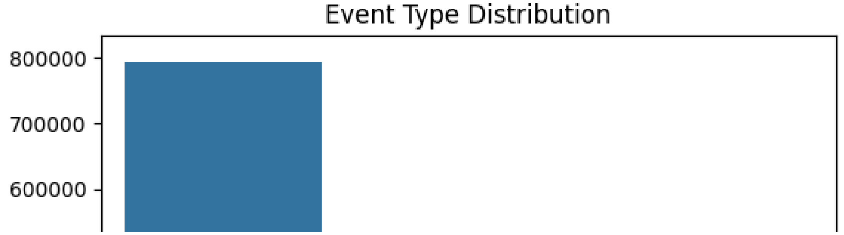
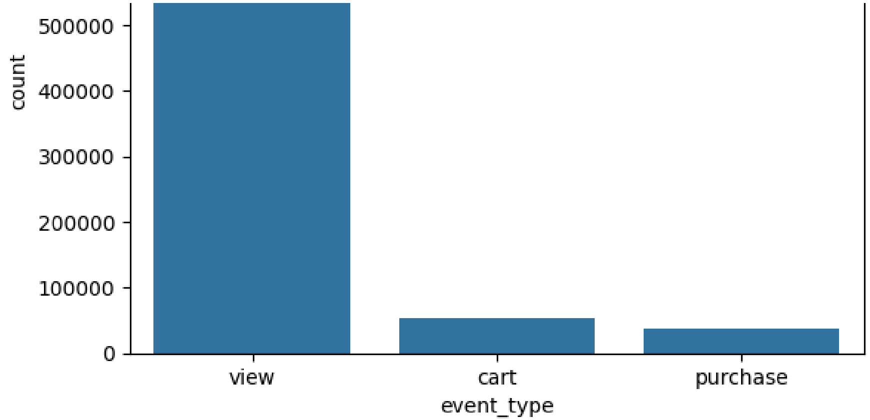
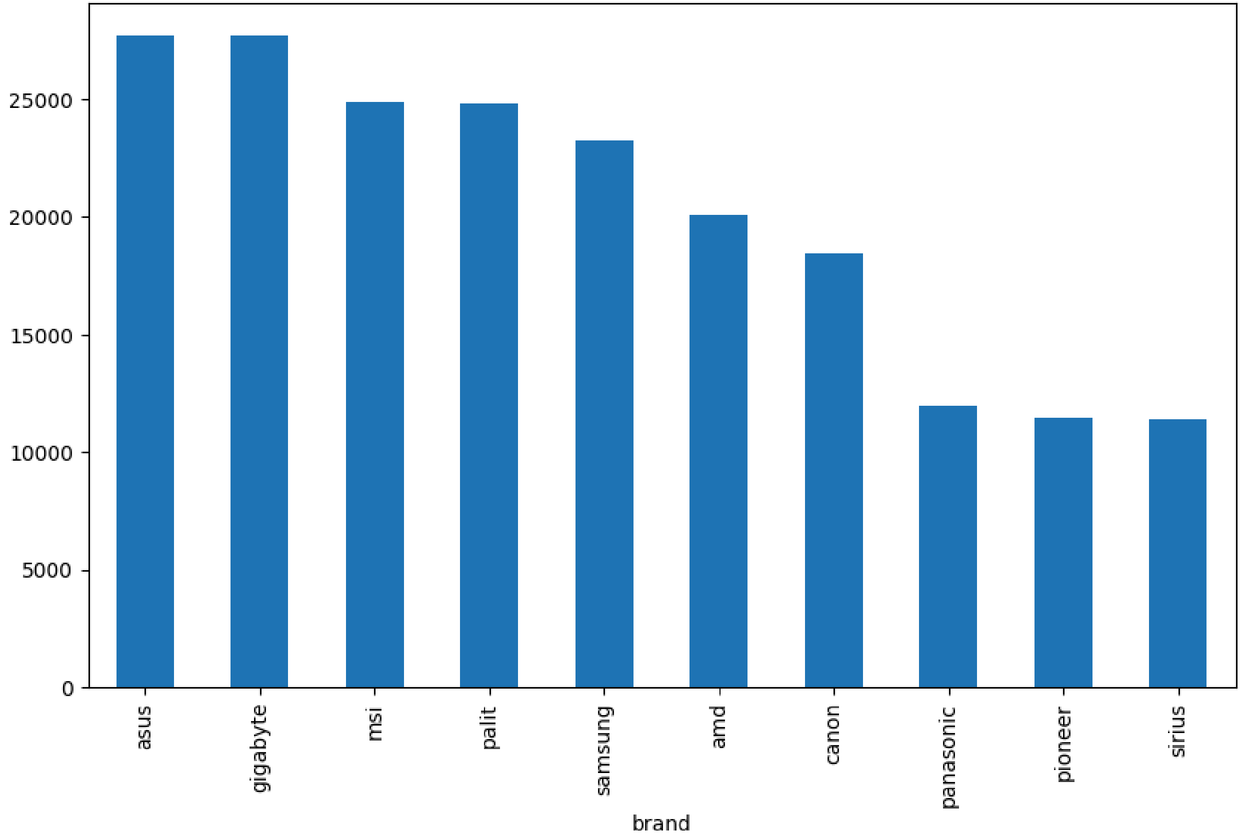
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

data = pd.read_csv('/content/events.csv')

data['event_time'] = pd.to_datetime(data['event_time'])


print(data.head())
print(data.info())
print(data.describe())

sns.countplot(x='event_type', data=data)
plt.title("Event Type Distribution")
plt.show()


top_brands = data['brand'].value_counts().head(10)
top_categories = data['category_code'].value_counts().head(10)

top_brands.plot(kind='bar', title="Top Brands", figsize=(10, 6))
plt.show()

top_categories.plot(kind='bar', title="Top Categories", figsize=(10, 6))
plt.show()

data['event_date'] = data['event_time'].dt.date
event_counts = data.groupby('event_date')['event_type'].count()
event_counts.plot(title="Events Over Time", figsize=(10, 6))
plt.show()

session_length = data.groupby('user_session')['event_time'].apply(lambda x: (x.max() - x.min()).seconds)
sns.histplot(session_length, bins=50, kde=True)
plt.title("Session Duration Distribution")
plt.xlabel("Session Duration (seconds)")
plt.show()
```

```
                 event_time event_type  product_id          category_id  \
0 2020-09-24 11:57:06+00:00       view     1996170  2144415922528452715
1 2020-09-24 11:57:26+00:00       view      139905  2144415926932472027
2 2020-09-24 11:57:27+00:00       view      215454  2144415927158964449
3 2020-09-24 11:57:33+00:00       view      635807  2144415923107266682
4 2020-09-24 11:57:36+00:00       view     3658723  2144415921169498184

                  category_code        brand   price              user_id  \
0          electronics.telephone          NaN   31.90  1515915625519388267
1    computers.components.cooler       zalman   17.16  1515915625519380411
2                            NaN          NaN    9.81  1515915625513238515
3  computers.peripherals.printer       pantum  113.81  1515915625519014356
4                            NaN  cameronsino   15.87  1515915625510743344

  user_session
0   LJuJVLEjPT
1   tdicluNnRY
2   4TMArHtXQy
3   aGFYrNgC08
4   aa4mmk0kwQ
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 885129 entries, 0 to 885128
Data columns (total 9 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   event_time     885129 non-null  datetime64[ns, UTC]
 1   event_type     885129 non-null  object
 2   product_id     885129 non-null  int64
 3   category_id    885129 non-null  int64
 4   category_code  648910 non-null  object
 5   brand          672765 non-null  object
 6   price          885129 non-null  float64
 7   user_id        885129 non-null  int64
 8   user_session   884964 non-null  object
dtypes: datetime64[ns, UTC](1), float64(1), int64(3), object(4)
memory usage: 60.8+ MB
None
         product_id   category_id          price       user_id
count  8.851290e+05  8.851290e+05  885129.000000  8.851290e+05
mean   1.906621e+06  2.144423e+18     146.328713  1.515916e+18
std    1.458708e+06  6.165105e+14     296.807683  3.747287e+07
min    1.020000e+02  2.144416e+18       0.220000  1.515916e+18
25%    6.988030e+05  2.144416e+18      26.460000  1.515916e+18
50%    1.452883e+06  2.144416e+18      65.710000  1.515916e+18
75%    3.721194e+06  2.144416e+18     190.490000  1.515916e+18
max    4.183880e+06  2.227847e+18   64771.060000  1.515916e+18
```

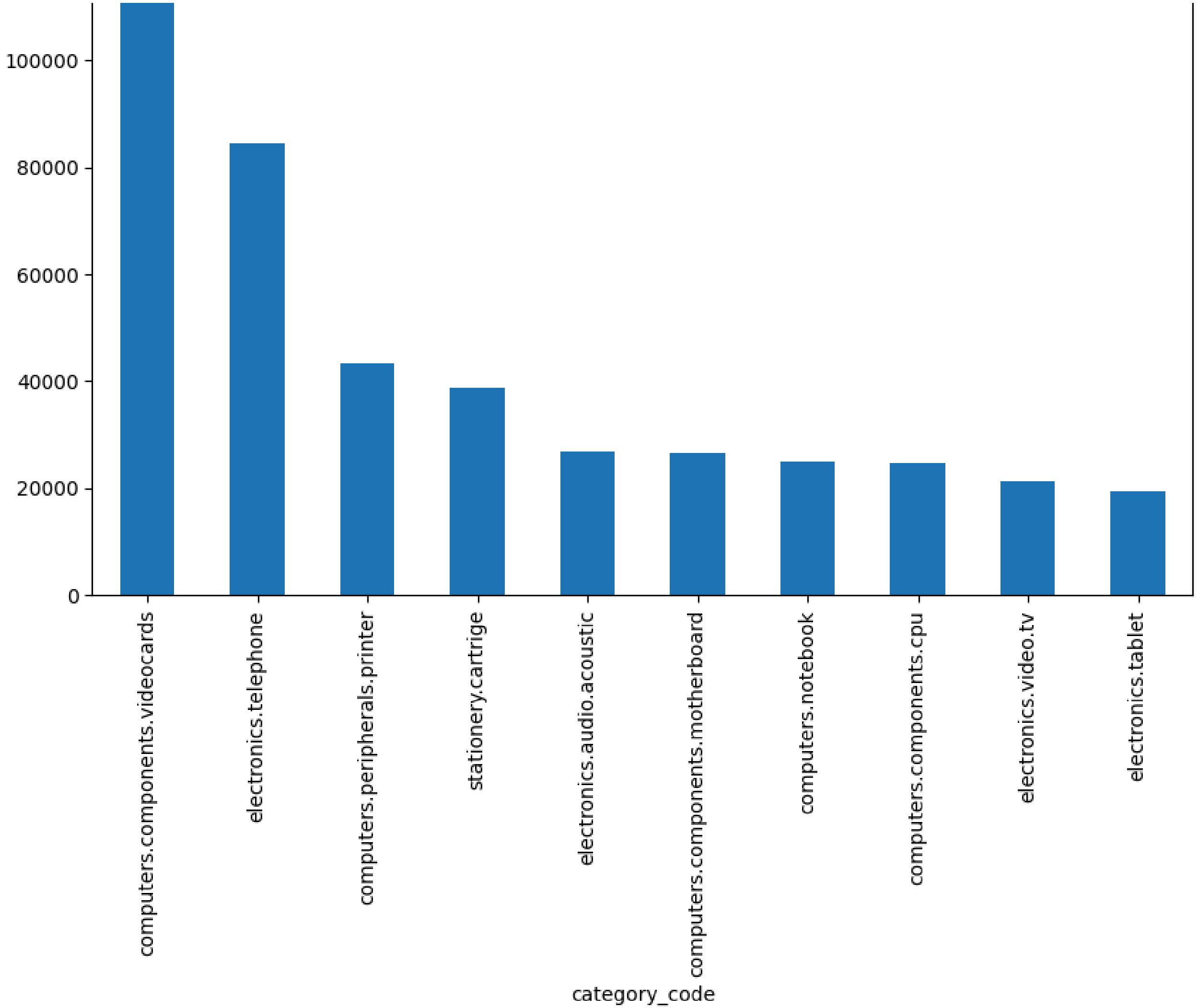


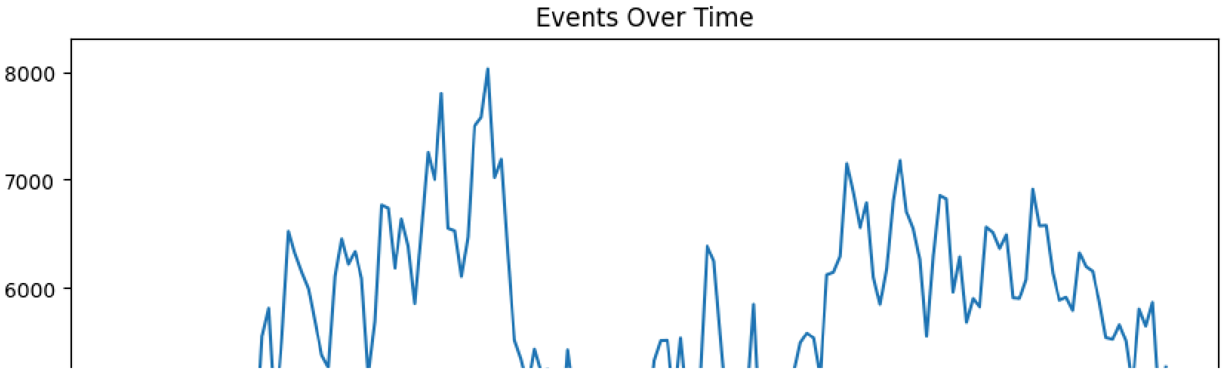


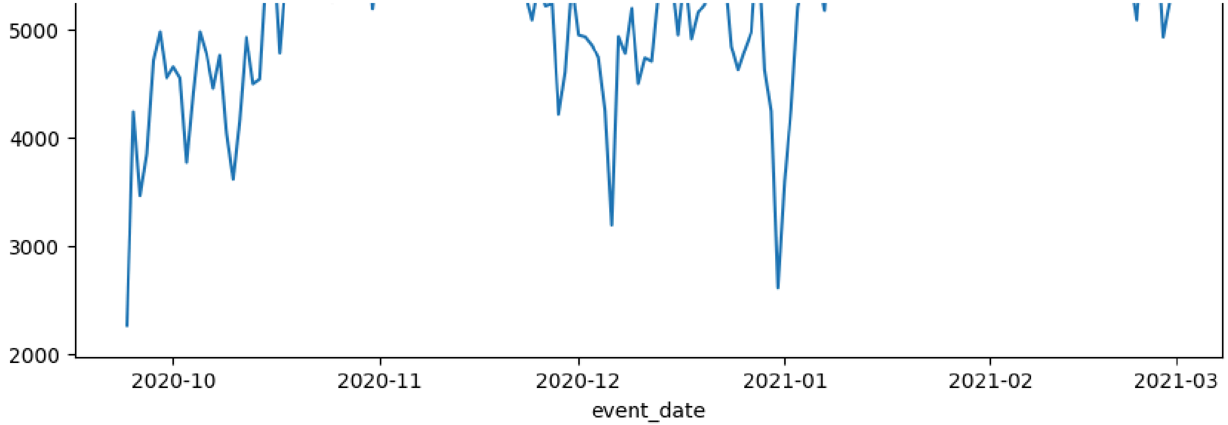
Top Brands

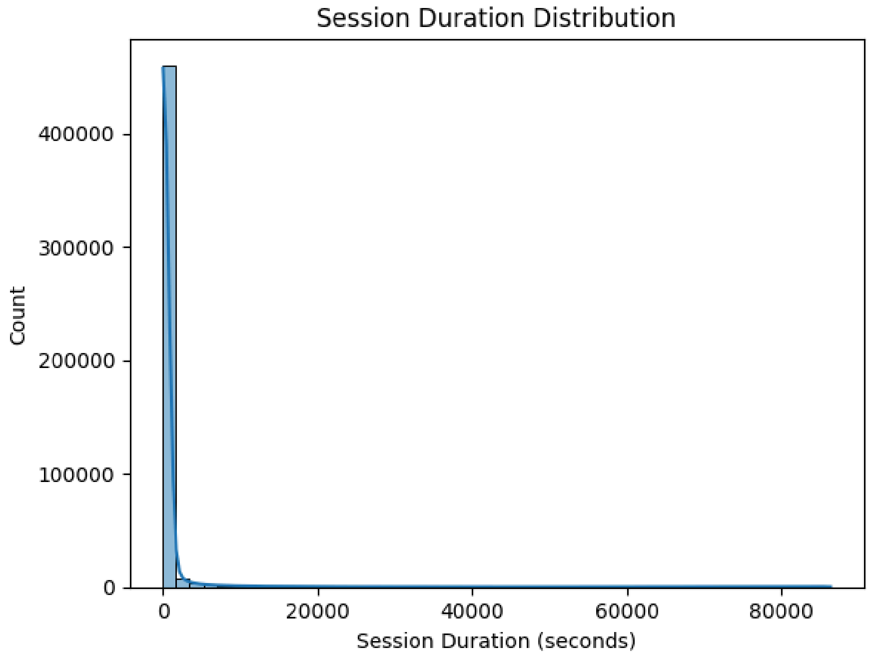


Top Categories


100000
80000
60000
40000
20000
0
computers.components.videocards
electronics.telephone
computers.peripherals.printer
stationery.cartrige
electronics.audio.acoustic
computers.components.motherboard
computers.notebook
computers.components.cpu
electronics.video.tv
electronics.tablet
category_code


Events Over Time
8000
7000
6000


5000
4000
3000
2000
2020-10
2020-11
2020-12
2021-01
2021-02
2021-03
event_date


Session Duration Distribution
Count
400000
300000
200000
100000
0
0
20000
40000
60000
80000
Session Duration (seconds)

```
from datetime import timedelta

# Define churn: No purchase in the last 30 days
data['is_purchase'] = data['event_type'] == 'purchase'
last_purchase_date = data.loc[data['is_purchase']].groupby('user_id')['event_time'].max()
max_date = data['event_time'].max()

# Churn threshold: 30 days since last purchase
churn_threshold = last_purchase_date + timedelta(days=30)
data['churn_status'] = data['user_id'].map(lambda x: x not in churn_threshold.index or churn_threshold[x] < max_date)

# Verify churn rate
churn_rate = data['churn_status'].mean()
print("The churn rate is {:.2%}".format(churn_rate))
```

```
The churn rate is 94.90%
```

```
# RFM Metrics
data['recency'] = data.groupby('user_id')['event_time'].transform(lambda x: (max_date - x.max()).days)
data['frequency'] = data.groupby('user_id')['event_type'].transform('count')
data['monetary'] = data.groupby('user_id')['price'].transform('sum')

# Behavioral patterns
data['view_to_cart_ratio'] = data.groupby('user_id')['event_type'].apply(
    lambda x: (x == 'cart').sum() / max((x == 'view').sum(), 1)
)
data['cart_to_purchase_ratio'] = data.groupby('user_id')['event_type'].apply(
    lambda x: (x == 'purchase').sum() / max((x == 'cart').sum(), 1)
)

# Preferences
data['top_brand'] = data.groupby('user_id')['brand'].transform(lambda x: x.mode()[0] if not x.mode().empty else 'Unknown')
data['top_category'] = data.groupby('user_id')['category_code'].transform(lambda x: x.mode()[0] if not x.mode().empty else 'Unknown')

# Final user-level dataset
user_features = data.groupby('user_id').agg({
    'recency': 'first',
    'frequency': 'first',
    'monetary': 'first',
    'view_to_cart_ratio': 'first',
    'cart_to_purchase_ratio': 'first',
    'top_brand': 'first',
    'top_category': 'first',
    'churn_status': 'first'
}).reset_index()

print(user_features.head())
```

```
               user_id  recency  frequency  monetary  view_to_cart_ratio  \
0  1515915625353226922      122          1     76.48                 NaN
1  1515915625353230067      145          1     28.98                 NaN
2  1515915625353230683       78         13    814.93                 NaN
3  1515915625353230922      149          1    274.40                 NaN
4  1515915625353234047       10         36   5481.90                 NaN

   cart_to_purchase_ratio top_brand                   top_category  \
0                     NaN     honor              electronics.clocks
1                     NaN    kester                         Unknown
2                     NaN  creative       electronics.audio.acoustic
3                     NaN       msi  computers.components.videocards
4                     NaN   samsung      electronics.audio.headphone

   churn_status
0          True
1          True
2          True
3          True
4          True
```

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, roc_auc_score, log_loss, brier_score_loss

# Prepare data for modeling
X = user_features[['recency', 'frequency', 'monetary', 'view_to_cart_ratio', 'cart_to_purchase_ratio', 'top_brand', 'top_category']]
X = pd.get_dummies(X, columns=['top_brand', 'top_category'], drop_first=True)
y = user_features['churn_status']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train Random Forest Classifier
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

# Model Evaluation
y_pred = model.predict(X_test)
y_prob = model.predict_proba(X_test)[:, 1]

# Classification Report
print("Classification Report:")
print(classification_report(y_test, y_pred))

# AUC Score
auc_score = roc_auc_score(y_test, y_prob)
print(f"AUC Score: {auc_score:.4f}")

# Log Loss
log_loss_value = log_loss(y_test, model.predict_proba(X_test))
print(f"Log Loss: {log_loss_value:.4f}")
```

```
# Brier Score Loss
brier_loss_value = brier_score_loss(y_test, y_prob)
print(f"Brier Score Loss: {brier_loss_value:.4f}")
import shap

# Explain model predictions using SHAP
explainer = shap.TreeExplainer(model)
shap_values = explainer.shap_values(X_test)

# SHAP Summary Plot
shap.summary_plot(shap_values[1], X_test)
```

```
Classification Report:
              precision    recall  f1-score   support

       False       0.43      0.29      0.35       886
        True       0.99      1.00      0.99     80571

    accuracy                           0.99     81457
   macro avg       0.71      0.64      0.67     81457
weighted avg       0.99      0.99      0.99     81457

AUC Score: 0.9670
```