In [2]:
```
import pandas as pd
df= pd.read_csv('ab_test_data.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	user_id	timestamp	test_group	conversion
0	7f6833e6-1141-4f20-b4b2-f1e31019b1fd	2023-07-04 4:40:56	a	0
1	e6a6e960-d3f3-4074-a516-ba1e609b211e	2023-07-06 0:26:45	b	0
2	4d3fbfa5-6847-410a-bac2-477f01d5f400	2023-07-10 20:24:34	b	0
3	361457d9-a044-48f7-981c-d67dc3861679	2023-07-20 7:04:50	b	0
4	285cd63d-7d03-427f-a062-1fa2dd2e77d6	2023-07-19 23:27:50	b	0

In [11]:
```
df[(df['test_group']== 'a')].count()
```

Out[11]:
```
user_id       10013
timestamp     10013
test_group    10013
conversion    10013
dtype: int64
```

In [12]:
```
df[(df['test_group']== 'b')].count()
```

Out[12]:
```
user_id       9985
timestamp     9985
test_group    9985
conversion    9985
dtype: int64
```

In [13]:
```
df.count()
```

Out[13]:
```
user_id       19998
timestamp     19998
test_group    19998
conversion    19998
dtype: int64
```

In [32]:
```
df[(df['test_group']=='a') & (df['conversion']== 0)].count()
```

Out[32]:
```
user_id       9402
timestamp     9402
test_group    9402
conversion    9402
dtype: int64
```

In [34]:
```
df[(df['test_group']=='a') & (df['conversion']== 1)].count()
```

Out[34]:
```
user_id       611
timestamp     611
test_group    611
conversion    611
dtype: int64
```

In [35]:
```
df[(df['test_group']=='b') & (df['conversion']== 0)].count()
```

Out[35]:
```
user_id       9096
timestamp     9096
test_group    9096
conversion    9096
dtype: int64
```

In [36]:
```
df[(df['test_group']=='b') & (df['conversion']== 1)].count()
```

Out[36]:
```
user_id       889
timestamp     889
test_group    889
conversion    889
dtype: int64
```

In [52]:
```
df["timestamp"]= pd.to_datetime(df["timestamp"])

dates= df.groupby("test_group").agg(start = ("timestamp", "min"), finish= ("time

dates["total_days"]= (dates["finish"] - dates["start"]).dt.days

print(dates)
```

```
                      start              finish  total_days
test_group
a          2023-07-03 01:46:15 2023-07-25 01:41:19          21
b          2023-07-03 01:42:34 2023-07-25 01:35:59          21
```

In [59]:
```
import pandas as pd

# Test verisini oluştur
test_data = pd.DataFrame({
    'test_group': ['a'] * 10013 + ['b'] * 9985,
    'conversion': [1] * 9402 + [0] * (10013 - 9402) +
                  [1] * 9096 + [0] * (9985 - 9096)
})

# Her şeyin doğru bir şekilde oluşturulup oluşturulmadığını kontrol edelim:
test_data.groupby('test_group').describe()
```

Out[59]:

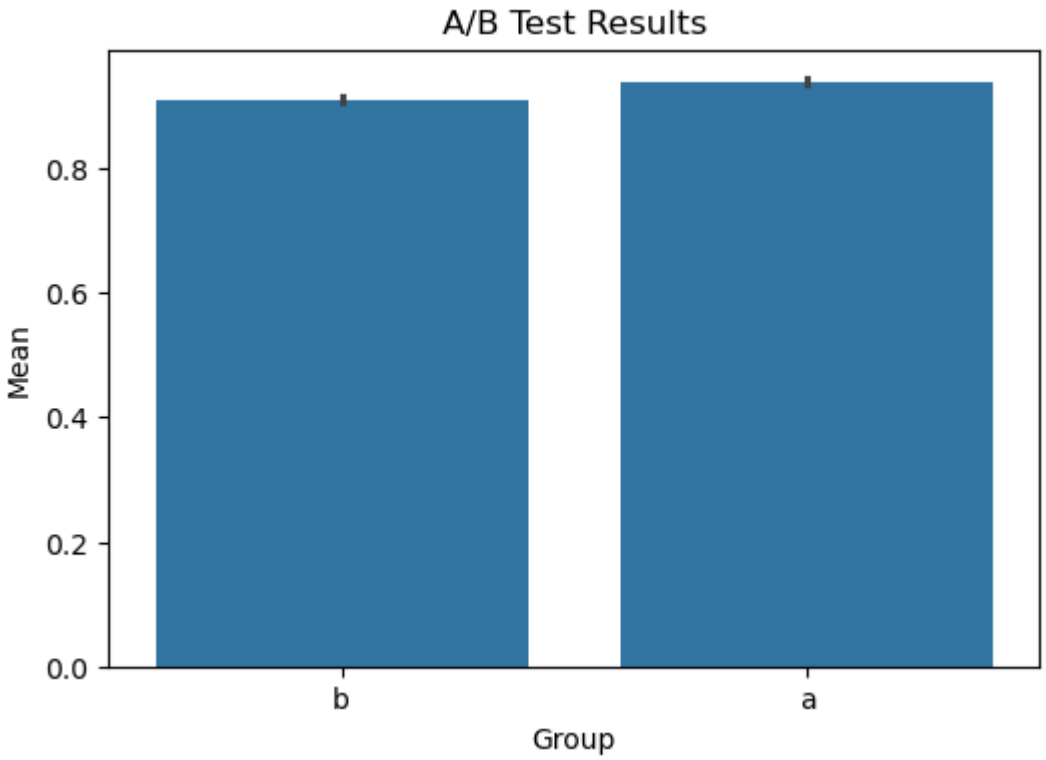
	conversion							
	count	mean	std	min	25%	50%	75%	max
test_group								
a	10013.0	0.938979	0.239380	0.0	1.0	1.0	1.0	1.0
b	9985.0	0.910966	0.284806	0.0	1.0	1.0	1.0	1.0

In [69]:
```
from scipy import stats

alpha = 0.05
```

```
statistic, pvalue = stats.ttest_ind(test_data[test_data['test_group'] == 'a']['c
                                    test_data[test_data['test_group'] == 'b']['c
                                    alternative='less')

print(f't-statistic: {round(statistic, 2)}, p-value: {round(pvalue, 2)}')

if pvalue < alpha:
    print('The difference is statistically significant, Null Hypothesis is rejec
else:
    print('The difference is insignificant, Null Hypothesis cannot be rejected.'
```

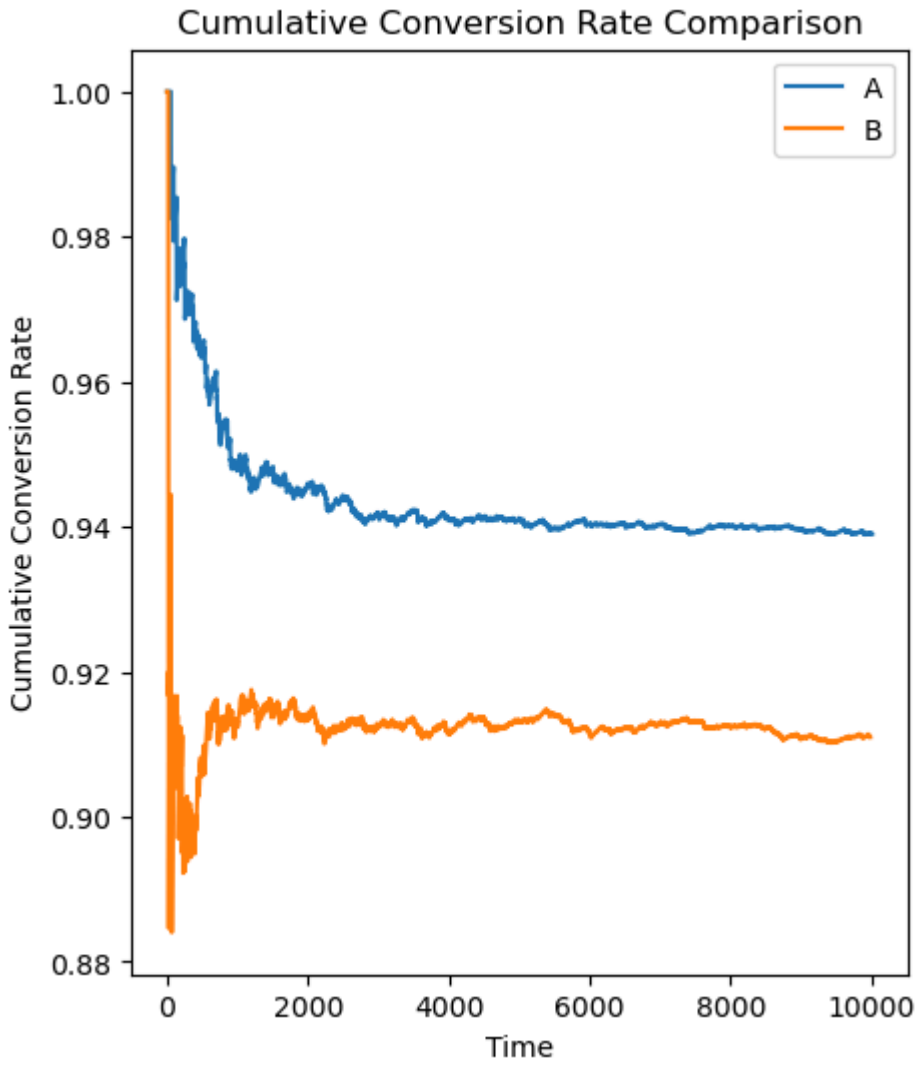
```
t-statistic: 7.53, p-value: 1.0
The difference is insignificant, Null Hypothesis cannot be rejected.
```

In [94]:
```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(6, 4))
sns.barplot(x=test_data['test_group'],
            y=test_data['conversion'],
            errorbar=('ci', 95)) # Confidence Intervals

plt.title('A/B Test Results')
plt.xlabel('Group')
plt.ylabel('Mean')

plt.show()
```



In [91]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Verileri karıştırıyoruz çünkü şu anki sıralama dönüşüm değerine göre yapılmış
# Gerçek verileri kullanmış olsaydık, burada tarih ve saat sıralaması yapılması
test_data = test_data.sample(frac=1).reset_index(drop=True)

# Kümülatif ortalamayı hesaplıyoruz - bu, zamanla dönüşüm değişimini gösterir
```

```
cumulative_metric_a = test_data[test_data['test_group'] == 'a']['conversion'].ex
cumulative_metric_b = test_data[test_data['test_group'] == 'b']['conversion'].ex

plt.figure(figsize=(5, 6))
plt.plot(cumulative_metric_a, label='A')
plt.plot(cumulative_metric_b, label='B')

plt.title('Cumulative Conversion Rate Comparison')
plt.xlabel('Time')
plt.ylabel('Cumulative Conversion Rate')

plt.legend()
plt.show()
```



In [ ]: