```
TASK 1
"""

import numpy as np
import pandas as pd
from scipy.stats import ttest_ind

#H0 : There is no significant difference between the homepage versions
#H1 : There is a significant difference between the homepage versions

np.random.seed(42)

# Generate data_a and data_b with different probabilities
data_a = np.random.choice([0, 1], size=100, p=[0.8, 0.2])
data_b = np.random.choice([0, 1], size=100, p=[0.75, 0.25])

# Generate ctr_a and ctr_b arrays with uniform random values between 0.05
and 0.15
ctr_a = np.random.uniform(0.05, 0.15, size=100)
ctr_b = np.random.uniform(0.05, 0.15, size=100)

# Generate avg_order_value_a and avg_order_value_b arrays with normal
random values
avg_order_value_a = np.random.normal(50, 10, size=100)
avg_order_value_b = np.random.normal(55, 10, size=100)

# Create a DataFrame to hold the data
df = pd.DataFrame({'Homepage A': data_a, 'Homepage B': data_b,
                   'CTR A': ctr_a, 'CTR B': ctr_b,
                   'Avg Order Value A': avg_order_value_a, 'Avg Order
Value B': avg_order_value_b})

# Calculate metrics
conversion_rate_a = df['Homepage A'].mean()
conversion_rate_b = df['Homepage B'].mean()
ctr_a_mean = df['CTR A'].mean()
ctr_b_mean = df['CTR B'].mean()
avg_order_value_a_mean = df['Avg Order Value A'].mean()
avg_order_value_b_mean = df['Avg Order Value B'].mean()
```

```
# Perform statistical tests
t_statistic, p_value = ttest_ind(df['Homepage A'], df['Homepage B'])

# Print the results
print("Conversion Rate A:", conversion_rate_a)
print("Conversion Rate B:", conversion_rate_b)
print("CTR A Mean:", ctr_a_mean)
print("CTR B Mean:", ctr_b_mean)
print("Avg Order Value A Mean:", avg_order_value_a_mean)
print("Avg Order Value B Mean:", avg_order_value_b_mean)
print("T-statistic:", t_statistic)
print("P-value:", p_value)

# Check for statistical significance
if p_value < 0.05:
    print("There is a significant difference between the homepage
versions.")
else:
    print("There is no significant difference between the homepage
versions.")

"""Report

Conclusion:
 There is a statistically significant difference' if p_value < 0.05 else
no statistically significant difference between the homepage versions.

Recommendation:
- If Homepage B has a higher conversion rate, click-through rate, and
average order value, implement it as the new default homepage.
- If Homepage A has a higher conversion rate, click-through rate, and
average order value, consider further iterations or testing with different
variations.
```

Task 2:

```
TASK 2
"""
!pip install pytesseract
import requests
from bs4 import BeautifulSoup
import re
```

```
import pytesseract
from PIL import Image
pytesseract.pytesseract.tesseract_cmd = r'/usr/bin/tesseract'

def scrape_email_addresses(domain_list):
    for domain in domain_list:
        print(f"Scraping {domain}...")
        try:
            # Fetch the homepage content
            response = requests.get(f"http://{domain}")
            if response.status_code != 200:
                print(f"Failed to fetch {domain}. Status code:
{response.status_code}")
                continue

            soup = BeautifulSoup(response.content, 'html.parser')

            # Find and follow the "Impressum" link
            impressum_link = None
            for link in soup.find_all('a', href=True):
                if 'impressum' in link['href'].lower():
                    impressum_link = link['href']
                    break

            if impressum_link:
                # Fetch the "Impressum" page content
                impressum_response = requests.get(impressum_link)
                if impressum_response.status_code != 200:
                    print(f"Failed to fetch Impressum page for {domain}.
Status code: {impressum_response.status_code}")
                    continue

                impressum_soup = BeautifulSoup(impressum_response.content,
'html.parser')

                # Extract email addresses from the "Impressum" page
                email_addresses = []
                email_pattern =
re.compile(r'[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}')
                for paragraph in impressum_soup.find_all('p'):
```

```
                    text = paragraph.get_text()
                    email_matches = email_pattern.findall(text)
                    email_addresses.extend(email_matches)

                # Extract email addresses from images using pytesseract
                for img_tag in
impressum_soup.find_all('/content/task2.png'):
                    img_url = img_tag.get('src')
                    if img_url.lower().endswith(('.jpg', '.jpeg',
'.png')):
                        image_response = requests.get(img_url,
stream=True)
                        if image_response.status_code == 200:
                            image = Image.open(image_response.raw)
                            email_from_image =
pytesseract.image_to_string(image)
                            email_matches =
email_pattern.findall(email_from_image)
                            email_addresses.extend(email_matches)

                # Remove duplicates from the list of email addresses
                email_addresses = list(set(email_addresses))

                # Print the extracted email addresses
                if email_addresses:
                    print(f"Email addresses found for {domain}:")
                    for email in email_addresses:
                        print(email)
                else:
                    print(f"No email addresses found for {domain}.")
            else:
                print(f"No 'Impressum' link found for {domain}.")

        except Exception as e:
            print(f"Error occurred while scraping {domain}: {e}")

if __name__ == "__main__":
    domains = ["peersociallending.com",
"kreditvergleich-kostenlos.net",
"matblog.de",
```

```
"malta-tours.de",
"wiseclerk.com",
"urlaub-in-thailand.com",
"findle.top",
"niederrheinzeitung.de",
"finanziell-umdenken.blogspot.com",
"midbio.org",
"klaudija.de",
"pc-welt.wiki",
"websitevalue.co.uk",
"freizeitcafe.info",
"ladenbau.de",
"bierspot.de",
"biboxs.com",
"finance-it-blog.de",
"guenstigerkreditvergleich.com",
"cloudbiz.one",
"frag-den-heimwerker.com",
"fintech-intel.com",
"selbst-schuld.com",
"eltemkredit.com",
"binoro.de",
"siteurl.org",
"frachiseportal.at",
"finlord.cz",
"vj-coach.de",
"mountainstatescfc.org",
"crowdstreet.de"]

    scrape_email_addresses(domains)

"""**DOMAIN NAME:** wiseclerk.com

emails found :


1.   info@p2p-kredite.com


**DOMAIN NAME:** freizeitcafe.info
```

```
emails found :


1.   christiangeradigital@gmail.com



**DOMAIN NAME:** ladenbau.de

emails found :


1.  info@ladenbau.de

2.   beratung@ladenbau.de

**DOMAIN NAME:** frag-den-heimwerker.com

emails found :


1.   info@frag-den-heimwerker.com
```

Task 3:

```
TASK 3
"""

import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt

data = pd.read_csv("/content/retail_services.csv")

data.info()

data.describe()

missing_values_df = data.isnull()
```

```
# Check if there are any columns with all True values (indicating all
elements are missing)
columns_with_all_missing = missing_values_df.all(axis=0)

# Get the column names where all elements are missing
cols_with_all_missing_values =
columns_with_all_missing[columns_with_all_missing].index

print(cols_with_all_missing_values)

#How has retail economic activity in the United States changed over the
past five years?

#getting all the retail columns
retail= [col for col in data.columns if
col.startswith('data.sales.retail')]
print(retail)

#finding max of time year
max = data["time.year"].max()
print(max)

#getting data for last 5 year
past_5_years = data[['time.year'] + retail].copy().query("`time.year` >=
2012")
past_5_years.head(10)

year = past_5_years.groupby('time.year')[retail].sum()

#visualising
fig, ax = plt.subplots(figsize=(12, 6))

# Plot lines for all four categories
sb.lineplot(data=past_5_years, x='time.year', y='data.sales.retail trade',
label='Retail Trade', ax=ax, color='red')
sb.lineplot(data=past_5_years, x='time.year', y='data.sales.retail trade
and food services', label='Retail Trade and Food Services', ax=ax,
color='blue')
```

```
sb.lineplot(data=past_5_years, x='time.year', y='data.sales.retail trade
and food services, ex auto', label='Retail Trade and Food Services, ex
Auto', ax=ax, color='green')
sb.lineplot(data=past_5_years, x='time.year', y='data.sales.retail trade,
ex auto', label='Retail Trade, ex Auto', ax=ax, color='orange')

# Set title and labels
plt.title('Retail Economic Activity in the United States Over the Past
Five Years')
plt.xlabel('Year')
plt.ylabel('Sales')
plt.legend()

#What are the key differences between the Advance Monthly Retail Trade
Survey (MARTS) and the Annual Retail Trade Survey (ARTS)?

#marts : The Advance Monthly Retail Trade Survey is conducted on a monthly
basis.
#arts  : The Annual Retail Trade Survey is conducted annually.

#marts : It covers a sample of retail and food service firms across
various industries.
#arts  : It aims for complete coverage of all retail establishments in the
United States.

#mart : The primary purpose is to provide timely data on current retail
sales trends.
#art  : It serves as a more comprehensive and detailed source of data for
the entire retail sector

#Can we identify any seasonal patterns or trends in monthly retail sales
data?
month_retail = pd.DataFrame(data.groupby('time.month name')[retail].sum())
month_retail = month_retail.reset_index()
month_retail
#visualise
plt.figure(figsize=[18, 6])
sb.lineplot(data=month_retail.sum(axis=1), color='red', marker='o',
linewidth=2)
```

```
plt.title('Retail Sale Monthly Trend', fontdict={'fontname': 'Monospace',
'fontsize': 20, 'fontweight': 'bold'})
plt.xlabel('Months', fontdict={'fontname': 'Monospace', 'fontsize': 15})
plt.ylabel('Sales', fontdict={'fontname': 'Monospace', 'fontsize': 15})
plt.show()

#How does e-commerce activity compare to traditional retail sales on a
quarterly basis?

#Are there any specific retail sectors that have shown significant growth
or decline in recent years?

#What is the overall contribution of retail trade to the United States'
GDP?
plt.figure(figsize=[18, 6])
plt.bar(data['time.year'], data['data.sales.retail trade'],
color='skyblue')
plt.title('Contribution of Retail Trade to the US GDP',
fontdict={'fontname': 'Monospace', 'fontsize': 20, 'fontweight': 'bold'})
plt.xlabel('Year', fontdict={'fontname': 'Monospace', 'fontsize': 15})
plt.ylabel('Contribution (Million USD)', fontdict={'fontname':
'Monospace', 'fontsize': 15})
plt.show()

#How do retail operating expenses vary across different types of
businesses?

#Can we identify any correlations between retail sales and macroeconomic
indicators such as unemployment rates or consumer sentiment?

#What are the main factors influencing fluctuations in monthly retail
sales data?
sb.heatmap(data=data.corr().round(2))
plt.show()

#Can we predict future retail sales based on historical data and other
relevant factors?
#Yes, it is possible to predict future retail sales based on historical
data and other
```

```
#relevant factors using various predictive modeling techniques.Predictive
modeling can
#provide valuable insights for decision-making in retail businesses,
allowing them to
#plan inventory, staffing, marketing strategies, and other operations
effectively.

#Is there a relationship between e-commerce sales and brick-and-mortar
retail sales?

#How accurate are the monthly estimates compared to the annual survey
data?

#What are the most significant challenges in reconciling the monthly and
annual data for retail economic activity?
#Reconciling monthly and annual retail data is challenging due to varying
reporting frequencies,
# seasonal patterns, and differences in data sources and definitions.
#Careful validation and data processing are required for accurate
comparisons.

#Can we identify any discrepancies between the retail data collected by
MARTS, MRTS, ARTS, and the Economic Census of Retail Trade?

#How do the retail sales patterns differ between rural and urban areas?
#dataset dont have data for this

#Are there any geographical variations in retail sales trends across
different states or regions?
#dataset dont have data for this

#Are there any geographical variations in retail sales trends across
different states or regions?
#dataset dont have data for this

#Can we identify any outliers or anomalies in the retail sales data that
require further investigation?
def count_outliers_iqr(column):
    # Convert column to numeric data type
    column = pd.to_numeric(column, errors='coerce')
```

```
    Q1 = column.quantile(0.25)
    Q3 = column.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers = (column < lower_bound) | (column > upper_bound)
    return outliers.sum()

# Handle missing values if necessary
data_filled = data.fillna(0)

# Calculate the number of outliers for each column
outliers_count = data_filled.apply(count_outliers_iqr)

print(outliers_count)

#How have retail operating expenses changed over the past decade?
#dataset dont have data for this

#What insights can we gain from comparing the detailed business operating
expenses collected in the Business Expenses Supplement with other retail
data sources?
```