```python
import numpy as np
import pandas as pd
np.random.seed(42)
N = 20000  # number of users



# 1. Basic identifiers
user_id = np.arange(1, N + 1)

# 2. Demographics
age = np.clip(np.random.normal(loc=35, scale=10, size=N), 18,
70).astype(int)
countries = ["US", "UK", "IN", "CA", "DE", "BR", "AU"]
country_probs = [0.35, 0.1, 0.2, 0.1, 0.1, 0.1, 0.05]
country = np.random.choice(countries, size=N, p=country_probs)

# 3. Subscription tenure (months)
months_subscribed = np.random.randint(1, 61, size=N)  # 1–60 months

# 4. Plan type
plan_types = ["basic", "standard", "premium"]
plan_probs = [0.4, 0.4, 0.2]
plan_type = np.random.choice(plan_types, size=N, p=plan_probs)

# 5. Payment method
payment_methods = ["credit_card", "paypal", "gift_card",
"bank_transfer"]
payment_probs = [0.5, 0.2, 0.1, 0.2]
payment_method = np.random.choice(payment_methods, size=N,
p=payment_probs)

# 6. Devices / profiles
# more devices for higher plans
num_devices = (
    np.where(plan_type == "basic", np.random.randint(1, 3, size=N),
    np.where(plan_type == "standard", np.random.randint(2, 4, size=N),
             np.random.randint(3, 6, size=N)))
)

# 7. Add-ons
has_4k = np.where(plan_type == "premium",
                  np.random.binomial(1, 0.7, size=N),
                  np.random.binomial(1, 0.05, size=N))
has_offline_downloads = np.random.binomial(1, 0.4, size=N)

# 8. Monthly price (roughly based on plan + add-ons)
base_price = np.where(plan_type == "basic", 9.99,
             np.where(plan_type == "standard", 14.99, 19.99))
monthly_price = (
    base_price
```

```
    + has_4k * 3.0
    + has_offline_downloads * 1.5
    + np.random.normal(0, 0.75, size=N)  # small noise
)
monthly_price = np.round(monthly_price, 2)

# 9. Engagement – watch hours per last 30 days
# More expensive plans tend to watch slightly more
plan_watch_factor = np.where(plan_type == "basic", 1.0,
                     np.where(plan_type == "standard", 1.15, 1.3))
base_watch = np.random.gamma(shape=2.0, scale=8.0, size=N)  # ~0–80
hours
watch_hours_30d = base_watch * plan_watch_factor
watch_hours_30d = np.clip(watch_hours_30d, 0, 120)

# 10. Engagement last 7 days – could have dropped/increased
eng_change_factor = np.random.normal(loc=0.25, scale=0.15, size=N)
# convert 30d to 7d baseline (~1/4) then apply change
watch_hours_7d = (watch_hours_30d / 4.0) * (1 + eng_change_factor)
watch_hours_7d = np.clip(watch_hours_7d, 0, None)

# 11. Days since last watch (if very low engagement, often high)
base_inactivity = np.random.randint(0, 15, size=N)
extra_inactivity = np.where(watch_hours_7d < 3, np.random.randint(0,
20,
size=N), 0)
days_since_last_watch = np.clip(base_inactivity + extra_inactivity, 0,
45)

# 12. Genre diversity & binge behaviour
genre_diversity = np.clip(
np.random.poisson(lam=3 + watch_hours_30d / 30, size=N), 1, 15
)
binge_sessions = np.clip(
    np.random.poisson(lam=watch_hours_30d / 15, size=N), 0, 20
)

# 13. Support tickets & payment failures
support_tickets = np.clip(
    np.random.poisson(lam=0.2 + (days_since_last_watch > 20) * 0.3,
size=N),
    0,
    10,
)
payment_failures = np.clip(np.random.poisson(lam=0.1 + (monthly_price
> 18) * 0.2, size=N),
    0,
    6,
)
```

```python
# 14. Discount usage
discount_used = np.random.binomial(1, 0.3, size=N)

# 15. Lifetime value & engagement drop ratio
lifetime_value = monthly_price * months_subscribed
engagement_drop_ratio = np.where(
    watch_hours_30d > 0,
    watch_hours_7d / (watch_hours_30d / 4.0),  # compare 7d vs "expected" 7d
    0,
)
# clip extreme ratios
engagement_drop_ratio = np.clip(engagement_drop_ratio, 0, 2.5)

# 16. Compute churn probability using a scoring formula
# Start with a base probability
base_p = 0.15
# More likely to churn if:
p = np.full(N, base_p)
# Low watch in 30 days
p += np.where(watch_hours_30d < 10, 0.12, 0)
p += np.where(watch_hours_30d < 5, 0.10, 0)
# Big engagement drop (ratio < 0.7)
p += np.where(engagement_drop_ratio < 0.7, 0.15, 0)
# Many days since last watch
p += (days_since_last_watch / 60.0)
# Many payment failures
p += payment_failures * 0.06
# Many support tickets
p += support_tickets * 0.03
# Very new users churn more
p += np.where(months_subscribed <= 3, 0.10, 0)
# Very long-tenure users churn less
p -= np.where(months_subscribed > 24, 0.06, 0)
# Premium users churn slightly less
p -= np.where(plan_type == "premium", 0.05, 0)
# ensure probabilities are between 0.01 and 0.8
p = np.clip(p, 0.01, 0.8)

# 17. Sample actual churn outcome from Bernoulli distribution
churn = np.random.binomial(1, p, size=N)

# 18. Build DataFrame
df = pd.DataFrame({
    "user_id": user_id,
    "age": age,
    "country": country,
    "months_subscribed": months_subscribed,
    "plan_type": plan_type,
    "payment_method": payment_method,
```

```
    "num_devices": num_devices,
    "has_4k": has_4k,
    "has_offline_downloads": has_offline_downloads,
    "monthly_price": monthly_price,
    "watch_hours_30d": watch_hours_30d,
    "watch_hours_7d": watch_hours_7d,
    "days_since_last_watch": days_since_last_watch,
    "genre_diversity": genre_diversity,
    "binge_sessions": binge_sessions,
    "support_tickets": support_tickets,
    "payment_failures": payment_failures,
    "discount_used": discount_used,
    "lifetime_value": lifetime_value,
    "engagement_drop_ratio": engagement_drop_ratio,
    "churn": churn
})

# 19. Save to CSV
df.to_csv("streaming_churn_20k.csv", index=False)
print("Saved dataset with shape:", df.shape)
print(df.head())
```

```
Saved dataset with shape: (20000, 21)
   user_id  age country  months_subscribed plan_type payment_method  \
0        1   39      US                 12     basic    credit_card
1        2   33      AU                 58  standard  bank_transfer
2        3   41      DE                 49     basic    credit_card
3        4   50      BR                 34  standard         paypal
4        5   32      CA                 49  standard         paypal

   num_devices  has_4k  has_offline_downloads  monthly_price  ...  \
0            1       0                      1          12.06  ...
1            2       1                      0          18.44  ...
2            1       0                      1          10.16  ...
3            2       0                      1          16.52  ...
4            2       0                      1          16.79  ...

   watch_hours_7d  days_since_last_watch  genre_diversity
binge_sessions  \
0        6.818861                      1                7
0
1        5.046324                      0                5
2
2        5.446575                      4                3
1
3        4.044258                      7                3
2
4        6.162215                     11                1
0
```

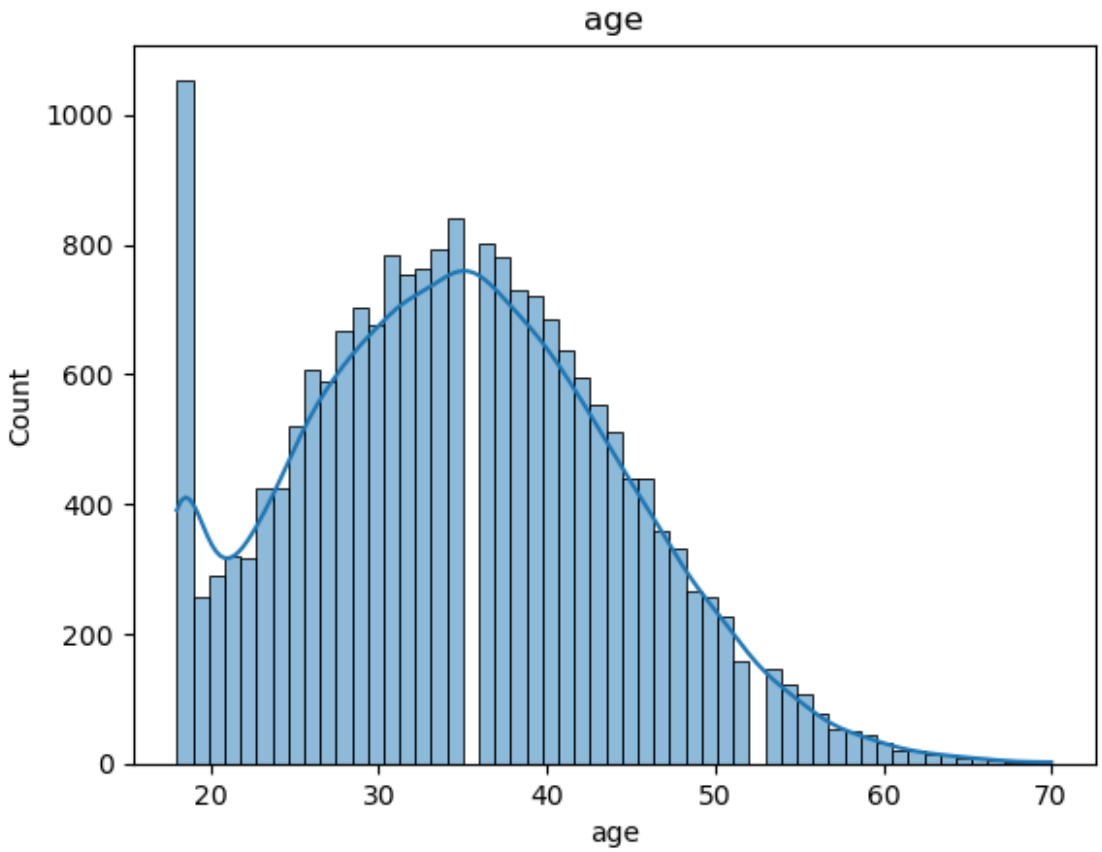
```
   support_tickets  payment_failures  discount_used  lifetime_value  \
0                1                 0              1          144.72   
1                0                 1              1         1069.52   
2                0                 0              0          497.84   
3                1                 0              0          561.68   
4                0                 0              0          822.71   

   engagement_drop_ratio  churn
0               1.331173      0
1               1.211439      0
2               1.072884      0
3               1.297427      0
4               1.421602      0

[5 rows x 21 columns]
```

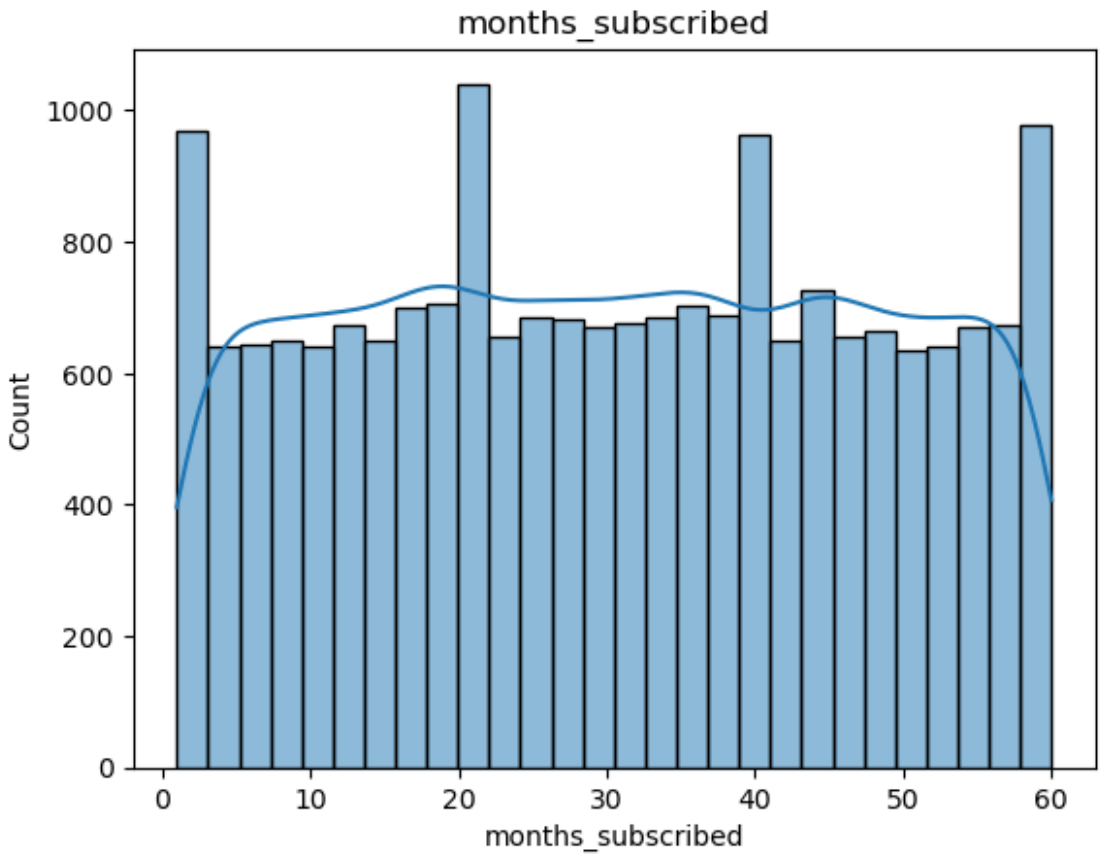
```python
#20 EDA BASIC CHECKING
import pandas as pd

df = pd.read_csv("streaming_churn_20k.csv")
df.info()
df.describe(include="all")
df["churn"].value_counts(normalize=True)
```

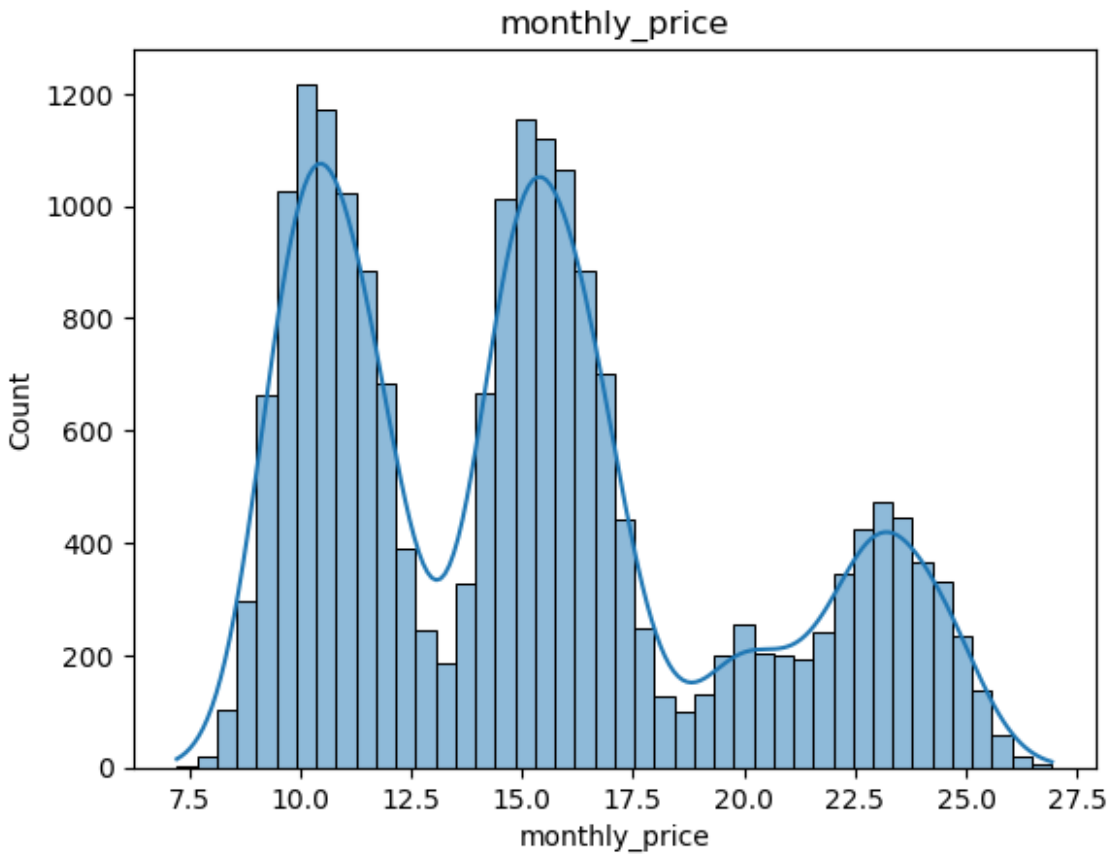
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20000 entries, 0 to 19999
Data columns (total 21 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   user_id                20000 non-null  int64  
 1   age                    20000 non-null  int64  
 2   country                20000 non-null  object 
 3   months_subscribed      20000 non-null  int64  
 4   plan_type              20000 non-null  object 
 5   payment_method         20000 non-null  object 
 6   num_devices            20000 non-null  int64  
 7   has_4k                 20000 non-null  int64  
 8   has_offline_downloads  20000 non-null  int64  
 9   monthly_price          20000 non-null  float64
 10  watch_hours_30d        20000 non-null  float64
 11  watch_hours_7d         20000 non-null  float64
 12  days_since_last_watch  20000 non-null  int64  
 13  genre_diversity        20000 non-null  int64  
 14  binge_sessions         20000 non-null  int64  
 15  support_tickets        20000 non-null  int64  
 16  payment_failures       20000 non-null  int64  
 17  discount_used          20000 non-null  int64  
 18  lifetime_value         20000 non-null  float64
 19  engagement_drop_ratio  20000 non-null  float64
 20  churn                  20000 non-null  int64  
```

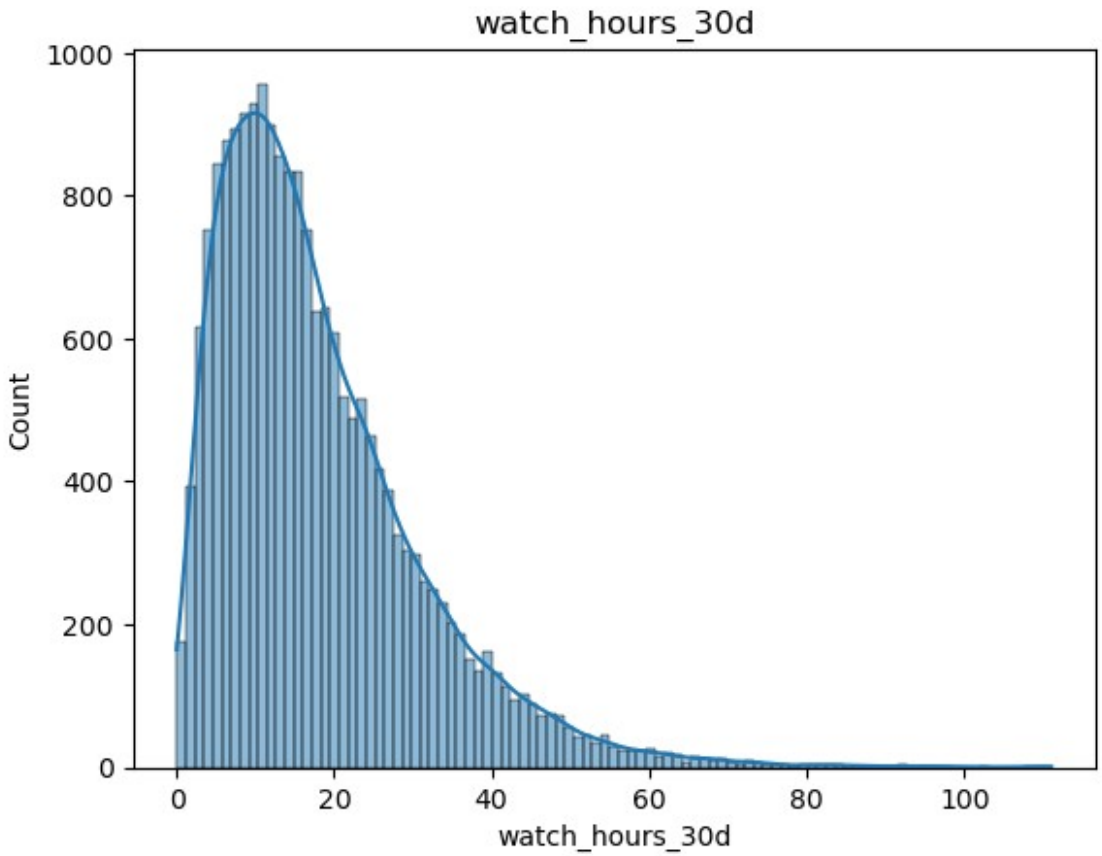
```
dtypes: float64(5), int64(13), object(3)
memory usage: 3.2+ MB

churn
0    0.6663
1    0.3337
Name: proportion, dtype: float64
```

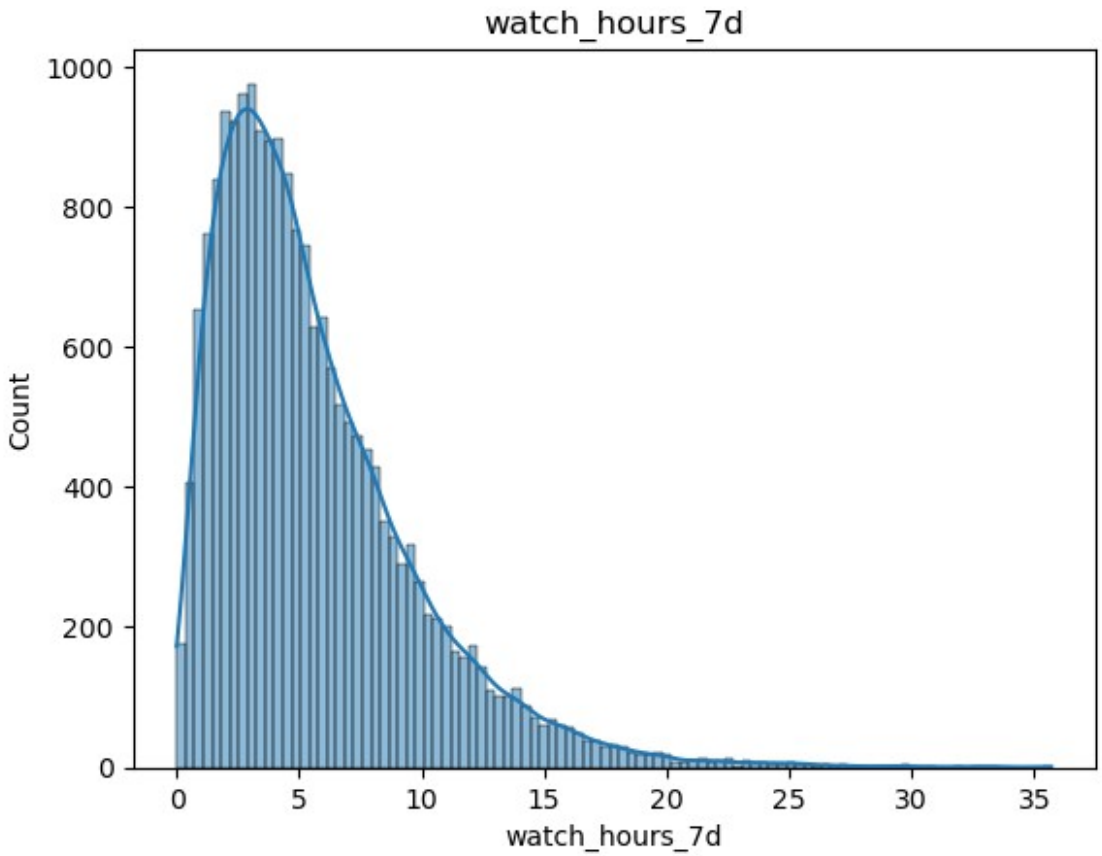
```
#21 UNIVARIATE EDA
import seaborn as sns
import matplotlib.pyplot as plt

num_cols =
["age","months_subscribed","monthly_price","watch_hours_30d",
            "watch_hours_7d","days_since_last_watch","lifetime_value",

"engagement_drop_ratio","payment_failures","support_tickets"]
cat_cols = ["country","plan_type","payment_method","churn"]

for col in num_cols:
    sns.histplot(df[col], kde=True)
    plt.title(col)
    plt.show()

for col in cat_cols:
    sns.countplot(data=df, x=col)
    plt.title(col)
    plt.show()
```

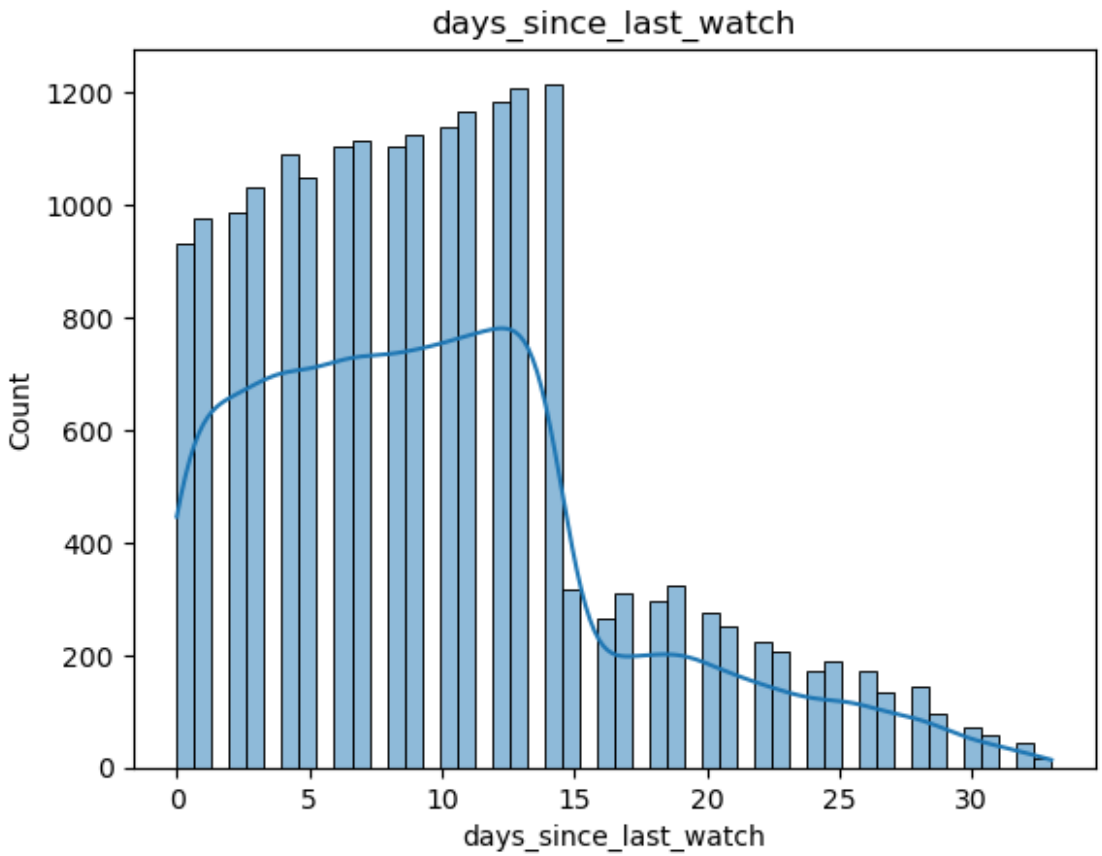

age
Count
1000
800
600
400
200
0
20
30
40
50
60
70
age

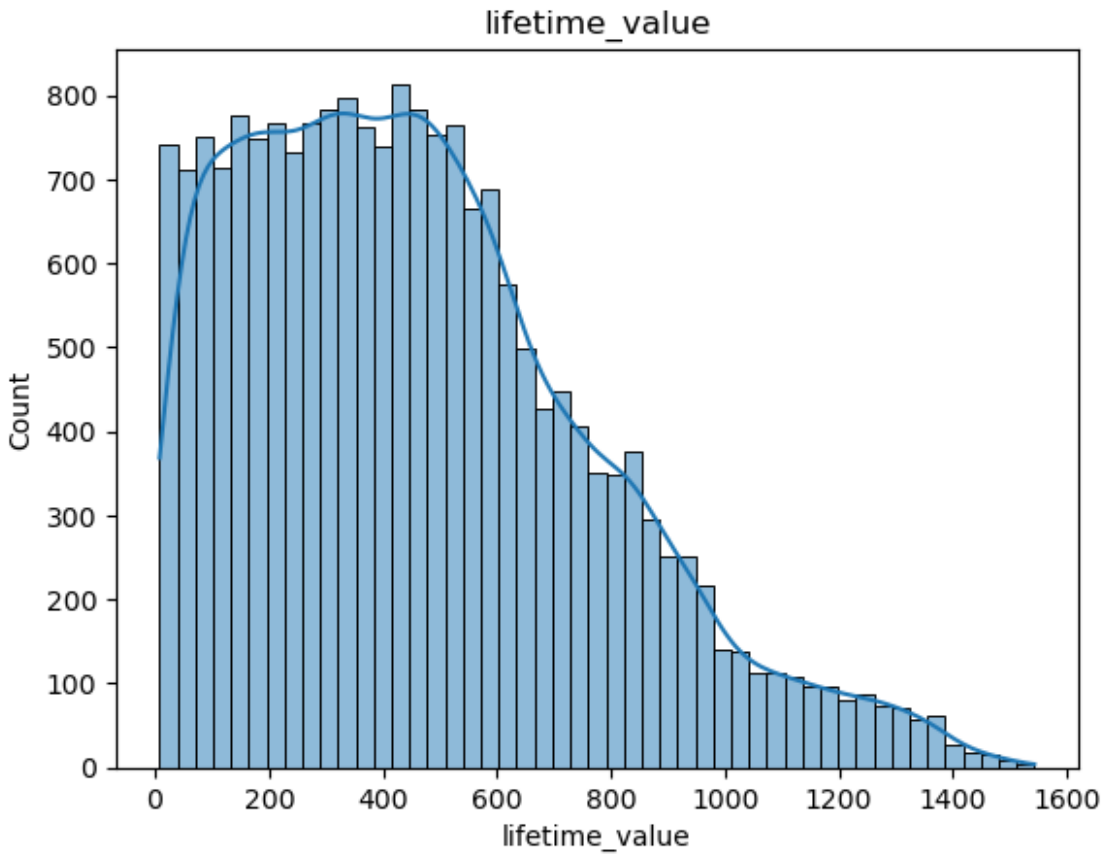

months_subscribed
Count
1000
800
600
400
200
0
0
10
20
30
40
50
60
months_subscribed

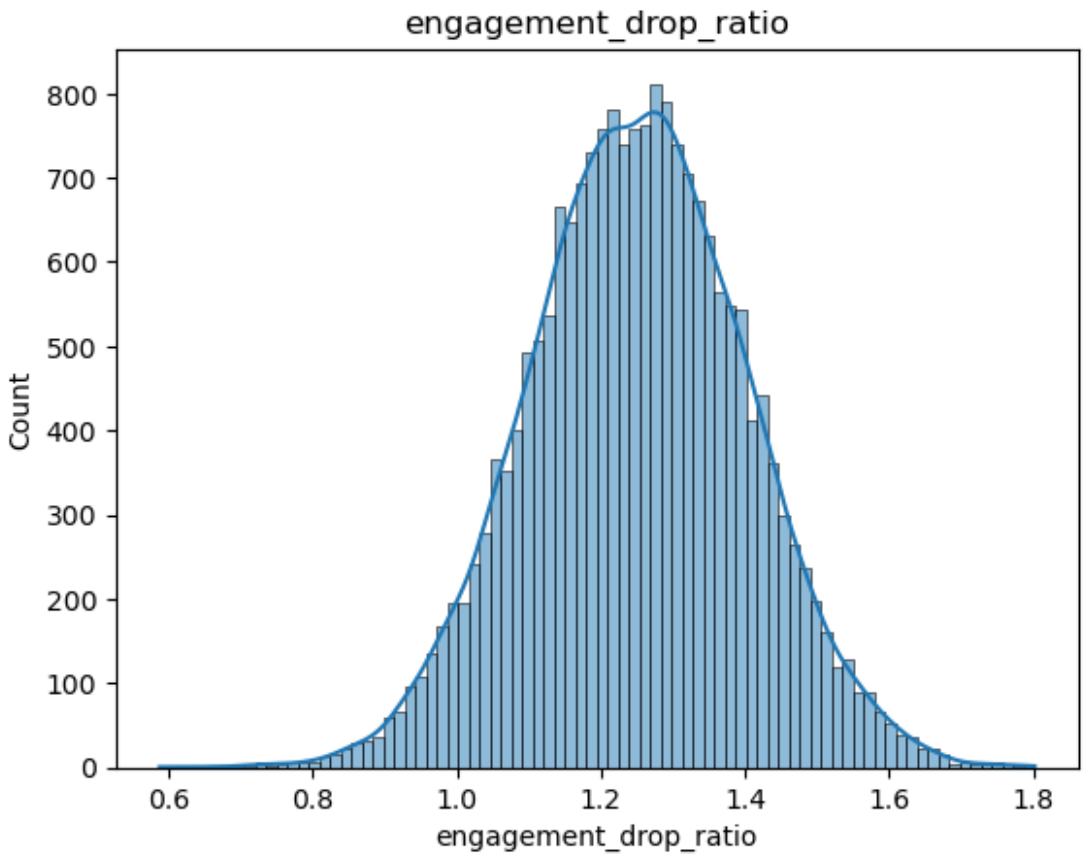

monthly_price
Count
0
200
400
600
800
1000
1200
7.5
10.0
12.5
15.0
17.5
20.0
22.5
25.0
27.5
monthly_price

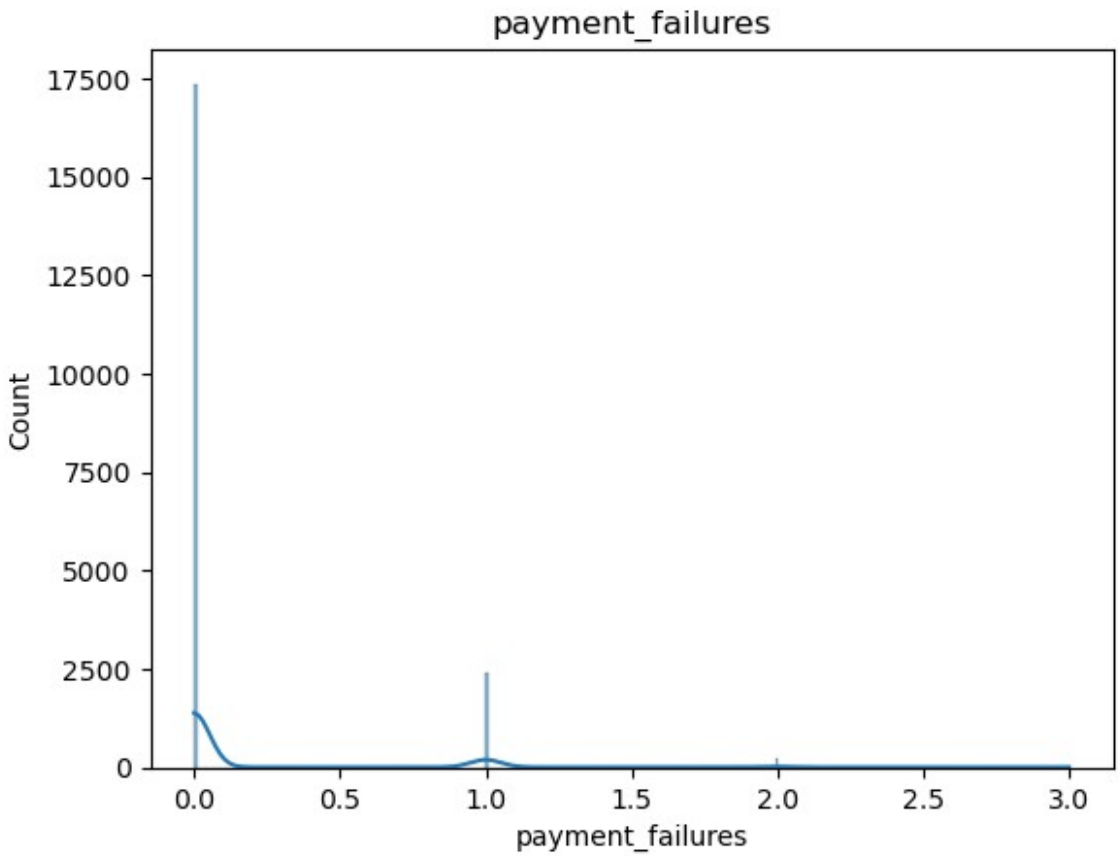

watch_hours_30d
1000
800
600
400
200
0
Count
0
20
40
60
80
100
watch_hours_30d

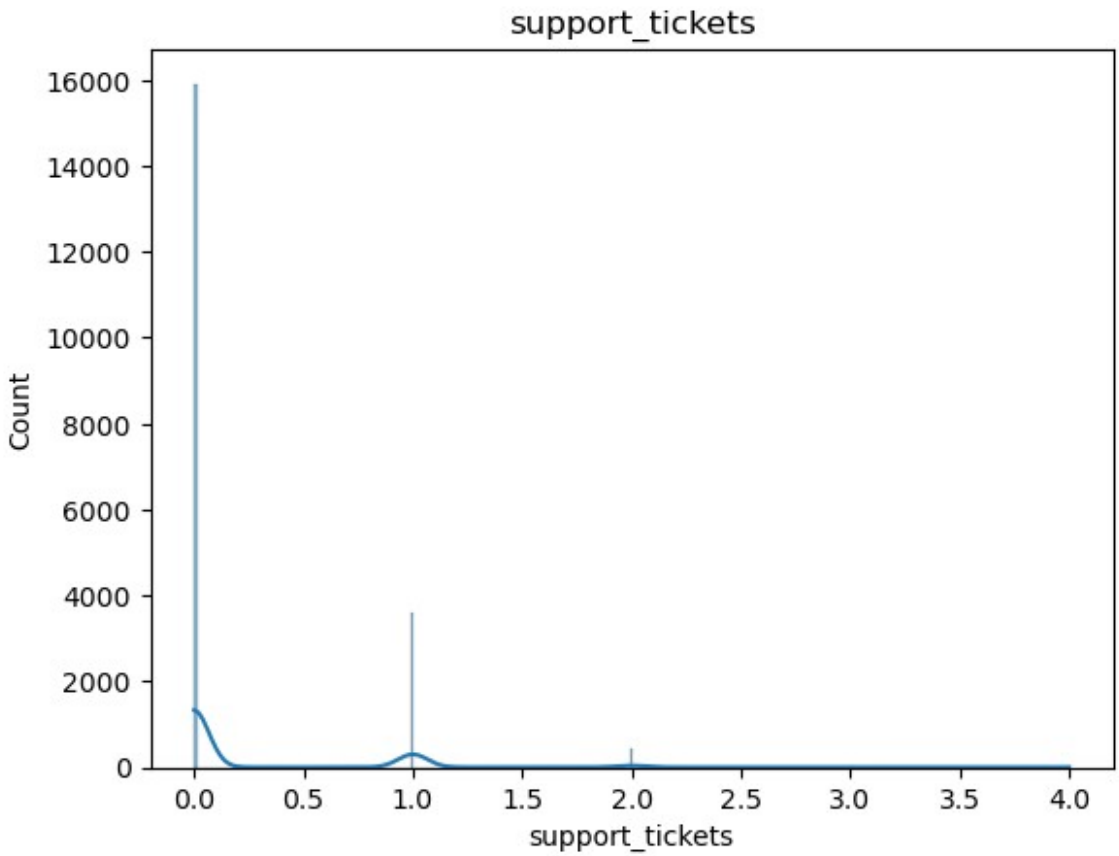

watch_hours_7d
Count
1000
800
600
400
200
0
0
5
10
15
20
25
30
35
watch_hours_7d

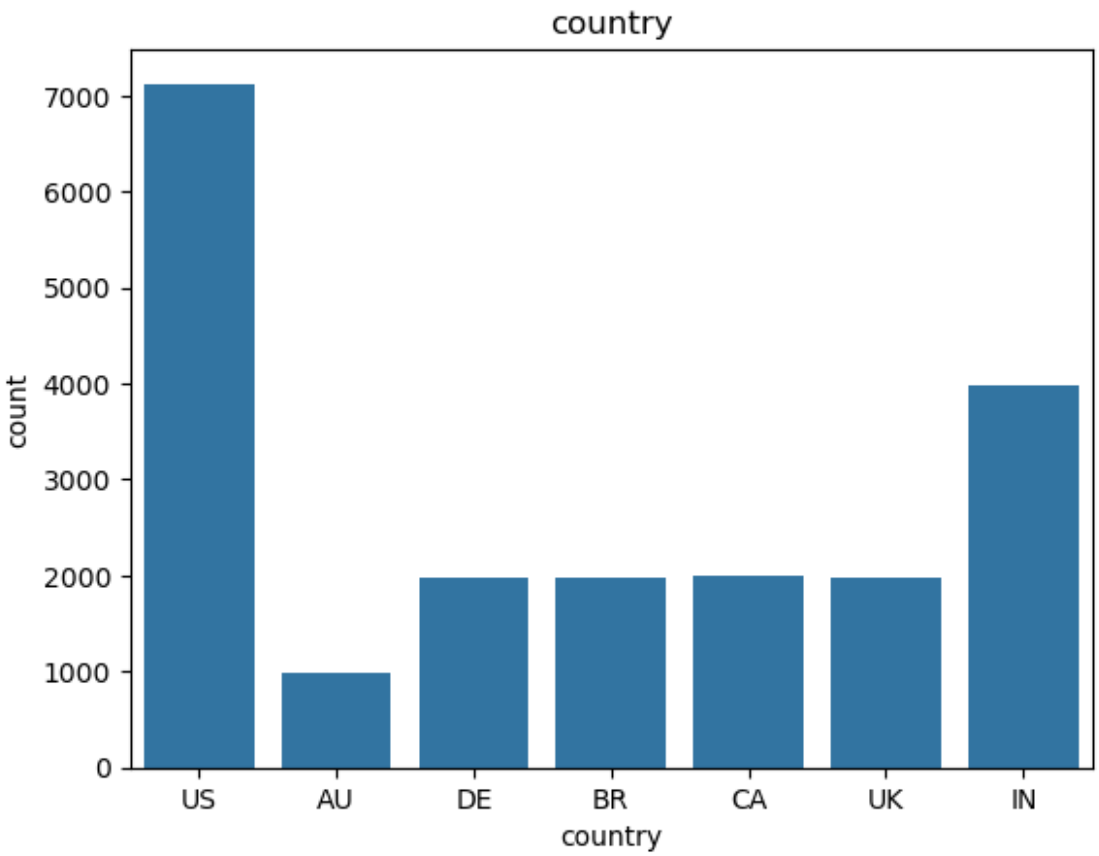

days_since_last_watch
Count
0
200
400
600
800
1000
1200
0
5
10
15
20
25
30
days_since_last_watch

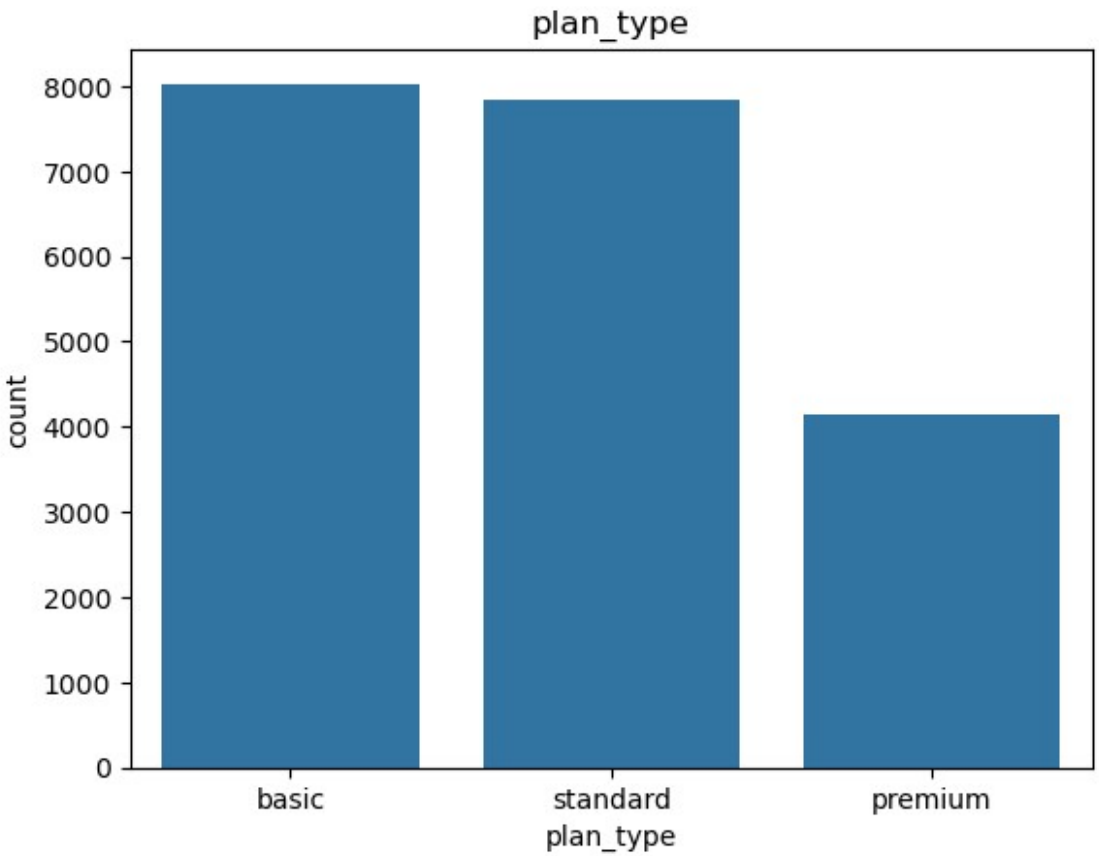

lifetime_value
Count
0
100
200
300
400
500
600
700
800
0
200
400
600
800
1000
1200
1400
1600
lifetime_value

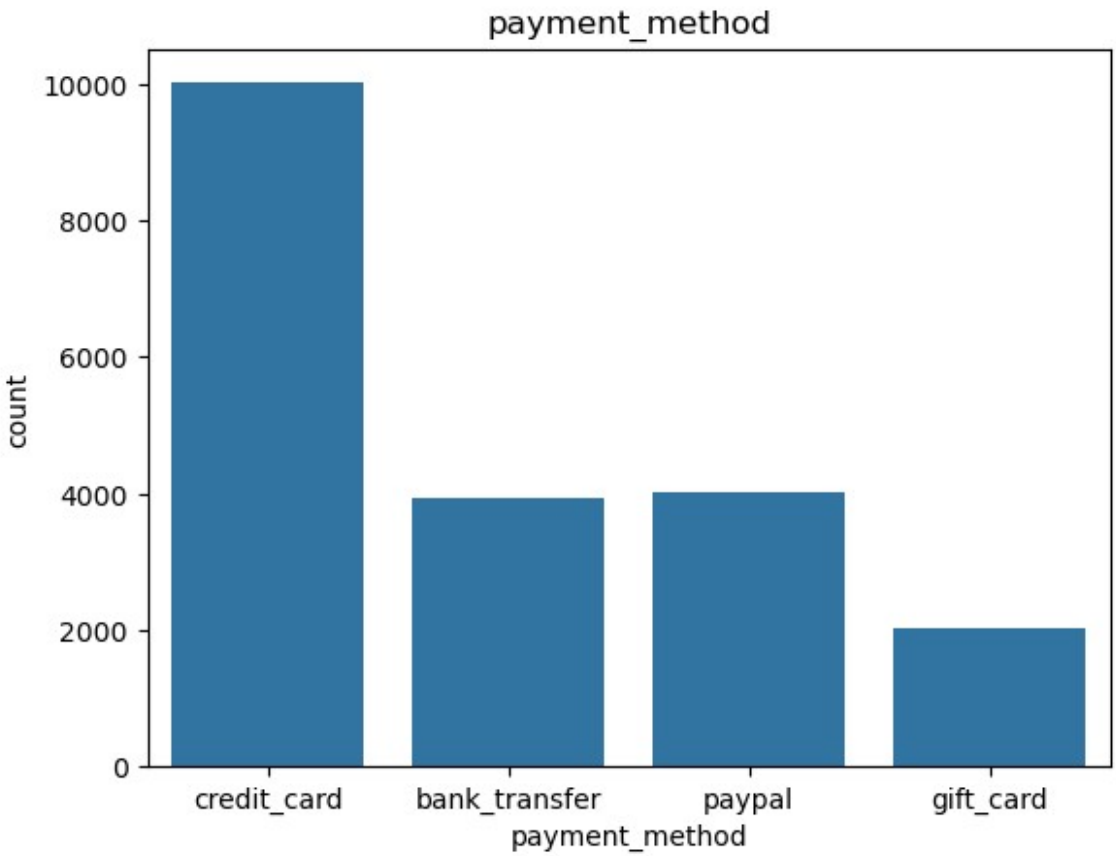

engagement_drop_ratio
Count
0
100
200
300
400
500
600
700
800
0.6
0.8
1.0
1.2
1.4
1.6
1.8
engagement_drop_ratio

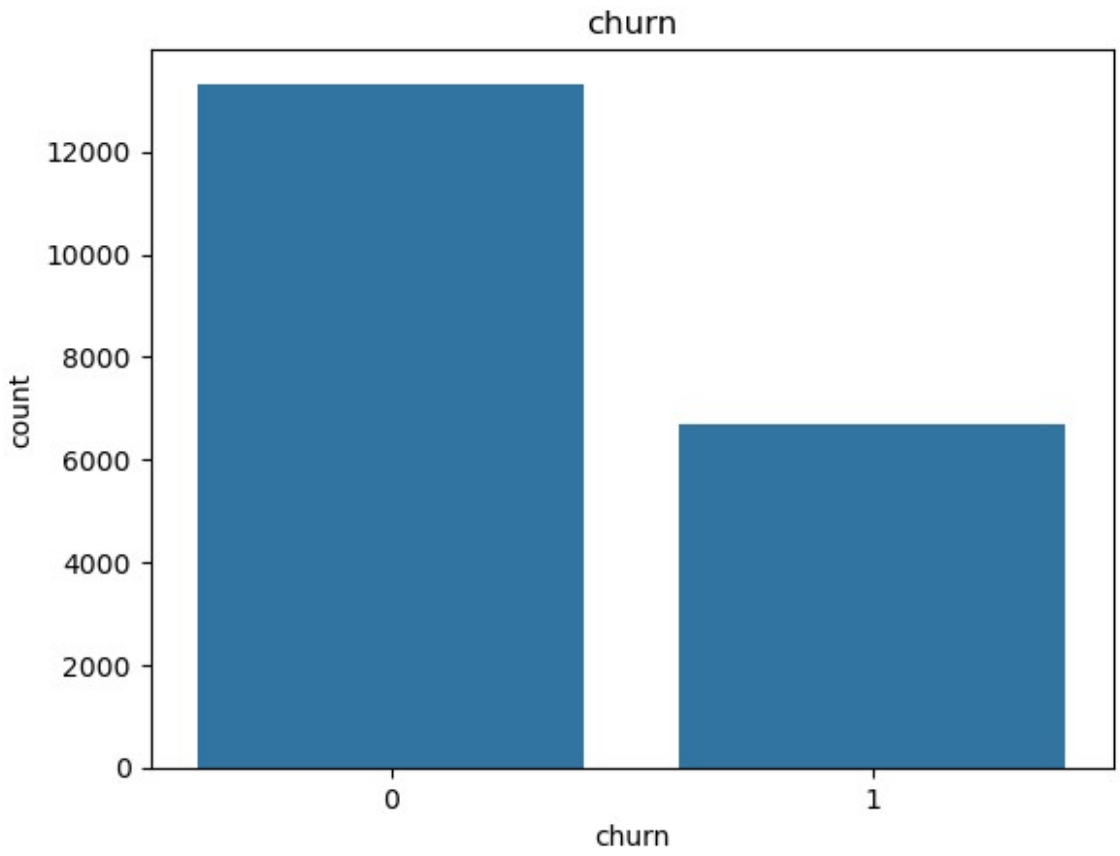

payment_failures
17500
15000
12500
10000
7500
5000
2500
0
Count
0.0
0.5
1.0
1.5
2.0
2.5
3.0
payment_failures


support_tickets
16000
14000
12000
10000
8000
6000
4000
2000
0
Count
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
support_tickets


country
7000
6000
5000
4000
3000
2000
1000
0
count
US
AU
DE
BR
CA
UK
IN
country


plan_type
8000
7000
6000
5000
4000
3000
2000
1000
0
count
basic
standard
premium
plan_type


payment_method
10000
8000
6000
4000
2000
0
count
credit_card
bank_transfer
paypal
gift_card
payment_method

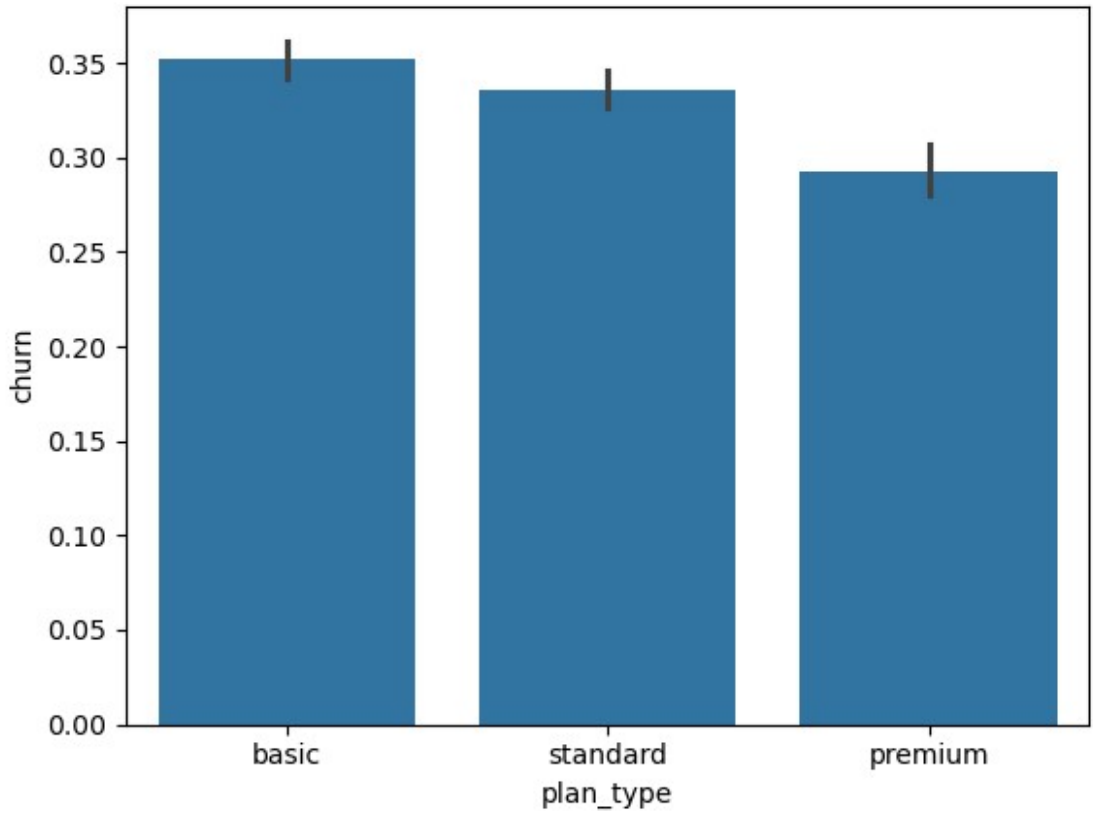


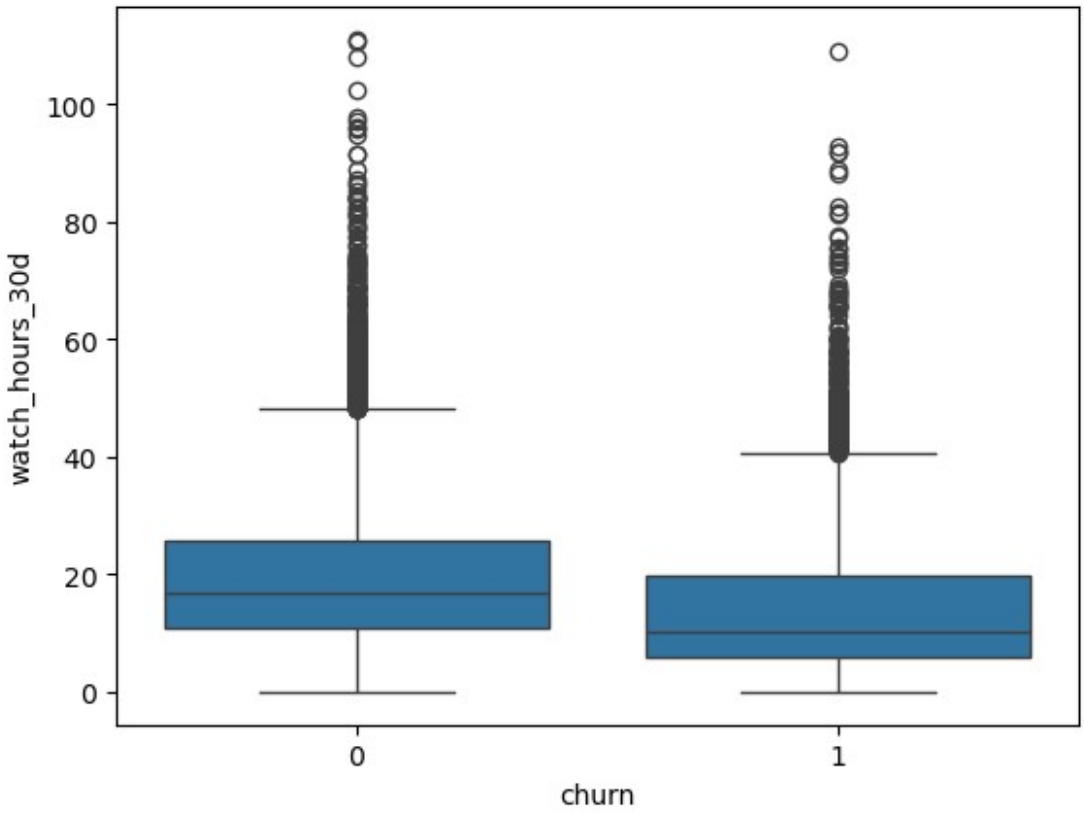

```
#22 SINGLE VS CHURN
# Churn rate by plan
print(df.groupby("plan_type")["churn"].mean())

# Churn rate by country
print(df.groupby("country")["churn"].mean())

# Churn vs engagement buckets
df["watch_bucket"] = pd.cut(df["watch_hours_30d"],
bins=[0,10,30,60,120],
                            labels=["0-10","10-30","30-60","60-120"])
print(df.groupby("watch_bucket")["churn"].mean())
```

```
plan_type
basic       0.352223
premium     0.292907
standard    0.336224
Name: churn, dtype: float64
country
AU    0.350663
BR    0.349391
CA    0.331994
DE    0.342119
```

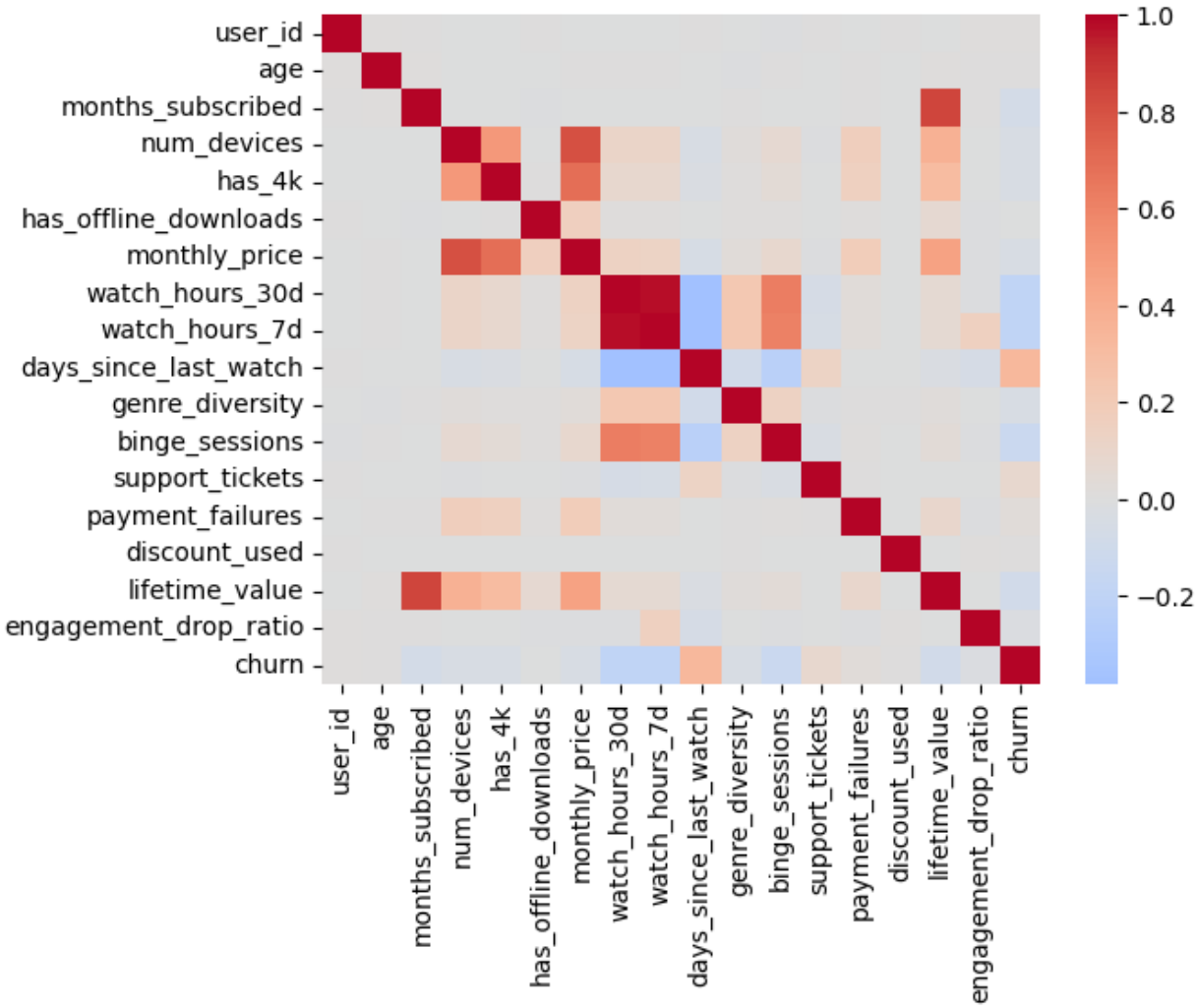
```
IN    0.325552
UK    0.322368
US    0.332865
Name: churn, dtype: float64
watch_bucket
0-10      0.539014
10-30     0.245915
30-60     0.235131
60-120    0.204762
Name: churn, dtype: float64

C:\Users\rithw\AppData\Local\Temp\ipykernel_1064\1764451689.py:11:
FutureWarning: The default of observed=False is deprecated and will be
changed to True in a future version of pandas. Pass observed=False to
retain current behavior or observed=True to adopt the future default
and silence this warning.
  print(df.groupby("watch_bucket")["churn"].mean())
```

```
sns.barplot(data=df, x="plan_type", y="churn")
plt.show()

sns.boxplot(data=df, x="churn", y="watch_hours_30d")
plt.show()
```






```
#23 CORRELATIONS
corr = df.corr(numeric_only=True)
sns.heatmap(corr, cmap="coolwarm", center=0)
plt.show()

corr["churn"].sort_values(ascending=False)
```




```
churn                    1.000000
days_since_last_watch    0.329457
support_tickets          0.080185
payment_failures         0.026488
user_id                  0.017029
discount_used            0.003831
age                      0.003822
has_offline_downloads    0.000081
engagement_drop_ratio   -0.018274
has_4k                  -0.039108
num_devices             -0.043888
genre_diversity         -0.044630
monthly_price           -0.045250
months_subscribed       -0.072744
lifetime_value          -0.086147
binge_sessions          -0.136070
watch_hours_30d         -0.195112
```

```
watch_hours_7d          -0.195669
Name: churn, dtype: float64
```