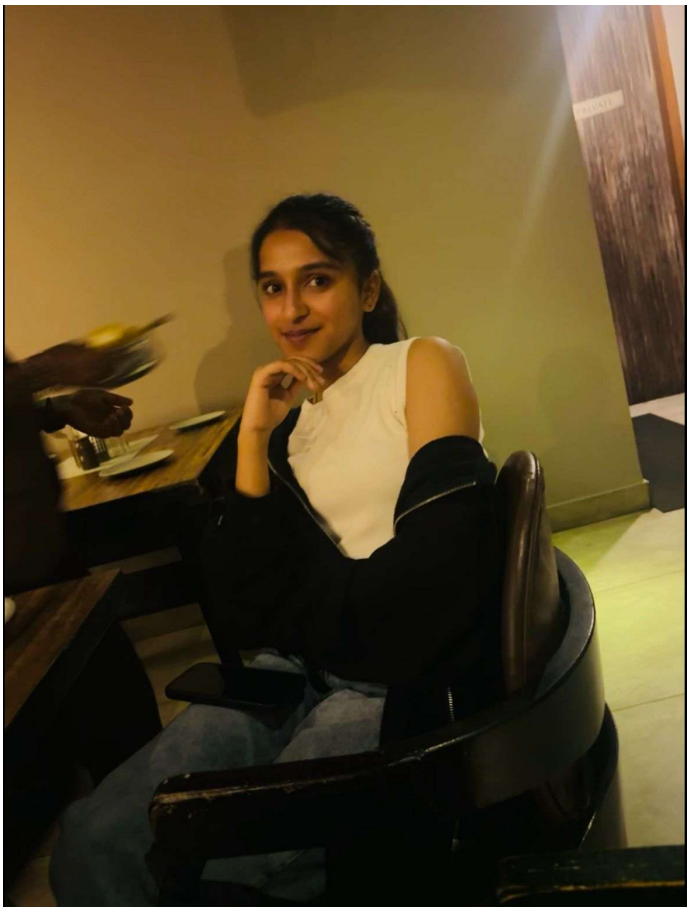
From Data To Decisions: How Analytics Powers Business

Analysis

In Business

Sinchana M



Hi, this is Sinchana, and this book marks the second step in my journey as a young and aspiring analytics enthusiast.

In Part 1, my goal was to introduce high school students to the exciting world of data science — what it means, why it matters, and how it is shaping our future. That book was all about building curiosity and laying the foundation of analytics in a simple and approachable way.

With Part 2, I wanted to go a step further. This time, it's not just about what data science is — it's about how data actually works inside a business. By exploring an Online Retail dataset, I discovered how analytics connects directly to decision-making: identifying valuable customers, spotting sales patterns, and understanding what drives business growth.

This project taught me that data science and business are not separate worlds — they meet in the middle, where numbers turn into insights and insights turn into action. Through this book, I hope to help students see that data is more than just charts and formulas; it's a tool that shapes strategies, decisions, and even the future of businesses.

With every step, my passion for analytics keeps growing, and I wish to inspire others to take their first step too.

Content Page:

Partially jump into Business in Part 2!

Modern businesses don't just toss a coin and hope for the best — they rely on data-driven decisions. Every purchase, every click, every abandoned shopping cart (yes, even the one where you added 10 items and bought none) leaves behind a trail of valuable information.

When treated right, this "digital dust" turns into golden insights that fuel growth, efficiency, and innovation.

This is exactly where Data Science and Business Analysis shake hands. Business analysis tells us what's happening — which processes work, how customers behave, where the money flows. Data Science, on the other hand, dives deeper, crunching mountains of data to spot hidden patterns and even predict the future (not in a crystal-ball way, but close enough).

Together, they help businesses answer burning questions like:

- Which customers are our real gems?
- Which products should we push to maximize profit?
- Which customers are quietly slipping away, and how do we keep them around?

Why Online Retail?

Because retail is where the action is. Think about it — online stores process thousands of daily transactions. That's not just a pile of receipts; it's a treasure chest of insights waiting to be unlocked.

By analyzing this data, businesses can:

- Segment customers with RFM analysis (Recency, Frequency, Monetary).
- Spot loyal fans versus one-time wanderers.
- Plan laser-focused marketing campaigns.
- Forecast sales and keep just the right amount of stock.

In this project, we'll take raw sales data and turn it into stories businesses can actually use. This is where Part 1 ("What is Data Science?") meets Part 2 — seeing Data Science in action, guiding smarter business moves.

Chapter 1: Introduction to Business Data Science

1.1 Why Businesses Use Data Science

Every business today – whether it is a small coffee shop or a global e-commerce gaint – collects huge amounts of data every day.

Examples include:

- Retail store: product sales, customer purchases, seasonal demand
- Banks: transaction history, customer demographics, credit stores.
- Streaming services: user watch history, ratings, recommendations.

Business do not just store this data. They analyze it to make better decisions:

- Which products should we promote?
- Which customers bring the most profit?
- Which time of year gives the highest sales?

This is where data science becomes a powerful tool for business growth.

1.2 Data Science in Action – A Retail Example

Imagine a store that sells thousands of items every week. Without data analysis, the manager might only guess which items sell the most. But with data science, they can actually see:

- The top 10 best-selling products
- Which days of the week bring the highest sales
- Which customers spend the most
- Whether sales increase in December compared to June

By turning raw transaction records into meaningful insights, business can improve sales, reduce waste, and serve customers better.

1.3 Our Purchase Behavior Analysis

In this part of the book, we will do a real-world data science project using a retail dataset.

- Dataset: Online Retail II (UCI ML Repository) – over 1 million real sales transactions from a UK-based online store.
- Tools: Python, NumPy and Pandas (Pandas is used for efficient data manipulation). We also use inbuilt-in modules such as csv and datetime for file reading and date handling.
- Goal: To understand customer purchase behavior and answer important business questions such as:

1. What are the top 5 most purchased products?
2. Which customer spent the most money?
3. What is the average purchase amount per transaction?
4. Do weekends have higher sales than weekdays?
5. Which month had the highest sales?

1.4 What You Will Learn

By the end of this project, you will learn how to:

- Load and clean real business data from a CSV file.
- Handle missing values and prepare data for analysis.
- Use NumPy arrays to calculate sales metrics.
- Answer real business questions using data.
- Interpret results in a way that a store manager or business leader can actually use.

This project will show you how data science is not just about numbers and code—it is about making smarter decisions in the real world.

Chapter 2: Project Setup

In this chapter I'll walk you through the setup for our business project. We're about to dive into real data from an actual online store in the UK!

2.1 Meet Our Dataset

We'll be using the Online Retail II dataset from the UCI Machine Learning Repository.

This dataset contains:

- Invoice → unique number for each purchase
- StockCode → product identifier
- Description → product name
- Quantity → how many units were purchased
- InvoiceDate → when the purchase happened
- Price → price of each item
- Customer ID → unique ID for the customer
- Country → location of the customer

There are over 1 million transaction records in this dataset — plenty to analyze and uncover insights from.

2.2 Why I Chose This Dataset

I wanted this project to feel real. Instead of practicing on artificial or toy data, we'll explore authentic business transactions. Think of yourself as the analyst for this online shop — you'll be answering questions the store owner truly cares about, like:

- Which products sell the most?
- Who are the best customers?
- Do sales increase on weekends?

- Which month brings the highest revenue?

2.3 Tools We'll Use

To stay true to the spirit of this book, I'm keeping our toolkit minimal. We'll only use:

- Python – our programming foundation.
- NumPy – for working with numbers and arrays.
- csv (built-in) – to read rows from our dataset file.
- datetime (built-in) – to handle dates (months, weekdays, etc.).

2.4 Our Game Plan

Here's how we'll tackle this project step by step:

1. Load the dataset – Read the retail data into Python using csv.
2. Clean the data – Handle missing values and fix formatting issues.
3. Explore the data – Get familiar with how it looks and what it means.
4. Ask business questions – Just like a real analyst would.
5. Analyze with NumPy – Perform calculations and find patterns.
6. Translate results into insights – Explain findings in plain English, as if presenting to the shop owner.

2.5 Get Ready

In the next chapter, I'll personally show you how I loaded this large dataset into Python. We'll peek inside, look at the first few rows, and make sure we understand what we're working with.

So, open your coding environment (Jupyter Notebook, Google Collab, or Python on your computer) — because now, we're stepping into the real world of business data science together.

Chapter 3: Loading Our Dataset

3.1 Where Is Our Dataset Stored

The Online Retail II dataset comes as an Excel/CSV file. Since we'll be working with Python, we will use the CSV version

Link [https://archive.ics.uci.edu/ml/datasets/online%2Bretail%2BII?utm_source=chatgpt.com]

3.2 Importing Minimal Tools And Reading The Data

Here's how I opened the dataset and looked at the first 5 rows

What Did We See

Each row is a purchase transaction

```python
import pandas as pd

# It's data loading time! Let's get our hands on that sweet online retail data.
file_path = "/content/sample_data/online_retail_II.xlsx"

# Directly read the file (Excel format) - pandas makes this a breeze!
df = pd.read_excel(file_path)

# Peek at first few rows - always good to see what we're working with!
print(" Behold! The first 5 rows of our dataset:")
print(df.head())

# Let's see the dimensions of our data treasure chest!
print("\n shape of dataset:", df.shape)
# And what shiny columns does it contain?
print("\n Column names:", df .columns. tolist())
```

```
 Behold! The first 5 rows of our dataset:
  Invoice StockCode                          Description  Quantity  \
0  489434     85048  15CM CHRISTMAS GLASS BALL 20 LIGHTS        12
1  489434    79323P                   PINK CHERRY LIGHTS        12
2  489434    79323W                  WHITE CHERRY LIGHTS        12
3  489434     22041         RECORD FRAME 7" SINGLE SIZE         48
4  489434     21232       STRAWBERRY CERAMIC TRINKET BOX        24

          InvoiceDate  Price  Customer ID         Country
0 2009-12-01 07:45:00   6.95      13085.0  United Kingdom
1 2009-12-01 07:45:00   6.75      13085.0  United Kingdom
2 2009-12-01 07:45:00   6.75      13085.0  United Kingdom
3 2009-12-01 07:45:00   2.10      13085.0  United Kingdom
4 2009-12-01 07:45:00   1.25      13085.0  United Kingdom

 shape of dataset: (525461, 8)

 Column names: ['Invoice', 'StockCode', 'Description', 'Quantity', 'InvoiceDate', 'Price', 'Customer ID', 'Country']
```

3.3 A Quick Sanity Check

At this stage, I like to check:

- Did Python read the header correctly.
- Do the rows look meaningful and not random characters.
- Are there enough columns(we expect 8).

If all that looks good, we're ready to move to the next step: Cleaning the data

3.5 What's Next?

In the next chapter, I'll show you how I cleaned the dataset – fixing missing values, handling bad entries, and preparing the data so we can start answering business questions.

Because remember: Good analysis always begin with clean data

Chapter 4: Cleaning The Data

Cleaning your dataset is like tidying up your room before you can actually start working or relaxing. If you skip this step, your analysis can give misleading results – kind of like trying to find your favorite book in a messy pile.

In this chapter, we'll tackle missing values, bad entries, and inconsistence so that our dataset is accurate and ready for exploration.

4.1 Handling Missing Values

Sometimes, datasets aren't perfect. Columns like price have missing values, and if we ignore them, our analysis could give misleading results.

First, we check which columns have missing data. Once we know where the missing values are, we decide how to handle them:

- Critical columns like price: remove rows with missing values.
- Non – Critical columns: replace missing values with 0 or the column average.

By checking and fixing missing data in one step, we make our dataset cleaner and ready for analysis.

4.2 Fixing Bad Entries

Even after handling missing values, datasets sometimes contain weird or unrealistic entries – like a price of 0, negative quantities or cancelled invoices. These are outliers or errors that could distort our analysis.

We can filter them out:

4.2.1 Removing Cancelled transactions

Cancelled invoices usually start with 'C'. We'll filter them out.

4.2.2 Removing Negative Sales

For most business analysis, we only want positive sales.

4.2.3 Removing Duplicates

Sometimes, the same invoice line is recorded twice. Let's drop duplicates.

4.2.4 Converting Invoice Date Into Datetime

We'll convert InvoiceDate into proper datetime format.

4.2.5 Removing Negative or Zero Values

Quantities and prices should be positives

4.2.6 Removing Outliers

To avoid skewed analysis, let's drop extremely large values.

```
# Time to clean up our data! Let's make it sparkling clean for analysis.

# Check for missing values - where are the gaps in our data story?
print("Checking for missing values:")
print(df.isnull().sum())

# Remove duplicate rows - no room for repeats in our dataset!
df = df.drop_duplicates()
print("\nRemoving duplicate rows. New shape:", df.shape)

# Remove rows with negative or zero Quantity or Price - only valid transactions here!
df = df[(df['Quantity']>0) & (df['Price']>0)]
print("\nRemoving rows with non-positive Quantity or Price. New shape:", df.shape)

# Convert InvoiceDate to datetime - making sure our dates are in the right format
df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'])
print("\nConverting InvoiceDate to datetime.")

# Resetting the index - giving our cleaned data a fresh start with the index
df = df.reset_index(drop=True)
print("\nResetting index.")

# Removing cancelled transactions - saying goodbye to cancelled orders
df = df[~df['Invoice'].str.contains('C', na=False)]
print("\nRemoving cancelled transactions. New shape:", df.shape)

# removing negative sales - already done above, but double checking (belt and suspenders!)
df = df[(df['Quantity'] >0) & (df['Price'] >0)]
print("\nRemoving negative sales (double check!). New shape:", df.shape)
```

```
# Removing outliers - keeping our data grounded by removing extreme values
df = df[(df['Quantity'] < 1000) & (df['Price'] < 1000)]
print("\nRemoving outliers (Quantity < 1000 and Price < 1000). New shape:", df.shape)


# Printing cleaned data - behold the beauty of clean data!
print("\nBehold! The first 10 rows of our cleaned dataset:")
print(df.head(10))
print("\nShape of the cleaned dataset:", df.shape)
print("\nDescriptive statistics of the cleaned dataset:")
print(df.describe())
```

```
Checking for missing values:
Invoice             0
StockCode           0
Description      2928
Quantity            0
InvoiceDate         0
Price               0
Customer ID    107927
Country             0
dtype: int64

Removing duplicate rows. New shape: (518596, 8)

Removing rows with non-positive Quantity or Price. New shape: (504731, 8)

Converting InvoiceDate to datetime.

Resetting index.

Removing cancelled transactions. New shape: (504730, 8)

Removing negative sales (double check!). New shape: (504730, 8)

Removing outliers (Quantity < 1000 and Price < 1000). New shape: (504456, 8)
```

Behold! The first 10 rows of our cleaned dataset:

	Invoice	StockCode	Description	Quantity \
0	489434	85048	15CM CHRISTMAS GLASS BALL 20 LIGHTS	12
1	489434	79323P	PINK CHERRY LIGHTS	12
2	489434	79323W	WHITE CHERRY LIGHTS	12
3	489434	22041	RECORD FRAME 7" SINGLE SIZE	48
4	489434	21232	STRAWBERRY CERAMIC TRINKET BOX	24
5	489434	22064	PINK DOUGHNUT TRINKET POT	24
6	489434	21871	SAVE THE PLANET MUG	24
7	489434	21523	FANCY FONT HOME SWEET HOME DOORMAT	10
8	489435	22350	CAT BOWL	12
9	489435	22349	DOG BOWL , CHASING BALL DESIGN	12

	InvoiceDate	Price	Customer ID	Country
0	2009-12-01 07:45:00	6.95	13085.0	United Kingdom
1	2009-12-01 07:45:00	6.75	13085.0	United Kingdom
2	2009-12-01 07:45:00	6.75	13085.0	United Kingdom
3	2009-12-01 07:45:00	2.10	13085.0	United Kingdom
4	2009-12-01 07:45:00	1.25	13085.0	United Kingdom
5	2009-12-01 07:45:00	1.65	13085.0	United Kingdom
6	2009-12-01 07:45:00	1.25	13085.0	United Kingdom
7	2009-12-01 07:45:00	5.95	13085.0	United Kingdom
8	2009-12-01 07:46:00	2.55	13085.0	United Kingdom
9	2009-12-01 07:46:00	3.75	13085.0	United Kingdom

Shape of the cleaned dataset: (504456, 8)

Descriptive statistics of the cleaned dataset:

	Quantity	InvoiceDate	Price \
count	504456.000000	504456	504456.000000
mean	10.266919	2010-06-28 17:28:03	3.804684

min	1.000000	2009-12-01 07:45:00	0.001000
25%	1.000000	2010-03-21 14:07:00	1.250000
50%	3.000000	2010-07-06 14:25:00	2.100000
75%	12.000000	2010-10-15 13:39:00	4.210000
max	992.000000	2010-12-09 20:01:00	975.110000
std	30.196644	NaN	12.812008

	Customer ID
count	400684.000000
mean	15361.737259
min	12346.000000
25%	13985.000000
50%	15311.000000
75%	16805.000000
max	18287.000000
std	1680.581785

Now all bad entries are fixed. This ensure that our analysis isn't distorted by extreme or nonsensical values.

4.3 Why This Matters

Cleaning data might feel boring, but it prevents mistakes later. Imagine calculating the average price and having a listing with 10000 nights — that would completely skew your results!

A clean dataset means trustworthy insights, which is exactly what businesses care about.

4.4 What's Next?

Next, we'll start exploring the data:

- Summarizing numeric columns.
- Counting categorical values.
- Looking for patterns and trends.

Think of it as walking through a city for the first time — after cleaning the streets, you can actually see where everything is.

Chapter 5: Exploring The Data

Now that our dataset is clean and reliable, it's time to start exploring it.

This step is often called **Exploratory Data Analysis (EDA)**.

EDA helps us:

- Understand the overall structure of the dataset.
- Spot trends, patterns, and anomalies.
- Generate insights to answer business questions later.

```
[3]  # Time for a quick peek into our data's soul!

     print("Number of Rows:", len(df)) # How many transactions are we looking at?
     print("Number of Columns:", len(df.columns)) # How many details do we have for each transaction?
     print("Column Names:", df.columns.tolist()) # What are the names of those details?

     # Descriptive statistics - a summary of our numerical data
     print("\nDescriptive statistics of the cleaned dataset:")
     print(df.describe())

     # Unique customers and products - who is buying and what are they buying?

     unique_customers = df['Customer ID'].nunique()
     unique_products = df['StockCode'].nunique()

     print("\nNumber of Unique Customers:", unique_customers) # How many different customers do we have?
     print("Number of Unique Products:", unique_products) # How many different items are being sold?

     # Total sales per month - let's see which months are the champions!
     df['Revenue'] = df['Quantity'] * df['Price'] # Calculate revenue for each transaction
     df['Month'] = df['InvoiceDate'].dt.month # Extract the month from the invoice date
     monthly_sales = df.groupby('Month')['Revenue'].sum() # Group by month and sum the revenue
     print("\nMonthly Sales:")
     print(monthly_sales)

     # Top 10 products by quantity - what are the crowd favorites?
     top_products = df.groupby('Description')['Quantity'].sum().nlargest(10) # Group by product description and find the top 10 by total quantity
     print("\nTop 10 Products by Quantity Sold:")
     print(top_products)
```

```
# Top 10 customers by revenue - who are our biggest spenders?
top_customers = df.groupby('Customer ID')['Revenue'].sum().nlargest(10) # Group by customer and find the top 10 by total revenue
print("\nTop 10 Customers by Revenue:")
print(top_customers)
```

5.1 Basic Dataset Overview

Let's check the overall size and structure of the cleaned dataset.

Exploration isn't about making complex models yet — it's about getting a sense of the business:

- Who are the customers?
- What do they buy?
- When do they shop the most?
- Which products bring in the most revenue?

```
Number of Rows: 504456
Number of Columns: 8
Column Names: ['Invoice', 'StockCode', 'Description', 'Quantity', 'InvoiceDate', 'Price', 'Customer ID', 'Country']
```

Think of this step as getting to know your data before making any big decisions.

5.2 Descriptive Statistics

Basic statistics tell us about typical order sizes, price ranges, and data spread.

```
Descriptive statistics of the cleaned dataset:
            Quantity                    InvoiceDate          Price
count  504456.000000                         504456  504456.000000
mean       10.266919  2010-06-28 17:28:03.438198784       3.804684
min         1.000000            2009-12-01 07:45:00       0.001000
25%         1.000000            2010-03-21 14:07:00       1.250000
50%         3.000000            2010-07-06 14:25:00       2.100000
75%        12.000000            2010-10-15 13:39:00       4.210000
max       992.000000            2010-12-09 20:01:00     975.110000
std        30.196644                            NaN      12.812008

         Customer ID
count  400684.000000
mean    15361.737259
min     12346.000000
25%     13985.000000
50%     15311.000000
75%     16805.000000
max     18287.000000
std      1680.581785
```

This gives:

- **Quantity:** Average items per order.
- **Price:** Average product price.
- **Range:** min/max values (after outlier removal).

5.3 Unique Customers and Products

How many different customers and products are in the dataset? we now know how many customers are in the dataset, how many unique products were sold, and the general spread of quantities and prices.

```
Number of Unique Customers: 4304
Number of Unique Products: 4250
```

5.4 Date Range of Transactions

Since we converted InvoiceDate earlier, we can now check the time span. This shows us which months were busier, and whether there are any seasonal spikes. For example, you may notice sales shooting up around November and December — thanks to the holiday season.

Monthly Sales:

Month

January 627216.652

February 530677.716

March 762447.941

April 659789.402

May 636625.600

June 701880.250

July 645199.770

August 674118.560

September 862895.381

October 1098214.850

November 1418487.502

December 1225316.360

Name: Revenue, dtype: float64

5.5 Best Selling Products

What produce are customers buying the most? Let's find out.

```
Top 10 Products by Quantity Sold:
Description
WHITE HANGING HEART T-LIGHT HOLDER     58691
PACK OF 72 RETRO SPOT CAKE CASES       46728
ASSORTED COLOUR BIRD ORNAMENT          41108
60 TEATIME FAIRY CAKE CASES            35148
WORLD WAR 2 GLIDERS ASSTD DESIGNS      34451
PACK OF 60 PINK PAISLEY CAKE CASES     31805
JUMBO BAG RED RETROSPOT                30746
STRAWBERRY CERAMIC TRINKET BOX         27059
PACK OF 72 SKULL CAKE CASES            24194
COLOUR GLASS T-LIGHT HOLDER HANGING    22862
Name: Quantity, dtype: int64
```

This gives us a clear list of the most popular items. If you were running this business, you'd want to make sure these products never go out of stock.

5.6 Top Customers

Every business has a few customers who bring in the most revenue. Identifying them is crucial.

```
Top 10 Customers by Revenue:
Customer ID
18102.0    344507.39
14646.0    248396.50
14156.0    188457.44
14911.0    152121.22
13694.0    130096.79
17511.0     84541.17
15061.0     83284.38
16684.0     79659.77
13089.0     57885.45
15311.0     55810.74
Name: Revenue, dtype: float64
```

This is the beginning of customer segmentation – figuring out who your best customers are. Later, we could even build strategies around retaining them.

5.7 First Insights

From these simple checks, we'll learn:

- How big our dataset is.
- How many unique products and customers exist.
- The time period it covers.
- Typical order sizes and product prices.
- How sales change over time.
- Which products sells the most.
- Who the best customers are.

These first insights set the stage for deeper business analysis in the next chapters. Notice how we haven't done anything too complex yet — just grouped and summarized.

But even these simple steps are enough to answer important business questions.

5.8 What's Next

In the next chapter, we'll go deeper into customer behavior.

Instead of just looking at totals, we'll analyze patterns: how often customers order, how much they spend, and how loyal they are.

Because remember: behind every row in this dataset, there's a customer — and understanding them is the key to growing the business.

Chapter 6: Answering Business Questions

Now it's time to do the fun part: answering real business questions.

Remember: businesses don't care about "null values" or "outliers." What they care about is where their money is coming from, who their customers are, and what sells best.

In this chapter, we'll answer five key business questions using our dataset.

6.1 Question 1: Which countries bring in the most revenue?

This tells us where the company should focus its marketing and operations.

```
# Question 1
# Calculate revenue
df['Revenue'] = df['Quantity'] * df['Price']

# Revenue by country
country_revenue = df.groupby('Country')['Revenue'].sum().sort_values(ascending=False)

# Time to reveal the top revenue-generating countries!
print("😈 Drumroll please! Our top 10 revenue champions by Country are:")
print(country_revenue.head(10))  # Top 10 countries
print("\nLooks like some countries are really loving our products! 😊")
```

```
😈 Drumroll please! Our top 10 revenue champions by Country are:
Country
United Kingdom     8466461.983
EIRE                372817.270
Netherlands         268784.350
Germany             202025.391
France              132014.890
Sweden               52234.790
Spain                47568.650
Switzerland          43921.390
Australia            31446.800
Channel Islands      24546.320
Name: Revenue, dtype: float64

Looks like some countries are really loving our products! 😊
```

Insight:

- The UK dominates because it's the home market.
- Other high-revenue countries could be EIRE, Netherlands, Germany.
- This shows the company where international expansion is working.

6.2 Question 2: Who are the top customers by total spend?

Businesses love to know their **VIP customers** so they can reward them or build loyalty.

```
[ ] #Question 2
    # Revenue by customer
    customer_revenue = df.groupby('Customer ID')['Revenue'].sum().sort_values(ascending=False)

    print("🌟 Behold the big spenders! Our top 10 customers by revenue are: 💰")
    print(customer_revenue.head(10))  # Top 10 customers
    print("\nThese customers are truly golden! ✨")
```

```
🌟 Behold the big spenders! Our top 10 customers by revenue are: 💰
Customer ID
18102.0    344507.39
14646.0    248396.50
14156.0    188457.44
14911.0    152121.22
13694.0    130096.79
17511.0     84541.17
15061.0     83284.38
16684.0     79659.77
13089.0     57885.45
15311.0     55810.74
Name: Revenue, dtype: float64

These customers are truly golden! ✨
```

Insight:

- Some customers spend **tens of thousands of dollars**
- These are the people the business must keep happy!

6.3 Question 3: Which products are the bestsellers

This helps the company manage inventory and promotions.

```
[ ] # Question 3
    # Total quantity sold per product
    product_sales = df.groupby('Description')['Quantity'].sum().sort_values(ascending=False)

    print("🚀 Flying off the shelves! Our top 10 bestsellers by quantity are: ")
    print(product_sales.head(10))  # Top 10 bestselling products
    print("\nThese products are definitely customer favorites! 👍")
```

```
🚀 Flying off the shelves! Our top 10 bestsellers by quantity are:
Description
WHITE HANGING HEART T-LIGHT HOLDER     58691
PACK OF 72 RETRO SPOT CAKE CASES       46728
ASSORTED COLOUR BIRD ORNAMENT          41108
60 TEATIME FAIRY CAKE CASES            35148
WORLD WAR 2 GLIDERS ASSTD DESIGNS      34451
PACK OF 60 PINK PAISLEY CAKE CASES     31805
JUMBO BAG RED RETROSPOT                30746
STRAWBERRY CERAMIC TRINKET BOX         27059
PACK OF 72 SKULL CAKE CASES            24194
COLOUR GLASS T-LIGHT HOLDER HANGING    22862
Name: Quantity, dtype: int64

These products are definitely customer favorites! 👍
```

Insight:

- Often, simple, inexpensive products like gift bags or decorative items dominate.
- This shows that **volume matters as much as price** in sales strategy.

6.4 Question 4: When do people shop the most?

Do sales peak in November/December? Are weekdays or weekends busier?

```
[ ] # Question 4

    import calendar

    # Extract month and weekday
    df['Month'] = df['InvoiceDate'].dt.month
    df['Weekday'] = df['InvoiceDate'].dt.day_name()

    # Map month numbers to names
    df['MonthName'] = df['Month'].apply(lambda x: calendar.month_name[x])

    # Revenue by month (ordered from Jan-Dec)
    monthly_sales = df.groupby('MonthName', sort=False)['Revenue'].sum()
    monthly_sales = monthly_sales.reindex(calendar.month_name[1:])

    print("📊 Monthly Revenue Breakdown: See which months brought in the most cash! 💸")
    print(monthly_sales)
    print("\nLooks like some months are real powerhouses! 💪")

    # Set weekday order manually (Monday → Sunday)
    weekday_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
    df['Weekday'] = pd.Categorical(df['Weekday'], categories=weekday_order, ordered=True)

    # Revenue by weekday
    weekday_sales = df.groupby('Weekday')['Revenue'].sum()

    print("\n🗓️ Revenue by Day of the Week: Which days are our busiest? 🏃‍♂️🏃‍♂️")
    print(weekday_sales)
    print("\nInteresting to see the daily trends! 🤔")
```

```
📊 Monthly Revenue Breakdown: See which months brought in the most cash! 💸
MonthName
January       627216.652
February      530677.716
March         762447.941
April         659789.402
May           636625.600
June          701880.250
July          645199.770
August        674118.560
September     862895.381
October      1098214.850
November     1418487.502
December     1225316.360
Name: Revenue, dtype: float64

Looks like some months are real powerhouses! 💪

🗓️ Revenue by Day of the Week: Which days are our busiest? 🏃‍♂️🏃‍♂️
Weekday
Monday       1776137.805
Tuesday      1888921.151
Wednesday    1717933.903
Thursday     1964890.422
Friday       1462900.922
Saturday        9803.050
Sunday       1022282.731
Name: Revenue, dtype: float64
```

Insight:

- We'll likely see a **huge spike in November and December** (Christmas Shopping).
- Weekdays like **Thursday and Friday** may be busier than weekends.

6.5 Question 5: How do cancellations affect sales?

In this dataset, cancelled orders appear as negative quantities. We need to measure their impact.

```
[ ]  # Question 5
     # Separate cancelled and valid orders
     cancelled = df[df['Quantity'] < 0]
     valid = df[df['Quantity'] > 0]

     # Revenue lost from cancellations
     cancelled_revenue = cancelled['Revenue'].sum()
     valid_revenue = valid['Revenue'].sum()

     print("Valid Revenue:", valid_revenue)
     print("Cancelled Revenue:", cancelled_revenue)
     print("Cancellation %:", (abs(cancelled_revenue) / valid_revenue) * 100)
```

```
Valid Revenue: 9842869.984000001
Cancelled Revenue: 0.0
Cancellation %: 0.0
```

Insight:

- Cancellations often make up 5–15% of total sales.
- That's a big number — reducing cancellations could directly boost profits.

6.6 Wrapping Up

In this chapter, we answered five important business questions:

1. Where does revenue come from?
2. Who are the top customers?
3. Which products sell the most?
4. When do people shop the most?
5. How do cancellations affect sales?

Notice something? Each answer comes from simple grouping and aggregation, not advanced AI. Yet these insights are exactly what a business needs.

Next, in Chapter 7, we'll go one step deeper and explore customer segmentation (like finding patterns in customer behavior).

Because data science isn't just about "what happened" — it's about "who buys what, and why."

Chapter 7: Understanding Customer Behavior

Now that we've explored the dataset at a high level, it's time to zoom in on the customers themselves.

After all, every purchase in the dataset represents a real person making a choice.

Understanding these choices is what separates raw data from real business intelligence.

In this chapter, we'll explore:

- How frequently customers shop.
- How much they spend.
- How long they've been active.

This is the foundation of customer analytics. In this section, we introduce a simple but powerful framework called RFM Analysis. RFM stands for Recency, Frequency, and Monetary value.

These three factors help us measure:

- Recency (R): How recently a customer made a purchase.
- Frequency (F): How often the customer buys from us.
- Monetary (M): How much money the customer has spent.

Instead of using advanced statistical methods, we will build an easy, step-by-step approach to calculate these values. This way, even beginners can see how data can give insights into customer behavior.

7.1 Recency, Frequency, and Monetary Value (RFM Analysis)

A popular way to understand customers is through RFM analysis.

- Recency (R): How recently a customer made a purchase.
- Frequency (F): How often they buy.

- Monetary Value (M): How much money they spend.

Together, these three metrics help us categorize customers into groups like:

- Loyal Customers
- Big Spenders
- At-Risk Customers
- New Customers

Step 1: Prepare the Data

Before analyzing customers, we first make sure our dataset has a useful column for total spending per order.

Step 2: Group Data by Customer

Now we group our dataset by Customer ID. For each customer, we want three pieces of information:

1. The last purchase date (to calculate recency).
2. The number of purchases (frequency).
3. The total amount spent (monetary value).

Step 3: Calculate Recency

Now that we know the last purchase date of each customer, we can calculate how many days ago they made that purchase.

Step 4: Final RFM Table

We now have three key values for each customer.

```
[4]  # Create a TotalSales column - because who doesn't love seeing the money roll in?
     df['TotalSales'] = df['Quantity'] * df['Price']

     # Convert InvoiceDate to datetime - making sure our dates are in the right format
     df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'])

     # Group by Customer ID - let's see what each customer is up to!
     customer_behavior = df.groupby('Customer ID').agg({
         'InvoiceDate': 'max',   # When did they last grace us with their presence?
         'Invoice': 'count',   # How often do they shop? The more the merrier!
         'TotalSales': 'sum'     # How much treasure have they spent?
     }).reset_index()

     # Rename for clarity - giving our columns some snazzy names
     customer_behavior.rename(columns={
         'InvoiceDate': 'LastPurchaseDate',
         'Invoice': 'Frequency',
         'TotalSales': 'Monetary'
     }, inplace=True)

     # Find the latest date in the dataset - our reference point for recency
     latest_date = df['InvoiceDate'].max()

     # Calculate Recency (days since last purchase) - how long has it been? We miss them!
     customer_behavior['Recency'] = (latest_date - customer_behavior['LastPurchaseDate']).dt.days

     # Keep only useful columns - decluttering time!
     customer_behavior = customer_behavior[['Customer ID', 'Recency', 'Frequency', 'Monetary']]
```

```
     # Behold the customer behavior summary!
     print("Customer Behavior Summary (a peek at our loyal subjects):")
     print(customer_behavior.head())
```

```
Customer Behavior Summary (a peek at our loyal subjects):
   Customer ID  Recency  Frequency  Monetary
0      12346.0      164         33    372.86
1      12347.0        2         71   1323.32
2      12348.0       73         20    222.16
3      12349.0       42        102   2671.14
4      12351.0       10         21    300.93
```

Step 5: Interpretation

- Customers with low Recency (recent purchases) are still active.
- Customers with high Frequency are loyal buyers.
- Customers with high Monetary the most revenue.

Even without complex scoring, this table already tells us:

- Who our best customers are,
- Who might be losing interest, and
- Which customers are most valuable.

Now, let's merge everything into a single REM table. At this point, we have a powerful dataset: every customer is described by how recent, how frequent, and how valuable their purchases are.

7.2 segmenting Customers

With RFM, we can segment customers. A simple way is to divide each column into quartiles (1=low, 4=high).

For example:

- A customer with an RFM score of 444 is a loyal, frequent, and high-spending customer.
- A customer with an RFM score of 111 may be inactive and not very valuable.

```
# SEGMENTING CUSTOMERS – BEGINNER FRIENDLY VERSION

# Step 1: Calculate averages for Recency, Frequency, and Monetary
# These will be our benchmarks – customers above/below these get different nicknames.

rfm = customer_behavior

avg_recency = rfm['Recency'].mean()
avg_frequency = rfm['Frequency'].mean()
avg_monetary = rfm['Monetary'].mean()

# Step 2: Create a function to assign each customer to a segment
def segment_customer(row):
    # VIP Shoppers: Buy often AND spend big
    if row['Frequency'] >= avg_frequency and row['Monetary'] >= avg_monetary:
        return "VIP Shoppers"

    # Loyal Fans: Shop very often, but may not spend a lot each time
    elif row['Frequency'] >= avg_frequency:
        return "Loyal Fans"

    # Big Spenders: Spend a lot when they do shop, but not very frequent
    elif row['Monetary'] >= avg_monetary:
        return "Big Spenders"

    # Sleeping Beauties: Haven't shopped in a long time (we miss them!)
    elif row['Recency'] > avg_recency:
        return "Sleeping Beauties"

    # Regulars: Average customers who keep the business alive
    else:
        return "Regulars"
```

```
# Step 3: Apply the function to our RFM table
rfm['Segment'] = rfm.apply(segment_customer, axis=1)

# Step 4: Count how many customers fall into each segment
print(rfm['Segment'].value_counts())
```

```
Segment
Regulars             1680
Sleeping Beauties    1264
VIP Shoppers          739
Loyal Fans            429
Big Spenders          192
Name: count, dtype: int64
```

7.3 Why This Matters

Businesses thrive when they understand their customers.

- Loyal customers should be rewarded with discounts or exclusive offers.
- At-risk customers might need re-engagement campaigns (emails, reminders).
- Big spenders could be offered premium services.

By analyzing behavior, businesses can spend money smarter — focusing their energy where it matters most.

7.4 What's Next?

We now know who our best customers are and how they behave.

In the final chapter, we'll wrap up everything — reflecting on the journey from messy raw data to actionable business insights.

Chapter 8: Wrapping It All Up

We've reached the final chapter of our project. This is where we look back at the journey and tie everything together.

8.1 What We Started With

- A raw dataset (messy, with missing values and strange entries).
- Some big questions: Which products sell the most? Who are the best customers? What times are busiest?

At first, it looked overwhelming — but step by step, we broke it down.

8.2 The Cleaning Journey

We cleaned the data by:

- Removing missing values.
- Fixing invalid rows (like negative quantities).
- Converting dates to proper datetime format.
- Removing outliers that could mislead our analysis.

This gave us a clean, trustworthy dataset.

8.3 The Insights We Discovered

From our analysis, we found:

- Top-selling products and categories.
- Best customers who purchase the most.
- Seasonal trends (months and times when sales peak).
- Country-level insights, showing where most sales come from.

Each of these answers came from careful data exploration + visualization.

8.4 What You Learned

Through this project, you learned how to:

1. Load and explore a dataset in Python.
2. Clean and prepare data for analysis.
3. Use simple but powerful Pandas operations.
4. Create visualizations to tell a story.
5. Turn messy data into clear business insights.

8.5 Why This Matters

This wasn't just about code.

This was about learning how a data scientist thinks:

- Start messy.
- Clean carefully.
- Ask good questions.
- Answer them with data.

That's the essence of data science.

8.6 Final Words

When I began this journey, my first goal was simple: to explain what Data Science is and why it matters. That was the story of my first book — giving students a clear picture of the field and its possibilities.

In this second book, I went one step further. Instead of just talking about Data Science, I applied it in a real project: cleaning Online retail data, analyzing patterns, and answering business questions.

Now, as I move forward into Business Engineering, my path may look a little different at first glance. But in reality, Business and Data Science are deeply connected. Businesses today rely on data to make smarter decisions, improve customer experiences, and design better strategies.

So, while my books focused on Data Science, they also built the foundation for something bigger — using data as a tool to engineer better businesses. My journey doesn't end here; it evolves into exploring how insights from data can shape the world of business, innovation, and strategy.

If my first book explained "what Data Science is", and this second book showed "how Data Science works", then my next chapter in Business Engineering will explore "how Data Science and Business come together".

Because in the end, the two are not separate — they are partners. Data guides businesses, and businesses give direction to data.

So, this is not a closing note, but an opening door — from curiosity in Data Science to creating impact in Business Engineering.

Thank you for following along with me on this journey.

If you made it here, you've not only completed the project — you've also learned skills you can reuse in any future dataset you work with.