In [2]:
```
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
```

In [3]:
```
import pandas as pd
from pathlib import Path
import pyarrow.parquet as pq

month = 1
year = 2023
path = Path("..") / "data" / "raw" / f"rides_{year}_{month:02}.parquet"

table = pq.read_table(path)
rides = table.to_pandas()
rides.head()
```

Out[3]:

	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	Ratec
0	2	2023-01-01 00:32:10	2023-01-01 00:40:36	1.0	0.97	
1	2	2023-01-01 00:55:08	2023-01-01 01:01:27	1.0	1.10	
2	2	2023-01-01 00:25:04	2023-01-01 00:37:49	1.0	2.51	
3	1	2023-01-01 00:03:48	2023-01-01 00:13:25	0.0	1.90	
4	2	2023-01-01 00:10:29	2023-01-01 00:21:19	1.0	1.43	

In [4]:
```
rides_cp = rides.copy()
rides_cp["duration"] = rides["tpep_dropoff_datetime"] - rides["tpep_pickup_dat
rides_cp.head()
```

Out[4]:

	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	Ratec
0	2	2023-01-01 00:32:10	2023-01-01 00:40:36	1.0	0.97	
1	2	2023-01-01 00:55:08	2023-01-01 01:01:27	1.0	1.10	
2	2	2023-01-01 00:25:04	2023-01-01 00:37:49	1.0	2.51	
3	1	2023-01-01 00:03:48	2023-01-01 00:13:25	0.0	1.90	
4	2	2023-01-01 00:10:29	2023-01-01 00:21:19	1.0	1.43	

In [5]:
```
rides_cp["duration"].describe().T
```

Out[5]:
```
count                   3066766
mean     0 days 00:15:40.139710
std      0 days 00:42:35.661074
min           -1 days +23:30:48
25%             0 days 00:07:07
50%             0 days 00:11:31
75%             0 days 00:18:18
max             6 days 23:09:11
Name: duration, dtype: object
```

In [6]:
```
rides_cp["duration"].quantile(0)
rides_cp["duration"].quantile(0.01)
rides_cp["duration"].quantile(0.995)
rides_cp["duration"].quantile(0.999)
```

Out[6]: Timedelta('-1 days +23:30:48')

Out[6]: Timedelta('0 days 00:00:47')

Out[6]: Timedelta('0 days 01:05:31')

Out[6]: Timedelta('0 days 02:55:49.290000')

In [7]:
```
duration_filter = (rides_cp["duration"] > pd.Timedelta(0)) & (rides_cp["durati
sum(~duration_filter)
```

Out[7]: 4001

In [8]:
```
rides_cp["total_amount"].describe().T
```

Out[8]:
```
count    3.066766e+06
mean     2.702038e+01
std      2.216359e+01
min     -7.510000e+02
25%      1.540000e+01
50%      2.016000e+01
75%      2.870000e+01
max      1.169400e+03
Name: total_amount, dtype: float64
```

In [9]:
```
rides_cp["total_amount"].quantile(0.0)
rides_cp["total_amount"].quantile(0.01)
rides_cp["total_amount"].quantile(0.995)
rides_cp["total_amount"].quantile(0.999)
```

Out[9]: -751.0

Out[9]: 5.5

Out[9]: 108.9

Out[9]: 167.01175000001678

In [10]:
```
rides_cp["total_amount"].max()
```

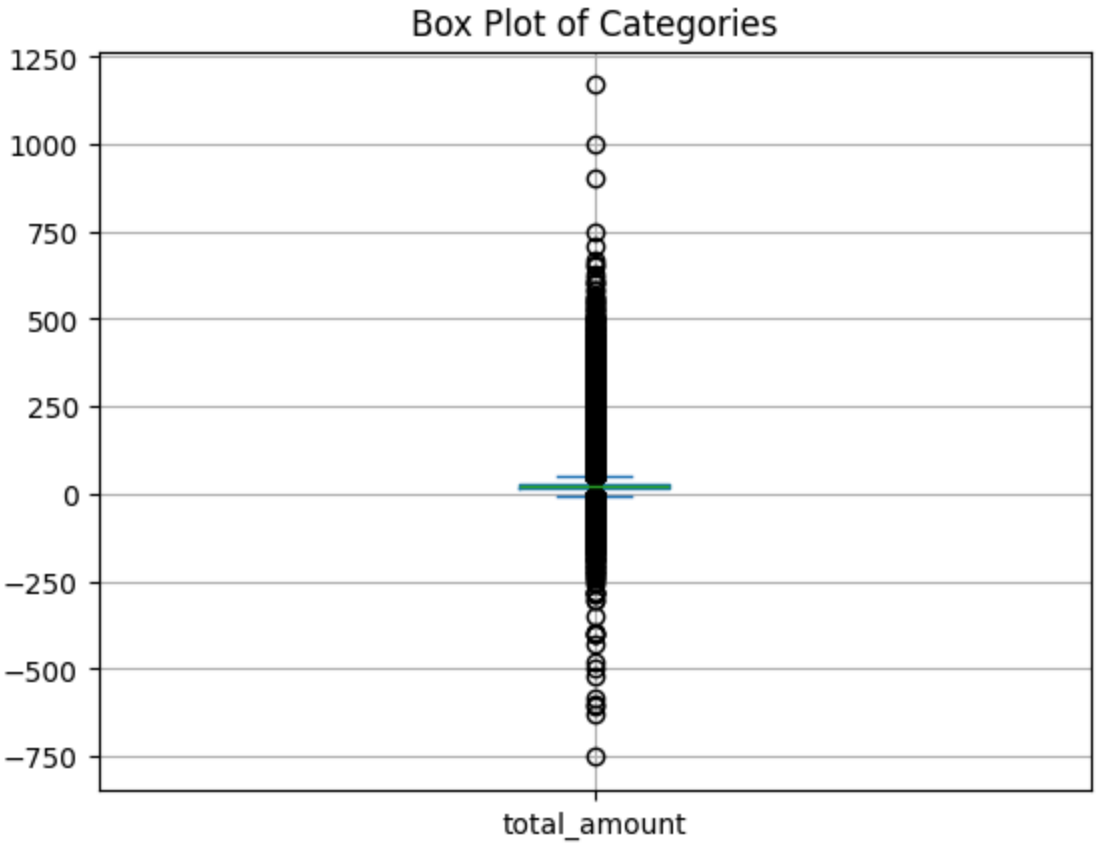
Out[10]:
```
1169.4
```

In [11]:
```
total_amount_filter = (rides_cp["total_amount"]  > 0) & (rides_cp["total_amour
sum(~total_amount_filter) / rides_cp.shape[0] * 100
```

Out[11]:
```
0.9403717140466537
```

In [12]:
```
rides_cp["total_amount"].plot.box(title="Box Plot of Categories", grid=True)
```

Out[12]:
```
<Axes: title={'center': 'Box Plot of Categories'}>
```



In [13]:
```
nyc_locations = ~rides_cp["PULocationID"].isin((1, 264, 265))
sum(~nyc_locations)
```

Out[13]:
```
42173
```

In [14]:
```
sorted_df = rides_cp.sort_values(by="tpep_pickup_datetime", ascending=True)

# Get the top 10 (smallest) and bottom 10 (largest) values
top_10 = sorted_df.head(10)
bottom_10 = sorted_df.tail(10)

top_10

bottom_10
```

Out[14]:

	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance
2138036	2	2008-12-31 23:01:42	2009-01-01 14:29:11	1.0	17.76
209091	2	2008-12-31 23:04:41	2009-01-01 19:55:36	1.0	0.00
10023	2	2022-10-24 17:37:47	2022-10-24 17:37:51	1.0	0.00
18219	2	2022-10-24 20:01:46	2022-10-24 20:01:48	1.0	0.00
21660	2	2022-10-24 21:45:35	2022-10-24 21:45:38	1.0	0.00
22489	2	2022-10-24 23:15:32	2022-10-24 23:15:42	1.0	0.00
24577	2	2022-10-25 00:42:10	2022-10-25 00:44:22	1.0	0.97
24578	2	2022-10-25 00:59:02	2022-10-25 01:09:02	1.0	2.33
31916	2	2022-10-25 03:45:46	2022-10-25 03:45:50	1.0	0.02
47843	2	2022-10-25 07:48:15	2022-10-25 07:48:18	2.0	0.76

Out[14]:

	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance
2993635	2	2023-02-01 00:00:01	2023-02-01 00:33:41	1.0	17.31
2993262	2	2023-02-01 00:00:18	2023-02-01 00:08:46	1.0	2.12
2993890	2	2023-02-01 00:00:20	2023-02-01 00:13:18	2.0	2.31
2992346	2	2023-02-01 00:00:24	2023-02-01 00:07:53	2.0	2.22
2994212	2	2023-02-01 00:00:35	2023-02-01 00:17:12	1.0	2.88
2994844	2	2023-02-01 00:00:40	2023-02-01 00:23:03	5.0	10.12
2993558	2	2023-02-01 00:00:55	2023-02-01 00:06:33	1.0	1.09
2992642	2	2023-02-01 00:01:10	2023-02-01 00:14:26	1.0	2.03
2929496	2	2023-02-01 00:13:10	2023-02-01 00:29:37	1.0	3.27
2929497	2	2023-02-01 00:56:53	2023-02-01 01:06:43	1.0	2.38

In [15]:
```
filter_date_range = (rides_cp["tpep_pickup_datetime"] >= "2023-01-01") & (ride
sum(~filter_date_range)
```

Out[15]: 48

In [16]:
```
final_filter = duration_filter & total_amount_filter & nyc_locations & filter_
numbers_dropped = final_filter.shape[0] - sum(final_filter) # numbers dropped
numbers_dropped
numbers_dropped/final_filter.shape[0] * 100
```

Out[16]: 73626

Out[16]: 2.400770062013209

In [17]:
```
rides = rides[final_filter]
rides = rides[["tpep_pickup_datetime", "PULocationID"]]
rides.rename(columns={
    "tpep_pickup_datetime": "pickup_datetime",
    "PULocationID": "pickup_location_id"
}, inplace=True)
rides.head()
year = 2023
month = 1
path = Path("..") / "data" / "processed" / f"rides_{year}_{month:02}.parquet"
rides.to_parquet(path, engine="pyarrow", index=False)
```

Out[17]:

	pickup_datetime	pickup_location_id
0	2023-01-01 00:32:10	161
1	2023-01-01 00:55:08	43
2	2023-01-01 00:25:04	48
3	2023-01-01 00:03:48	138
4	2023-01-01 00:10:29	107

In [18]:
```
rides[rides["pickup_location_id"] == 2]
```

Out[18]:

	pickup_datetime	pickup_location_id
2687593	2023-01-28 17:03:38	2

In [ ]:

In [ ]:

In [ ]: