```
1 import pandas as pd
2 import numpy as np
3 from scipy import stats
4 import matplotlib.pyplot as plt
5
```

```
1 df = pd.read_csv("marketing_AB.csv")
2 df.head()
3
```

	Unnamed: 0	user id	test group	converted	total ads	most ads day	most ads hour
0	0	1069124	ad	False	130	Monday	20
1	1	1119715	ad	False	93	Tuesday	22
2	2	1144181	ad	False	21	Tuesday	18
3	3	1435133	ad	False	355	Tuesday	10
4	4	1015700	ad	False	276	Friday	14

```
1 df.columns
2
```

```
Index(['Unnamed: 0', 'user id', 'test group', 'converted', 'total ads',
       'most ads day', 'most ads hour'],
      dtype='object')
```

```
1 df = df.drop(columns=['Unnamed: 0'])
2
```

```
1 df['test group'].value_counts()
2
```

test group	count
ad	564577
psa	23524

dtype: int64

```
control = df[df['test group'] == 'control']
test = df[df['test group'] == 'ad']

```

```
# H0: Conversion rate of control = conversion rate of ad group
# H1: Conversion rate of control ≠ conversion rate of ad group

alpha = 0.05

```

```
control_rate = control['converted'].mean()
test_rate = test['converted'].mean()

print("Control Conversion Rate:", control_rate)
print("Ad Group Conversion Rate:", test_rate)


```

```
Control Conversion Rate: nan
Ad Group Conversion Rate: 0.025546559636683747
```

```
t_stat, p_value = stats.ttest_ind(
    control['converted'],
    test['converted'],
    equal_var=False
)

print("T-statistic:", t_stat)
print("P-value:", p_value)

```

```
T-statistic: nan
P-value: nan
/usr/local/lib/python3.12/dist-packages/scipy/_lib/deprecation.py:234: SmallSamp
  return f(*args, **kwargs)
```

```
if p_value < alpha:
    print("Reject H0 → Statistically significant difference")
else:
    print("Fail to reject H0 → No significant difference")

```

```
Fail to reject H0 → No significant difference
```

```
diff = test_rate - control_rate

se = np.sqrt(
    control['converted'].var()/len(control) +
```

```
    test['converted'].var()/len(test)
)

ci_low = diff - 1.96 * se
ci_high = diff + 1.96 * se

print("Mean Difference:", diff)
print("95% CI:", (ci_low, ci_high))

```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
/tmp/ipython-input-3586394585.py in <cell line: 0>()
      2
      3 se = np.sqrt(
----> 4     control['converted'].var()/len(control) +
      5     test['converted'].var()/len(test)
      6 )

ZeroDivisionError: float division by zero
```

Next steps: Explain error

```
print("Control size:", len(control))
print("Test size:", len(test))

```

```
Control size: 0
Test size: 564577
```

```
control = df[df['test group'] == 'control']
test = df[df['test group'] == 'test']

```

```
if len(control) > 0 and len(test) > 0:
    diff = test_rate - control_rate

    se = np.sqrt(
        control['converted'].var(ddof=1)/len(control) +
        test['converted'].var(ddof=1)/len(test)
    )

    ci_low = diff - 1.96 * se
    ci_high = diff + 1.96 * se

    print("Mean Difference:", diff)
    print("95% CI:", (ci_low, ci_high))
```

```
else:
    print("One of the groups is empty — cannot compute CI")

```

```
One of the groups is empty — cannot compute CI
```

```
df['test group'].value_counts()

```

test group	count
ad	564577
psa	23524

dtype: int64

```
control = df[df['test group'] == 'psa']
test = df[df['test group'] == 'ad']

```

```
print("Control size:", len(control))
print("Test size:", len(test))

```

```
Control size: 23524
Test size: 564577
```

```
control_rate = control['converted'].mean()
test_rate = test['converted'].mean()

print("Control conversion rate:", control_rate)
print("Ad conversion rate:", test_rate)

```

```
Control conversion rate: 0.01785410644448223
Ad conversion rate: 0.025546559636683747
```

```
diff = test_rate - control_rate

se = np.sqrt(
    control['converted'].var(ddof=1)/len(control) +
    test['converted'].var(ddof=1)/len(test)
)

ci_low = diff - 1.96 * se
ci_high = diff + 1.96 * se

```

```
print("Mean Difference:", diff)
print("95% Confidence Interval:", (ci_low, ci_high))

```

```
ıce: 0.007692453192201517
:e Interval: (np.float64(0.005950865393694846), np.float64(0.009434040990708189))
```

```
Start coding or generate with AI.
```