In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
from sklearn.preprocessing import StandardScaler
from IPython.display import display, HTML
import re
```

In [3]:
```
data = pd.read_excel('common_dataset_touch_features_offset.xlsx')
df = data.copy()
```

In [4]:
```
def getting_primary_info(df):
    print("-------------------------------------------------------")
    print("Veri setinin şekli", df.shape)
    print("-------------------------------------------------------")
    print("Veri seti değişken tipleri:\n", df.dtypes)
    print("-------------------------------------------------------")
    print("Veri setinin ilk 5 satırı")
    display(HTML(df.head().to_html()))
    print("-------------------------------------------------------")
    print("Veri setinin istatistiki verileri")
    description = df.describe()
    display(HTML(description.to_html()))
    print("-------------------------------------------------------")
getting_primary_info(df)
```

```
-------------------------------------------------------
Veri setinin şekli (2056, 3206)
-------------------------------------------------------
Veri seti değişken tipleri:
 user_id       int64
touch_type    int64
touch          bool
finger         bool
palm           bool
              ...
3196          int64
3197          int64
3198          int64
3199          int64
3200          int64
Length: 3206, dtype: object
-------------------------------------------------------
Veri setinin ilk 5 satırı
```

	user_id	touch_type	touch	finger	palm	fist	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
0	0	0	False	False	False	False	4	0	12	-16	-2	0	-25	6	-6	5	4	-20	9	-14	-7	-8	-32	6	-25	15	-1	-25	18	-8	19	-4	19	12	13	-30	-69	34	22	5	6	7	50	-15	-10	-1
1	0	0	False	False	False	False	-52	-2	17	1	15	-9	-22	-17	-8	-25	31	9	-6	9	-25	-2	1	-3	1	1	0	0	8	22	-2	-7	11	32	15	11	-10	50	-45	16	28	-14	4	1	7	-12
2	0	0	False	False	False	False	2	7	14	-5	16	12	-17	-22	-13	3	9	-20	-14	4	-32	25	-2	40	-24	-28	-16	12	7	23	-6	20	27	-24	24	-19	-75	13	-45	18	41	-17	9	-37	40	-6
3	0	0	False	False	False	False	0	6	10	8	-2	2	1	-4	9	-26	32	-9	14	-23	-1	23	-33	32	-24	-31	-24	14	-25	9	21	-1	-28	-21	8	34	1	-2	5	-25	-1	31	11	-61	3	3
4	0	0	False	False	False	False	2	5	3	0	17	10	-19	-19	-13	1	4	-22	17	8	-1	-1	-3	-3	1	0	-2	-29	24	-14	21	-5	-29	-2	29	-17	-55	37	-20	-8	45	-21	20	-32	40	5

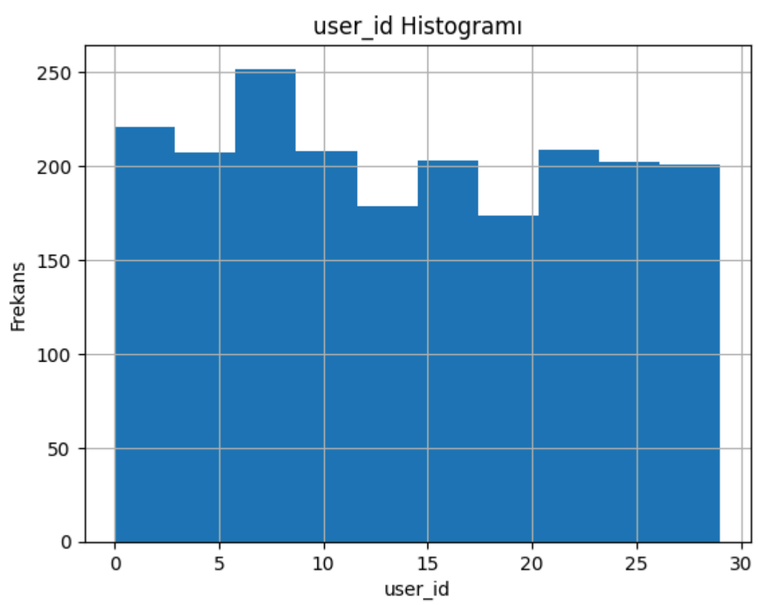
```
-------------------------------------------------------
Veri setinin istatistiki verileri
```

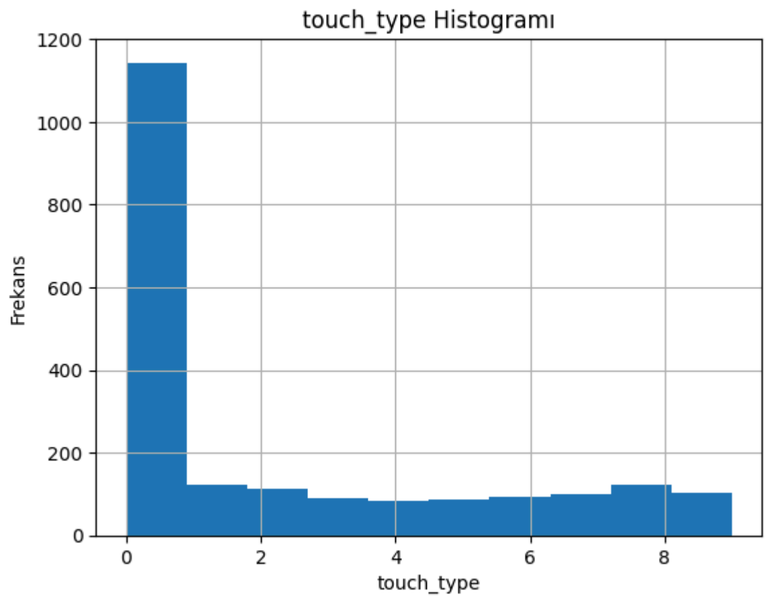
	user_id	touch_type	1	2	3	4	5	6	7	8	9	10	11	12	13	14	
count	2056.000000	2056.000000	2056.000000	2056.000000	2056.000000	2056.000000	2056.000000	2056.000000	2056.000000	2056.000000	2056.000000	2056.000000	2056.000000	2056.000000	2056.000000	2056.000000	2056.000
mean	14.144455	2.208658	6.383268	-0.736868	-1.372568	-1.951848	-0.771401	-0.479086	0.716926	0.893482	-4.085117	7.923152	-5.986868	-6.070039	0.230058	-1.211089	2.72
std	8.776936	3.059943	37.765997	7.140939	8.932638	10.529769	12.400623	9.899473	13.253678	14.909682	16.658413	16.579057	17.316836	14.343372	16.491068	16.753783	19.89
min	0.000000	0.000000	-62.000000	-22.000000	-23.000000	-27.000000	-29.000000	-28.000000	-31.000000	-34.000000	-37.000000	-38.000000	-36.000000	-40.000000	-37.000000	-41.000000	-39.00
25%	7.000000	0.000000	-2.000000	-6.000000	-8.000000	-9.000000	-9.000000	-7.000000	-7.000000	-9.000000	-18.000000	-3.000000	-22.000000	-17.000000	-10.000000	-15.000000	-5.00
50%	14.000000	0.000000	3.000000	-1.000000	-1.000000	-1.000000	-1.000000	0.000000	0.000000	0.000000	-3.000000	5.000000	-4.000000	-6.000000	0.000000	-1.000000	0.00
75%	22.000000	4.000000	53.000000	5.000000	4.000000	6.000000	9.000000	7.000000	11.000000	13.000000	7.000000	23.000000	2.000000	2.000000	14.000000	11.000000	26.00
max	29.000000	9.000000	70.000000	21.000000	21.000000	23.000000	41.000000	22.000000	70.000000	46.000000	32.000000	35.000000	39.000000	33.000000	34.000000	38.000000	62.00

```
-------------------------------------------------------
```

In [8]:
```
import matplotlib.pyplot as plt

for column in ["user_id", "touch_type"]:
    df[column].hist()
    plt.title(f'{column} Histogramı')
    plt.xlabel(column)
    plt.ylabel('Frekans')
    plt.show()
```





In [9]:
```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier

X, y = df.drop(columns = ["user_id","touch_type","touch","finger","palm","fist"]), df["user_id"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=47)

gb_model = GradientBoostingClassifier()
gb_model.fit(X_train, y_train)

y_pred_gb = gb_model.predict(X_test)
accuracy_gb = accuracy_score(y_test, y_pred_gb)
print("Gradient Boosting Accuracy:", accuracy_gb)
```

```
Gradient Boosting Accuracy: 0.9320388349514563
```

In [4]:
```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier
X, y = df.drop(columns = ["user_id","touch_type","touch","finger","palm","fist"]), df["touch"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=47)

gb_model = GradientBoostingClassifier()
gb_model.fit(X_train, y_train)

y_pred_gb = gb_model.predict(X_test)
accuracy_gb = accuracy_score(y_test, y_pred_gb)
print("Gradient Boosting Accuracy:", accuracy_gb)
```

```
Gradient Boosting Accuracy: 0.9077669902912622
```

In [9]:
```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier
X, y = df.drop(columns = ["user_id","touch_type","touch","finger","palm","fist"]), df["touch_type"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=47)

gb_model = GradientBoostingClassifier()
gb_model.fit(X_train, y_train)

y_pred_gb = gb_model.predict(X_test)
```

We couldn't calculate the remaining outputs and training scores due to the size of the data. Sorry about that.

```
accuracy_gb = accuracy_score(y_test, y_pred_gb)
print("Gradient Boosting Accuracy:", accuracy_gb)
```

```
Gradient Boosting Accuracy: 0.7475728155339806
```

In [ ]: