In [ ]:
```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt

sns.set(rc={'figure.figsize':(20.7,8.27)})
sns.set_style("whitegrid")
sns.color_palette("dark")
plt.style.use("fivethirtyeight")
```

Load Dataset

In [ ]:
```
raw_ecommerce = pd.read_csv('dataset/Dataset.csv')
```

In [ ]:
```
raw_ecommerce.columns = raw_ecommerce.columns.str.lower()
```

EDA

Descriptive Statistics

In [ ]:
```
raw_ecommerce.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12946 entries, 0 to 12945
Data columns (total 18 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   administrative           12835 non-null  float64
 1   administrative_duration  12313 non-null  float64
 2   informational            12946 non-null  int64  
 3   informational_duration   12946 non-null  float64
 4   productrelated           12946 non-null  int64  
 5   productrelated_duration  12307 non-null  float64
 6   bouncerates              12872 non-null  float64
 7   exitrates                12946 non-null  float64
 8   pagevalues               12946 non-null  float64
 9   specialday               12946 non-null  float64
 10  month                    12946 non-null  object 
 11  operatingsystems         12422 non-null  float64
 12  browser                  12946 non-null  int64  
 13  region                   12946 non-null  int64  
 14  traffictype              12946 non-null  int64  
 15  visitortype              12946 non-null  object 
 16  weekend                  12946 non-null  bool   
 17  revenue                  12946 non-null  bool   
dtypes: bool(2), float64(9), int64(5), object(2)
memory usage: 1.6+ MB
```

In [ ]:
```
raw_ecommerce.isna().sum()
```

Out[ ]:
```
administrative             111
administrative_duration    633
informational                0
informational_duration       0
productrelated               0
productrelated_duration    639
bouncerates                 74
exitrates                    0
pagevalues                   0
specialday                   0
month                        0
operatingsystems           524
browser                      0
region                       0
traffictype                  0
visitortype                  0
weekend                      0
revenue                      0
dtype: int64
```

In [ ]:
```
raw_ecommerce.duplicated().sum()
```

Out[ ]:
```
711
```

terdapat **12946** baris data, dengan jumlah attribut 18. Dari 18 attribut, dideteksi ada 5 attribut yang memiliki nilai kosong. dan terdapat **711** data duplikat

In [ ]:
```
cats = ['month','weekend','specialday','region','operatingsystems','browser','traff
nums = ['administrative','administrative_duration','informational','informational_d
        'productrelated','productrelated_duration',
        'bouncerates','exitrates','pagevalues']
```

In [ ]:
```
raw_ecommerce.describe()
```

Out[ ]:

	administrative	administrative_duration	informational	informational_duration	productrelate
count	12835.000000	12313.000000	12946.000000	12946.000000	12946.00000
mean	2.303857	80.370267	0.498841	34.136048	31.65765
std	3.314427	175.494016	1.263276	140.022848	44.20263
min	0.000000	0.000000	0.000000	0.000000	0.00000
25%	0.000000	0.000000	0.000000	0.000000	7.00000
50%	1.000000	7.000000	0.000000	0.000000	18.00000
75%	4.000000	92.933333	0.000000	0.000000	38.00000
max	27.000000	3398.750000	24.000000	2549.375000	705.00000

In [ ]:
```
raw_ecommerce[cats].astype(str).describe()
```

Out[ ]:

	month	weekend	specialday	region	operatingsystems	browser	traffictype	visitort
count	12946	12946	12946	12946	12946	12946	12946	12
unique	10	2	6	9	9	13	20	
top	May	False	0.0	1	2.0	2	2	Returning_Vi
freq	3533	9929	11636	5031	6673	8360	4100	11

In [ ]:

```
raw_ecommerce['revenue'].value_counts() / len(raw_ecommerce['revenue'])*100
```

Out[ ]:

```
False    84.489418
True     15.510582
Name: revenue, dtype: float64
```

1. Descriptive Statistics Insight

- A. Apakah ada kolom dengan tipe data kurang sesuai, atau nama kolom dan isinya kurang sesuai?
- B. Apakah ada kolom yang memiliki nilai kosong? Jika ada, apa saja?
- C. Apakah ada kolom yang memiliki nilai summary agak aneh? (min/mean/median/max/unique/top/freq)

A.

- tipe data kolom operating system dapat menggunakan tipe data int,\
- tipe data kolom month juga dapat menggunakan int. kolom lainnya sudah sesuai.

B.

Terdapat 12.946 baris data, dengan jumlah fitur 18. Dari 18 fitur tersebut, ada 5 fitur yang memiliki nilai null diantaranya:

1. Administrative `111` null data
2. Administrative_Duration `633` null data
3. ProductRelated_Duration `639` null data
4. BounceRates `74` null data
5. OperatingSystems `524` null data

Selain nilai null, juga terdapat 711 data *duplicated*

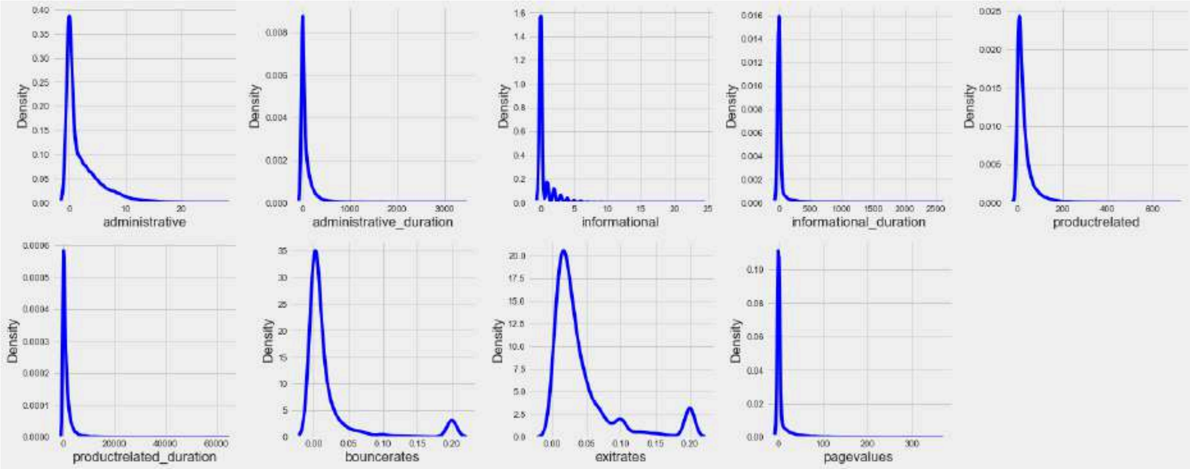
C.

- Untuk fitur numerik (nums) terdapat outlier pada masing-masing fiturnya, dan sebaran nilai masing-masing fitur merupakan sebaran positively skewed, karena nilai mean yang lebih besar dari nilai median nya.

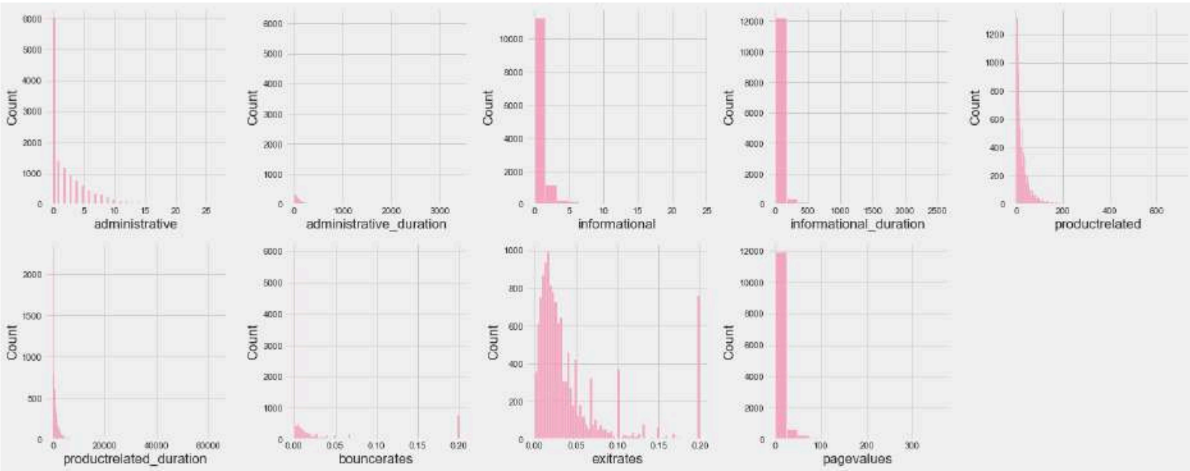
- Sedangkan untuk fitur kategorikal (cats), fitur revenue dipilih sebagai target. tetapi atribut ini memiliki imbalances, dimana nilai False/Not Buyer terdapat sebanyak 10.938 data, sehingga perlu untuk disesuaikan ketika proses training.

Univariate Analysis

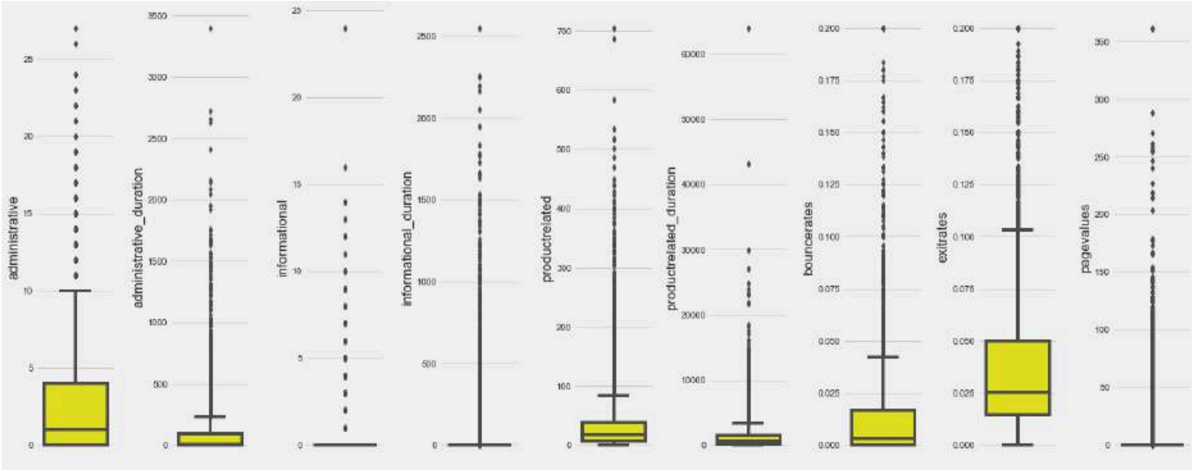
In [ ]:
```
for i in range(0, len(nums)):
    plt.subplot(2,5, i+1)
    sns.kdeplot(x=raw_ecommerce[nums[i]], color='blue')
plt.tight_layout()
```



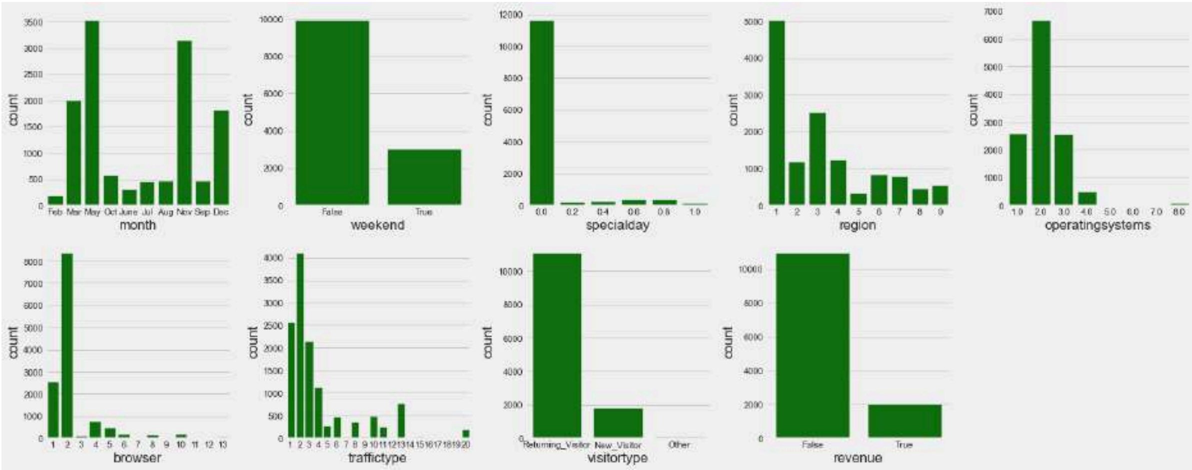
In [ ]:
```
for i in range(0, len(nums)):
    plt.subplot(2,5, i+1)
    sns.histplot(x=raw_ecommerce[nums[i]], color='#f78fb3')
plt.tight_layout()
```



In [ ]:
```
for i in range(0, len(nums)):
    plt.subplot(1, len(nums), i+1)
    sns.boxplot(data=raw_ecommerce, y=nums[i], color='yellow')
plt.tight_layout()
```



In [ ]:
```
for i in range(0, len(cats)):
    plt.subplot(2,5, i+1)
    sns.countplot(x=raw_ecommerce[cats[i]], color='green')
plt.tight_layout()
```



2. Univariate Analysis Insight

Gunakan visualisasi untuk melihat distribusi masing-masing kolom (feature maupun target). Tuliskan hasil observasinya, misalnya jika ada suatu kolom yang distribusinya menarik (misal skewed, bimodal, ada outlier, ada nilai yang mendominasi, kategorinya terlalu banyak, dsb). Jelaskan juga apa yang harus di-follow up saat data pre-processing.

---

untuk kolom numerikal berikut ini memiliki distribusi positively skewed dan juga memiliki outlier:

- 'administrative'
- 'administrative_duration'
- 'informational'
- 'informational_duration'
- 'productrelated'
- 'productrelated_duration'
- 'bouncerates'
- 'exitrates'
- 'pagevalues'

Untuk tahap preprocessing dapat dilakukan, handling outlier dan feature transformation.

Untuk kolom kategorikal :

- 'month' : jumlah data didominasi bulan: May, Nov, Mar, Dec
- 'weekend' : didominasi oleh nilai 'False'
- 'specialday' : kunjungan situs mayoritas dilakukan saat, jauh dari specialday (hari khusus)
- 'region' : observasi menunjukan user region 1 mendominasi
- 'operatingsystem' : yang digunakan banyak user 2, 1, 3, 4
- 'browser' : jenis 2 mendominasi data dari 13 jenis browser
- 'traffictype' : jenis traffic yang paling banyak membawa user merupakan traffic 2, 1, 3
- 'visitortype' : kunjungan mayoritas dilakukan oleh returning_visitor
- 'revenue' : sebanyak 84.48% dari kunjungan tidak melakakukan pembelian / tidak menghasilkan pendapatan

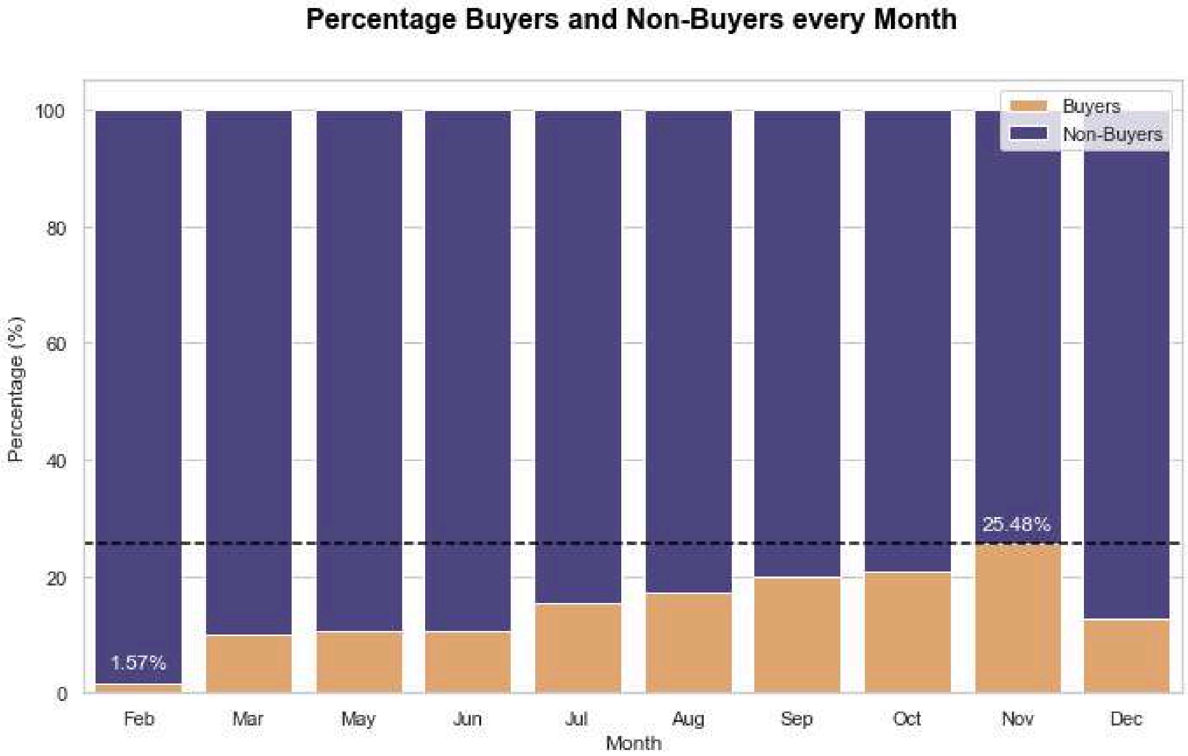
Untuk kolom revenue sebagai target perlu dilakukan imbalances handling\ kolom visitortype dan month, dapat dilakukan feature encoding agar dapat dilakukan algoritma korelasi\

Multivariate Analysis

In [ ]:

```
# groupby month
month_revenue       = raw_ecommerce.groupby(['month', 'revenue'])['revenue'].count(

# ubah ke pivot
df_pivot = month_revenue.pivot_table(index='month', columns='revenue', values='cour
df_pivot = df_pivot.reset_index()
df_pivot.columns = ['month', 'non buyer', 'buyer']

# sorted bulan agar berurutan
df_pivot.loc[df_pivot['month'] == 'June', 'month'] = 'Jun'
month_order = ['Feb', 'Mar', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'
df_pivot['month'] = pd.Categorical(df_pivot['month'], categories=month_order, order
df_pivot_sorted = df_pivot.sort_values(by='month')

# ubah month menjadi index
df_pivot_sorted.set_index('month', inplace=True)

# Menghitung total untuk setiap bulan
df_pivot_sorted['total'] = df_pivot_sorted['non buyer'] + df_pivot_sorted['buyer']

# Menghitung persentase untuk setiap kategori (False dan True)
df_pivot_sorted['non buyer_percent'] = (df_pivot_sorted['non buyer'] / df_pivot_sor
df_pivot_sorted['buyer_percent'] = (df_pivot_sorted['buyer'] / df_pivot_sorted['tot

# Menggambar stacked bar plot
sns.set(style="whitegrid")
plt.figure(figsize=(10, 6))

sns.barplot(x=df_pivot_sorted.index, y=df_pivot_sorted['buyer_percent'], color='Sar
sns.barplot(x=df_pivot_sorted.index, y=df_pivot_sorted['non buyer_percent'], botton

plt.xlabel('Month')
plt.ylabel('Percentage (%)')
plt.title('Percentage Buyers and Non-Buyers every Month', color='black', fontsize=1

#adding horizontal line
plt.axhline(y=df_pivot_sorted.loc['Nov','buyer_percent'],color='Black',ls='--')
```

```
#adding text
plt.text(8, df_pivot_sorted.loc['Nov', 'buyer_percent'] + 2, '25.48%', ha='center',
plt.text(0, df_pivot_sorted.loc['Feb', 'buyer_percent'] + 2, '1.57%', ha='center',

plt.legend()

plt.show()
```



In [ ]:
```
df_pivot
```

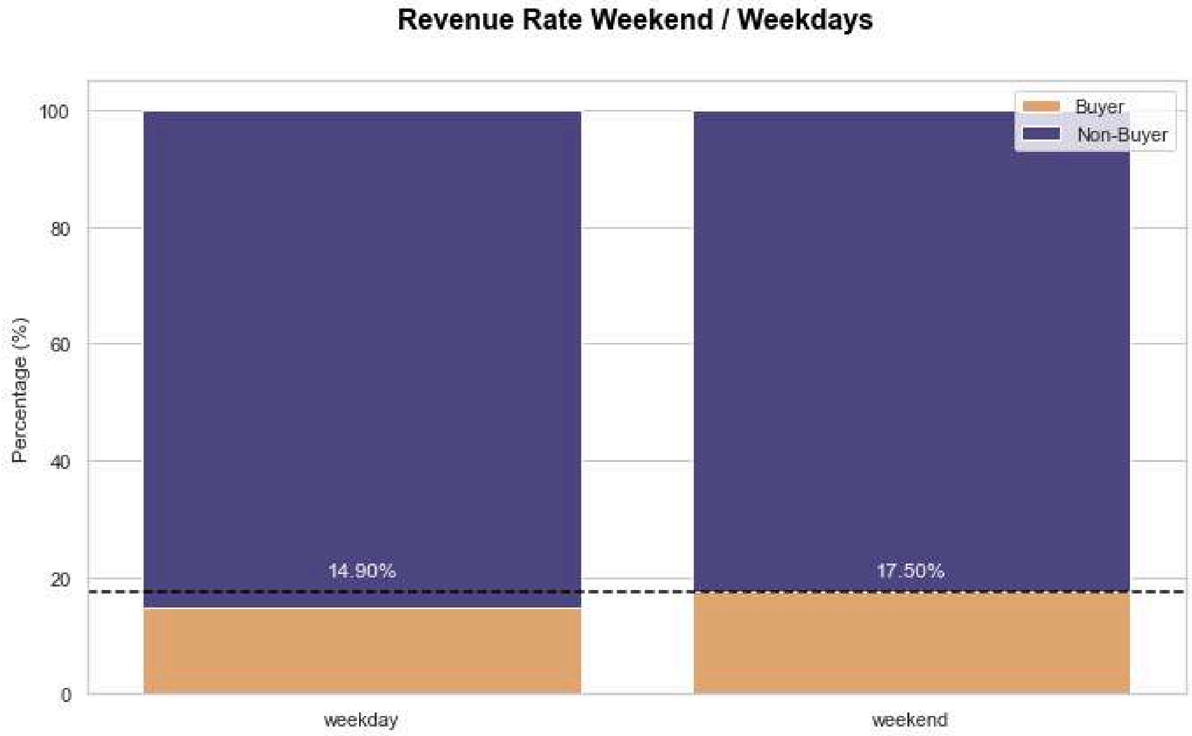
Out[ ]:

	month	non buyer	buyer
0	Aug	382	79
1	Dec	1588	228
2	Feb	188	3
3	Jul	381	70
4	Jun	275	33
5	Mar	1796	201
6	May	3154	379
7	Nov	2348	803
8	Oct	455	119
9	Sep	371	93

Kunjungan user pada platform, yang menghasilkan revenue didominasi pada bulan **November** `25,48% Revenue Rate` , Sementara bulan **Februari** memiliki kunjungan yang menghasilkan revenue yang paling sedikit `1.57% Revenue Rate` (3 buyer).

In [ ]:
```
import matplotlib.pyplot as plt
from matplotlib import cm
```

```
weekend_revenue = raw_ecommerce.groupby(['weekend', 'revenue'])['revenue'].count().
weekend_revenue['weekend'] = weekend_revenue['weekend'].map({False: 'weekday', True
weekend_revenue['revenue'] = weekend_revenue['revenue'].map({False: 'Non-Buyer', Tr

# creating pivot table
weekend_revenue_pivot = weekend_revenue.pivot_table(index='weekend', columns='rever

#changing names
weekend_revenue_pivot.columns = ['Buyers','Non-Buyers']

# adding column total customer
weekend_revenue_pivot['total'] = weekend_revenue_pivot.sum(axis=1)

# Calculate revenue rate for weekends and weekdays
weekend_revenue_pivot['buyer_pct'] = weekend_revenue_pivot['Buyers'] / weekend_reve

weekend_revenue_pivot['non_buyer_pct'] = weekend_revenue_pivot['Non-Buyers'] /  wee

# Creating stacked bar plot
sns.set(style="whitegrid")
plt.figure(figsize=(10, 6))

sns.barplot(x=weekend_revenue_pivot.index, y=weekend_revenue_pivot['buyer_pct'], co
sns.barplot(x=weekend_revenue_pivot.index, y=weekend_revenue_pivot['non_buyer_pct']
            color='DarkSlateBlue', label='Non-Buyer')

plt.xlabel('')
plt.ylabel('Percentage (%)')
plt.title('Revenue Rate Weekend / Weekdays', color='black', fontsize=16, loc='cente

#adding horizontal line
plt.axhline(y=weekend_revenue_pivot.loc['weekend','buyer_pct'],color='Black',ls='--

#adding text
#adding text
plt.text(1, weekend_revenue_pivot.loc['weekend','buyer_pct'] + 2, '17.50%', ha='cer
plt.text(0, weekend_revenue_pivot.loc['weekend','buyer_pct'] + 2, '14.90%', ha='cer

plt.legend()

plt.show()
```



In [ ]:
```
weekend_revenue_pivot
```

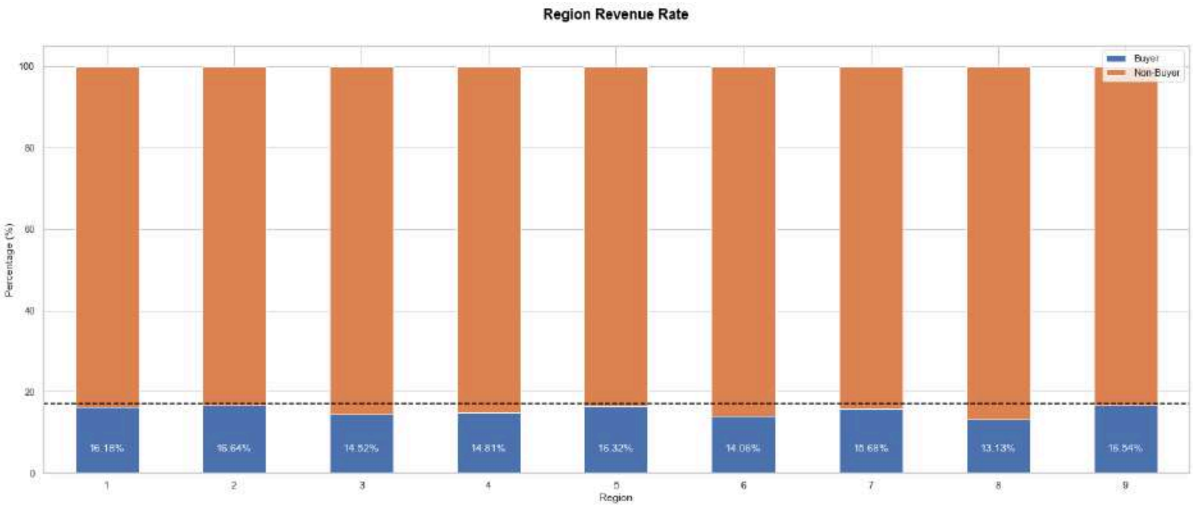
Out[ ]:

	Buyers	Non-Buyers	total	buyer_pct	non_buyer_pct
weekend					
weekday	1480	8449	9929	14.905831	85.094169
weekend	528	2489	3017	17.500829	82.499171

Kunjungan user pada weekday lebih tinggi dari weekend tetapi revenue rate weekend > weekday `17.5% /14.9%`

In [ ]:
```
region_revenue    = raw_ecommerce.groupby(['region','revenue'])['revenue'].count().
reg_pivot         = region_revenue.pivot_table(index='region',columns='revenue', va
reg_pivot.columns = ['Non-Buyer','Buyer']

#calculate revenue rate by region
reg_pivot_pct = reg_pivot.div(reg_pivot.sum(axis=1), axis=0) * 100
reg_pivot_pct = reg_pivot_pct[['Buyer','Non-Buyer']]

# plotting
reg_pivot_pct.plot(kind='bar', stacked=True)

# Add labels and title
plt.xlabel('Region')
plt.ylabel('Percentage (%)')
plt.title('Region Revenue Rate', color='black', fontsize=16, loc='center', weight='
plt.xticks(rotation=0)

# Add legend
plt.legend()

# add horizontal line
plt.axhline(y=reg_pivot_pct.loc[2,'Buyer'] +.5,color='Black',ls='--')

# Add percentages on top of each bar
for index, value in enumerate(reg_pivot_pct['Buyer']):
    plt.text(index, 5, s=f'{round(value,2)}%', ha='center', va='bottom', color='Whi
```

```python
# Display the chart
plt.show()
```



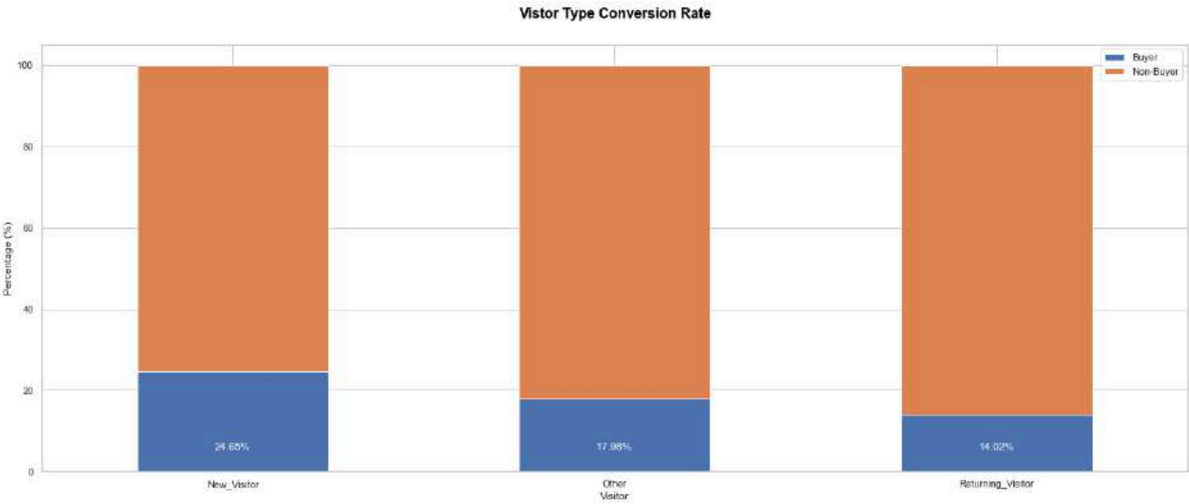
In [ ]:
```python
reg_pivot.sum(axis=1)
```

Out[ ]:
```
region
1    5031
2    1190
3    2528
4    1229
5     337
6     839
7     797
8     457
9     538
dtype: int64
```

Region 1 memiliki pengunjung paling banyak diantara region lainnya. akan tetapi revenue rate region 2 (16.64%) menjadi paling tinggi diantara region lainnya.

In [ ]:
```python
#Group df
visitor_df             = raw_ecommerce.groupby(['visitortype', 'revenue'])\
                         ['revenue'].count().reset_index(name='cnt').sort_values(by=
visitor_pivot          = visitor_df.pivot_table(index='visitortype',columns='revenue
visitor_pivot.columns  = ['Non-Buyer','Buyer']
visitor_pivot          = visitor_pivot[['Buyer','Non-Buyer']]
visitor_rev_pct        = visitor_pivot.div(visitor_pivot.sum(axis=1), axis=0)*100

#Plot
visitor_rev_pct.plot(kind='bar', stacked=True)

# Add labels and title
plt.xlabel('Visitor')
plt.ylabel('Percentage (%)')
plt.title('Vistor Type Conversion Rate', color='black', fontsize=16, loc='center',
plt.xticks(rotation=0)

# Add legend
plt.legend()


#Add percentages on top of each bar
for index, value in enumerate(visitor_rev_pct['Buyer']):
    plt.text(index, 5, s=f'{round(value,2)}%', ha='center', va='bottom', color='Whi
```

```
# Display the chart
plt.show()
```



In [ ]:
```
visitor_rev_pct
```

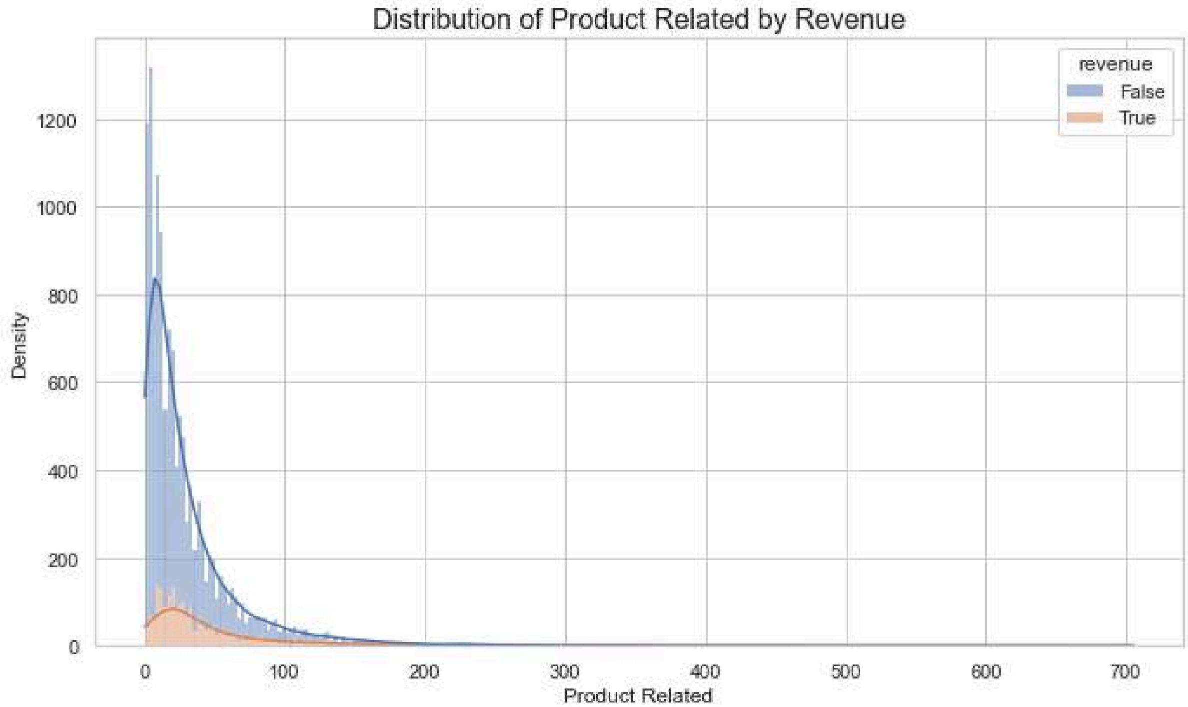
Out[ ]:

	Buyer	Non-Buyer
visitortype		
New_Visitor	24.649860	75.350140
Other	17.977528	82.022472
Returning_Visitor	14.017341	85.982659

Sesi dilakukan mayoritas oleh Returning Visitors. namun, persentase Buyer pada Returning Visitors secara signifikan lebih sedikit dari Non-Buyers. pada New visitor, proporsi Buyers mendekati proporsi Non-Buyers. hal ini menunjukan bahwa

Returning Visitor lebih banyak sesi kunjungannya, tetapi New Visitors mempunyai purchase rate yang lebih tinggi `24.65%`.

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

# Create a distribution plot for 'productrelated'
plt.figure(figsize=(10, 6))
sns.histplot(data=raw_ecommerce, x='productrelated', hue='revenue', kde=True, multi

plt.xlabel('Product Related')
plt.ylabel('Density')
plt.title('Distribution of Product Related by Revenue', fontsize=16)

#plt.legend(title='Revenue')

plt.show()
```

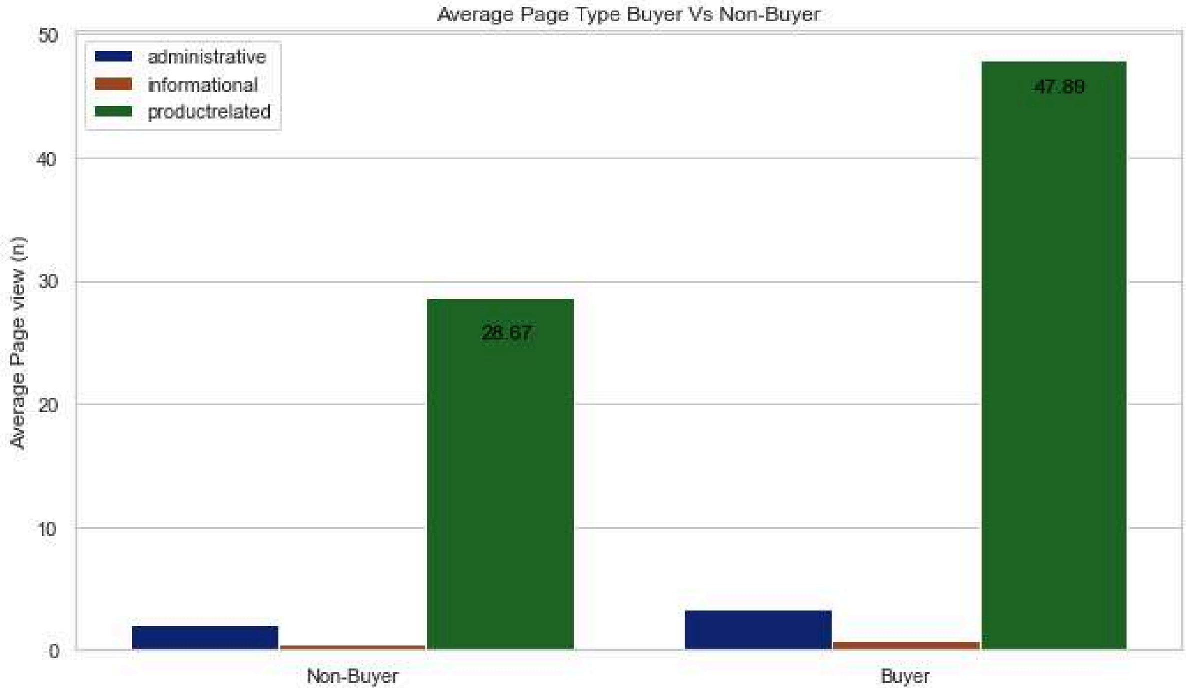


In [ ]:
```
#grouping df based on revenue and agregating page type mean
page_cnt = raw_ecommerce.groupby(['revenue'])[['administrative','informational','pr

#change revenue column value
page_cnt.loc[page_cnt['revenue']==True, 'revenue'] = 'Buyer'
page_cnt.loc[page_cnt['revenue']==False, 'revenue'] = 'Non-Buyer'
page_cnt =  page_cnt[['revenue','productrelated','administrative','informational']]

page_cnt
```

Out[ ]:

	revenue	productrelated	administrative	informational
0	Non-Buyer	28.676632	2.103486	0.447065
1	Buyer	47.895916	3.393879	0.780876

In [ ]:
```
#creating melted df for visualization
melted_pagetype = page_cnt.melt(id_vars='revenue', value_vars=['administrative','ir
                            var_name='PageType', value_name='Value')

#plotting
plt.figure(figsize=(10, 6))
sns.barplot(data=melted_pagetype, x='revenue', y='Value', hue='PageType', palette='
plt.xlabel('')
plt.ylabel('Average Page view (n)')
plt.legend(loc='upper left')
plt.title('Average Page Type Buyer Vs Non-Buyer')

#adding text
plt.text(0+.33, 25, '28.67', ha='right', va='bottom', color='Black')
plt.text(1+.33, 45, '47.89', ha='right', va='bottom', color='Black')
plt.show()
```



In [ ]:
```
melted_pagetype
```

Out[ ]:

	revenue	PageType	Value
0	Non-Buyer	administrative	2.103486
1	Buyer	administrative	3.393879
2	Non-Buyer	informational	0.447065
3	Buyer	informational	0.780876
4	Non-Buyer	productrelated	28.676632
5	Buyer	productrelated	47.895916

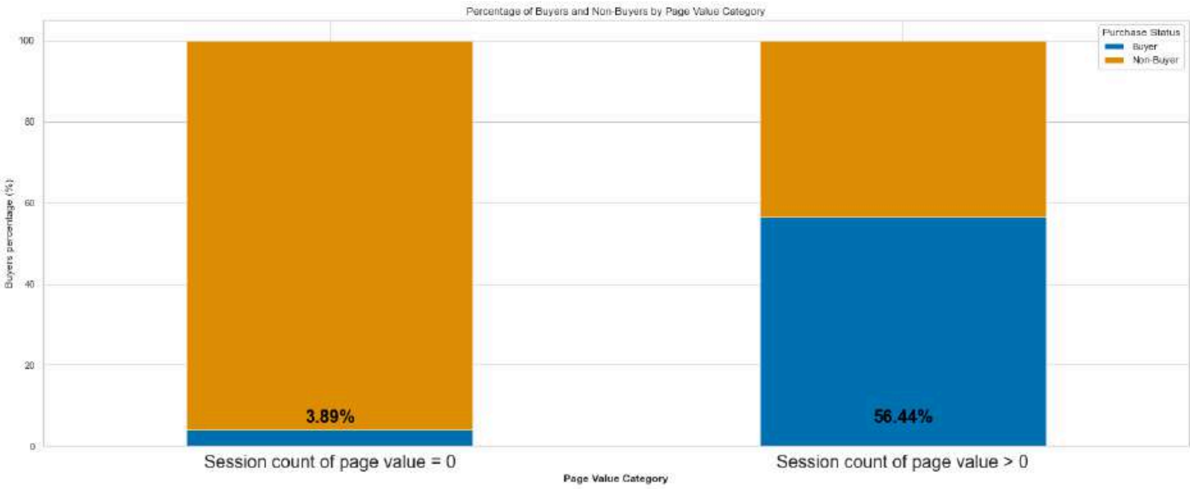
Barplot menunjukan bahwa pengunjung yang memutuskan untuk melakukan pembelian **Buyer**, memiliki nilai rata-rata yang lebih tinggi dari **Non-Buyer**. dalam melihat halaman productrelated

In [ ]:
```
pg_val_rev_true = raw_ecommerce[(raw_ecommerce['pagevalues'] >0) & (raw_ecommerce['
pg_nonval_rev_true = raw_ecommerce[(raw_ecommerce['pagevalues'] == 0) & (raw_ecomme
pg_val_rev_false = raw_ecommerce[(raw_ecommerce['pagevalues'] >0) & (raw_ecommerce[
pg_nonval_rev_false = raw_ecommerce[(raw_ecommerce['pagevalues'] == 0) & (raw_ecomm

# Creating dictionary
pg_rev_data = {
    'Session count of page value = 0': [pg_nonval_rev_true, pg_nonval_rev_false],
    'Session count of page value > 0': [pg_val_rev_true, pg_val_rev_false]}

# Creating the DataFrame
pg_rev = pd.DataFrame(pg_rev_data, index=['Buyer', 'Non-Buyer'])
pg_rev
```

Out[ ]:

	Session count of page value = 0	Session count of page value > 0
Buyer	392	1616
Non-Buyer	9691	1247

In [ ]:
```
# Calculate the percentages across the columns
pg_rev_percent = (pg_rev.div(pg_rev.sum(axis=0), axis=1) * 100).T

# Plotting
colors = sns.color_palette('colorblind')[0:2]
pg_rev_percent.plot(kind='bar', stacked=True,color=colors)

# Adjusting the legend to the upper right
plt.legend(title='Purchase Status', loc='upper right')

# Rotating x-axis labels to horizontal
plt.xticks(rotation=0)

# Adding labels and title
plt.xlabel('Page Value Category',fontweight='bold')
plt.ylabel('Buyers percentage (%)')
plt.title('Percentage of Buyers and Non-Buyers by Page Value Category')
plt.xticks(fontsize=20)

#adding percentage
for index, value in enumerate(pg_rev_percent['Buyer']):
    plt.text(index, 5, s=f'{round(value,2)}%', ha='center', va='bottom', color='Bla

# Show the plot
plt.show()
```



In [ ]:
```
pg_rev_percent
```

Out[ ]:

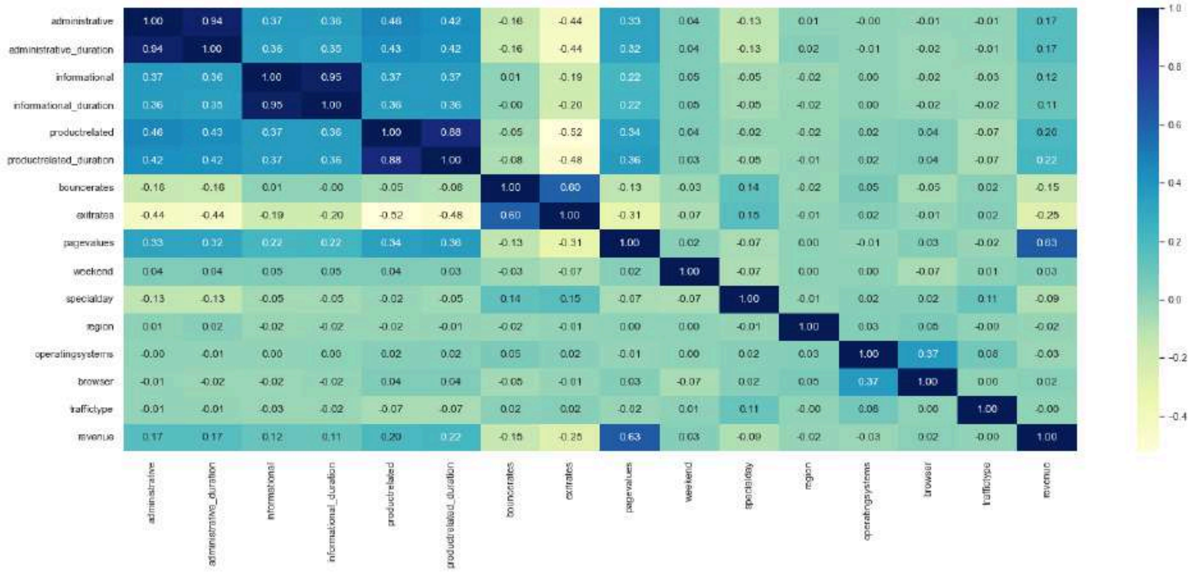
	Buyer	Non-Buyer
Session count of page value = 0	3.887732	96.112268
Session count of page value > 0	56.444289	43.555711

Dalam pembelian ketika session melibatkan pagevalues > 0 purchase rate tinggi `56.44%`. Sebaliknya, sesi dengan pagevalues nol menunjukkan purchase rate yang lebih rendah `3.88%`.

In [ ]:
```
kor = nums +['weekend', 'specialday', 'region', 'operatingsystems', 'browser', 'tra
```

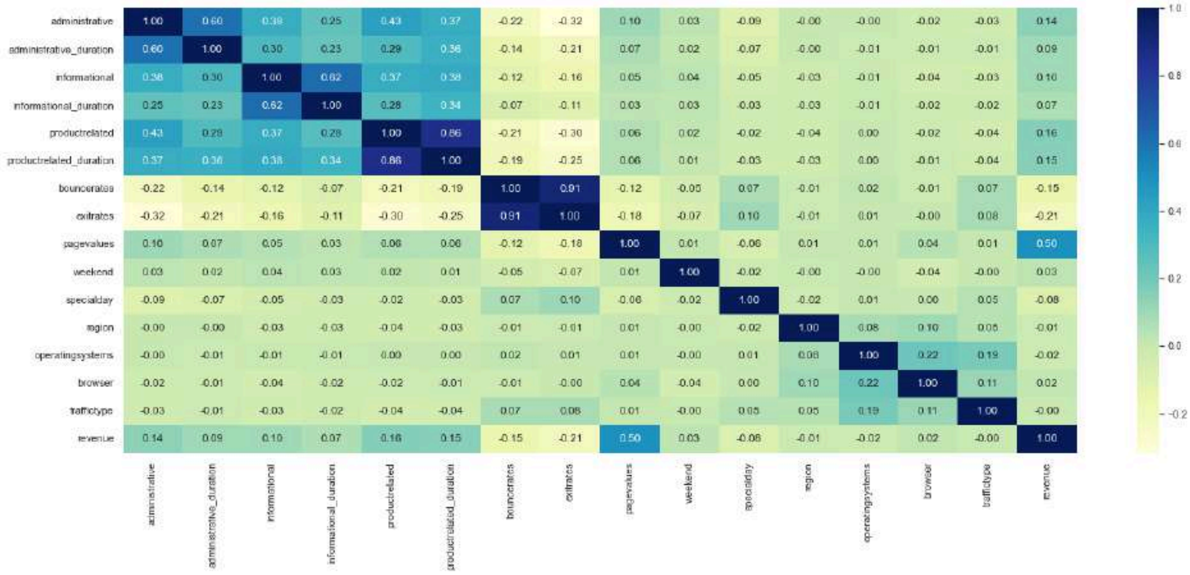
In [ ]:
```
#spearman correlation method
sns.heatmap(raw_ecommerce[kor].corr(method='spearman'), cmap='YlGnBu',annot=True,fm
```

Out[ ]: <AxesSubplot:>




```
In [ ]: #pearson correlation method
sns.heatmap(raw_ecommerce[kor].corr(), cmap='YlGnBu',annot=True,fmt='.2f')
```

Out[ ]: <AxesSubplot:>



3. Multivariate Analysis Insight

Lakukan multivariate analysis (seperti correlation heatmap dan category plots, sesuai yang diajarkan di kelas). Tuliskan hasil observasinya, seperti:

- A. Bagaimana korelasi antara masing-masing feature dan label. Kira-kira feature mana saja yang paling relevan dan harus dipertahankan?
- B. Bagaimana korelasi antar-feature, apakah ada pola yang menarik? Apa yang perlu dilakukan terhadap feature itu?

Tuliskan juga jika memang tidak ada feature yang saling berkorelasi

---

3A.

fitur :

- productrelated_duration

- administrative
- exitrates
- pagevalues

memiliki korelasi dengan target

`pagevalues` menjadi fitur yang memiliki korelasi sangat relevan dengan target (0.63)

3B.

berdasarkan hasil korelasi heatmap yang ditampilkan, terdapat korelasi yang tinggi antara fitur :

- productrelated dengan productrelated_duration (0.88)
- administrative dengan administrative_duration (0.94)
- informational dengan informational_duration (0.95)
- bounce_rates dengan exitrates (0.60)
- operatingsystem dengan browser (0.37)

maka antara salah satu fitur yang berkorelasi tinggi, akan di drop berdasarkan korelasi yang rendah terhadap target **revenue**.

fitur **pagevalues** memiliki korelasi yang tinggi/*relevan* terhadap target. sebesar (0.63)

ada kemungkinan fitur month dan visitortype berkorelasi tinggi terhadap target, maka perlu encoding untuk tahap preprocessing dan melihat korelasinya

4. Business Insight & Reccomendation

Insight

- Region 1 memiliki pengunjung paling banyak diantara region lainnya. akan tetapi revenue rate region 2 `16.64%` menjadi paling tinggi diantara region lainnya.

- Kunjungan user pada platform, yang menghasilkan revenue didominasi pada bulan November `25,48%` Revenue Rate, Sementara bulan Februari memiliki kunjungan yang menghasilkan revenue yang paling sedikit `1.57%` Revenue Rate (3 buyer).

- Bulan May memiliki kunjungan yang paling banyak diantara yang lain terdapat total kunjungan 3533 akan tetapi, hanya 379 dari total kunjungan yang menghasilkan revenue.

- Kunjungan user pada weekday lebih tinggi dari weekend tetapi revenue rate weekend > weekday `17.5% /14.9%`

- Sesi dilakukan mayoritas oleh Returning Visitors. namun, persentase Buyer pada Returning Visitors secara signifikan lebih sedikit dari Non-Buyers. pada New visitor, proporsi Buyers mendekati proporsi Non-Buyers. hal ini menunjukan bahwa Returning Visitor lebih banyak sesi kunjungannya, tetapi New Visitors mempunyai purchase rate yang lebih tinggi `24.65%`.

- Pengunjung yang memutuskan untuk melakukan pembelian **Buyer**, memiliki nilai rata-rata yang lebih tinggi dari **Non-Buyer**. dalam melihat halaman productrelated `47.89 / 28.67` .
- ketika session melibatkan pagevalues > 0 purchase rate tinggi `56.44%` . Sebaliknya, sesi dengan pagevalues nol menunjukkan purchase rate yang lebih rendah `3.88%` .

Business Recommendation

- untuk region yang masih rendah nilai revenue_rate nya, tim marketing dapat menampilkan halaman web yang memiliki pagavalues > 0, dan juga menampilkan rekomendasi yang relevan dengan halaman web yang yang dikunjungi user (product related). strategi marketing tersebut dapat dilakukan pada weekend, dikarenakan disaat weekend revenue_rate lebih tinggi dibandingkan weekday. maka hal ini dapat membantu meningkatkan revenue platform e-commerce.

Metrics

- Revenue

Data Preprocessing

In [ ]:
```
clean_data = raw_ecommerce.copy()
```

Handle Missing Value

In [ ]:
```
import pandas as pd

def info_missing_value(data):
    """
    Calculate missing data statistics and return the missing data DataFrame along w

    Parameters:
        data (pandas.DataFrame): The input DataFrame to analyze.

    Returns:
        pandas.DataFrame: A DataFrame containing the missing data statistics, inclu
    """

    # Calculate the total count of missing values for each column
    total = data.isna().sum().sort_values(ascending=False)
    percent = (data.isnull().sum() / len(data) * 100).sort_values(ascending=False)

    missing_data = pd.DataFrame(total, columns=['Total'])
    missing_data['Percent'] = percent

    # Remove the percentage calculation for data length
    missing_data = missing_data[missing_data.index != 'Data Length']

    # Add a row for data length
    missing_data = pd.concat([pd.DataFrame([[len(data), None]], columns=['Total', '

    return missing_data
```

```
info_missing_value(clean_data)
```

Out[ ]:

	Total	Percent
Data Length	12946	NaN
productrelated_duration	639	4.935888
administrative_duration	633	4.889541
operatingsystems	524	4.047582
administrative	111	0.857408
bouncerates	74	0.571605
weekend	0	0.000000
visitortype	0	0.000000
traffictype	0	0.000000
region	0	0.000000
browser	0	0.000000
specialday	0	0.000000
month	0	0.000000
pagevalues	0	0.000000
exitrates	0	0.000000
productrelated	0	0.000000
informational_duration	0	0.000000
informational	0	0.000000
revenue	0	0.000000

In [ ]:

```
# handle missing value by filling nan with respective median
null_feature = ['productrelated_duration', 'administrative_duration', 'operatingsys

clean_data[null_feature] = clean_data[null_feature].fillna(clean_data[null_feature]

info_missing_value(clean_data)
```

Out[ ]:

	Total	Percent
Data Length	12946	NaN
administrative	0	0.0
administrative_duration	0	0.0
weekend	0	0.0
visitortype	0	0.0
traffictype	0	0.0
region	0	0.0
browser	0	0.0
operatingsystems	0	0.0
month	0	0.0
specialday	0	0.0
pagevalues	0	0.0
exitrates	0	0.0
bouncerates	0	0.0
productrelated_duration	0	0.0
productrelated	0	0.0
informational_duration	0	0.0
informational	0	0.0
revenue	0	0.0

Handle Duplicate Data

In [ ]:
```
def handle_duplicates(data):
    """
    Handles duplicates in a given DataFrame by dropping them and returns the sum of
    the data length before handling duplicates, and the data length after handling

    Args:
        data: A pandas DataFrame representing the dataset.

    Returns:
        A tuple containing the following elements:
        - duplicates_sum (int): The sum of duplicated data.
        - data_length_before (int): The length of the dataset before handling dupli
        - data_length_after (int): The length of the dataset after handling duplica
    """
    data_length_before = len(data)
    data.drop_duplicates(inplace=True)
    data_length_after = len(data)
    duplicates_sum = data_length_before - data_length_after

    return (f'Duplicate :{duplicates_sum}', f'Origin Length :{data_length_before}',
```

In [ ]:
```
handle_duplicates(clean_data)
```

Out[ ]:
```
('Duplicate :717', 'Origin Length :12946', 'Droped Duplicate Length :12229')
```

In [ ]:
```
clean_data.duplicated().sum()
```

Out[ ]:
```
0
```

Split Data

In [ ]:
```
# split data. features to x, target to y
#x = clean_data.drop(columns='revenue').copy()
#y = clean_data['revenue'].copy()
```

Train Test Split

In [ ]:
```
# splitting data to train and test data
from sklearn.model_selection import train_test_split

# x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.2, random_st

train_size = 0.8
train_index = int(len(clean_data) * train_size)
train_df, test_df = clean_data.iloc[:train_index], clean_data.iloc[train_index:]

print(train_df.shape)
print(test_df.shape)
```

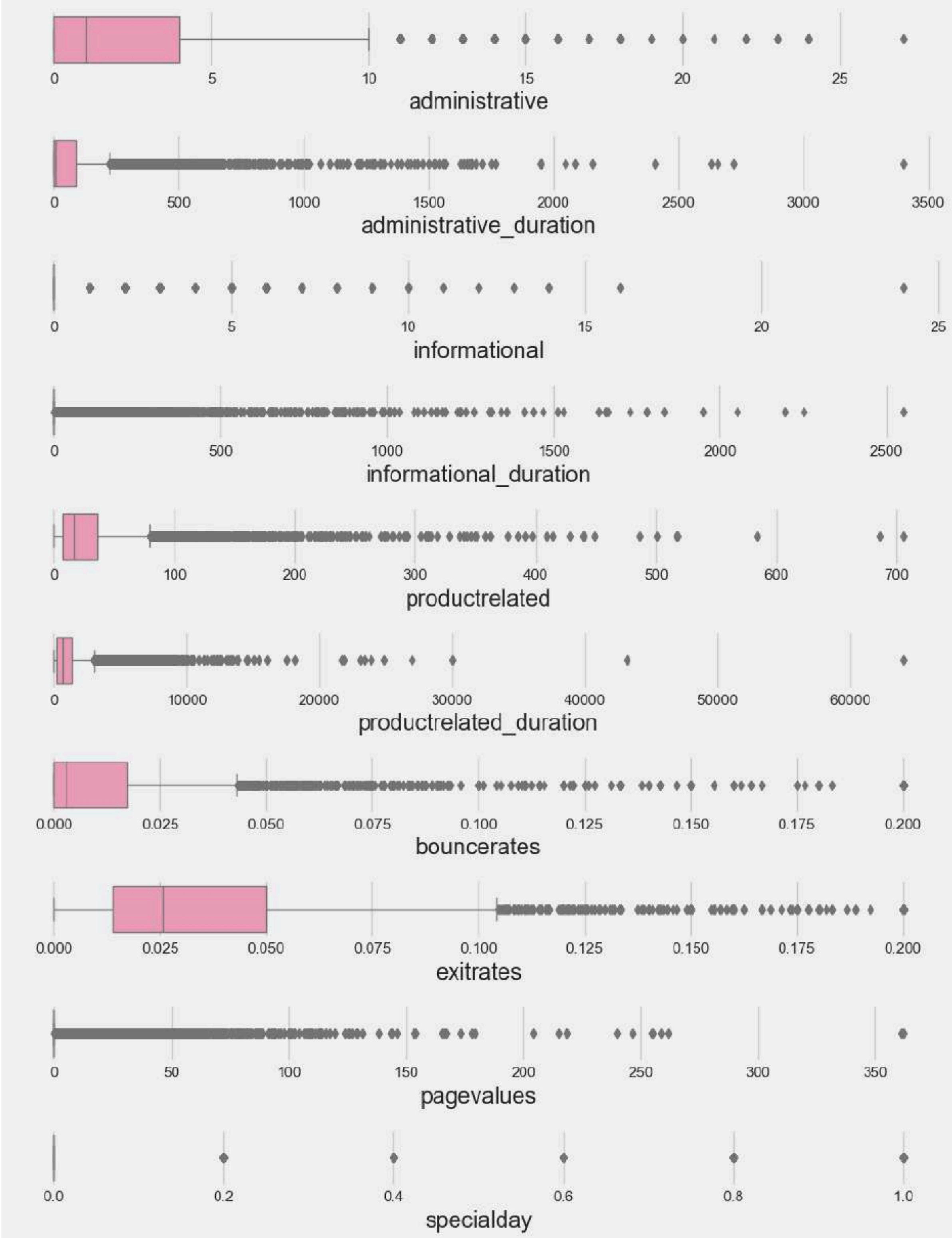
```
(9783, 18)
(2446, 18)
```

Handle outlier

In [ ]:
```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 9783 entries, 0 to 9859
Data columns (total 18 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   administrative           9783 non-null   float64
 1   administrative_duration  9783 non-null   float64
 2   informational            9783 non-null   int64  
 3   informational_duration   9783 non-null   float64
 4   productrelated           9783 non-null   int64  
 5   productrelated_duration  9783 non-null   float64
 6   bouncerates              9783 non-null   float64
 7   exitrates                9783 non-null   float64
 8   pagevalues               9783 non-null   float64
 9   specialday               9783 non-null   float64
 10  month                    9783 non-null   object 
 11  operatingsystems         9783 non-null   float64
 12  browser                  9783 non-null   int64  
 13  region                   9783 non-null   int64  
 14  traffictype              9783 non-null   int64  
 15  visitortype              9783 non-null   object 
 16  weekend                  9783 non-null   bool   
 17  revenue                  9783 non-null   bool   
dtypes: bool(2), float64(9), int64(5), object(2)
memory usage: 1.3+ MB
```

In [ ]:
```
# separating data type features
numerical_feature   = ['administrative','administrative_duration','informational','
                       'productrelated','productrelated_duration','bouncerates','ex

categorical_feature = ['month','operatingsystems','browser','region','traffictype',

# defining function to check boxplot of the data
def plot_outliner(data):
    plt.figure(figsize=(10, 15))
    plt.subplots_adjust(hspace=0.5)
    for i in range(0, len(numerical_feature)):
        plt.subplot(12, 1, i+1)
        sns.boxplot(x=data[numerical_feature[i]], color='#f78fb3', linewidth=1)
        plt.xlabel(numerical_feature[i])
    plt.tight_layout()

plot_outliner(train_df)
```



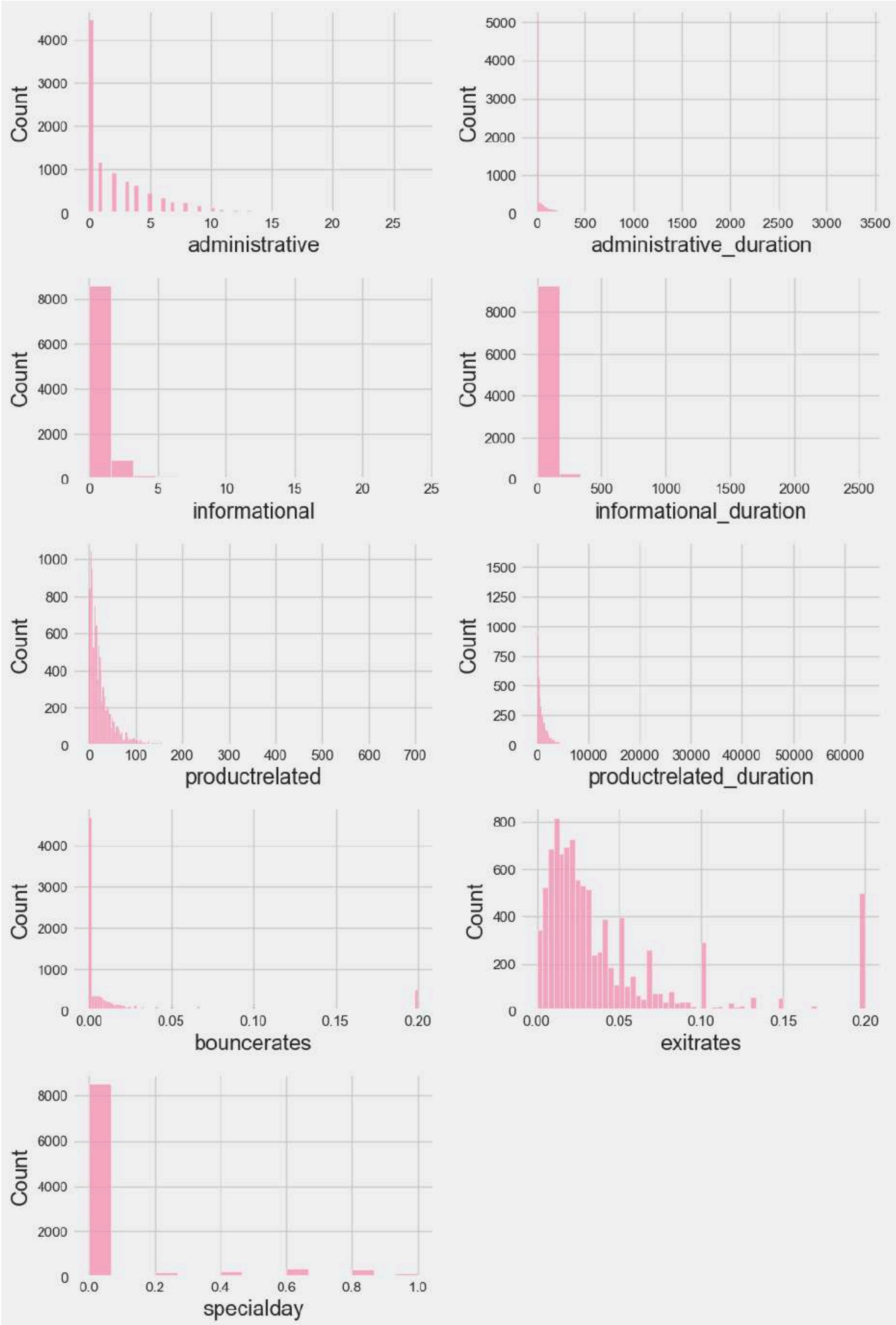
All features in `x_train` that are numerical type, have outliers. for `pagevalues` outliers wont be handled, because it's a special case it might have useful information for model

In [ ]:
```
numerical_feature
```

Out[ ]:
```
['administrative',
 'administrative_duration',
 'informational',
 'informational_duration',
 'productrelated',
 'productrelated_duration',
 'bouncerates',
 'exitrates',
 'pagevalues',
 'specialday']
```

In [ ]:
```
outlier_features = ['administrative',
 'administrative_duration',
 'informational',
 'informational_duration',
 'productrelated',
 'productrelated_duration',
 'bouncerates',
 'exitrates',
 'specialday']
```

In [ ]:
```
#Distribution
def plot_distribution(data):
    plt.figure(figsize=(10, 20))
    features = outlier_features
    for i in range(0, len(features)):
        plt.subplot(7, 2, i+1)
        sns.histplot(x=data[features[i]], color='#f78fb3')
        plt.xlabel(features[i])

    plt.tight_layout()

plot_distribution(train_df)
```



ZSCORE Outlier Handling

In [ ]:

```
# zscore handling

import numpy as np
from scipy import stats

def outlier_zscore(data,feature,threshold=3):
```

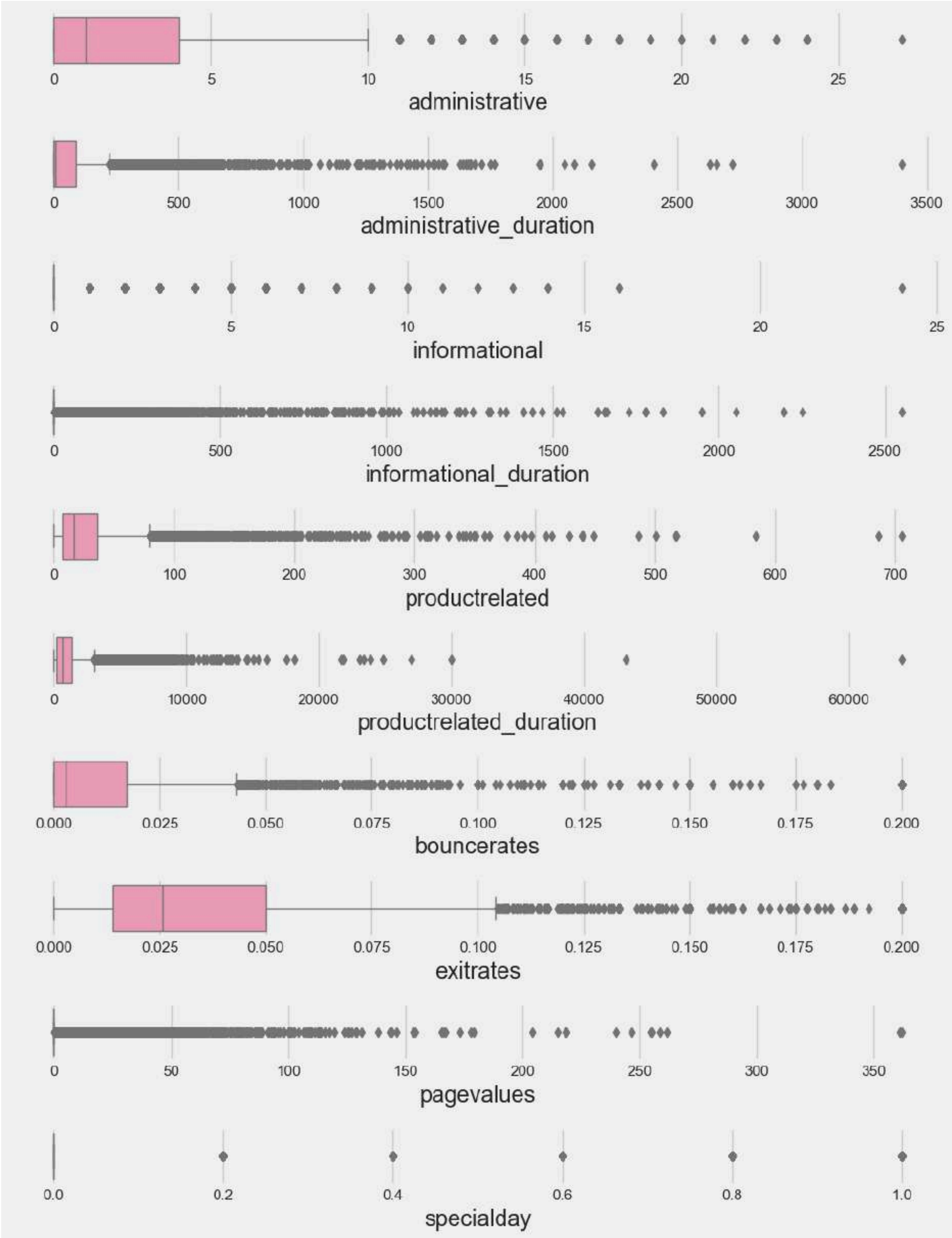
```
    for f in feature:
        z_score          = np.abs(stats.zscore(data[f]))
        filtered_outlier = (z_score < threshold)
        data             = data[filtered_outlier]

    return data

# Before handling outliers
data_length_before = len(train_df)

# Handle outliers
data_after_zs = outlier_zscore(train_df, outlier_features, threshold=2)

# After handling outliers
data_length_after = len(data_after_zs)

print("Data length before handling outliers:", data_length_before)
print("Data length after handling outliers:", data_length_after)
```

```
Data length before handling outliers: 9783
Data length after handling outliers: 6176
```

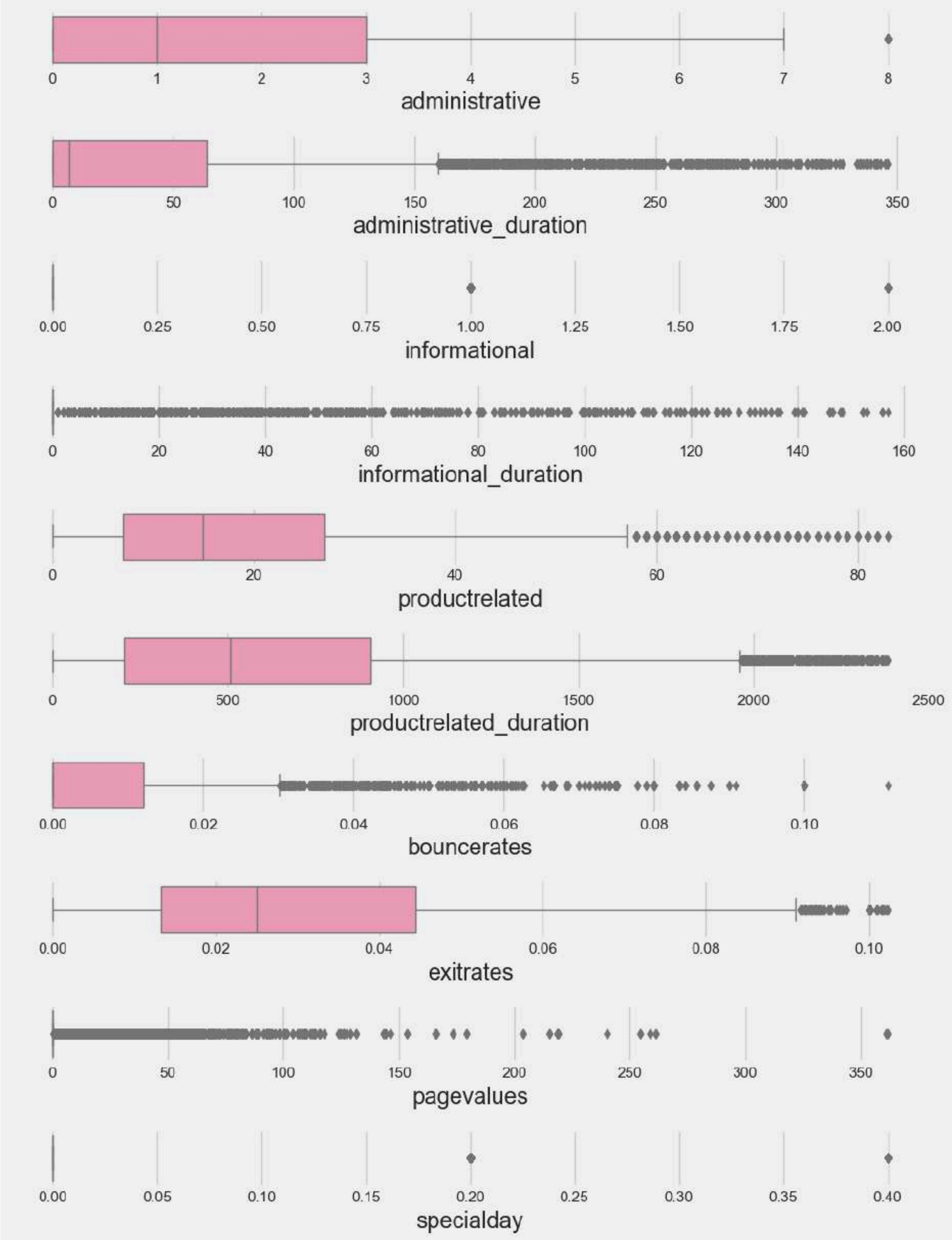
Outlier Handling Results

In [ ]:

```
data_processed = data_after_zs

# checking the results
plot_outliner(train_df)
plot_outliner(data_processed)
```



0 5 10 15 20 25
administrative
0 500 1000 1500 2000 2500 3000 3500
administrative_duration
0 5 10 15 20 25
informational
0 500 1000 1500 2000 2500
informational_duration
0 100 200 300 400 500 600 700
productrelated
0 10000 20000 30000 40000 50000 60000
productrelated_duration
0.000 0.025 0.050 0.075 0.100 0.125 0.150 0.175 0.200
bouncerates
0.000 0.025 0.050 0.075 0.100 0.125 0.150 0.175 0.200
exitrates
0 50 100 150 200 250 300 350
pagevalues
0.0 0.2 0.4 0.6 0.8 1.0
specialday



Feature Encoding

Encode Categorical Feature after train_test_split. to prevent data leakage

In [ ]:
```
categorical_feature
```

Out[ ]:
```
['month',
 'operatingsystems',
 'browser',
 'region',
 'traffictype',
 'visitortype',
 'weekend']
```

In [ ]:
```
encode_features = categorical_feature

train_processed  = pd.get_dummies(data_processed, columns=['month','operatingsystem
test_processed   = pd.get_dummies(test_df, columns=['month','operatingsystems','bro

print("Training : ", train_processed.shape)
print("Testing : ", test_processed.shape)
```

```
Training :  (6176, 73)
Testing :  (2446, 57)
```

In [ ]:
```
test_df['traffictype'].nunique()
```

Out[ ]:
```
11
```

In [ ]:
```
data_processed['traffictype'].nunique()
```

Out[ ]:
```
19
```

In [ ]:
```
print(train_processed.shape)
train_processed.info()
```

```
(6007, 74)
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6007 entries, 1 to 9858
Data columns (total 74 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   administrative            6007 non-null   float64
 1   administrative_duration   6007 non-null   float64
 2   informational             6007 non-null   int64  
 3   informational_duration    6007 non-null   float64
 4   productrelated            6007 non-null   int64  
 5   productrelated_duration   6007 non-null   float64
 6   bouncerates               6007 non-null   float64
 7   exitrates                 6007 non-null   float64
 8   pagevalues                6007 non-null   float64
 9   specialday                6007 non-null   float64
 10  weekend                   6007 non-null   int32  
 11  revenue                   6007 non-null   int32  
 12  month_2                   6007 non-null   uint8  
 13  month_3                   6007 non-null   uint8  
 14  month_5                   6007 non-null   uint8  
 15  month_6                   6007 non-null   uint8  
 16  month_7                   6007 non-null   uint8  
 17  month_8                   6007 non-null   uint8  
 18  month_9                   6007 non-null   uint8  
 19  month_10                  6007 non-null   uint8  
 20  month_11                  6007 non-null   uint8  
 21  month_12                  6007 non-null   uint8  
 22  operatingsystems_1        6007 non-null   uint8  
 23  operatingsystems_2        6007 non-null   uint8  
 24  operatingsystems_3        6007 non-null   uint8  
 25  operatingsystems_4        6007 non-null   uint8  
 26  operatingsystems_5        6007 non-null   uint8  
 27  operatingsystems_6        6007 non-null   uint8  
 28  operatingsystems_7        6007 non-null   uint8  
 29  operatingsystems_8        6007 non-null   uint8  
 30  browser_1                 6007 non-null   uint8  
 31  browser_2                 6007 non-null   uint8  
 32  browser_3                 6007 non-null   uint8  
 33  browser_4                 6007 non-null   uint8  
 34  browser_5                 6007 non-null   uint8  
 35  browser_6                 6007 non-null   uint8  
 36  browser_7                 6007 non-null   uint8  
 37  browser_8                 6007 non-null   uint8  
 38  browser_10                6007 non-null   uint8  
 39  browser_11                6007 non-null   uint8  
 40  browser_12                6007 non-null   uint8  
 41  browser_13                6007 non-null   uint8  
 42  region_1                  6007 non-null   uint8  
 43  region_2                  6007 non-null   uint8  
 44  region_3                  6007 non-null   uint8  
 45  region_4                  6007 non-null   uint8  
 46  region_5                  6007 non-null   uint8  
 47  region_6                  6007 non-null   uint8  
 48  region_7                  6007 non-null   uint8  
 49  region_8                  6007 non-null   uint8  
 50  region_9                  6007 non-null   uint8  
 51  traffictype_1             6007 non-null   uint8  
 52  traffictype_2             6007 non-null   uint8  
 53  traffictype_3             6007 non-null   uint8  
 54  traffictype_4             6007 non-null   uint8  
 55  traffictype_5             6007 non-null   uint8  
 56  traffictype_6             6007 non-null   uint8  
 57  traffictype_7             6007 non-null   uint8  
```

```
 58  trafficytype_8                 6007 non-null   uint8
 59  traffictype_9                 6007 non-null   uint8
 60  traffictype_10                6007 non-null   uint8
 61  traffictype_11                6007 non-null   uint8
 62  traffictype_12                6007 non-null   uint8
 63  traffictype_13                6007 non-null   uint8
 64  traffictype_14                6007 non-null   uint8
 65  traffictype_15                6007 non-null   uint8
 66  traffictype_16                6007 non-null   uint8
 67  traffictype_17                6007 non-null   uint8
 68  traffictype_18                6007 non-null   uint8
 69  traffictype_19                6007 non-null   uint8
 70  traffictype_20                6007 non-null   uint8
 71  visitortype_New_Visitor       6007 non-null   uint8
 72  visitortype_Other             6007 non-null   uint8
 73  visitortype_Returning_Visitor 6007 non-null   uint8
dtypes: float64(8), int32(2), int64(2), uint8(62)
memory usage: 926.9 KB
```

Feature Transformation

Transform numerical feature

In [ ]:
```
# Standardization
from sklearn.preprocessing import StandardScaler
ss = StandardScaler()

for n in numerical_feature:
    scaler_train = ss.fit(train_processed[[n]])
    scaler_test  = ss.fit(test_processed[[n]])
    train_processed[n] = scaler_train.transform(train_processed[[n]])
    test_processed[n] = scaler_test.transform(test_processed[[n]])
```

In [ ]:
```
train_processed.describe()
```

Out[ ]:

	administrative	administrative_duration	informational	informational_duration	productrelate
count	6007.000000	6007.000000	6007.000000	6007.000000	6007.0000(
mean	-0.265440	-0.266467	-0.311049	-0.235908	-0.4463
std	0.651270	0.316383	0.439808	0.112935	0.2579!
min	-0.687721	-0.460119	-0.442959	-0.264269	-0.7469
25%	-0.687721	-0.460119	-0.442959	-0.264269	-0.6187
50%	-0.687721	-0.460119	-0.442959	-0.264269	-0.5088
75%	-0.105254	-0.165781	-0.442959	-0.264269	-0.3439
max	4.554487	0.962025	4.613267	0.619286	2.3670!

8 rows × 74 columns

Spliting Train and Test Label

In [ ]:
```
x_train = train_processed.drop('revenue', axis=True)
x_test = test_processed.drop('revenue', axis=True)
y_train = train_processed['revenue']
y_test = test_processed['revenue']
```

```
print(x_train.shape, y_train.shape)
print(x_test.shape, y_test.shape)
```

```
(6176, 72) (6176,)
(2446, 56) (2446,)
```

Feature Selection

In [ ]:
```
from sklearn.feature_selection import SelectKBest, mutual_info_classif
```

In [ ]:
```
def select_best_features(X, y, k=10):
    """
    Selects the top 'k' features from a dataframe based on their relevance to the t

    Parameters:
        df (DataFrame)     : The input dataframe containing the features and target
        target_column (str): The name of the target variable column.
        k (int, optional)  : The number of top features to select. Defaults to 10.

    Returns:
        list: A list of the best feature(s) based on their relevance to the target
    """
    # Perform feature selection using SelectKBest and f_classif
    selector = SelectKBest(score_func=mutual_info_classif, k=k)
    selector.fit(X, y)

    # Get the indices of the k best features
    best_feature_indices = selector.get_support(indices=True)

    # Get the names of the best features
    best_features = list(X.columns[best_feature_indices])

    return best_features
```

In [ ]:
```
selected_feature = select_best_features(x_train, y_train, k=10)
selected_feature
```

Out[ ]:
```
['administrative_duration',
 'productrelated_duration',
 'bouncerates',
 'exitrates',
 'pagevalues',
 'month_Oct',
 'operatingsystems_3.0',
 'browser_7',
 'region_8',
 'traffictype_9']
```

In [ ]:
```
def select_best_features_fc(X, y):
    class_labels = np.unique(y)
    fisher_scores = []

    for feature in X.columns:
        feature_values = X[feature]
        feature_fisher_score = 0

        for label in class_labels:
            class_mask = (y == label)
            feature_values_class = feature_values[class_mask]
            mean_diff = np.abs(np.mean(feature_values_class) - np.mean(feature_valu
            std_within_class = np.std(feature_values_class)
```

```
            if std_within_class == 0:  # Avoid division by zero
                continue

            fisher_score_class = (mean_diff ** 2) / std_within_class
            feature_fisher_score += fisher_score_class

        fisher_scores.append(round(feature_fisher_score, 2))

    return fisher_scores
```

In [ ]:
```
fisher_score_result = select_best_features_fc(x_train, y_train)
```

In [ ]:
```
feature_fc = pd.DataFrame({'Feature': x_train.columns, 'Fisher Score': fisher_score
feature_fc.sort_values(by='Fisher Score', ascending=False, inplace=True)
print(feature_fc.head(10))
```

```
                        Feature  Fisher Score
8                    pagevalues          1.15
7                     exitrates          0.46
6                   bouncerates          0.21
72  visitortype_Returning_Visitor        0.05
70        visitortype_New_Visitor        0.05
51                 traffictype_2          0.05
52                 traffictype_3          0.03
0                administrative          0.03
5         productrelated_duration        0.03
23           operatingsystems_3          0.02
```

In [ ]:
```
#Feature Selection Using Mutual Info Methode
x_train_selected = x_train[selected_feature]
```

Class imbalances

In [ ]:
```
# pip install imbalanced-learn
from imblearn.over_sampling import SMOTE

# SMOTE (Synthetic Minority Over-sampling Technique)
smote = SMOTE(random_state=42)
X_train_smote, y_train_smote = smote.fit_resample(x_train, y_train)
```

In [ ]:
```
y_train.value_counts()
```

Out[ ]:
```
revenue
False    5267
True      909
Name: count, dtype: int64
```

In [ ]:
```
len(x_train)
```

Out[ ]:
```
6176
```

In [ ]:
```
y_train_smote.value_counts()
```

Out[ ]:
```
revenue
False    5267
True     5267
Name: count, dtype: int64
```

In [ ]:
```
len(y_train_smote)
```

Out[ ]:
```
10534
```

In [ ]:

```

```