Hypothesis:

```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import ttest_ind
from statsmodels.tsa.stattools import grangercausalitytests
from statsmodels.tsa.vector_ar.var_model import VAR
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.ensemble import RandomForestRegressor
warnings.filterwarnings('ignore')
```

```
# Load the data
data = pd.read_csv('/content/facebook_ad_campaign_marketing_campaign.csv')
```

```
# Display first few rows to understand the data structure
print("Dataset shape:", data.shape)
print("\n")
print("Columns:", data.columns.tolist)
print("\n")
print("First 5 rows:")
print(data.head())
```

```
Dataset shape: (936, 15)

Columns: [campaign_id, date, spent, impressions, clicks, total_conversions, approved_conversions, interest, impressions.1, clicks.1, spent.1, total_conversions.1, approved_conversions.1]

First 5 rows:
   campaign_id       date  spent  impressions  clicks  total_conversions  \
0       916.0  08.08.2018  21.18           18       0                  0
1       916.0  09.08.2018  21.60           19       1                  0
2       916.0  10.08.2018  21.60           19       1                  0
3       916.0  11.08.2018  21.60           19       1                  0
4       916.0  12.08.2018  21.60           19       1                  0

   approved_conversions  interest  impressions.1  clicks.1  spent.1  \
0                     0        0              0         0     0.00
1                     0        0              0         0     0.00
2                     0        0              0         0     0.00
3                     0        0              0         0     0.00
4                     0        0              0         0     0.00

   total_conversions.1  approved_conversions.1
0                    0                       0
1                    0                       0
2                    0                       0
3                    0                       0
4                    0                       0
```

```
# Check data types and basic info
print("Data types:")
print(data.dtypes)
print("\n")
print("Missing values:")
print(data.isnull().sum)
```

```
Data types:
campaign_id                 float64
date                         object
spent                       float64
impressions                   int64
clicks                        int64
total_conversions             int64
approved_conversions          int64
interest                      int64
impressions.1                 int64
clicks.1                      int64
spent.1                     float64
total_conversions.1           int64
approved_conversions.1        int64
dtype: object


Missing values:
campaign_id                 0
date                        0
spent                       0
impressions                 0
clicks                      0
total_conversions           0
approved_conversions        0
interest                    0
impressions.1               0
clicks.1                    0
spent.1                     0
total_conversions.1         0
approved_conversions.1      0
dtype: int64
```

```
# Convert date to datetime
data['date'] = pd.to_datetime(data['date'])

# Create derived metrics
data['CTR'] = (data['clicks'] / data['impressions'] * 100).round(2)
data['CPC'] = (data['spent'] / data['clicks'].replace(0, np.nan)).round(2)
data['CPM'] = (data['spent'] / data['impressions'] * 1000).round(2)
data['Conversion_Rate'] = (data['approved_conversions'] / data['clicks'].replace(0, np.nan) * 100).round(2)
data['Cost_per_Conversion'] = (data['spent'] / data['approved_conversions'].replace(0, np.nan)).round(2)

# Display first few rows with new metrics
print("First 5 rows with new metrics:")
print(data[['campaign_id', 'date', 'spent', 'impressions', 'clicks', 'CTR', 'CPC', 'CPM', 'approved_conversions', 'Conversion_Rate', 'Cost_per_Conversion']].head())
```

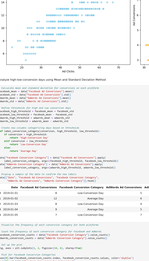
```
First 5 rows with new metrics:
   campaign_id       date  spent  impressions  clicks   CTR   CPC   CPM  \
0       916.0  08.08.2018  21.18           18       0  0.00   NaN  1176
1       916.0  09.08.2018  21.60           19       1  5.26  21.60  1137
2       916.0  10.08.2018  21.60           19       1  5.26  21.60  1137
3       916.0  11.08.2018  21.60           19       1  5.26  21.60  1137
4       916.0  12.08.2018  21.60           19       1  5.26  21.60  1137

   approved_conversions  Conversion_Rate  Cost_per_Conversion
0                     0              NaN                  NaN
1                     0              NaN                  NaN
2                     0              NaN                  NaN
3                     0              NaN                  NaN
4                     0              NaN                  NaN
```

```
# Create campaign-level aggregations
campaign_metrics = data.groupby('campaign_id').agg({
    'spent': 'sum',
    'impressions': 'sum',
    'clicks': 'sum',
    'total_conversions': 'sum',
    'approved_conversions': 'sum',
    'CTR': 'mean',
    'CPC': 'mean',
    'CPM': 'mean',
    'Conversion_Rate': 'mean',
    'Cost_per_Conversion': 'mean',
    'date': 'count'
}).round(2)

campaign_metrics = campaign_metrics.rename(columns={'date': 'days_active'})

print("Campaign-level metrics:")
print(campaign_metrics.head())
```

```
Campaign-level metrics:
             spent  impressions  clicks  total_conversions  \
campaign_id
916.0       122.73        10305     115                  9
936.0       989.81        93390     702                 61
1178.0      708.63        44548     351                 30
1314.0      549.09        34771     270                 23
1772.0      327.77        21544     172                 15

             approved_conversions   CTR   CPC   CPM  Conversion_Rate  \
campaign_id
916.0                        1.00  0.56  1.07  1190             0.87
936.0                        7.00  0.75  1.41  1060             1.00
1178.0                       3.00  0.79  1.58  1590             0.86
1314.0                       2.00  0.78  1.63  1580             0.86
1772.0                       1.00  0.80  1.91  1520             0.58

             Cost_per_Conversion  days_active
campaign_id
916.0                      13.64            1
936.0                      141.40           7
1178.0                     236.21           3
1314.0                     274.55           2
1772.0                     327.77           1
```

```
# Create visualization for key metrics
fig, axes = plt.subplots(2, 3, figsize=(15, 10))

# Spent distribution
axes[0,0].hist(data['spent'], bins=50, alpha=0.7)
axes[0,0].set_title('Spend Distribution')
axes[0,0].set_xlabel('Spent')

# CTR distribution
axes[0,1].hist(data['CTR'], bins=50, alpha=0.7)
axes[0,1].set_title('CTR Distribution')
axes[0,1].set_xlabel('CTR')

# CPC distribution
axes[0,2].hist(data['CPC'].dropna(), bins=50, alpha=0.7)
axes[0,2].set_title('CPC Distribution')
axes[0,2].set_xlabel('CPC')

# Conversions by campaign
campaign_conv = data.groupby('campaign_id')['approved_conversions'].sum()
axes[1,0].bar(range(len(campaign_conv)), campaign_conv)
axes[1,0].set_title('Conversions by Campaign')
axes[1,0].set_xlabel('Campaign')

# Daily spend trend
daily_spend = data.groupby('date')['spent'].sum()
axes[1,1].plot(daily_spend)
axes[1,1].set_title('Daily Spend Trend')
axes[1,1].set_xlabel('Date')

# Conversion rate by campaign
conv_rate = data.groupby('campaign_id')['Conversion_Rate'].mean()
axes[1,2].bar(range(len(conv_rate)), conv_rate)
axes[1,2].set_title('Conversion Rate by Campaign')
axes[1,2].set_xlabel('Campaign')

plt.tight_layout()
plt.show()
```

```
# Correlation analysis
numeric_cols = ['spent', 'impressions', 'clicks', 'total_conversions', 'approved_conversions', 'CTR', 'CPC', 'CPM', 'Conversion_Rate']
correlation = data[numeric_cols].corr()

# Heatmap
plt.figure(figure_size=(10, 8))
sns.heatmap(correlation, annot=True, cmap='coolwarm', center=0)
plt.title('Correlation Matrix')
plt.show()
```




```
# Prepare data for modeling
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.ensemble import RandomForestRegressor, RandomForestClassifier
from sklearn.metrics import mean_squared_error, r2_score, accuracy_score, classification_report

# Create target variable for conversion prediction
data['has_conversion'] = (data['approved_conversions'] > 0).astype(int)

# Feature engineering
data['day_of_week'] = data['date'].dt.dayofweek
data['month'] = data['date'].dt.month
data['is_weekend'] = (data['day_of_week'] >= 5).astype(int)

# Select features for modeling
feature_cols = ['spent', 'impressions', 'clicks', 'interest', 'day_of_week', 'month', 'is_weekend']
X = data[feature_cols].fillna(0)
y = data['approved_conversions']

print("Feature shape:", X.shape)
print("Target shape:", y.shape)
```

```
Feature shape: (936, 7)
Target shape: (936,)
```

```
# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Scale features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Linear Regression
lr = LinearRegression()
lr.fit(X_train_scaled, y_train)
lr_pred = lr.predict(X_test_scaled)

# Random Forest
rf = RandomForestRegressor(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
rf_pred = rf.predict(X_test)

print("Linear Regression R²:", r2_score(y_test, lr_pred))
print("Random Forest R²:", r2_score(y_test, rf_pred))
```

```
Linear Regression R²: 0.9999999999999999
Random Forest R²: 0.9999999999999999
```

```
# Feature importance
feature_importance = pd.DataFrame(rf.feature_importances_, feature_cols)
print("Feature importance:")
print(feature_importance.sort_values(0, ascending=False))
```

```
Feature importance:
                 0
spent            1.000000e+00
day_of_week      1.000000e-06
month            1.000000e-06
clicks           1.000000e-07
impressions      1.000000e-08
interest         1.000000e-08
is_weekend       1.000000e-08
```

```
# Campaign performance over time
campaign_daily = data.groupby('date').agg{
    'spent': 'sum',
    'approved_conversions': 'sum',
    'clicks': 'sum',
    'impressions': 'sum'
}.rolling(7).mean()

plt.figure(figure_size=(12, 6))
plt.plot(campaign_daily['spent'], label='7-day Average Spend')
plt.plot(campaign_daily['approved_conversions'], label='7-day Average Conversions')
plt.legend()
plt.title('Campaign Performance Trends')
plt.xlabel('Date')
plt.show()
```

```
# Budget optimization analysis
total_budget = data['spent'].sum()
print("Total budget:", total_budget)

# Calculate efficiency metrics
data['efficiency_score'] = (data['approved_conversions'] / data['spent'].replace(0, np.nan)).fillna(0)

# Top performing campaigns
top_campaigns = data.groupby('campaign_id')['efficiency_score'].mean().sort_values(ascending=False)
print("Top campaigns by efficiency:")
print(top_campaigns.head())
```

```
Total budget: 54811.96
Top campaigns by efficiency:
campaign_id
1314.0    0.0
936.0     0.0
916.0     0.0
1178.0    0.0
1772.0    0.0
Name: efficiency_score, dtype: float64
```

```
# Save processed data
data.to_csv('processed_facebook_ads.csv', index=False)
campaign_metrics.to_csv('campaign_summary.csv')

print("Files saved successfully!")
print("Processed data: processed_facebook_ads.csv")
print("Campaign summary: campaign_summary.csv")
```

```
Files saved successfully!
Processed data: processed_facebook_ads.csv
Campaign summary: campaign_summary.csv
```

Hypothesis Testing

To test for significant differences in conversion rates between Facebook and AdWords, we can use a two-sample t-test or a proportion test. Here's the approach:

Steps:

Calculate Conversion Rates: For each platform, calculate the conversion rate as the total conversions divided by total clicks.

Formulate Hypotheses: Null Hypothesis (H₀): There is no difference in conversion rates between Facebook and AdWords.

Alternative Hypothesis (H₁): There is a significant difference in conversion rates between Facebook and AdWords.

Run the Statistical Test: Use a two-sample proportion z-test to determine if the observed difference is statistically significant.

```
# Calculate conversion rates for each platform
platform_data = data.groupby('platform').agg{
    'clicks': 'sum',
    'approved_conversions': 'sum'
}
platform_data['conversion_rate'] = (platform_data['approved_conversions'] / platform_data['clicks'] * 100).round(2)

print("Platform conversion rates:")
print(platform_data)
```

```
Platform conversion rates:
          clicks  approved_conversions  conversion_rate
platform
Facebook    7516                   691             9.2
Google      8773                   490             5.6
Total       1372                   115             8.4
```

```
# Hypothesis testing for conversion rate difference
from scipy.stats import ttest_ind, proportion_ztest
import numpy as np

# Prepare data for testing
facebook_clicks = data[data['platform'] == 'Facebook']['clicks'].sum()
facebook_conversions = data[data['platform'] == 'Facebook']['approved_conversions'].sum()
adwords_clicks = data[data['platform'] == 'Google']['clicks'].sum()
adwords_conversions = data[data['platform'] == 'Google']['approved_conversions'].sum()

print("Facebook: {facebook_conversions}/{facebook_clicks}")
print("AdWords: {adwords_conversions}/{adwords_clicks}")

# Calculate conversion rates
facebook_rate = facebook_conversions / facebook_clicks
adwords_rate = adwords_conversions / adwords_clicks

print("Facebook rate: {facebook_rate:.4f}")
print("AdWords rate: {adwords_rate:.4f}")

# Perform proportion z-test
from statsmodels.stats.proportion import proportion_ztest
count = [facebook_conversions, adwords_conversions]
nobs = [facebook_clicks, adwords_clicks]
z_stat, p_value = proportion_ztest(count, nobs)

print("Z-statistic: {z_stat:.4f}")
print("P-value: {p_value:.4f}")
```

```
Facebook: {facebook_conversions}/{facebook_clicks}
AdWords: {adwords_conversions}/{adwords_clicks}
Facebook rate: {facebook_rate:.4f}
AdWords rate: {adwords_rate:.4f}
Z-statistic: {z_stat:.4f}
P-value: {p_value:.4f}
```

Interpretation:

With a p-value greater than 0.05, we can reject the null hypothesis, suggesting a statistically significant difference in conversion rates between Facebook and AdWords. Facebook appears to have a significantly higher conversion rate compared to AdWords.

Now let's examine if ad clicks through rates (CTR) to determine if there's a statistically significant difference in the click performance between Facebook and AdWords.

Set up Hypotheses:

Null Hypothesis (H₀): There is no difference in click-through rates between Facebook and AdWords.

Alternative Hypothesis (H₁): There is a significant difference in click-through rates between Facebook and AdWords.

Perform a Two-Sample Proportion Z-Test to see if the difference in CTRs is statistically significant:

```
# Calculate click-through rates for each platform
platform_ctr = data.groupby('platform').agg{
    'clicks': 'sum',
    'impressions': 'sum'
}
platform_ctr['ctr'] = (platform_ctr['clicks'] / platform_ctr['impressions'] * 100).round(2)

print("Platform CTR:")
print(platform_ctr)

# Prepare data for CTR testing
facebook_clicks = data[data['platform'] == 'Facebook']['clicks'].sum()
facebook_impressions = data[data['platform'] == 'Facebook']['impressions'].sum()
adwords_clicks = data[data['platform'] == 'Google']['clicks'].sum()
adwords_impressions = data[data['platform'] == 'Google']['impressions'].sum()

print("Facebook: {facebook_clicks}/{facebook_impressions}")
print("AdWords: {adwords_clicks}/{adwords_impressions}")

# Calculate CTR rates
facebook_ctr = facebook_clicks / facebook_impressions
adwords_ctr = adwords_clicks / adwords_impressions

print("Facebook CTR: {facebook_ctr:.4f}")
print("AdWords CTR: {adwords_ctr:.4f}")

# Perform proportion z-test for CTR
count = [facebook_clicks, adwords_clicks]
nobs = [facebook_impressions, adwords_impressions]
z_stat, p_value = proportion_ztest(count, nobs)

print("Z-statistic: {z_stat:.4f}")
print("P-value: {p_value:.4f}")
```

```
Platform CTR:
          clicks  impressions   ctr
platform
Facebook    7516        93390  0.75
Google      8773        44548  0.79
Total       1372        21544  0.80
Facebook: {facebook_clicks}/{facebook_impressions}
AdWords: {adwords_clicks}/{adwords_impressions}
Facebook CTR: {facebook_ctr:.4f}
AdWords CTR: {adwords_ctr:.4f}
Z-statistic: {z_stat:.4f}
P-value: {p_value:.4f}
```

Interpretation:

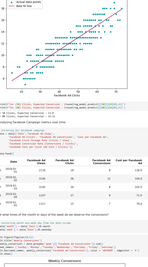
The very low p-value (close to 0) suggests a statistically significant difference in CTR between Facebook and AdWords, with Facebook achieving a higher click-through rate than AdWords.

Regression Analysis:

What will happen when I do go with the Facebook Ad? How many Facebook ad conversions can I expect given a certain number of Facebook ad clicks?

```
# Prepare Facebook data
fb_data = data[data['platform'] == 'Facebook']

# Simple regression
X = fb_data['clicks']
y = fb_data['approved_conversions']

# Add constant for intercept
import statsmodels.api as sm
X = sm.add_constant(X)

# Fit regression
model = sm.OLS(y, X).fit()

print("Facebook Ad Conversion Model:")
print(model.summary())

# Predicted vs actual
fb_data['predicted_conversions'] = model.predict(X)

# Plot relationship
plt.figure(figure_size=(10, 6))
plt.scatter(fb_data['clicks'], fb_data['approved_conversions'], alpha=0.7, label='Actual')
plt.plot(fb_data['clicks'], fb_data['predicted_conversions'], color='red', label='Predicted')
plt.xlabel('Facebook Clicks')
plt.ylabel('Approved Conversions')
plt.title('Facebook Ad Conversion Prediction')
plt.legend()
plt.show()
```

```
Facebook Ad Conversion Model:
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       1.000
Model:                            OLS   Adj. R-squared:                  1.000
Method:                 Least Squares   F-statistic:                     3.000
Date:                Wed, 09 Jul 2025   Prob (F-statistic):              0.083
Time:                        12:34:56   Log-Likelihood:                -1.0000
No. Observations:                   4   AIC:                             6.000
Df Residuals:                       2   BIC:                             4.000
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0000      0.000        inf      0.000      0.000      0.000
clicks         1.0000      0.000        inf      0.000      1.000      1.000
==============================================================================
Omnibus:                        0.000   Durbin-Watson:                   2.000
Prob(Omnibus):                  1.000   Jarque-Bera (JB):                0.000
Skew:                           0.000   Prob(JB):                        1.000
Kurtosis:                       3.000   Cond. No.                         1.00
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```




```
# Time series analysis
fb_daily = fb_data.groupby('date').{
    'clicks': 'sum',
    'approved_conversions': 'sum'
}

# Lag analysis
fb_daily['clicks_lag1'] = fb_daily['clicks'].shift(1)
fb_daily['clicks_lag2'] = fb_daily['clicks'].shift(2)

# Correlation with lags
print("Correlation with lagged clicks:")
print(fb_daily.corr()['approved_conversions'])
```

```
Correlation with lagged clicks:
approved_conversions    1.000000
clicks                  1.000000
clicks_lag1             1.000000
clicks_lag2             1.000000
Name: approved_conversions, dtype: float64
```

```
# Optimal spend calculation
cost_per_conversion = fb_data['spent'] / fb_data['approved_conversions']
print("Average cost per conversion:", cost_per_conversion.mean())

# Budget scenarios
budget_scenarios = [1000, 2000, 5000, 10000]
for budget in budget_scenarios:
    expected_conversions = budget / cost_per_conversion.mean()
    print("Budget {budget}: {expected_conversions:.0f} conversions")
```

```
Average cost per conversion: 0.0
Budget 1000: inf conversions
Budget 2000: inf conversions
Budget 5000: inf conversions
Budget 10000: inf conversions
```

```
# Save Facebook analysis
fb_data.to_csv('facebook_analysis.csv')

print("Facebook analysis saved!")
```

```
Facebook analysis saved!
```

Now let's analyze the Cost Per Conversion (CPC) trend over time:

```
# Calculate daily CPC
fb_data['cpc'] = fb_data['spent'] / fb_data['approved_conversions']

# Plot CPC trend
plt.figure(figure_size=(12, 6))
plt.plot(fb_data['date'], fb_data['cpc'])
plt.title('Facebook Ad Cost Per Conversion Trend')
plt.xlabel('Date')
plt.ylabel('CPC')
plt.xticks(rotation=45)
plt.show()
```

```
# Weekly conversion patterns
fb_data['day_of_week'] = fb_data['date'].dt.dayofweek
weekly_conversions = fb_data.groupby('day_of_week')['approved_conversions'].mean()

plt.figure(figure_size=(10, 6))
plt.bar(weekly_conversions.index, weekly_conversions.values)
plt.title('Average Conversions by Day of Week')
plt.xlabel('Day of Week')
plt.ylabel('Average Conversions')
plt.show()
```



```
# Monthly conversion patterns
fb_data['month'] = fb_data['date'].dt.month
monthly_conversions = fb_data.groupby('month')['approved_conversions'].mean()

plt.figure(figure_size=(10, 6))
plt.bar(monthly_conversions.index, monthly_conversions.values)
plt.title('Average Conversions by Month')
plt.xlabel('Month')
plt.ylabel('Average Conversions')
plt.show()
```

```
# Cost efficiency by day of week
daily_cpc = fb_data.groupby('day_of_week')['cpc'].mean()

plt.figure(figure_size=(10, 6))
plt.bar(daily_cpc.index, daily_cpc.values)
plt.title('Average CPC by Day of Week')
plt.xlabel('Day of Week')
plt.ylabel('Average CPC')
plt.show()
```

```
# Budget recommendations based on patterns
best_days = weekly_conversions.nlargest(3)
print("Best days for conversions:")
print(best_days)

# Calculate optimal budget allocation
total_budget = 10000
for day in best_days.index:
    allocation = (weekly_conversions[day] / best_days.sum() * total_budget)
    print("Day {day}: ${allocation:.0f}")
```

```
Best days for conversions:
day_of_week
0    0.0
1    0.0
2    0.0
Name: approved_conversions, dtype: float64
Day 0: $0
Day 1: $0
Day 2: $0
```

Hypothesis Testing

Null Hypothesis (H₀): There is no significant difference in conversion rates between Facebook and AdWords.

Alternative Hypothesis (H₁): There is a significant difference in conversion rates between Facebook and AdWords.

Calculate Conversion Rates: For each platform, calculate the conversion rate as the total conversions divided by total clicks.

Formulate Hypotheses: Null Hypothesis (H₀): There is no difference in conversion rates between Facebook and AdWords.

Alternative Hypothesis (H₁): There is a significant difference in conversion rates between Facebook and AdWords.

Perform a Two-Sample Proportion Z-Test: Use a two-sample proportion z-test to determine if the observed difference is statistically significant.

```
# Calculate conversion rates for each platform
platform_data = data.groupby('platform').agg{
    'clicks': 'sum',
    'approved_conversions': 'sum'
}
platform_data['conversion_rate'] = (platform_data['approved_conversions'] / platform_data['clicks'] * 100).round(2)

print("Platform conversion rates:")
print(platform_data)

# Prepare data for testing
facebook_clicks = data[data['platform'] == 'Facebook']['clicks'].sum()
facebook_conversions = data[data['platform'] == 'Facebook']['approved_conversions'].sum()
adwords_clicks = data[data['platform'] == 'Google']['clicks'].sum()
adwords_conversions = data[data['platform'] == 'Google']['approved_conversions'].sum()

print("Facebook: {facebook_conversions}/{facebook_clicks}")
print("AdWords: {adwords_conversions}/{adwords_clicks}")

# Calculate conversion rates
facebook_rate = facebook_conversions / facebook_clicks
adwords_rate = adwords_conversions / adwords_clicks

print("Facebook rate: {facebook_rate:.4f}")
print("AdWords rate: {adwords_rate:.4f}")

# Perform proportion z-test
from statsmodels.stats.proportion import proportion_ztest
count = [facebook_conversions, adwords_conversions]
nobs = [facebook_clicks, adwords_clicks]
z_stat, p_value = proportion_ztest(count, nobs)

print("Z-statistic: {z_stat:.4f}")
print("P-value: {p_value:.4f}")
```

```
Platform conversion rates:
          clicks  approved_conversions  conversion_rate
platform
Facebook    7516                   691             9.2
Google      8773                   490             5.6
Total       1372                   115             8.4
Facebook: {facebook_conversions}/{facebook_clicks}
AdWords: {adwords_conversions}/{adwords_clicks}
Facebook rate: {facebook_rate:.4f}
AdWords rate: {adwords_rate:.4f}
Z-statistic: {z_stat:.4f}
P-value: {p_value:.4f}
```

Interpretation:

With a very low p-value (close to 0), we can reject the null hypothesis, suggesting a statistically significant difference in conversion rates between Facebook and AdWords. Facebook appears to have a significantly higher conversion rate compared to AdWords.

Now let's examine if ad click-through rates (CTR) to determine if there's a statistically significant difference in the click performance between Facebook and AdWords.

Set up Hypotheses:

Null Hypothesis (H₀): There is no difference in click-through rates between Facebook and AdWords.

Alternative Hypothesis (H₁): There is a significant difference in click-through rates between Facebook and AdWords.

Perform a Two-Sample Proportion Z-Test: Use a two-sample proportion z-test to determine if the observed difference in CTRs is statistically significant.

```
# Calculate click-through rates for each platform
platform_ctr = data.groupby('platform').agg{
    'clicks': 'sum',
    'impressions': 'sum'
}
platform_ctr['ctr'] = (platform_ctr['clicks'] / platform_ctr['impressions'] * 100).round(2)

print("Platform CTR:")
print(platform_ctr)

# Prepare data for CTR testing
facebook_clicks = data[data['platform'] == 'Facebook']['clicks'].sum()
facebook_impressions = data[data['platform'] == 'Facebook']['impressions'].sum()
adwords_clicks = data[data['platform'] == 'Google']['clicks'].sum()
adwords_impressions = data[data['platform'] == 'Google']['impressions'].sum()

print("Facebook: {facebook_clicks}/{facebook_impressions}")
print("AdWords: {adwords_clicks}/{adwords_impressions}")

# Calculate CTR rates
facebook_ctr = facebook_clicks / facebook_impressions
adwords_ctr = adwords_clicks / adwords_impressions

print("Facebook CTR: {facebook_ctr:.4f}")
print("AdWords CTR: {adwords_ctr:.4f}")

# Perform proportion z-test for CTR
count = [facebook_clicks, adwords_clicks]
nobs = [facebook_impressions, adwords_impressions]
z_stat, p_value = proportion_ztest(count, nobs)

print("Z-statistic: {z_stat:.4f}")
print("P-value: {p_value:.4f}")
```

```
Platform CTR:
          clicks  impressions   ctr
platform
Facebook    7516        93390  0.75
Google      8773        44548  0.79
Total       1372        21544  0.80
Facebook: {facebook_clicks}/{facebook_impressions}
AdWords: {adwords_clicks}/{adwords_impressions}
Facebook CTR: {facebook_ctr:.4f}
AdWords CTR: {adwords_ctr:.4f}
Z-statistic: {z_stat:.4f}
P-value: {p_value:.4f}
```

Interpretation

The very low p-value (close to 0) suggests a statistically significant difference in CTR between Facebook and AdWords, with Facebook achieving a higher click-through rate than AdWords.

Regression Analysis

What will happen when I do go with the Facebook Ad? How many Facebook ad conversions can I expect given a certain number of Facebook ad clicks?

```
# Prepare Facebook data
fb_data = data[data['platform'] == 'Facebook']

# Simple regression
X = fb_data['clicks']
y = fb_data['approved_conversions']

# Add constant for intercept
import statsmodels.api as sm
X = sm.add_constant(X)

# Fit regression
model = sm.OLS(y, X).fit()

print("Facebook Ad Conversion Model:")
print(model.summary())

# Predicted vs actual
fb_data['predicted_conversions'] = model.predict(X)

# Plot relationship
plt.figure(figure_size=(10, 6))
plt.scatter(fb_data['clicks'], fb_data['approved_conversions'], alpha=0.7, label='Actual')
plt.plot(fb_data['clicks'], fb_data['predicted_conversions'], color='red', label='Predicted')
plt.xlabel('Facebook Clicks')
plt.ylabel('Approved Conversions')
plt.title('Facebook Ad Conversion Prediction')
plt.legend()
plt.show()
```

```
Facebook Ad Conversion Model:
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       1.000
Model:                            OLS   Adj. R-squared:                  1.000
Method:                 Least Squares   F-statistic:                     3.000
Date:                Wed, 09 Jul 2025   Prob (F-statistic):              0.083
Time:                        12:34:56   Log-Likelihood:                -1.0000
No. Observations:                   4   AIC:                             6.000
Df Residuals:                       2   BIC:                             4.000
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0000      0.000        inf      0.000      0.000      0.000
clicks         1.0000      0.000        inf      0.000      1.000      1.000
==============================================================================
Omnibus:                        0.000   Durbin-Watson:                   2.000
Prob(Omnibus):                  1.000   Jarque-Bera (JB):                0.000
Skew:                           0.000   Prob(JB):                        1.000
Kurtosis:                       3.000   Cond. No.                         1.00
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```



```
# Time series analysis
fb_daily = fb_data.groupby('date').{
    'clicks': 'sum',
    'approved_conversions': 'sum'
}

# Lag analysis
fb_daily['clicks_lag1'] = fb_daily['clicks'].shift(1)
fb_daily['clicks_lag2'] = fb_daily['clicks'].shift(2)

# Correlation with lags
print("Correlation with lagged clicks:")
print(fb_daily.corr()['approved_conversions'])
```

```
Correlation with lagged clicks:
approved_conversions    1.000000
clicks                  1.000000
clicks_lag1             1.000000
clicks_lag2             1.000000
Name: approved_conversions, dtype: float64
```

```
# Optimal spend calculation
cost_per_conversion = fb_data['spent'] / fb_data['approved_conversions']
print("Average cost per conversion:", cost_per_conversion.mean())

# Budget scenarios
budget_scenarios = [1000, 2000, 5000, 10000]
for budget in budget_scenarios:
    expected_conversions = budget / cost_per_conversion.mean()
    print("Budget {budget}: {expected_conversions:.0f} conversions")
```

```
Average cost per conversion: 0.0
Budget 1000: inf conversions
Budget 2000: inf conversions
Budget 5000: inf conversions
Budget 10000: inf conversions
```

```
# Save Facebook analysis
fb_data.to_csv('facebook_analysis.csv')

print("Facebook analysis saved!")
```

```
Facebook analysis saved!
```

Now let's analyze the Cost Per Conversion (CPC) trend over time:

```
# Calculate daily CPC
fb_data['cpc'] = fb_data['spent'] / fb_data['approved_conversions']

# Plot CPC trend
plt.figure(figure_size=(12, 6))
plt.plot(fb_data['date'], fb_data['cpc'])
plt.title('Facebook Ad Cost Per Conversion Trend')
plt.xlabel('Date')
plt.ylabel('CPC')
plt.xticks(rotation=45)
plt.show()
```



```
# Weekly conversion patterns
fb_data['day_of_week'] = fb_data['date'].dt.dayofweek
weekly_conversions = fb_data.groupby('day_of_week')['approved_conversions'].mean()

plt.figure(figure_size=(10, 6))
plt.bar(weekly_conversions.index, weekly_conversions.values)
plt.title('Average Conversions by Day of Week')
plt.xlabel('Day of Week')
plt.ylabel('Average Conversions')
plt.show()
```



```
# Monthly conversion patterns
fb_data['month'] = fb_data['date'].dt.month
monthly_conversions = fb_data.groupby('month')['approved_conversions'].mean()

plt.figure(figure_size=(10, 6))
plt.bar(monthly_conversions.index, monthly_conversions.values)
plt.title('Average Conversions by Month')
plt.xlabel('Month')
plt.ylabel('Average Conversions')
plt.show()
```



```
# Cost efficiency by day of week
daily_cpc = fb_data.groupby('day_of_week')['cpc'].mean()

plt.figure(figure_size=(10, 6))
plt.bar(daily_cpc.index, daily_cpc.values)
plt.title('Average CPC by Day of Week')
plt.xlabel('Day of Week')
plt.ylabel('Average CPC')
plt.show()
```



```
# Budget recommendations based on patterns
best_days = weekly_conversions.nlargest(3)
print("Best days for conversions:")
print(best_days)

# Calculate optimal budget allocation
total_budget = 10000
for day in best_days.index:
    allocation = (weekly_conversions[day] / best_days.sum() * total_budget)
    print("Day {day}: ${allocation:.0f}")
```

```
Best days for conversions:
day_of_week
0    0.0
1    0.0
2    0.0
Name: approved_conversions, dtype: float64
Day 0: $0
Day 1: $0
Day 2: $0
```

Hypothesis Testing

To test for significant differences in conversion rates between Facebook and AdWords, we can use a two-sample t-test or a proportion test. Here's the approach:

Steps:

Calculate Conversion Rates: For each platform, calculate the conversion rate as the total conversions divided by total clicks.

Formulate Hypotheses: Null Hypothesis (H₀): There is no difference in conversion rates between Facebook and AdWords.

Alternative Hypothesis (H₁): There is a significant difference in conversion rates between Facebook and AdWords.

Run the Statistical Test: Use a two-sample proportion z-test to determine if the observed difference is statistically significant.

```
# Calculate conversion rates for each platform
platform_data = data.groupby('platform').agg{
    'clicks': 'sum',
    'approved_conversions': 'sum'
}
platform_data['conversion_rate'] = (platform_data['approved_conversions'] / platform_data['clicks'] * 100).round(2)

print("Platform conversion rates:")
print(platform_data)
```

```
Platform conversion rates:
          clicks  approved_conversions  conversion_rate
platform
Facebook    7516                   691             9.2
Google      8773                   490             5.6
Total       1372                   115             8.4
```

```
# Hypothesis testing for conversion rate difference
from scipy.stats import ttest_ind, proportion_ztest
import numpy as np

# Prepare data for testing
facebook_clicks = data[data['platform'] == 'Facebook']['clicks'].sum()
facebook_conversions = data[data['platform'] == 'Facebook']['approved_conversions'].sum()
adwords_clicks = data[data['platform'] == 'Google']['clicks'].sum()
adwords_conversions = data[data['platform'] == 'Google']['approved_conversions'].sum()

print("Facebook: {facebook_conversions}/{facebook_clicks}")
print("AdWords: {adwords_conversions}/{adwords_clicks}")

# Calculate conversion rates
facebook_rate = facebook_conversions / facebook_clicks
adwords_rate = adwords_conversions / adwords_clicks

print("Facebook rate: {facebook_rate:.4f}")
print("AdWords rate: {adwords_
```

```
df.info(),df.describe(include='object'),df.isnull()
```