E-commerce Sales Analysis and Insights

April 20, 2025

0.1 Dataset Overview

This dataset is pulled **directly from Google BigQuery** using a custom SQL query.
It combines session-level user activity with product, account and traffic metadata.

Key Information: - **Source**: Google BigQuery (tables: `session`, `session_params`, `account_session`, `account`, `order`, `product`) - **Extraction**: Python + `google-cloud-bigquery` client - **Period covered**: November 1, 2020 – January 27, 2021 - **Rows × Columns**: each row = one purchased product in a session; ~33 k × 18 columns - **Goal**: analyze sales trends across regions, devices, channels and user types

0.2 Data Preprocessing

- Date formatting
- Missing value analysis
- Basic column classification (numeric, categorical, datetime)

[1]:
```
# Install connector if needed
!pip install --upgrade pandas-gbq --quiet

# Imports
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from google.cloud import bigquery
from statsmodels.tsa.seasonal import seasonal_decompose
from scipy.stats import pearsonr, mannwhitneyu, kruskal
import plotly.graph_objects as go

# Create BigQuery client
client = bigquery.Client(project="data-analytics-mate")

# SQL query
query = """
SELECT
  s.date,
  s.ga_session_id,
  sp.continent, sp.country, sp.device, sp.browser,
  sp.mobile_model_name, sp.operating_system, sp.language,
  sp.name AS model_name,
  sp.channel,
  acs.account_id, a.is_verified, a.is_unsubscribed,
  p.category, p.name AS product_name, p.price, p.short_description
FROM `DA.session` AS s
JOIN `DA.session_params` AS sp ON s.ga_session_id = sp.ga_session_id
LEFT JOIN `DA.account_session`  AS acs ON s.ga_session_id = acs.ga_session_id
LEFT JOIN `DA.account`          AS a   ON acs.account_id     = a.id
JOIN `DA.order`                AS o   ON s.ga_session_id     = o.ga_session_id
JOIN `DA.product`              AS p   ON o.item_id           = p.item_id
"""

# Load into DataFrame
df = client.query(query).to_dataframe()

# Preview
display(df.head())

# High-level facts
print(f"Dataset shape: {df.shape[0]} rows × {df.shape[1]} columns")
df['date'] = pd.to_datetime(df['date'], errors='coerce')
print(f"Date range: {df['date'].min().date()} → {df['date'].max().date()}")
```

```
C:\Users\Alex\AppData\Local\Programs\Python\Python313\Lib\site-
packages\google\auth\_default.py:76: UserWarning: Your application has
authenticated using end user credentials from Google Cloud SDK without a quota
project. You might receive a "quota exceeded" or "API not enabled" error. See
the following page for troubleshooting:
https://cloud.google.com/docs/authentication/adc-troubleshooting/user-creds.
  warnings.warn(_CLOUD_SDK_CREDENTIALS_WARNING)
```

```
        date  ga_session_id continent        country  device browser  \
0 2020-11-01     6260592731  Americas  United States  mobile  Chrome
1 2020-11-02     9287517789      Asia          India  mobile  Chrome
2 2020-11-02     8202854533  Americas  United States  mobile  Chrome
3 2020-11-02     2401617513  Americas  United States  mobile  Chrome
4 2020-11-03     6315247329  Americas  United States  mobile  Chrome

  mobile_model_name operating_system language  model_name         channel  \
0           Pixel 3              Web     None   (organic)  Organic Search
1           Pixel 3              Web     None    (direct)          Direct
2           Pixel 3              Web    en-us   (organic)  Organic Search
3        Pixel 4 XL          Android     None  (referral)     Paid Search
4        Pixel 4 XL              Web    en-us   (organic)  Organic Search

   account_id  is_verified  is_unsubscribed                    category  \
0        <NA>         <NA>             <NA>  Bookcases & shelving units
```

```
1        <NA>         <NA>             <NA>                Chairs
2        <NA>         <NA>             <NA>        Sofas & armchairs
3        <NA>         <NA>             <NA>        Nursery furniture
4        <NA>         <NA>             <NA>                  Beds

            product_name   price  \
0                VITTSJÖ   609.0
1  NORDVIKEN / HENRIKSDAL  2675.0
2                LIDHULT  5810.0
3                SMÅGÖRA    95.0
4                 HEMNES   995.0

                             short_description
0  Shelving unit with laptop table, 202x36x175 cm
1             Table and 4 chairs, 152/223x95 cm
2                          Corner sofa, 5-seat
3                         Shelf unit, 29x88 cm
4                         Bed frame, 90x200 cm

Dataset shape: 33538 rows × 18 columns
Date range: 2020-11-01 → 2021-01-27
```

[2]:
```python
# Convert 'date' if not already
df['date'] = pd.to_datetime(df['date'], errors='coerce')

# Column type summary
print("Column data types:\n", df.dtypes, "\n")

# Classify columns
numeric_cols     = df.select_dtypes(include='number').columns.tolist()
categorical_cols = df.select_dtypes(include='object').columns.tolist()
datetime_cols    = df.select_dtypes(include='datetime').columns.tolist()

print(f"Numeric columns ({len(numeric_cols)}):     {numeric_cols}")
print(f"Categorical columns ({len(categorical_cols)}): {categorical_cols}")
print(f"Datetime columns ({len(datetime_cols)}):     {datetime_cols}\n")

# Missing values: absolute & percent
missing_counts   = df.isnull().sum()
missing_percents = (df.isnull().mean() * 100).round(2)

missing_df = pd.DataFrame({
    'Missing': missing_counts[missing_counts>0],
    '% of total': missing_percents[missing_percents>0]
}).sort_values('Missing', ascending=False)

print("Columns with missing data:\n", missing_df)
```

```
Column data types:
 date                    datetime64[ns]
ga_session_id                    Int64
continent                       object
country                         object
device                          object
browser                         object
mobile_model_name               object
operating_system                object
language                        object
model_name                      object
channel                         object
account_id                       Int64
is_verified                      Int64
is_unsubscribed                  Int64
category                        object
product_name                    object
price                          float64
short_description               object
dtype: object

Numeric columns (5):     ['ga_session_id', 'account_id', 'is_verified',
'is_unsubscribed', 'price']
Categorical columns (12): ['continent', 'country', 'device', 'browser',
'mobile_model_name', 'operating_system', 'language', 'model_name', 'channel',
'category', 'product_name', 'short_description']
Datetime columns (1):     ['date']

Columns with missing data:
                  Missing  % of total
account_id          30757       91.71
is_verified         30757       91.71
is_unsubscribed     30757       91.71
language            11007       32.82
```

0.3 Geographic Sales Analysis

- Sales by continent and country
- Number of sessions by region
- Top countries and continents (interactive bar charts)

[4]:
```python
# Aggregate sales by continent and country
sales_by_continent = df.groupby('continent')['price'].sum().
 ↪sort_values(ascending=False)
sales_by_country = df.groupby('country')['price'].sum().
 ↪sort_values(ascending=False)

# Top 3 continents and top 5 countries with highest sales
top_3_continents = sales_by_continent.head(3)
```

```
top_5_countries = sales_by_country.head(5)

# Display the results
print("Top 3 continents by sales:\n", top_3_continents)
print("\nTop 5 countries by sales:\n", top_5_countries)

# Visualize sales by continent using Plotly
fig = px.bar(sales_by_continent.reset_index(), x='continent', y='price',
             title='Total Sales by Continent', labels={'price': 'Total Sales␣
 ↪($)', 'continent': 'Continent'},
             color='continent', template='plotly_white')
fig.update_layout(autosize=True, height=600, width=1050)
fig.show()

# Visualize top countries by sales using Plotly
fig = px.bar(sales_by_country.head(10).reset_index(), x='country', y='price',
             title='Top 10 Countries by Sales', labels={'price': 'Total Sales␣
 ↪($)', 'country': 'Country'},
             color='country', template='plotly_white')
fig.update_layout(autosize=True, height=600, width=1050)
fig.show()
```

```
Top 3 continents by sales:
 continent
Americas    17665280.0
Asia         7601298.3
Europe       5934624.2
Name: price, dtype: float64

Top 5 countries by sales:
 country
United States     13943553.9
India              2809762.0
Canada             2437921.0
United Kingdom      938317.9
France              710692.8
Name: price, dtype: float64
```

0.4 Product Insights

- Top categories by sales
- Category breakdown by top-selling country
- Interactive bar charts

[5]:
```
# Aggregate sales by product category
sales_by_category = df.groupby('category')['price'].sum().
 ↪sort_values(ascending=False)
```

```
# Top 10 product categories by total sales
top_10_categories = sales_by_category.head(10)

# Display results
print("Top 10 product categories by total sales:\n", top_10_categories)

# Build an interactive bar chart for the top 10 categories using Plotly
fig = px.bar(top_10_categories.reset_index(), x='category', y='price',
             title='Top 10 Product Categories by Total Sales',
             labels={'price': 'Total Sales ($)', 'category': 'Product␣
 ↪Category'},
             color='category', template='plotly_white')
fig.show()

# Now let's break down sales by the top-selling country
top_country = df.groupby('country')['price'].sum().idxmax()  # Identify the␣
 ↪top-selling country
df_top_country = df[df['country'] == top_country]

# Aggregate sales by category for the top-selling country
sales_by_category_top_country = df_top_country.groupby('category')['price'].
 ↪sum().sort_values(ascending=False)
top_10_categories_top_country = sales_by_category_top_country.head(10)

# Display results
print(f"\nTop 10 product categories in {top_country} by total sales:\n",␣
 ↪top_10_categories_top_country)

# Build an interactive bar chart for the top 10 categories in the top-selling␣
 ↪country using Plotly
fig = px.bar(top_10_categories_top_country.reset_index(), x='category',␣
 ↪y='price',
             title=f'Top 10 Product Categories in {top_country} by Total Sales',
             labels={'price': 'Total Sales ($)', 'category': 'Product␣
 ↪Category'},
             color='category', template='plotly_white')
fig.show()
```

```
Top 10 product categories by total sales:
 category
Sofas & armchairs                   8388254.5
Chairs                              6147748.8
Beds                                4919725.0
Bookcases & shelving units          3640818.1
Cabinets & cupboards                2336499.5
Outdoor furniture                   2142222.2
Tables & desks                      1790307.5
```

```
Chests of drawers & drawer units     906562.5
Bar furniture                        735503.0
Children's furniture                 467697.0
Name: price, dtype: float64


Top 10 product categories in United States by total sales:
 category
Sofas & armchairs                   3707144.5
Chairs                              2619773.8
Beds                                2213058.0
Bookcases & shelving units          1567606.9
Cabinets & cupboards                 994545.5
Outdoor furniture                    929245.2
Tables & desks                       777865.0
Chests of drawers & drawer units     382388.0
Bar furniture                        330805.0
Children's furniture                 207575.0
Name: price, dtype: float64
```

0.5 Device and Technology Insights

- Sales by device type and model
- Sales by browser
- Heatmaps: device vs traffic channel

[11]:
```
# Aggregate sales by device type
sales_by_device = df.groupby('device')['price'].sum().
 ↪sort_values(ascending=False)

# Display sales by device type
print("Sales by Device Type:\n", sales_by_device)

# Plot sales by device type
fig = px.bar(sales_by_device.reset_index(), x='device', y='price',
             title='Sales by Device Type', labels={'price': 'Total Sales ($)',␣
 ↪'device': 'Device Type'},
             color='device', template='plotly_white')
fig.update_layout(autosize=True, height=600)  # Set to auto width and fixed␣
 ↪height
fig.show()

# Aggregate sales by mobile model
sales_by_mobile_model = df.groupby('mobile_model_name')['price'].sum().
 ↪sort_values(ascending=False)

# Display sales by mobile model
print("Sales by Mobile Model:\n", sales_by_mobile_model)
```

```python
# Plot sales by mobile model
fig = px.bar(sales_by_mobile_model.reset_index(), x='mobile_model_name',␣
 ↪y='price',
             title='Sales by Mobile Model', labels={'price': 'Total Sales ($)',␣
 ↪'mobile_model_name': 'Mobile Model'},
             color='mobile_model_name', template='plotly_white')
fig.update_layout(autosize=True, height=600)  # Set to auto width and fixed␣
 ↪height
fig.show()

# Create a pivot table for device type vs traffic channel
pivot_table = pd.pivot_table(df, index='channel', columns='device',␣
 ↪values='ga_session_id', aggfunc='nunique', fill_value=0)

# Reset the pivot table to long format for Plotly
pivot_long = pivot_table.reset_index().melt(id_vars='channel',␣
 ↪var_name='device', value_name='sessions')

# Create an interactive heatmap to visualize device vs channel
fig = px.density_heatmap(pivot_long, x='device', y='channel', z='sessions',
                         title='Number of Sessions by Traffic Channel and␣
 ↪Device Type',
                         labels={'sessions': 'Number of Sessions'},␣
 ↪color_continuous_scale='YlGnBu', text_auto=True)
fig.update_layout(xaxis_title='Device Type', yaxis_title='Traffic Channel',␣
 ↪hovermode='closest', height=600)
fig.show()
```

```
Sales by Device Type:
 device
desktop    18864039.0
mobile     12384225.8
tablet       723466.3
Name: price, dtype: float64
```

```
Sales by Mobile Model:
 mobile_model_name
Chrome        8899523.9
<Other>       6535330.8
Safari        6491062.1
iPhone        6420776.3
ChromeBook    1830458.7
Edge           697222.3
iPad           448854.2
Firefox        421066.9
Pixel 4 XL     118287.7
Pixel 3        109148.2
```

```
Name: price, dtype: float64
```

0.6 Traffic Source Analysis

- Sales share by traffic channel
- Channel distribution: pie chart and heatmap
- Sessions per channel (with Kruskal-Wallis Test)

[7]:
```
# Aggregate sales by traffic channel
sales_by_channel = df.groupby('channel')['price'].sum()

# Calculate the percentage of total sales for each traffic channel
sales_by_channel_percentage = (sales_by_channel / sales_by_channel.sum()) * 100

# Display results
print("Sales by traffic channel (as percentage of total sales):\n",␣
 ↪sales_by_channel_percentage)

# Create a pie chart to visualize the sales distribution across channels
fig = px.pie(
    sales_by_channel_percentage.reset_index(),
    names='channel',
    values='price',
    title='Sales Distribution by Traffic Channel (%)',
    template='plotly_white',
    hole=0.3  # Make it a donut chart
)
fig.update_traces(textinfo='percent+label', hoverinfo='label+percent')
fig.show()

# Create a heatmap for sessions by traffic channel and device type
# Filter out rows where 'channel' or 'device' is missing (e.g., '(not set)' or␣
 ↪NaN)
filtered_df = df[(df['channel'] != '(not set)') & (df['device'] != '(not set)')]

# Create a pivot table to count sessions by traffic channel and device type
pivot_table = pd.pivot_table(
    filtered_df,
    index='channel',
    columns='device',
    values='ga_session_id',
    aggfunc='nunique',  # Count unique sessions
    fill_value=0  # Replace missing values with 0 for better readability
)

# Reset the pivot table to a long format for Plotly
pivot_long = pivot_table.reset_index().melt(id_vars='channel',␣
 ↪var_name='device', value_name='sessions')
```

```python
# Create an interactive heatmap using Plotly Express
fig = px.density_heatmap(
    pivot_long,
    x='device',
    y='channel',
    z='sessions',
    title='Number of Sessions by Traffic Channel and Device Type',
    labels={'sessions': 'Number of Sessions'},
    color_continuous_scale='YlGnBu',  # Color scale for intensity
    text_auto=True  # Display values on the heatmap
)

# Update layout for better readability
fig.update_layout(
    xaxis_title='Device Type',
    yaxis_title='Traffic Channel',
    hovermode='closest'
)

fig.show()

# Perform Kruskal-Wallis Test for sessions across traffic channels


# Group sessions by traffic channel
channels = df['channel'].unique()
channel_sessions = [df[df['channel'] == channel]['ga_session_id'].nunique() for␣
  ↪channel in channels]

# Perform the Kruskal-Wallis test (non-parametric test for multiple groups)
statistic, p_value = kruskal(*channel_sessions)

# Output the test results
print(f"Kruskal-Wallis Test Results:")
print(f"Statistic: {statistic:.2f}")
print(f"P-value: {p_value:.4f}")

if p_value < 0.05:
    print("The difference in sessions across traffic channels is statistically␣
  ↪significant (p < 0.05).")
else:
    print("There is no statistically significant difference in sessions across␣
  ↪traffic channels (p >= 0.05).")
```

```
Sales by traffic channel (as percentage of total sales):
 channel
```

```
Direct            23.442345
Organic Search    35.760189
Paid Search       26.620546
Social Search      7.919827
Undefined          6.257093
Name: price, dtype: float64
```

```
Kruskal-Wallis Test Results:
Statistic: 4.00
P-value: 0.4060
There is no statistically significant difference in sessions across traffic
channels (p >= 0.05).
```

0.7 User Behavior Analysis

- Registered vs unregistered users
- Email verification and subscription rates
- Sales by user type
- Histograms and statistical tests (Mann-Whitney U)

[12]:
```
# Separate sales by user type (registered vs unregistered)
registered_sales = df[df['account_id'].notna()]['price']  # Sales from␣
 ↪registered users
unregistered_sales = df[df['account_id'].isna()]['price']  # Sales from␣
 ↪unregistered users

# Histogram for registered users
fig_registered = px.histogram(registered_sales, nbins=30, title="Sales␣
 ↪Distribution for Registered Users", labels={'value': 'Total Sales'},␣
 ↪color_discrete_sequence=['blue'])
fig_registered.update_layout(xaxis_title="Total Sales", yaxis_title="Frequency")
fig_registered.show()

# Histogram for unregistered users
fig_unregistered = px.histogram(unregistered_sales, nbins=30, title="Sales␣
 ↪Distribution for Unregistered Users", labels={'value': 'Total Sales'},␣
 ↪color_discrete_sequence=['orange'])
fig_unregistered.update_layout(xaxis_title="Total Sales",␣
 ↪yaxis_title="Frequency")
fig_unregistered.show()

# Perform Mann-Whitney U test
statistic, p_value = mannwhitneyu(registered_sales, unregistered_sales)

# Output test results
print(f"Mann-Whitney U Test Results:")
print(f"Statistic: {statistic:.2f}")
print(f"P-value: {p_value:.4f}")
```

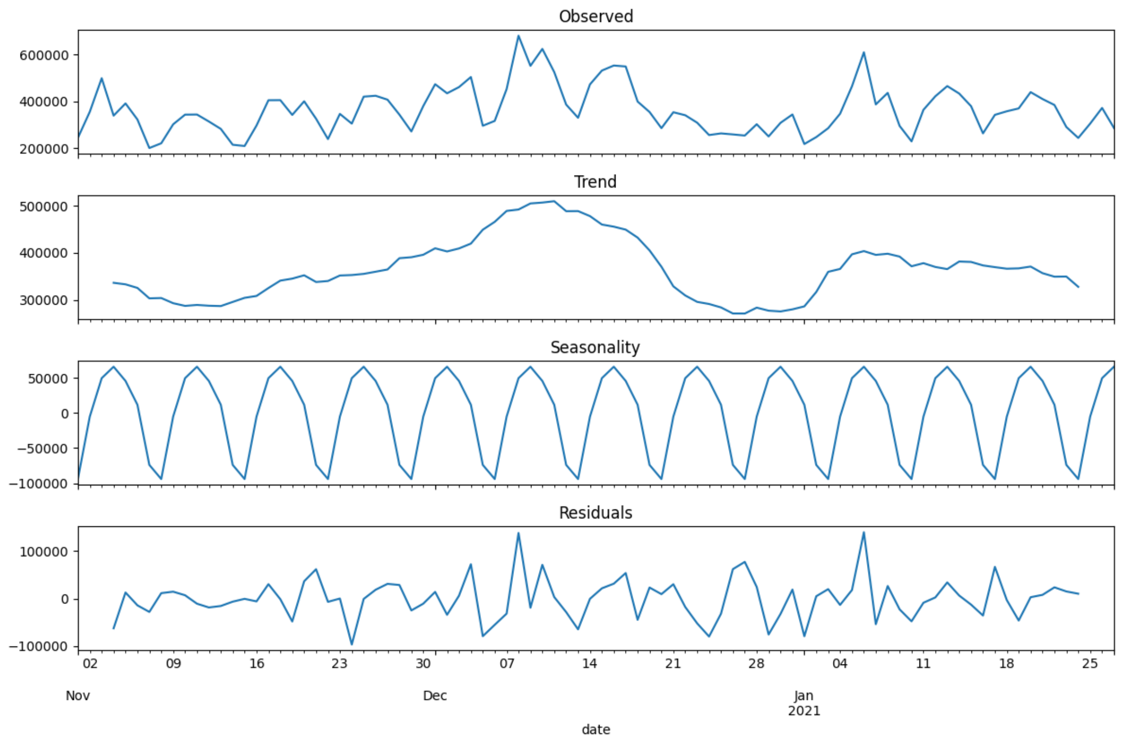
```
# Interpretation of results
if p_value < 0.05:
    print("There is a statistically significant difference between the sales of␣
 ↪registered and unregistered users (p < 0.05).")
else:
    print("There is no statistically significant difference between the sales␣
 ↪of registered and unregistered users (p >= 0.05).")

# Analyze email verification rate
verified_users = df[df['is_verified'] == 1]['account_id'].nunique()  # Verified␣
 ↪users
total_registered_users = df['account_id'].notna().sum()  # Total registered␣
 ↪users

# Percentage of verified users
verified_percentage = (verified_users / total_registered_users) * 100
print(f"Percentage of registered users who verified their email:␣
 ↪{verified_percentage:.2f}%")

# Analyze subscription rates
unsubscribed_users = df[df['is_unsubscribed'] == 1]['account_id'].nunique()  #␣
 ↪Unsubscribed users
unsubscribed_percentage = (unsubscribed_users / total_registered_users) * 100
print(f"Percentage of registered users who unsubscribed:␣
 ↪{unsubscribed_percentage:.2f}%")

# Email verification status
verification_data = {'Status': ['Verified', 'Not Verified'], 'Count':␣
 ↪[verified_users, total_registered_users - verified_users]}
df_verification = pd.DataFrame(verification_data)

fig_verification = go.Figure(data=[go.Pie(labels=df_verification['Status'],␣
 ↪values=df_verification['Count'], hole=0.3)])
fig_verification.update_layout(title='Email Verification Status of Registered␣
 ↪Users')
fig_verification.show()

# Subscription status
subscription_data = {'Status': ['Unsubscribed', 'Subscribed'], 'Count':␣
 ↪[unsubscribed_users, total_registered_users - unsubscribed_users]}
df_subscription = pd.DataFrame(subscription_data)

fig_subscription = go.Figure(data=[go.Pie(labels=df_subscription['Status'],␣
 ↪values=df_subscription['Count'], hole=0.3)])
fig_subscription.update_layout(title='Subscription Status of Registered Users')
fig_subscription.show()
```

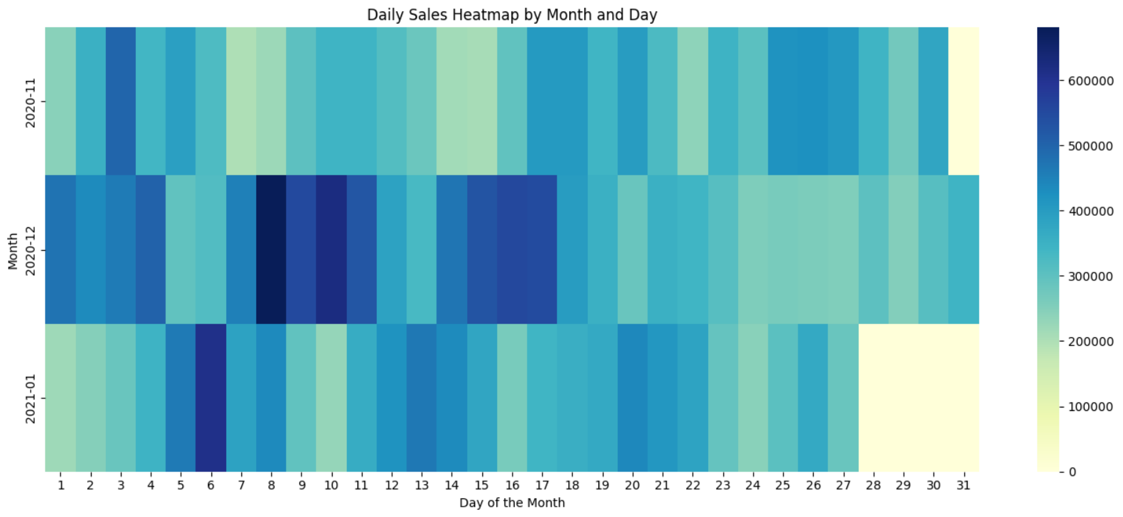
```
Mann-Whitney U Test Results:
Statistic: 41771375.00
P-value: 0.0416
There is a statistically significant difference between the sales of registered
and unregistered users (p < 0.05).
Percentage of registered users who verified their email: 71.52%
Percentage of registered users who unsubscribed: 16.07%
```

0.8 Time Series & Seasonality

- Daily and monthly sales trends
- Sales by day of the week
- Time series decomposition
- Seasonal heatmaps

[9]:
```
# Daily sales trend
daily_sales = df.groupby('date')['price'].sum().reset_index()

fig = px.line(
    daily_sales, x='date', y='price',
    title='Daily Sales Trend',
    labels={'date': 'Date', 'price': 'Total Sales ($)'},
    template='plotly_white'
)
fig.update_xaxes(rangeslider_visible=True)
fig.show()

# Monthly sales trend
monthly_sales = df.groupby(df['date'].dt.to_period('M'))['price'].sum().
  ↪reset_index()
monthly_sales['Month'] = monthly_sales['date'].dt.to_timestamp()
monthly_sales = monthly_sales.rename(columns={'price': 'Total Sales'})

fig = px.line(
    monthly_sales, x='Month', y='Total Sales',
    title='Monthly Sales Trend',
    labels={'Month': 'Month', 'Total Sales': 'Total Sales ($)'},
    template='plotly_white'
)
fig.update_traces(mode='lines+markers')
fig.show()

# Sales by day of the week
daily_sales['Day of Week'] = daily_sales['date'].dt.day_name()
avg_sales_by_day = daily_sales.groupby('Day of Week')['price'].mean().reindex([
    'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday'
]).reset_index().rename(columns={'price': 'Average Sales'})

fig = px.bar(
    avg_sales_by_day, x='Day of Week', y='Average Sales',
    title='Average Sales by Day of the Week',
    labels={'Average Sales': 'Avg Sales ($)'},
    template='plotly_white'
)
fig.show()

# Time Series Decomposition (weekly seasonality)
ts = daily_sales.set_index('date')['price'].asfreq('D').fillna(0)
decomp = seasonal_decompose(ts, model='additive', period=7)

# Plot decomposition with matplotlib
fig, axs = plt.subplots(4, 1, figsize=(12, 8), sharex=True)
decomp.observed.plot(ax=axs[0], title='Observed')
decomp.trend.plot(ax=axs[1], title='Trend')
decomp.seasonal.plot(ax=axs[2], title='Seasonality')
decomp.resid.plot(ax=axs[3], title='Residuals')
plt.tight_layout()
plt.show()

# Seasonal heatmap (days by month)
heatmap_data = daily_sales.copy()
heatmap_data['Month'] = heatmap_data['date'].dt.strftime('%Y-%m')
heatmap_data['Day'] = heatmap_data['date'].dt.day

pivot = heatmap_data.pivot_table(index='Month', columns='Day', values='price',␣
 ↪fill_value=0)

plt.figure(figsize=(14, 6))
sns.heatmap(pivot, cmap='YlGnBu')
plt.title('Daily Sales Heatmap by Month and Day')
plt.xlabel('Day of the Month')
plt.ylabel('Month')
plt.tight_layout()
plt.show()
```





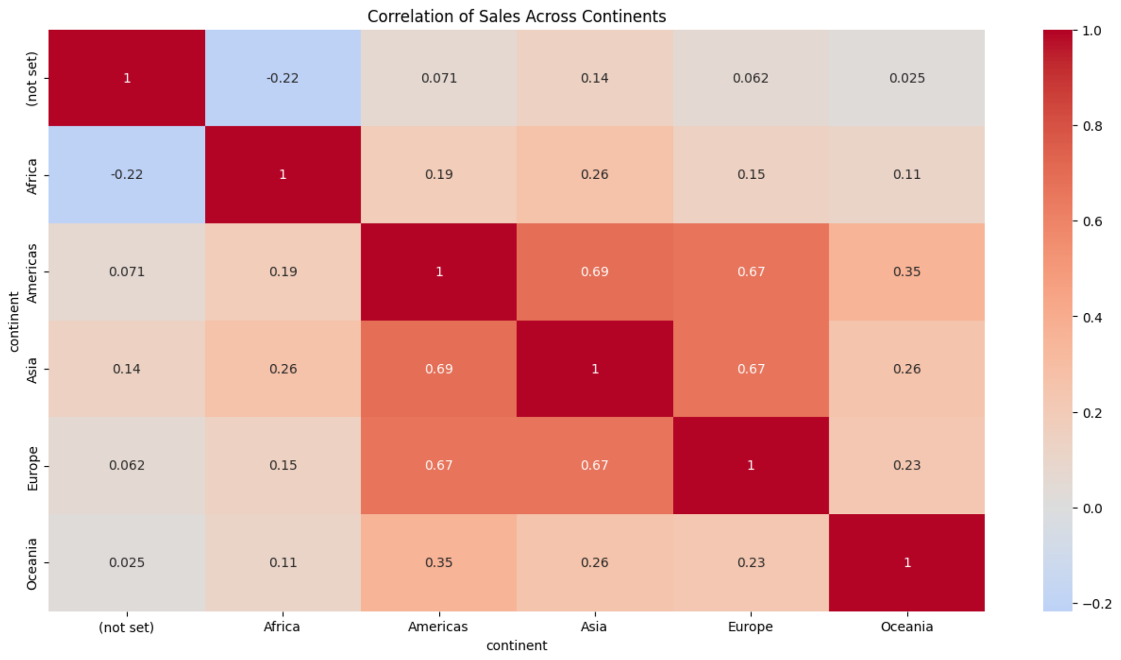
0.9 Correlation Analysis

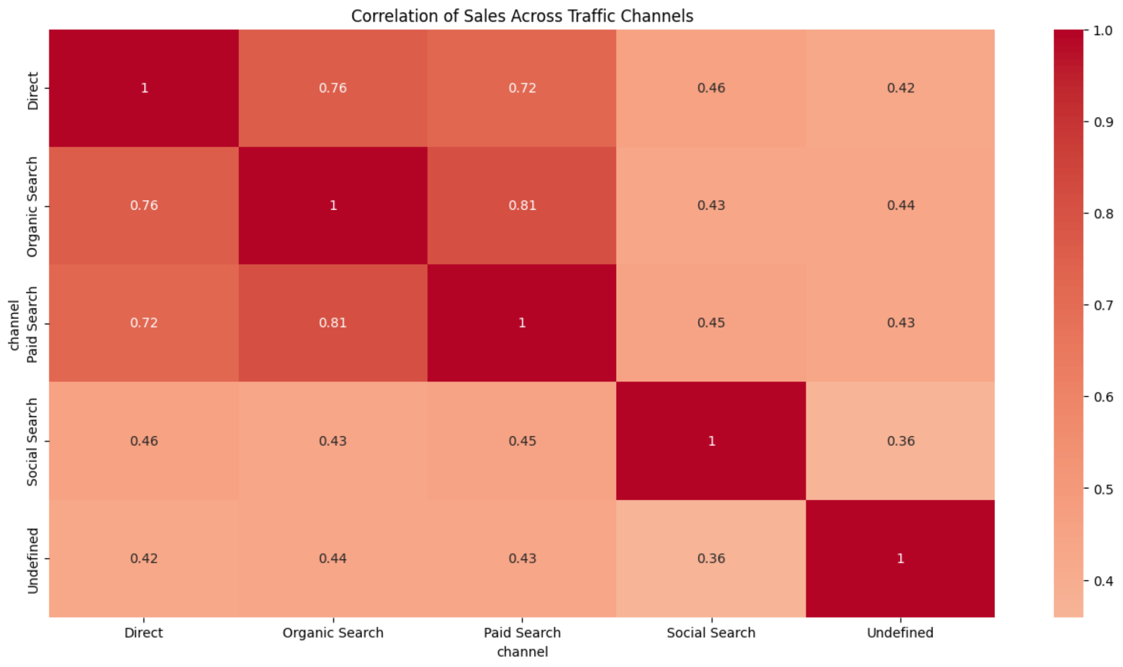
- Sessions vs sales (Pearson correlation)
- Sales correlation across continents, channels, and product categories
- Correlation heatmaps

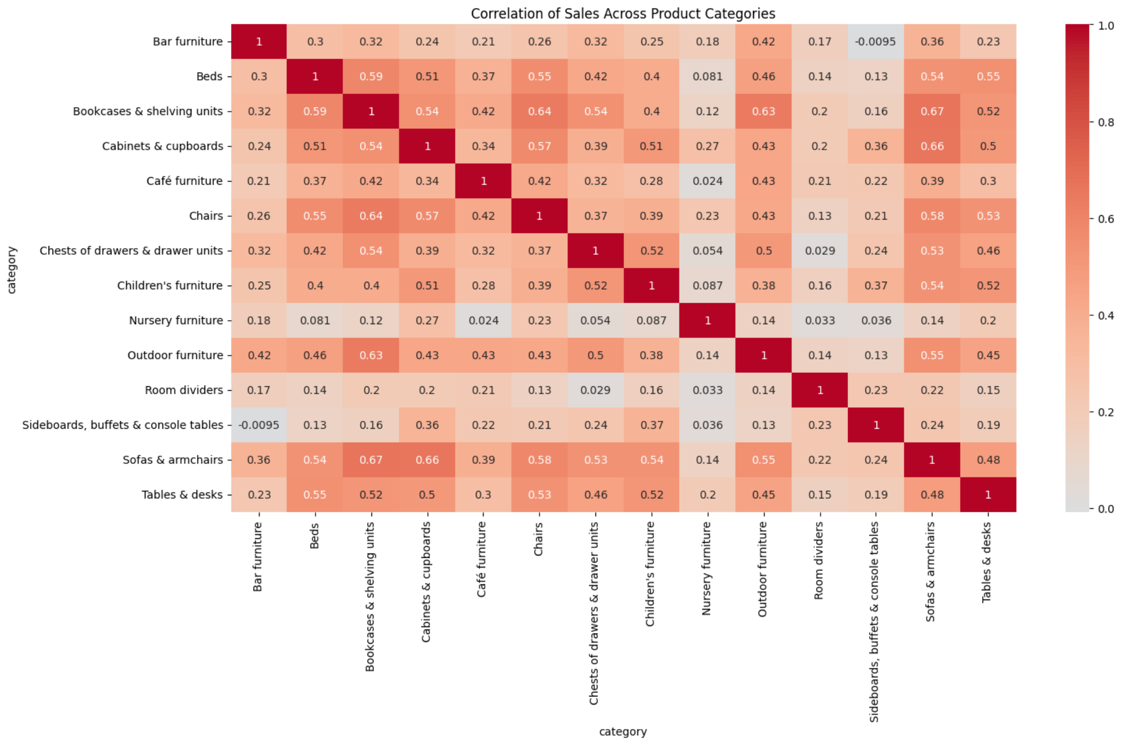
[10]:
```
# Correlation between sessions and sales (daily)
daily_stats = df.groupby('date').agg(
    daily_sales=('price', 'sum'),
    daily_sessions=('ga_session_id', 'nunique')
).reset_index()

# Pearson correlation
r, p_value = pearsonr(daily_stats['daily_sessions'], daily_stats['daily_sales'])
print("Pearson Correlation between sessions and sales:")
print(f"r = {r:.4f}, p-value = {p_value:.4f}")

# Scatter plot with trendline (using Plotly for interactivity and wider display)
fig = px.scatter(
    daily_stats,
    x='daily_sessions', y='daily_sales',
    trendline='ols',
    title='Correlation Between Daily Sessions and Sales',
    labels={'daily_sessions': 'Sessions', 'daily_sales': 'Sales ($)'},
    template='plotly_white'
)
fig.update_layout(
    autosize=True,
    height=600,
)
fig.show()

# Pivot tables of sales by continent, channel, and category
sales_by_continent = df.groupby(['date', 'continent'])['price'].sum().unstack().
 ↪fillna(0)
sales_by_channel = df.groupby(['date', 'channel'])['price'].sum().unstack().
 ↪fillna(0)
sales_by_category = df.groupby(['date', 'category'])['price'].sum().unstack().
 ↪fillna(0)

# Correlation matrices
corr_continent = sales_by_continent.corr()
corr_channel = sales_by_channel.corr()
corr_category = sales_by_category.corr()

# Heatmaps
plt.figure(figsize=(16, 8))
sns.heatmap(corr_continent, annot=True, cmap='coolwarm', center=0)
plt.title('Correlation of Sales Across Continents')
plt.show()

plt.figure(figsize=(16, 8))  # Same for other heatmaps
sns.heatmap(corr_channel, annot=True, cmap='coolwarm', center=0)
```

```
plt.title('Correlation of Sales Across Traffic Channels')
plt.show()

plt.figure(figsize=(16, 8))  # Same for the third heatmap
sns.heatmap(corr_category, annot=True, cmap='coolwarm', center=0)
plt.title('Correlation of Sales Across Product Categories')
plt.show()
```

```
Pearson Correlation between sessions and sales:
r = 0.9642, p-value = 0.0000
```




Correlation of Sales Across Traffic Channels
channel
Direct
Organic Search
Paid Search
Social Search
Undefined
Direct: 1 0.76 0.72 0.46 0.42
Organic Search: 0.76 1 0.81 0.43 0.44
Paid Search: 0.72 0.81 1 0.45 0.43
Social Search: 0.46 0.43 0.45 1 0.36
Undefined: 0.42 0.44 0.43 0.36 1
1.0
0.9
0.8
0.7
0.6
0.5
0.4


Correlation of Sales Across Product Categories
category
Bar furniture: 1 0.3 0.32 0.24 0.21 0.26 0.32 0.25 0.18 0.42 0.17 -0.0095 0.36 0.23
Beds: 0.3 1 0.59 0.51 0.37 0.55 0.42 0.4 0.081 0.46 0.14 0.13 0.54 0.55
Bookcases & shelving units: 0.32 0.59 1 0.54 0.42 0.64 0.54 0.4 0.12 0.63 0.2 0.16 0.67 0.52
Cabinets & cupboards: 0.24 0.51 0.54 1 0.34 0.57 0.39 0.51 0.27 0.43 0.2 0.36 0.66 0.5
Café furniture: 0.21 0.37 0.42 0.34 1 0.42 0.32 0.28 0.024 0.43 0.21 0.22 0.39 0.3
Chairs: 0.26 0.55 0.64 0.57 0.42 1 0.37 0.39 0.23 0.43 0.13 0.21 0.58 0.53
Chests of drawers & drawer units: 0.32 0.42 0.54 0.39 0.32 0.37 1 0.52 0.054 0.5 0.029 0.24 0.53 0.46
Children's furniture: 0.25 0.4 0.4 0.51 0.28 0.39 0.52 1 0.087 0.38 0.16 0.37 0.54 0.52
Nursery furniture: 0.18 0.081 0.12 0.27 0.024 0.23 0.054 0.087 1 0.14 0.033 0.036 0.14 0.2
Outdoor furniture: 0.42 0.46 0.63 0.43 0.43 0.43 0.5 0.38 0.14 1 0.14 0.13 0.55 0.45
Room dividers: 0.17 0.14 0.2 0.2 0.21 0.13 0.029 0.16 0.033 0.14 1 0.23 0.22 0.15
Sideboards, buffets & console tables: -0.0095 0.13 0.16 0.36 0.22 0.21 0.24 0.37 0.036 0.13 0.23 1 0.24 0.19
Sofas & armchairs: 0.36 0.54 0.67 0.66 0.39 0.58 0.53 0.54 0.14 0.55 0.22 0.24 1 0.48
Tables & desks: 0.23 0.55 0.52 0.5 0.3 0.53 0.46 0.52 0.2 0.45 0.15 0.19 0.48 1
1.0
0.8
0.6
0.4
0.2
0.0

0.10 Key Findings & Recommendations

- Summary of analytical insights
- Actionable recommendations (marketing, inventory, engagement)
- Observations on user behavior and seasonality

0.10.1 Summary of Analytical Insights

- **Sales Distribution**
  - The majority of revenue comes from the United States, followed by India and Canada.
  - Organic Search and Paid Search channels drive over 60% of total sales.
- **Top Products and Devices**
  - Sofas & Armchairs and Chairs are the most purchased categories.
  - Desktop devices contribute ~59% of sales; mobile ~39%.
- **User Behavior**
  - Registered users make more frequent but smaller purchases.
  - Unregistered users generate higher order values per transaction.
  - Email verification rate is ~71%, while ~16% of users unsubscribe.
- **Seasonality**
  - Clear spike in sales around New Year; lower activity in early January.
  - Higher average sales on Tuesdays and Wednesdays.
- **Correlations**
  - Daily sessions and sales have strong positive correlation (r   0.96).
  - Similar sales patterns are observed across product categories and traffic channels.

0.10.2 Actionable Recommendations

Marketing: - Focus campaign spend on Organic and Paid Search — the top-performing channels. - Introduce mid-week promotions (Tue/Wed) to leverage higher purchase activity. - Launch retargeting campaigns for unregistered users to encourage account creation.

Inventory & Merchandising: - Prioritize stock for Sofas, Chairs, and Beds — the highest-selling categories. - Prepare extra inventory ahead of holidays (December). - Use product bundles to encourage cross-category purchases (e.g. Sofa + Armchair).

User Engagement: - Offer loyalty incentives to increase order value for registered users. - Add in-site nudges for email verification to improve communication rates. - Promote newsletter opt-ins with first-order discounts or exclusive offers.

0.10.3 Observations on Behavior & Seasonality

- Weekday patterns indicate strong business-hour engagement — align campaigns accordingly.
- Mobile traffic drives ~40% of sales — ensure mobile checkout is fast and frictionless.
- Correlated demand across product categories allows for bundling and smart recommendations.

0.11 Conclusion

- Business insights recap
- Future steps and improvements

0.11.1 Key Analytical Insights:

- **Sales by Country and Channel:**
  - The United States is the largest contributor to total sales, followed by India and Canada.
  - Over 60% of sales come from Organic Search and Paid Search channels.
- **Top Products and Devices:**
  - The most popular product categories are "Sofas & Armchairs" and "Chairs".
  - Desktop devices account for ~59% of sales, while Mobile accounts for ~39%.
- **User Behavior:**
  - Registered users tend to make more frequent, lower-value purchases.
  - Unregistered users make fewer but larger transactions.
  - 71% of registered users verify their email, while 16% unsubscribe.
- **Seasonality and Days of the Week:**
  - There is a clear sales peak during the New Year period, followed by a drop in early January.
  - Tuesdays and Wednesdays see the highest sales, while weekends experience lower activity.
- **Correlations:**
  - There is a strong positive correlation between daily sessions and sales (r 0.96).
  - Similar sales patterns are observed across product categories and traffic channels.

0.11.2 Actionable Recommendations:

Marketing: - Increase investment in Organic and Paid Search, as these channels drive the most sales. - Target promotional efforts on Tuesdays and Wednesdays to capitalize on mid-week activity. - Launch retargeting campaigns for unregistered users to encourage registration.

Inventory & Merchandising: - Prioritize stock for Sofas, Chairs, and Beds — the top-selling categories. - Prepare for increased demand around holidays (December). - Use product bundles to boost average order value.

User Engagement: - Implement loyalty programs to increase sales from registered users. - Encourage email verification to improve communication and engagement. - Optimize the mobile checkout experience, as mobile sales contribute ~40%.

0.11.3 Observations on Behavior & Seasonality:

- Weekday patterns indicate that most sales occur on weekdays, especially Tuesday and Wednesday.
- Sales surge during the holiday period, suggesting that holiday-specific campaigns are highly effective.

- Mobile users contribute significantly to sales — ensure the mobile site is optimized for seamless user experience.