03/07/2017 2:24:47 AM	03/07/2017 2:24:50 AM	Credit Card	
22672	John Erickson	03/03/2017 11:09:16 PM	03/03/2017 11:09:35 PM	Debit Card	

147 rows × 9 columns

In [26]:
```
# Q. trip_distance의 이상치를 확인합니다.
data['trip_distance'].sort_values() #자연스러운 분포의 끝자락, 혹은 outliner? 아님)
```

Out[26]:
```
3764      0.00
13064     0.00
5620      0.00
1277      0.00
5632      0.00
         ...
30       30.83
10293    31.95
6066     32.72
13863    33.92
9282     33.96
Name: trip_distance, Length: 22662, dtype: float64
```

In [27]:
```
data.sort_values('trip_distance') #전체 데이터를 모두 열어서 trip_distance 확인
#정렬이 뒤죽박죽임,,
```

Out[27]:

	passenger_name	tpep_pickup_datetime	tpep_dropoff_datetime	payment_method	p
3764	Tiffany Washington DDS	04/28/2017 8:43:59 PM	04/28/2017 8:44:08 PM	Cash	
13064	Dylan Olson	10/10/2017 9:53:00 AM	10/10/2017 9:53:00 AM	Cash	
5620	Angela Webb	03/07/2017 6:02:37 AM	03/07/2017 6:03:31 AM	Cash	
1277	Joseph Aguilar	02/28/2017 5:46:44 AM	02/28/2017 5:46:49 AM	Credit Card	
5632	Jacqueline Allison	01/29/2017 8:16:21 PM	01/29/2017 8:16:21 PM	Debit Card	
...	...	...	...	...	...
30	David Burton	11/06/2017 8:30:50 PM	11/07/2017 12:00:00 AM	Credit Card	
10293	Emily Stevens	09/11/2017 11:41:04 AM	09/11/2017 12:18:58 PM	Cash	
6066	Tina Knight	06/13/2017 12:30:22 PM	06/13/2017 1:37:51 PM	Debit Card	
13863	William Yates	05/19/2017 8:20:21 AM	05/19/2017 9:20:30 AM	Credit Card	
9282	Samantha Frederick	06/18/2017 11:33:25 PM	06/19/2017 12:12:38 AM	Cash	

22662 rows × 9 columns

In [28]:
```
data.sort_values('trip_distance').iloc[150:170]
#순서가 어차피 뒤죽박죽이니 iloc; 0.xxx 으로 처리해주기
```

Out[28]:

	passenger_name	tpep_pickup_datetime	tpep_dropoff_datetime	payment_method	p
9190	Valerie Vasquez	03/31/2017 5:29:19 AM	03/31/2017 5:29:32 AM	Cash	
3611	James Chen	11/19/2017 7:17:16 AM	11/19/2017 7:17:19 AM	Cash	
5503	Mike Bishop	08/08/2017 11:28:54 PM	08/08/2017 11:29:00 PM	Credit Card	
19646	Michael Solomon	12/13/2017 12:19:29 PM	12/13/2017 12:19:39 PM	Credit Card	
5760	Samuel Cooper	01/03/2017 8:15:23 PM	01/03/2017 8:15:39 PM	Debit Card	
325	Valerie Mullen	01/14/2017 7:04:51 PM	01/14/2017 7:05:01 PM	Cash	
14470	Leah Carrillo	09/09/2017 1:29:37 PM	09/09/2017 1:29:57 PM	Credit Card	
16829	Jeffrey Jackson	05/02/2017 12:18:59 AM	05/02/2017 12:19:02 AM	Credit Card	
13496	Amber Boyd	01/15/2017 5:04:18 AM	01/15/2017 5:04:21 AM	Cash	
15348	Michael Ferguson	01/17/2017 1:18:24 PM	01/17/2017 1:18:31 PM	Debit Card	
16351	Nathan Salazar	05/13/2017 5:42:22 PM	05/13/2017 5:42:45 PM	Cash	
19371	Amanda Taylor	03/24/2017 8:59:58 PM	03/24/2017 9:00:06 PM	Cash	
4543	Tammy Hansen	01/10/2017 6:25:47 PM	01/10/2017 6:42:09 PM	Debit Card	
5431	Jeffrey Sullivan	12/09/2017 11:56:56 AM	12/09/2017 11:58:13 AM	Cash	
20135	Aaron Montoya	07/03/2017 3:00:37 PM	07/03/2017 3:00:58 PM	Credit Card	
3160	Beth Young	03/16/2017 5:51:31 AM	03/16/2017 5:51:43 AM	Cash	
13633	Jamie Williams	01/02/2017 7:16:30 PM	01/02/2017 7:19:30 PM	Debit Card	
21090	Renee Garza	08/13/2017 4:09:35 PM	08/13/2017 4:10:56 PM	Credit Card	
13994	Calvin Guzman	08/14/2017 10:03:24 PM	08/14/2017 10:03:35 PM	Debit Card	
1512	Amy Robertson	03/25/2017 4:37:43 AM	03/25/2017 4:37:49 AM	Credit Card	

In [29]:
```
data = data[data['trip_distance'] != 0] #trip_distance가 0 인 경우 삭제
```

In [30]:
```
data.sort_values('trip_distance')
```

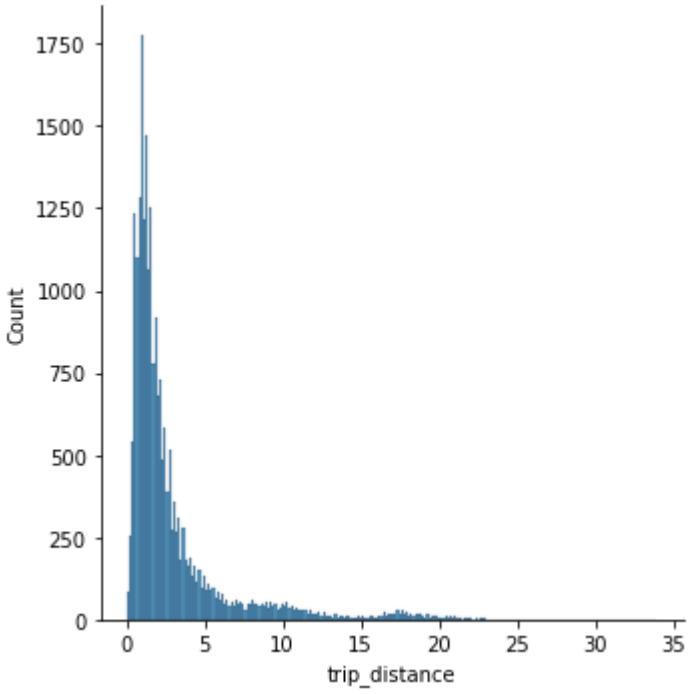
Out[30]:

	passenger_name	tpep_pickup_datetime	tpep_dropoff_datetime	payment_method	p
2987	Christine Harper	11/24/2017 4:32:18 AM	11/24/2017 4:32:23 AM	Credit Card	
19646	Michael Solomon	12/13/2017 12:19:29 PM	12/13/2017 12:19:39 PM	Credit Card	
8199	Steven Brooks	05/16/2017 1:33:23 PM	05/16/2017 1:33:37 PM	Cash	
5503	Mike Bishop	08/08/2017 11:28:54 PM	08/08/2017 11:29:00 PM	Credit Card	
3611	James Chen	11/19/2017 7:17:16 AM	11/19/2017 7:17:19 AM	Cash	
...	...	...	...	...	...
30	David Burton	11/06/2017 8:30:50 PM	11/07/2017 12:00:00 AM	Credit Card	
10293	Emily Stevens	09/11/2017 11:41:04 AM	09/11/2017 12:18:58 PM	Cash	
6066	Tina Knight	06/13/2017 12:30:22 PM	06/13/2017 1:37:51 PM	Debit Card	
13863	William Yates	05/19/2017 8:20:21 AM	05/19/2017 9:20:30 AM	Credit Card	
9282	Samantha Frederick	06/18/2017 11:33:25 PM	06/19/2017 12:12:38 AM	Cash	

22515 rows × 9 columns

In [31]:
```
sns.displot(data['trip_distance'])#데이터를 플롯하여서 확인하기, outliner가 아니라 tr
```

Out[31]:
```
<seaborn.axisgrid.FacetGrid at 0x76b8c615b190>
```



In [32]:
```
data.describe() #0 혹은 -120 등 fare_amount 값에 이상치가 표함되어 있는 것을 확인할 수
```

Out[32]:

	passenger_count	trip_distance	fare_amount	tip_amount	tolls_amount
count	22515.000000	22515.000000	22515.000000	22515.000000	22515.000000
mean	1.645969	2.931924	12.958055	1.829513	0.309625
std	1.285783	3.657290	12.701799	2.767054	1.387300
min	1.000000	0.010000	-120.000000	0.000000	0.000000
25%	1.000000	1.000000	6.500000	0.000000	0.000000
50%	1.000000	1.630000	9.500000	1.360000	0.000000
75%	2.000000	3.090000	14.500000	2.450000	0.000000
max	6.000000	33.960000	999.990000	200.000000	19.100000

In [33]:
```
data[data['fare_amount'] < 0] # 셀 수 있을 정도인 것을 확인함
```

Out[33]:

	passenger_name	tpep_pickup_datetime	tpep_dropoff_datetime	payment_method	p
316	Tiffany Johnson	12/13/2017 2:02:39 AM	12/13/2017 2:03:08 AM	Cash	
1648	Debbie Holmes	07/05/2017 11:02:23 AM	07/05/2017 11:03:00 AM	Credit Card	
4425	Bobby Wilson	11/16/2017 8:13:30 PM	11/16/2017 8:14:50 PM	Cash	
5450	Alejandro Williams	04/06/2017 12:50:26 PM	04/06/2017 12:52:39 PM	Debit Card	
5760	Samuel Cooper	01/03/2017 8:15:23 PM	01/03/2017 8:15:39 PM	Debit Card	
8206	Stephanie Summers	10/28/2017 8:39:36 PM	10/28/2017 8:41:59 PM	Credit Card	
11206	Austin Fields	07/09/2017 7:20:59 AM	07/09/2017 7:23:50 AM	Debit Card	
12946	Patrick Herring	04/08/2017 12:00:16 AM	04/08/2017 11:15:57 PM	Cash	
14716	Stefanie Warner	12/24/2017 10:37:58 PM	12/24/2017 10:41:08 PM	Debit Card	
17604	Tyler Lowe	03/24/2017 7:31:13 PM	03/24/2017 7:34:49 PM	Cash	
18567	Selena Mann	05/22/2017 3:51:20 PM	05/22/2017 3:52:22 PM	Cash	
20319	Tyler Robinson	09/09/2017 10:59:51 PM	09/09/2017 11:02:06 PM	Debit Card	
20700	Nicole Pierce	02/24/2017 12:38:17 AM	02/24/2017 12:42:05 AM	Cash	

In [34]:
```
len(data[data['fare_amount'] < 0 ]) #해당 갯수가 몇개인지를 체크, 13건이니 그냥 바로 저
```

Out[34]:
```
13
```

In [35]:
```
data = data[data['fare_amount'] > 0 ] #0보다 큰 것만 남기자, 덮어쓰면서
```

In [36]:
```
# Q. fare_amount의 이상치 데이터 개수를 확인합니다.
# (fare_amount가 0 이하인 경우)
data.sort_values('fare_amount') #999는 이상한 값이고 이것이 확인되었음
```

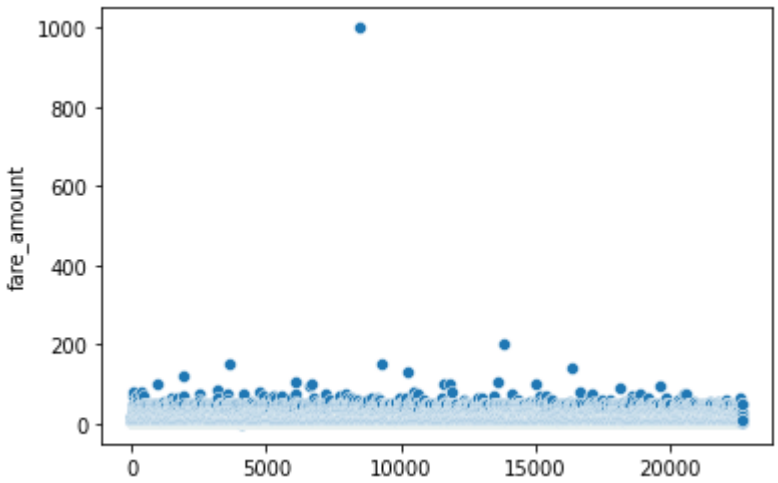
Out[36]:

	passenger_name	tpep_pickup_datetime	tpep_dropoff_datetime	payment_method	p
4063	Phillip Gonzalez	08/12/2017 8:49:29 PM	08/12/2017 9:18:50 PM	Cash	
14470	Leah Carrillo	09/09/2017 1:29:37 PM	09/09/2017 1:29:57 PM	Credit Card	
2987	Christine Harper	11/24/2017 4:32:18 AM	11/24/2017 4:32:23 AM	Credit Card	
16351	Nathan Salazar	05/13/2017 5:42:22 PM	05/13/2017 5:42:45 PM	Cash	
6702	Yvonne Brooks	08/26/2017 7:33:22 AM	08/26/2017 7:34:18 AM	Debit Card	
...	...	...	...	...	
16381	Erica Hernandez	11/30/2017 10:41:11 AM	11/30/2017 11:31:45 AM	Cash	
9282	Samantha Frederick	06/18/2017 11:33:25 PM	06/19/2017 12:12:38 AM	Cash	
3584	Matthew Chavez	01/01/2017 11:53:01 PM	01/01/2017 11:53:42 PM	Credit Card	
13863	William Yates	05/19/2017 8:20:21 AM	05/19/2017 9:20:30 AM	Credit Card	
8478	Alexis Hanson	02/06/2017 5:50:10 AM	02/06/2017 5:51:08 AM	Credit Card	

22499 rows × 9 columns

In [37]:
```
# Q. fare_amount의 scatter plot을 그립니다.

sns.scatterplot(x = data.index, y = data['fare_amount'])
```

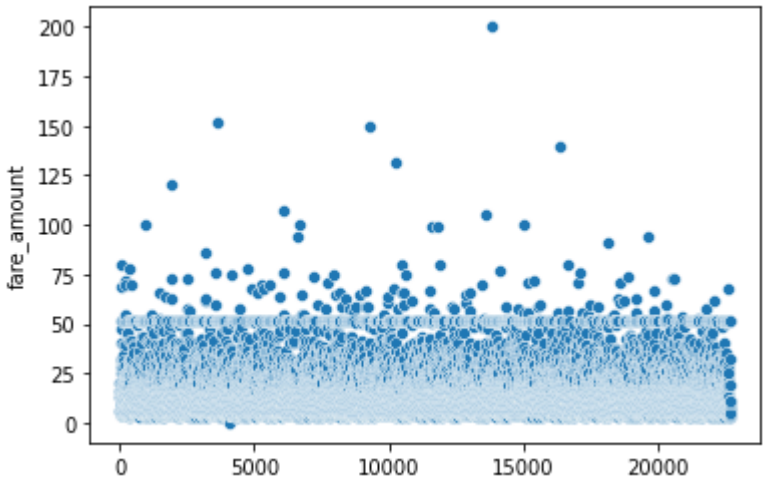
Out[37]:
```
<AxesSubplot:ylabel='fare_amount'>
```



In [38]:
```
 #fare_amount의 이상치를 제거합니다.

data = data[data['fare_amount'] < 300]
```

In [39]:
```
sns.scatterplot(x = data.index, y = data['fare_amount'])
```

Out[39]:
```
<AxesSubplot:ylabel='fare_amount'>
```



In [40]:
```
# fare_amount가 150을 초과한다면 150으로 변환 (이상치 처리)

def fare_func(x):
    if x > 150:
        return 150
    else:
        return x
```

In [41]:
```
data['fare_amount'].apply(fare_func) #한줄 한줄 확인되는 것을 확인하기 위해서
```

Out[41]:
```
0        13.0
1        16.0
2         6.5
3        20.5
4        16.5
         ...
22696     4.0
22697    52.0
22698     4.5
22699    10.5
22700    11.0
Name: fare_amount, Length: 22498, dtype: float64
```

In [42]:
```
data['fare_amount'] = data['fare_amount'].apply(lambda x: 150 if x > 150 els
```

In [43]:
```
data.sort_values('fare_amount')#150으로 변환된 것 확인 가능
```

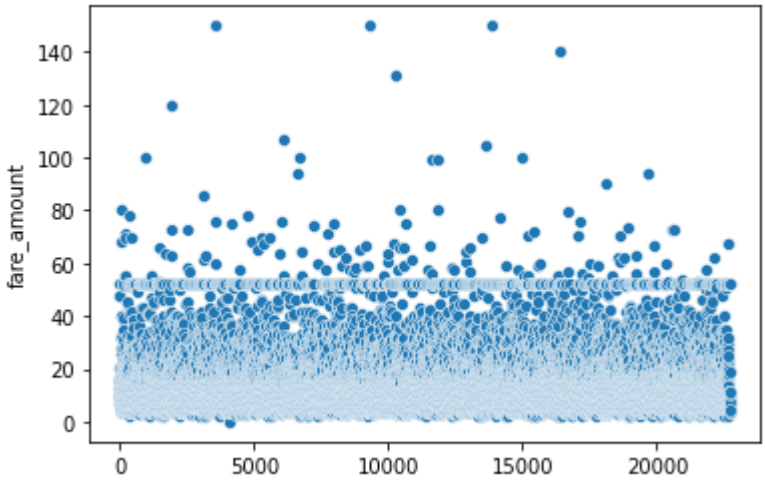
Out[43]:

	passenger_name	tpep_pickup_datetime	tpep_dropoff_datetime	payment_method	p
4063	Phillip Gonzalez	08/12/2017 8:49:29 PM	08/12/2017 9:18:50 PM	Cash	
16829	Jeffrey Jackson	05/02/2017 12:18:59 AM	05/02/2017 12:19:02 AM	Credit Card	
19371	Amanda Taylor	03/24/2017 8:59:58 PM	03/24/2017 9:00:06 PM	Cash	
15501	Julie Ferguson	12/29/2017 9:06:34 PM	12/29/2017 9:07:19 PM	Cash	
1077	Kyle Johnson	04/12/2017 8:51:58 PM	04/12/2017 8:52:07 PM	Cash	
...	...	...	...	...	...
10293	Emily Stevens	09/11/2017 11:41:04 AM	09/11/2017 12:18:58 PM	Cash	
16381	Erica Hernandez	11/30/2017 10:41:11 AM	11/30/2017 11:31:45 AM	Cash	
13863	William Yates	05/19/2017 8:20:21 AM	05/19/2017 9:20:30 AM	Credit Card	
3584	Matthew Chavez	01/01/2017 11:53:01 PM	01/01/2017 11:53:42 PM	Credit Card	
9282	Samantha Frederick	06/18/2017 11:33:25 PM	06/19/2017 12:12:38 AM	Cash	

22498 rows × 9 columns

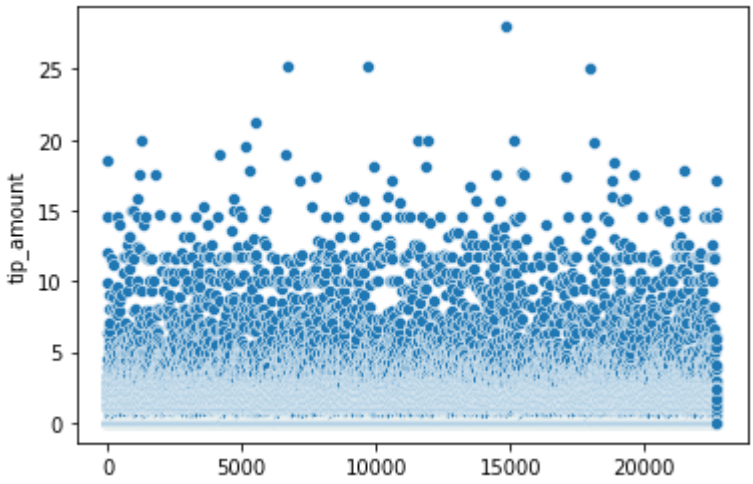
In [44]:
```
sns.scatterplot(x = data.index, y = data['fare_amount'])
```

Out[44]:
```
<AxesSubplot:ylabel='fare_amount'>
```



In [98]:
```
sns.scatterplot(x = data.index, y = data['tip_amount'])
#outliner을 찾으려고 하는 거니까 축에는 index (y값 위주 파악)
```

Out[98]:
```
<AxesSubplot:ylabel='tip_amount'>
```



In [46]:
```
data[data['tip_amount'] > 40] #팁을 이렇게 내는게 outliner 인 것은 맞음.. 평범하진 않
```

Out[46]:

	passenger_name	tpep_pickup_datetime	tpep_dropoff_datetime	payment_method	p
986	Elaine Horton	08/23/2017 6:23:26 PM	08/23/2017 7:18:29 PM	Cash	
6066	Tina Knight	06/13/2017 12:30:22 PM	06/13/2017 1:37:51 PM	Debit Card	
13863	William Yates	05/19/2017 8:20:21 AM	05/19/2017 9:20:30 AM	Credit Card	

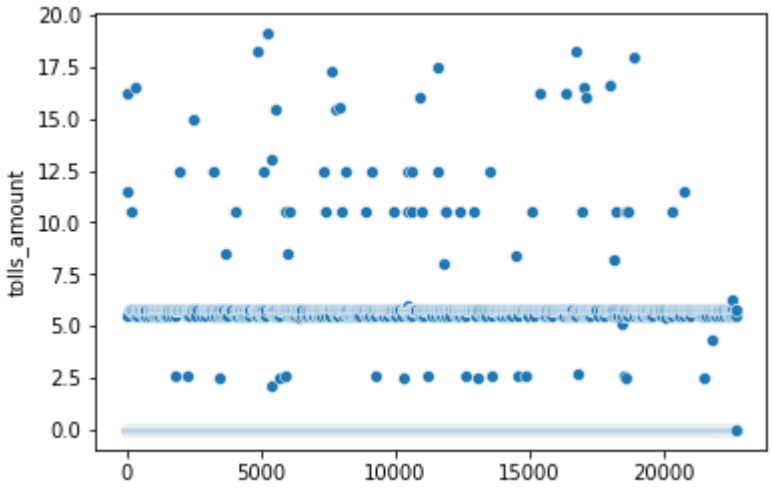
In [47]:
```
#tip_amount의 이상치를 제거합니다.

data = data[data['tip_amount'] < 40]
```

In [48]:
```
len(data) #얼마나 줄어들었는지 확인
```

Out[48]:
```
22495
```

In [49]:
```
sns.scatterplot(x = data.index, y = data['tolls_amount']) #outliner가 없다고 
```

Out[49]:
```
<AxesSubplot:ylabel='tolls_amount'>
```



In [50]:
```
data.head(30) #사실 빅데이터인 경우라면 30개 set만 보고 판단은 어려움
```

Out[50]:

	passenger_name	tpep_pickup_datetime	tpep_dropoff_datetime	payment_method	pass
0	Pamela Duffy	03/25/2017 8:55:43 AM	03/25/2017 9:09:47 AM	Debit Card	
1	Michelle Foster	04/11/2017 2:53:28 PM	04/11/2017 3:19:58 PM	Debit Card	
2	Tina Combs	12/15/2017 7:26:56 AM	12/15/2017 7:34:08 AM	Debit Card	
3	Anthony Ray	05/07/2017 1:17:59 PM	05/07/2017 1:48:14 PM	Cash	
4	Brianna Johnson	04/15/2017 11:32:20 PM	04/15/2017 11:49:03 PM	Debit Card	
5	Justin Smith	03/25/2017 8:34:11 PM	03/25/2017 8:42:11 PM	Debit Card	
6	Tonya Moreno	05/03/2017 7:04:09 PM	05/03/2017 8:03:47 PM	Cash	
7	Hannah Foley	08/15/2017 5:41:06 PM	08/15/2017 6:03:05 PM	Debit Card	
8	Katie Whitney	02/04/2017 4:17:07 PM	02/04/2017 4:29:14 PM	Cash	
9	Amanda Jones	11/10/2017 3:20:29 PM	11/10/2017 3:40:55 PM	Cash	
10	Cory Jensen	03/04/2017 11:58:00 AM	03/04/2017 12:13:12 PM	Cash	
11	Jamie Brown	03/05/2017 7:15:30 PM	03/05/2017 7:52:18 PM	Debit Card	
12	Ryan Reyes	06/09/2017 7:00:26 PM	06/09/2017 7:20:11 PM	Debit Card	
13	Jessica Mooney	11/06/2017 11:35:05 PM	11/06/2017 11:42:57 PM	Credit Card	
14	Heidi May	02/22/2017 3:18:31 PM	02/22/2017 3:42:50 PM	Cash	
15	Anthony Richard	06/02/2017 6:41:39 AM	06/02/2017 6:57:47 AM	Credit Card	
16	Sarah Gross	08/15/2017 7:48:08 PM	08/15/2017 8:00:37 PM	Cash	
18	Susan Robinson	07/10/2017 1:36:31 PM	07/10/2017 1:48:43 PM	Cash	
19	Cynthia Mendoza	04/10/2017 6:12:58 PM	04/10/2017 6:17:39 PM	Cash	
20	Zachary James	03/05/2017 4:01:07 AM	03/05/2017 4:14:11 AM	Credit Card	
21	Marissa Scott	12/30/2017 11:52:44 PM	12/30/2017 11:58:57 PM	Debit Card	
22	Jacqueline Mclean DVM	10/11/2017 12:34:49 PM	10/11/2017 1:22:38 PM	Debit Card	
23	Krista Stewart	01/06/2017 8:12:07 PM	01/06/2017 8:18:37 PM	Cash	
24	Mike Taylor	06/27/2017 12:08:22 AM	06/27/2017 12:13:45 AM	Credit Card	
25	Heather Johnson	02/13/2017 10:29:33 AM	02/13/2017 10:34:11 AM	Cash	
26	Tiffany Ramirez	01/14/2017 7:58:42 PM	01/14/2017 8:05:59 PM	Debit Card	
27	James Taylor	11/04/2017 1:27:59 AM	11/04/2017 1:44:05 AM	Credit Card	
28	Gabriela Bryan	11/24/2017 10:48:13 AM	11/24/2017 10:52:57 AM	Cash	
29	Janet Hogan MD	11/22/2017 10:24:17 AM	11/22/2017 10:38:52 AM	Cash	
30	David Burton	11/06/2017 8:30:50 PM	11/07/2017 12:00:00 AM	Credit Card	

In [52]:
```
data['payment_method'].unique() #payment_method의 고유값들을 보여주고 있음
```

Out[52]:
```
array(['Debit Card', 'Cash', 'Credit Card'], dtype=object)
```

In [53]:
```
data['payment_method'].nunique() #고유값들의 갯수를 알려줌
```

Out[53]:
```
3
```

In [54]:
```
data['payment_method'].value_counts() #각각 몇번인지를 보여주고 있음
```

Out[54]:
```
Cash           11094
Debit Card      5729
Credit Card     5672
Name: payment_method, dtype: int64
```

In [63]:
```
# Q. 'Debit Card'와 'Credit Card' 항목을 'Card'로 변환합니다.
# (힌트: replace() 메서드를 사용합니다.)

data['payment_method'].replace({'Debit Card' : 'Card', ' Credit Card' : 'Ca
```

Out[63]:
```
0        Card
1        Card
2        Card
3        Cash
4        Card
         ...
22696    Cash
22697    Cash
22698    Card
22699    Card
22700    Cash
Name: payment_method, Length: 22495, dtype: object
```

In [64]:
```
data['payment_method'] = data['payment_method'].replace({'Debit Card' : 'Ca
```

In [65]:
```
data['payment_method'].value_counts() #다시 한번 value counts
```

Out[65]:
```
Cash           11094
Card            5729
Credit Card     5672
Name: payment_method, dtype: int64
```

In [67]:
```
#chat gpt에 넣어서 credit card 앞에 공백이 있어서 제대로 안 되고 있었다는 것을 확인, 다시 넣
data['payment_method'] = data['payment_method'].replace({'Debit Card' : 'Ca
```

In [68]:
```
data['payment_method'].value_counts() #다시 한번 value counts, 행을 두개 붙이던 그
```

Out[68]:
```
Card    11401
Cash    11094
Name: payment_method, dtype: int64
```

In [69]:
```
example = 'Susan Robinson'
```

In [70]:
```
example.split()
```

Out[70]:
```
['Susan', 'Robinson']
```

In [71]:
```
data['passenger_name']
```

Out[71]:
```
0             Pamela Duffy
1          Michelle Foster
2               Tina Combs
3              Anthony Ray
4          Brianna Johnson
                  ...
22696       Austin Johnson
22697     Monique Williams
22698          Drew Graves
22699    Jonathan Copeland
22700      Benjamin Miller
Name: passenger_name, Length: 22495, dtype: object
```

In [72]:
```
#판다스 시리즈에서 split쓰기 (data type)
data['passenger_name'].str.split()
```

Out[72]:
```
0             [Pamela, Duffy]
1          [Michelle, Foster]
2               [Tina, Combs]
3              [Anthony, Ray]
4          [Brianna, Johnson]
                  ...
22696       [Austin, Johnson]
22697     [Monique, Williams]
22698          [Drew, Graves]
22699    [Jonathan, Copeland]
22700      [Benjamin, Miller]
Name: passenger_name, Length: 22495, dtype: object
```

In [73]:
```
data['passenger_name'].str.split(expand = True)
#middle name이 포함되던 격
```

Out[73]:

	0	1	2	3
0	Pamela	Duffy	None	None
1	Michelle	Foster	None	None
2	Tina	Combs	None	None
3	Anthony	Ray	None	None
4	Brianna	Johnson	None	None
...	...	...	...	...
22696	Austin	Johnson	None	None
22697	Monique	Williams	None	None
22698	Drew	Graves	None	None
22699	Jonathan	Copeland	None	None
22700	Benjamin	Miller	None	None

22495 rows × 4 columns

In [75]:
```
data['passenger_first_name'] = data['passenger_name'].str.split(expand = Tru
```

In [76]:
```
data.head()
```

Out[76]:

	passenger_name	tpep_pickup_datetime	tpep_dropoff_datetime	payment_method	passe
0	Pamela Duffy	03/25/2017 8:55:43 AM	03/25/2017 9:09:47 AM	Card	
1	Michelle Foster	04/11/2017 2:53:28 PM	04/11/2017 3:19:58 PM	Card	
2	Tina Combs	12/15/2017 7:26:56 AM	12/15/2017 7:34:08 AM	Card	
3	Anthony Ray	05/07/2017 1:17:59 PM	05/07/2017 1:48:14 PM	Cash	
4	Brianna Johnson	04/15/2017 11:32:20 PM	04/15/2017 11:49:03 PM	Card	

In [77]:
```
data.info() #datatype 형태 바꾸기
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 22495 entries, 0 to 22700
Data columns (total 10 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   passenger_name        22495 non-null  object 
 1   tpep_pickup_datetime  22495 non-null  object 
 2   tpep_dropoff_datetime 22495 non-null  object 
 3   payment_method        22495 non-null  object 
 4   passenger_count       22495 non-null  int64  
 5   trip_distance         22495 non-null  float64
 6   fare_amount           22495 non-null  float64
 7   tip_amount            22495 non-null  float64
 8   tolls_amount          22495 non-null  float64
 9   passenger_first_name  22495 non-null  object 
dtypes: float64(4), int64(1), object(5)
memory usage: 1.9+ MB
```

In [79]:
```
pd.to_datetime(data['tpep_pickup_datetime']) #datetime64[ns]을 확인할 수 있음
```

Out[79]:
```
0       2017-03-25 08:55:43
1       2017-04-11 14:53:28
2       2017-12-15 07:26:56
3       2017-05-07 13:17:59
4       2017-04-15 23:32:20
                ...        
22696   2017-02-24 17:37:23
22697   2017-08-06 16:43:59
22698   2017-09-04 14:54:14
22699   2017-07-15 12:56:30
22700   2017-03-02 13:02:49
Name: tpep_pickup_datetime, Length: 22495, dtype: datetime64[ns]
```

In [80]:
```
data['tpep_pickup_datetime'] = pd.to_datetime(data['tpep_pickup_datetime'])
```

In [81]:
```
data['tpep_dropoff_datetime'] = pd.to_datetime(data['tpep_dropoff_datetime'])
```

In [82]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 22495 entries, 0 to 22700
Data columns (total 10 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   passenger_name         22495 non-null  object
 1   tpep_pickup_datetime   22495 non-null  datetime64[ns]
 2   tpep_dropoff_datetime  22495 non-null  datetime64[ns]
 3   payment_method         22495 non-null  object
 4   passenger_count        22495 non-null  int64
 5   trip_distance          22495 non-null  float64
 6   fare_amount            22495 non-null  float64
 7   tip_amount             22495 non-null  float64
 8   tolls_amount           22495 non-null  float64
 9   passenger_first_name   22495 non-null  object
dtypes: datetime64[ns](2), float64(4), int64(1), object(3)
memory usage: 1.9+ MB
```

In [87]:
```
data['tpep_dropoff_datetime'] - data['tpep_pickup_datetime'] #오타가 자주나는 편
```

Out[87]:
```
0       0 days 00:14:04
1       0 days 00:26:30
2       0 days 00:07:12
3       0 days 00:30:15
4       0 days 00:16:43
              ...
22696   0 days 00:03:16
22697   0 days 00:40:48
22698   0 days 00:04:08
22699   0 days 00:11:56
22700   0 days 00:13:20
Length: 22495, dtype: timedelta64[ns]
```

In [88]:
```
data['travel_time'] = data['tpep_dropoff_datetime'] - data['tpep_pickup_date
```

In [89]:
```
data.head()
```

Out[89]:

	passenger_name	tpep_pickup_datetime	tpep_dropoff_datetime	payment_method	passe
0	Pamela Duffy	2017-03-25 08:55:43	2017-03-25 09:09:47	Card	
1	Michelle Foster	2017-04-11 14:53:28	2017-04-11 15:19:58	Card	
2	Tina Combs	2017-12-15 07:26:56	2017-12-15 07:34:08	Card	
3	Anthony Ray	2017-05-07 13:17:59	2017-05-07 13:48:14	Cash	
4	Brianna Johnson	2017-04-15 23:32:20	2017-04-15 23:49:03	Card	

In [91]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 22495 entries, 0 to 22700
Data columns (total 11 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   passenger_name        22495 non-null  object
 1   tpep_pickup_datetime  22495 non-null  datetime64[ns]
 2   tpep_dropoff_datetime 22495 non-null  datetime64[ns]
 3   payment_method        22495 non-null  object
 4   passenger_count       22495 non-null  int64
 5   trip_distance         22495 non-null  float64
 6   fare_amount           22495 non-null  float64
 7   tip_amount            22495 non-null  float64
 8   tolls_amount          22495 non-null  float64
 9   passenger_first_name  22495 non-null  object
 10  travel_time           22495 non-null  timedelta64[ns]
dtypes: datetime64[ns](2), float64(4), int64(1), object(3), timedelta64[ns]
(1)
memory usage: 2.1+ MB
```

In [92]:
```
data['travel_time'].dt.seconds
```

Out[92]:
```
0         844
1        1590
2         432
3        1815
4        1003
         ...
22696     196
22697    2448
22698     248
22699     716
22700     800
Name: travel_time, Length: 22495, dtype: int64
```

In [93]:
```
data['travel_time'] = data['travel_time'].dt.seconds
```

In [94]:
```
data.head() #feacture enginnering
```

Out[94]:

	passenger_name	tpep_pickup_datetime	tpep_dropoff_datetime	payment_method	passe
0	Pamela Duffy	2017-03-25 08:55:43	2017-03-25 09:09:47	Card	
1	Michelle Foster	2017-04-11 14:53:28	2017-04-11 15:19:58	Card	
2	Tina Combs	2017-12-15 07:26:56	2017-12-15 07:34:08	Card	
3	Anthony Ray	2017-05-07 13:17:59	2017-05-07 13:48:14	Cash	
4	Brianna Johnson	2017-04-15 23:32:20	2017-04-15 23:49:03	Card	

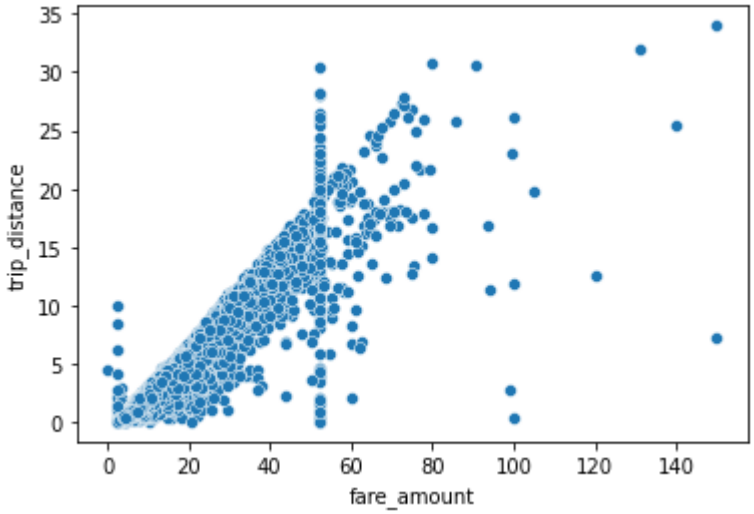
In [95]:
```
data['fare_amount'] + data['tip_amount'] + data['tolls_amount']
```

Out[95]:
```
0        15.76
1        20.00
2         7.95
3        26.89
4        16.50
         ...
22696     4.00
22697    72.40
22698     4.50
22699    12.20
22700    13.35
Length: 22495, dtype: float64
```

In [97]:
```
data['total_amount'] = data['fare_amount'] + data['tip_amount'] + data['tol
```

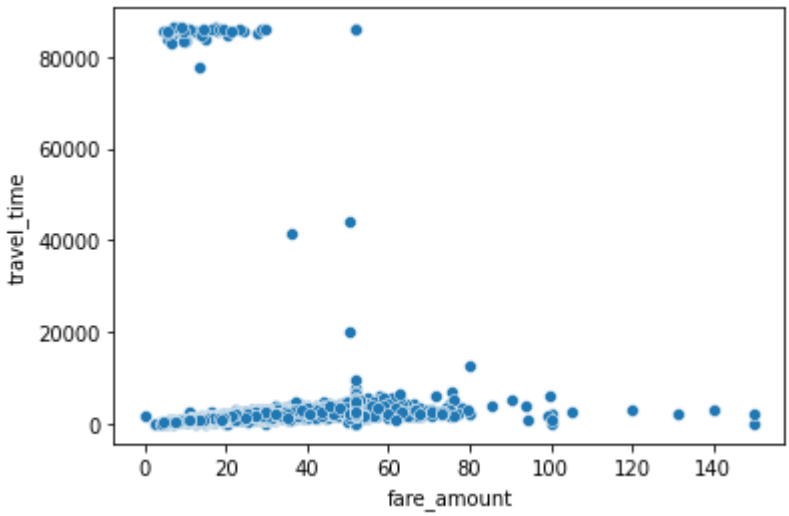
In [99]:
```
sns.scatterplot(x = data['fare_amount'], y = data['trip_distance'])
#양의 상관관계에 있음, treshold가 있는 것을 추측해 볼 수 있음 (-> 데이터의 분포를 이해하는 것도
```

Out[99]:
```
<AxesSubplot:xlabel='fare_amount', ylabel='trip_distance'>
```



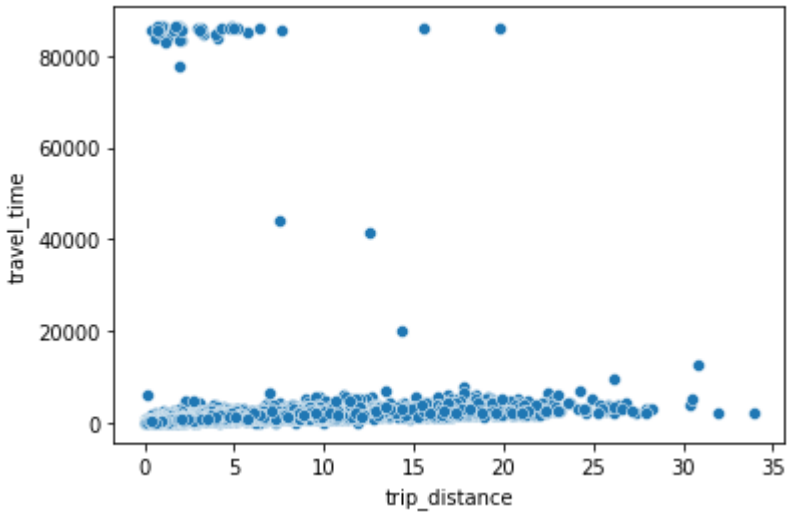
In [101…
```
sns.scatterplot(x = data['fare_amount'], y = data['travel_time'])
#80,000 부근에 처리해야 하는 데이터들이 보임
```

Out[101]:
```
<AxesSubplot:xlabel='fare_amount', ylabel='travel_time'>
```



In [103…
```
sns.scatterplot(x = data['trip_distance'], y = data['travel_time'])
#여기도 좌측 상단에 있는 데이터들이 이상함
```

Out[103]:
```
<AxesSubplot:xlabel='trip_distance', ylabel='travel_time'>
```



In [105…
```
data[data['travel_time'] > 60000]
```

Out[105]:

	passenger_name	tpep_pickup_datetime	tpep_dropoff_datetime	payment_method
699	Scott Garcia	2017-06-10 21:55:01	2017-06-11 21:45:51	Card
926	Michael Perez	2017-02-09 23:24:58	2017-02-10 23:24:31	Cash
1012	James Anderson	2017-12-08 07:17:20	2017-12-09 07:07:22	Cash
1201	Carla Allen	2017-11-12 19:52:44	2017-11-13 19:37:35	Card
1357	Jamie Collins	2017-04-17 21:26:49	2017-04-18 20:46:13	Cash
1760	Ronald Kidd	2017-12-28 23:58:24	2017-12-29 23:38:45	Cash
4602	Brandon Miller	2017-12-20 08:24:34	2017-12-21 07:39:27	Cash
5372	Catherine Ray	2017-12-13 19:40:05	2017-12-14 19:31:09	Cash
5480	Patricia Galvan	2017-09-19 13:16:13	2017-09-20 12:36:12	Card
6495	Travis Tucker	2017-06-27 16:52:07	2017-06-28 16:49:57	Cash
6753	Justin Rosales	2017-06-14 11:51:18	2017-06-15 11:49:20	Card
7014	Alex Cummings	2017-12-20 08:23:16	2017-12-21 08:19:56	Cash
7171	Michael Allen	2017-04-09 07:55:14	2017-04-10 07:02:02	Card
7941	Benjamin Ortiz	2017-06-30 20:36:00	2017-07-01 20:34:28	Cash
8197	David Crane	2017-02-12 02:21:07	2017-02-13 00:00:00	Card
8714	Rhonda Castillo	2017-06-18 09:21:07	2017-06-19 08:59:45	Card
8871	Kathleen Welch	2017-07-12 21:55:00	2017-07-13 21:50:48	Card
9210	Renee Bowman	2017-09-22 09:20:53	2017-09-23 09:04:02	Card
9358	Donna Summers	2017-11-05 01:23:08	2017-11-05 01:06:09	Cash
10212	Dennis Goodwin	2017-06-30 22:39:13	2017-07-01 22:33:12	Cash
10931	Jesse Ward DVM	2017-04-02 17:28:22	2017-04-03 17:23:29	Cash
11674	Jesus Smith	2017-03-18 14:58:31	2017-03-19 14:31:35	Card
12564	Danielle Porter	2017-08-09 20:44:58	2017-08-10 20:25:53	Card
13149	Jennifer Graham	2017-11-05 01:52:31	2017-11-06 01:04:34	Cash
13798	Rebecca Hawkins	2017-07-08 02:37:56	2017-07-09 02:28:47	Cash
14324	Jonathan Sanchez	2017-12-24 13:03:22	2017-12-25 12:47:27	Card
14411	James Church	2017-09-16 01:41:48	2017-09-17 01:16:28	Cash
15000	Jennifer Wilson	2017-07-31 14:04:25	2017-08-01 14:03:16	Cash
15165	Katelyn Greer	2017-04-13 23:41:09	2017-04-14 23:39:42	Cash
15581	Ashley Holmes	2017-05-10 18:53:53	2017-05-11 18:53:02	Cash
17396	Jason Smith	2017-12-14 11:48:00	2017-12-15 10:59:44	Card
18425	Jason Ryan	2017-03-11 11:35:13	2017-03-12 11:27:17	Cash
18545	Matthew Velez	2017-07-01 20:30:04	2017-07-02 20:22:59	Card
18652	Shannon Mitchell	2017-10-05 10:51:03	2017-10-06 10:48:01	Card
19166	Dwayne Wilcox	2017-12-27 22:08:53	2017-12-28 21:53:04	Cash
20689	Christina Barajas	2017-03-03 20:33:11	2017-03-04 20:24:27	Cash
21368	Natasha Ingram	2017-12-14 17:21:37	2017-12-15 17:19:53	Card

	passenger_name	tpep_pickup_datetime	tpep_dropoff_datetime	payment_method
21420	Sandra White	2017-05-26 19:40:47	2017-05-27 19:37:17	Card
21513	Gregory Wong	2017-07-02 15:45:27	2017-07-03 15:41:54	Card
21650	Jessica Hudson	2017-04-28 10:30:55	2017-04-29 10:19:39	Cash
22280	Jacob Hayes	2017-05-18 20:00:55	2017-05-19 19:50:14	Card

In [106…
```
data = data[data['travel_time'] > 60000]
```

In [107…
```
#정답은 없다...
```

In [ ]:
```
```