comportement des clients dans le temps. Le choix de Python a été motivé par la volonté d'effectuer rapidement l'analyse, tout en démontrant notre polyvalence technique et notre capacité à générer des insights significatifs grâce à des méthodes analytiques avancées.

---

5 Executive Summary (English)

This report aims to provide a clear, and competent analysis of the online retail data, focusing on sales patterns, product popularity, and revenue insights, alongside technical considerations for report optimization.

We present a detailed analysis of the sales recorded between December 1, 2010, and December 9, 2011, by a UK-based non-store online retail specializing in unique all-occasion gifts. With a dataset featuring over 541,909 instances across six features, our exploration focuses on sales dynamics, revenue generation, product popularity, customer behavior, and geographical distribution of sales. The analysis was facilitated by Power BI, employing specific visualizations and data manipulation techniques to uncover trends and insights that can drive business strategies and operational efficiencies.

6 Key components of the Power BI dashboard

The dashboard incorporates a selective range of visuals to maintain clarity and performance, adhering to best practices with no more than eight visuals per page. These visuals include:

- Weekday filters to observe sales patterns across different days of the week.
- A ranking of top revenue-generating products, highlighting the `Regency Cakestand 3 Tier` as the highest revenue product contributor.
- Hourly product sales volume distribution, pinpointing peak sales times between 10h and 15h, with a peak at 12h.
- Global sales distribution by country, showcasing the UK as the primary market followed by significant contributions from the European Union.

7 Data Insights

The `Regency Cakestand 3 Tier` emerges as the top revenue generator, while `Paper Craft, Little Birdie` leads in unit sales, primarily in December 2011. This suggests a seasonal influence on product popularity, with significant sales concentration around the holiday season.

In terms of sales volume distribution, the concentration of sales between 10 AM and 3 PM, peaking at 12 PM, could indicate a consumer tendency to shop during lunch breaks or midday free time, highlighting potential timing for targeted promotions or marketing efforts. Again, the spike in sales and quantities in November, preceding Christmas, underscores the critical importance of the holiday season for the retail's business. This suggests the need for strategic stock planning, marketing, and promotional activities to capitalize on this peak shopping period.

When we look also into the Weekday filters, the absence of sales on Saturdays might reflect operational aspects of the retail or the data collection process. It warrants further investigation into operational hours or data logging practices.

In terms of the global sales distribution, while the UK dominates the sales distribution, notable markets within the European Union, including France, Germany, and the Netherlands, signify the importance of the EU as a secondary market. This geographic insight can guide market expansion strategies and localized marketing campaigns.

Finally, we have a total sales revenue of £10419165.38. Using the historical conversion rate from sterling to euros on October 31, 2011, of 0.8731, the total sales revenue in euros is approximately €9096973.73.

8 Final comments about the Power BI dashboard and conclusions

The analysis of the online retail dataset through Power BI visualizations offers valuable insights into sales trends, customer preferences, and market dynamics. The strategic use of visuals, optimization of report performance, and detailed exploration of data narratives provide a foundation for informed decision-making and strategic planning.

In conclusion, the key to boosting sales performance for our UK-based online retail platform lies in several strategic measures. These involve harnessing the power of data from seasonal sales patterns, daily sales volumes, and regional market studies.

Critical to this strategy is capitalizing on increased consumer demand during the festive season, specifically for high-selling and high-profit items such as the `Regency Cakestand 3 Tier` and `Paper Craft, Little Birdie`. This can be achieved by ensuring ample stock availability and initiating targeted marketing drives during these high-demand periods.

Furthermore, fine-tuning promotional schedules to coincide with the busiest shopping hours, especially between 10 AM and 3 PM, can maximize sales opportunities. Lastly, considering strategic growth or customized marketing initiatives in emerging European markets like France, Germany, and the Netherlands could significantly enhance sales performance. Therefore, through these measures, we can drive a substantial increase in sales.

8.1 Appendix A: Data Cleaning and Pre-Processing:

In our analysis of the online retail dataset, maintaining the quality and relevance of the data was vital. This was achieved through comprehensive data cleaning and pre-processing procedures. Even though Power BI's Power Query is a powerful tool for data manipulation, we chose to use Python, specifically the Pandas library, for this stage of the process for several reasons.

Python provides a significant advantage in terms of flexibility and precision, allowing for intricate data processing tasks that exceed the capabilities of point-and-click interfaces. This decision to use Python also underlines our expertise in utilizing a variety of data analysis tools, thereby highlighting our versatility and adaptability in data science. This choice ultimately enabled us to handle the data more efficiently and effectively.

The initial step involved removing duplicate entries to ensure that our analysis was based on unique transaction records. This was followed by identifying and excluding entries with negative Unit Prices. Upon closer inspection, entries marked with an "A" at the beginning of the InvoiceNo were identified as adjustments for bad debt, distinguished from regular transactions and cancellations (marked by a "C"). These entries were deemed irrelevant for a sales performance analysis and thus excluded.

Further scrutiny revealed that 1.72% of transactions were cancellations (InvoiceNo starting with "C") and entries with negative quantities typically represented returned products. Given the scope of the analysis, focused on sales performance and customer behavior, and the relatively small percentage of such transactions, these were also removed. This decision was not made lightly but was justified by the specific objectives of this exercise and the need to streamline the dataset for a "quick" analysis, as requested.

[1]:
```
import pandas as pd # Pandas library for data analysis.

df = pd.read_excel('Online Retail.xlsx', engine='openpyxl') # Opening the Excel␣
 ↪document as the DataFrame df.

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    541909 non-null  object
 1   StockCode    541909 non-null  object
 2   Description  540455 non-null  object
 3   Quantity     541909 non-null  int64
 4   InvoiceDate  541909 non-null  datetime64[ns]
 5   UnitPrice    541909 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      541909 non-null  object
dtypes: datetime64[ns](1), float64(2), int64(1), object(4)
memory usage: 33.1+ MB
```

[2]:
```
df = df.drop_duplicates() # To eliminate duplicates.
df.duplicated().sum() # To check.
```

[2]:
```
0
```

[3]:
```
df.describe()
```

[3]:

	Quantity	UnitPrice	CustomerID
count	536641.000000	536641.000000	401604.000000
mean	9.620029	4.632656	15281.160818
std	219.130156	97.233118	1714.006089
min	-80995.000000	-11062.060000	12346.000000
25%	1.000000	1.250000	13939.000000
50%	3.000000	2.080000	15145.000000
75%	10.000000	4.130000	16784.000000
max	80995.000000	38970.000000	18287.000000

[4]:
```
negative_unitprice_df = df[df['UnitPrice'] < 0] # To check.

# Print:
print(negative_unitprice_df.head())
```

```
       InvoiceNo StockCode      Description  Quantity         InvoiceDate  \
299983   A563186         B  Adjust bad debt         1 2011-08-12 14:51:00
299984   A563187         B  Adjust bad debt         1 2011-08-12 14:52:00

        UnitPrice  CustomerID         Country
299983  -11062.06         NaN  United Kingdom
299984  -11062.06         NaN  United Kingdom
```

[6]:
```
# Remove rows where UnitPrice is negative:
df = df[df['UnitPrice'] >= 0]

# Filter out rows where Quantity is less than 0:
df = df[df['Quantity'] >= 0]
```

[7]:
```
# Check if there are any InvoiceNo starting with 'C' at this point (I think␣
 ↪not):
has_c_start = df['InvoiceNo'].str.startswith('C').any()
print(f"It is {has_c_start} that there are InvoiceNo that start with a 'C'.")
```

```
It is False that there are InvoiceNo that start with a 'C'.
```

[8]:
```
# Indeed, by removing the orders with negative quantities, all the returns or␣
 ↪canceled orders were also removed.
```

In examining product descriptions, unique cases such as entries with non-capitalized letters typically indicated damaged or returned goods. These were excluded to maintain a focus on regular sales transactions. Additionally, entries where the product description was missing (NaN), or with ?, often coincided with missing CustomerIDs and a Unit Price of 0.0. This pattern suggested these entries might represent **free samples**, which were not pertinent to the analysis of sales performance and were therefore removed.

[11]:
```
criteria_df_with_description = df[
    (df['UnitPrice'] == 0.0) &
    (df['CustomerID'].isna()) &
    (~df['Description'].isna())
]

# Extract unique values in the Description column:
unique_descriptions = criteria_df_with_description['Description'].unique()

# Check if descriptions are not capitalized:
non_capitalized_descriptions = [desc for desc in unique_descriptions if not␣
 ↪desc[0].isupper()]
```

```
print(non_capitalized_descriptions)
```

```
['amazon', '4 TRADITIONAL SPINNING TOPS', 'amazon sales', '?', 'found', 'wrongly
sold (22719) barcode', 'rcvd be air temp fix for dotcom sit', 'did  a credit
and did not tick ret', 'adjustment', 'returned', 'mailout ', 'mailout', 'on
cargo order', 'incorrectly credited C550456 see 47', 'to push order througha s
stock was ', 'came coded as 20713', 'alan hodge cant mamage this section',
'dotcom', 'test', 'taig adjust', 'allocate stock for dotcom orders ta', 'add
stock to allocate online orders', 'for online retail orders', 'found box',
'damaged', 'website fixed', 'michel oops', 'wrongly coded 20713', 'wrongly
marked 23343', 'wrongly coded 23343', 'check', 'wrongly marked', 'had been put
aside', 'amazon adjust', 'dotcomstock', 'dotcom adjust', 'check?']
```

[12]:
```
# Check if when Description is NaN, then CustomerID is NaN and UnitPrice is 0.0:
condition_met = df[(df['Description'].isnull()) & (~(df['CustomerID'].isnull()␣
↪& (df['UnitPrice'] == 0.0)))].empty

print(f"It is {condition_met} that whenever Description is NaN, CustomerID is␣
↪also NaN, and UnitPrice is 0.0.")
```

```
It is True that whenever Description is NaN, CustomerID is also NaN, and
UnitPrice is 0.0.
```

Finally, for the purpose of the Power BI dashboard, the data processing included calculating total sales by multiplying quantity and unit price, and deriving additional time features from the InvoiceDate, such as YearMonth, WeekdayName, and Hour, to facilitate deeper analysis.

This comprehensive pre-processing not only ensured the dataset was optimally prepared for analysis but also highlighted the practical benefits of using Python for data manipulation tasks. The clean and refined dataset was then utilized in Power BI for visualization and further analysis. The final table, reflecting these preprocessing steps, can be reviewed within the Power BI document, providing transparency and accessibility to the processed data.

8.2 Appendix B: Cohort Analysis Using Python:

The decision to perform a cohort analysis using Python, specifically leveraging the Pandas and Seaborn libraries, was a strategic choice aimed at complementing the insights gained from the Power BI dashboard. While Power BI possesses the capability to analyze customer behavior over time, the choice to utilize Python was driven by personal preference and the desire to showcase a breadth of data analysis skills. This approach allowed for a detailed exploration of customer retention and spending patterns, offering a perspective on how customers interact with the online retail over time.

8.2.1 Methodology

- Data Preparation: The analysis began by refining the dataset to exclude rows with missing CustomerID values to ensure accuracy in tracking customer transactions over time.

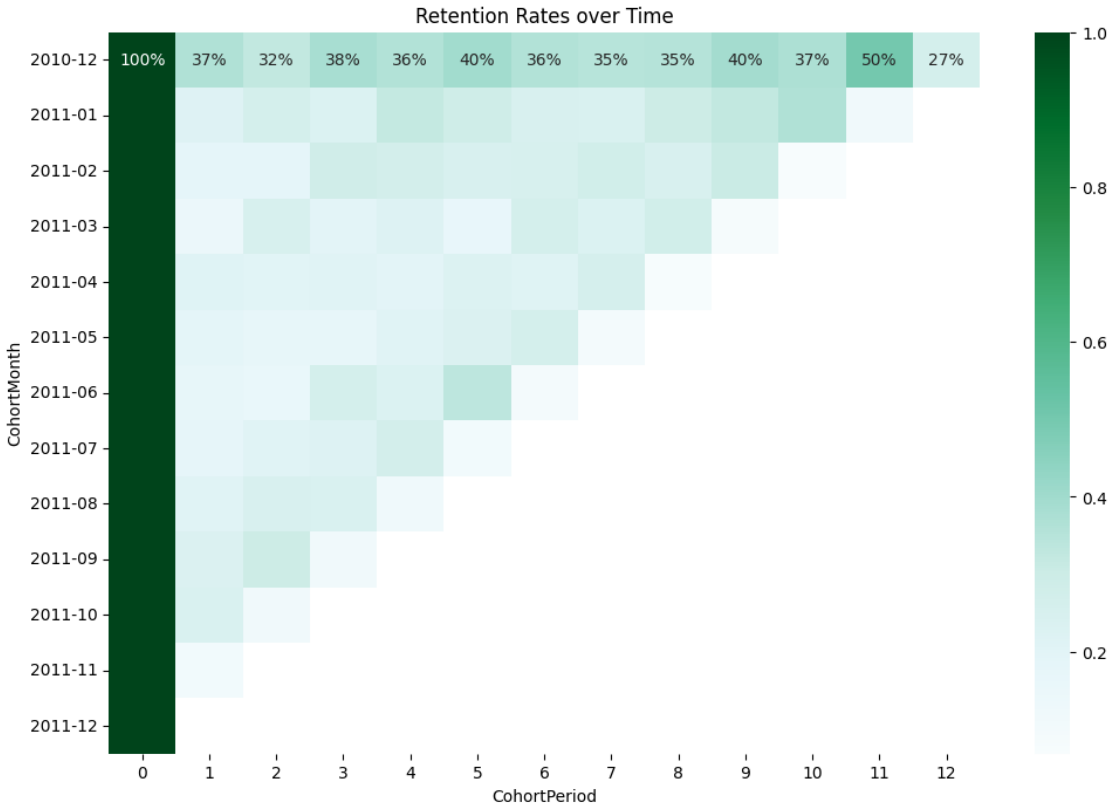
- Defining Cohorts: Cohorts were established based on the month of each customer's first purchase (CohortMonth) and the invoice month of each transaction (InvoiceMonth). This approach enables tracking of customer engagement from their initial purchase.
- Calculating Cohort Periods: A function call `diff_month` was implemented to calculate the difference in months between each transaction and the customer's first purchase (CohortPeriod). This metric is crucial for understanding the timeline of customer engagement post their initial interaction.
- Constructing the Cohort Table: A pivot table was created to count unique customers in each cohort across different periods following their first purchase, facilitating the analysis of customer retention over time.
- Retention Rate Calculation: The retention rate was calculated by dividing the number of active customers in each subsequent period by the initial cohort size. This rate is a key indicator of customer loyalty and engagement over time.
- Visualization: A heatmap was generated to visually represent retention rates over time, with each row representing a cohort defined by the month of the customers' first purchase. This visualization aids in identifying patterns in customer retention and engagement.

[15]:
```
dfRegularCohort = df.copy() # To prepare a df copy.

# Exclude rows where CustomerID is NaN and create a clean copy to work with:
dfRegularCohortClean = dfRegularCohort.dropna(subset=['CustomerID']).copy()

# Proceed with setting InvoiceMonth and CohortMonth:
dfRegularCohortClean['InvoiceMonth'] = dfRegularCohortClean['InvoiceDate'].dt.
 ↪to_period('M')
dfRegularCohortClean['CohortMonth'] = dfRegularCohortClean.
 ↪groupby('CustomerID')['InvoiceMonth'].transform('min')
```

[16]:
```
# Compute cohort period:
def diff_month(d1, d2):
    return (d1.year - d2.year) * 12 + d1.month - d2.month # (Year d1 - Year␣
 ↪d2)*12 month + (Month of Date 1 - Month of Date 2)

dfRegularCohortClean['CohortPeriod'] = dfRegularCohortClean.apply(lambda row:␣
 ↪diff_month(row['InvoiceMonth'].to_timestamp(), row['CohortMonth'].
 ↪to_timestamp()), axis=1)
```

[17]:
```
customer_cohort = dfRegularCohortClean.pivot_table(index='CohortMonth',␣
 ↪columns='CohortPeriod', values='CustomerID', aggfunc='nunique')

# Retention Rate Calculation:
cohort_size = customer_cohort.iloc[:,0]
retention = customer_cohort.divide(cohort_size, axis=0)
```

[19]:
```
# For DataViz:
import matplotlib.pyplot as plt
import seaborn as sns

# Final DataViz:
plt.figure(figsize=(12, 8))
plt.title('Retention Rates over Time')
sns.heatmap(data=retention, annot=True, fmt='.0%', cmap='BuGn')
plt.show()
```



[20]:
```
# Now a table:
customer_cohort = dfRegularCohortClean.pivot_table(index='CohortMonth',
                                                   columns='CohortPeriod',
                                                   values='CustomerID',
                                                   aggfunc='nunique')

# Get the initial cohort size (number of customers in the first month of the␣
 ↪cohort):
cohort_size = customer_cohort.iloc[:, 0]

# Calculate retention by dividing the cohort counts by the cohort size:
```

```
retention = customer_cohort.divide(cohort_size, axis=0)

# Convert the index to a more readable format if necessary:
retention.index = retention.index.strftime('%Y-%m')

# Round the retention rates for easier interpretation:
retention_percentage = retention.round(3) * 100

print(retention_percentage)
```

```
CohortPeriod     0     1     2     3     4     5     6     7     8     9   \
CohortMonth
2010-12       100.0  36.6  32.3  38.4  36.3  39.8  36.3  34.9  35.4  39.5
2011-01       100.0  22.1  26.6  23.0  32.1  28.8  24.7  24.2  30.0  32.6
2011-02       100.0  18.7  18.7  28.4  27.1  24.7  25.3  27.9  24.7  30.5
2011-03       100.0  15.0  25.2  19.9  22.3  16.8  26.8  23.0  27.9   8.6
2011-04       100.0  21.3  20.3  21.0  19.7  22.7  21.7  26.0   7.3   NaN
2011-05       100.0  19.0  17.3  17.3  20.8  23.2  26.4   9.5   NaN   NaN
2011-06       100.0  17.4  15.7  26.4  23.1  33.5   9.5   NaN   NaN   NaN
2011-07       100.0  18.1  20.7  22.3  27.1  11.2   NaN   NaN   NaN   NaN
2011-08       100.0  20.7  24.9  24.3  12.4   NaN   NaN   NaN   NaN   NaN
2011-09       100.0  23.4  30.1  11.4   NaN   NaN   NaN   NaN   NaN   NaN
2011-10       100.0  24.0  11.5   NaN   NaN   NaN   NaN   NaN   NaN   NaN
2011-11       100.0  11.1   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN
2011-12       100.0   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN

CohortPeriod    10    11    12
CohortMonth
2010-12       37.4  50.3  26.6
2011-01       36.5  11.8   NaN
2011-02        6.8   NaN   NaN
2011-03        NaN   NaN   NaN
2011-04        NaN   NaN   NaN
2011-05        NaN   NaN   NaN
2011-06        NaN   NaN   NaN
2011-07        NaN   NaN   NaN
2011-08        NaN   NaN   NaN
2011-09        NaN   NaN   NaN
2011-10        NaN   NaN   NaN
2011-11        NaN   NaN   NaN
2011-12        NaN   NaN   NaN
```

The cohort analysis revealed initial drop-offs in customer engagement following the first month, a common pattern reflecting the challenge of sustaining new customer interest. Subsequent fluctuations in retention rates highlighted changes in customer engagement, potentially aligning with marketing efforts, seasonal trends, or other factors influencing re-engagement. In our Power BI dashboard, you can corroborate critical importance of the holiday season for the retail's business.

Notably, certain cohorts exhibited strong retention rates nearly a year after the first purchase, indicating successful re-engagement strategies or high customer satisfaction levels.

This analysis exemplifies the utility of Python for conducting intricate data analysis tasks that complement the visualizations provided by Power BI. By employing Python, we were able to perform a detailed cohort analysis that enriched our understanding of customer behavior over time. The decision to use Python, in this case, was motivated by the desire for a quick turnaround for the dashboard, coupled with the intention to showcase technical versatility and the ability to derive profound insights through advanced analytical techniques.

[ ]: