From Theory to Practice: Python and Analytics in Accounting Fraud Detection

David Olsen, PhD
Utah Tech University

Cindy Greenman, PhD, CFE
Utah Tech University

Abstract

The increasing complexity and volume of financial transactions have highlighted the limitations of traditional fraud detection methods in accounting. As such, the integration of data analytics into accounting education has become crucial to equipping the next generation of accounting professionals with the necessary skills to combat evolving fraudulent activities.

Accounting students today are being trained to become "bilingual" professionals, fluent in both accounting principles and the analytical capabilities provided by modern data tools. By exposing accounting students to different analytical tools and teaching them how to parse financial data to identify and quantify fraudulent activities, college accounting programs can prepare their graduates to become effective forensic accountants, auditors, and advisors.

Ultimately, the integration of data analytics into accounting education is not just about teaching technical skills, but about cultivating a mindset that is a combination of analytical problem-solving using data and financial expertise.

This case integrates data analytics into accounting education by simulating a real-world scenario that allow students to build their data analytic skills that are crucial for discovering fraudulent activity. It tests the students' critical thinking, their proficiency in using Python or Excel, and their ability to communicate complex findings effectively and efficiently.

I. Case Introduction

Background Information

FinForensix Generator represents a comprehensive approach to generating invoice data for an educational module focused on fraud detection within financial transactions. This Python script is a practical tool for simulating real-world invoicing environments with both legitimate and fraudulent transactions, designed to aid students in understanding the complexities of financial fraud detection.

There are many advantages of the case study, including:

Educational Value:

The program provides a hands-on learning experience for students, offering insights into the practical aspects of financial fraud detection. By generating data that includes both legitimate and fraudulent transactions, students can apply theoretical knowledge to real-world scenarios, enhancing their understanding and investigative skills.

Customizability:

With parameters for the number of invoices, date range, and transaction amounts, educators can tailor the dataset to suit various teaching objectives, from introductory courses to advanced fraud detection techniques. The inclusion of Benford's Law in the dataset generation further allows for the exploration of statistical methods in fraud examination.

Realism and Complexity:

By incorporating elements such as random payment dates, customer names, and genders, along with fraudulent behaviors, the program creates a complex dataset that mirrors the intricacies of real financial transactions. This complexity challenges students to employ critical thinking and analytical skills in distinguishing between legitimate and fraudulent activities.

Scalability:
The program is designed to generate a large volume of invoice data, making it suitable for both individual and group projects. The scalability ensures that the dataset can be used for comprehensive analysis, allowing students to explore various fraud detection methods and tools.
Integration with Technological Tools:
The final dataset is saved as a CSV file, facilitating easy integration with data analysis tools such as Excel, Google Sheets, or specialized software. This flexibility supports a wide range of instructional strategies and allows for the incorporation of technology into the learning process.

II. Case Requirements

The program leverages several Python libraries, including pandas for data manipulation, numpy for numerical calculations, random for generating random numbers and choices, and datetime for handling dates and times. It is structured around key functions that collectively produce a dataset of invoices that mimic real-world financial transactions with an element of fraud incorporated. The functions include:

1. Name and Gender Generation:
The generate_name_gender function creates a dictionary mapping names to genders based on a predefined set of male names. This function is essential for simulating realistic customer data by assigning genders based on first names, allowing for a more detailed analysis of transactions.

2. Unique Invoice Number Generation:
generate_unique_invoice_number ensures that each invoice has a distinct identifier by generating random alphanumeric strings and checking against previously generated numbers to avoid duplicates. This function mimics real-world scenarios where each transaction is uniquely identifiable.

3. Benford's Law Numbers:
The benford_law_numbers function generates invoice amounts that comply with Benford's Law, a principle often used in fraud detection to identify anomalous numerical data. This feature introduces students to practical applications of mathematical laws in identifying irregularities in financial records.

4. Invoice Data Generation:
The core of the program is the generate_invoice_data function, which simulates the generation of invoice transactions over a specified date range with amounts within a given range. It incorporates the logic for identifying fraudulent transactions, adjusting invoice amounts, and generating additional fraudulent transactions such as duplicates or unusually frequent transactions.

Technical Requirements and Prerequisites

To ensure the effective implementation of this fraud detection case study, it is crucial to establish clear technical requirements and prerequisites for students and educators. The case study is designed to integrate practical data analysis skills with theoretical knowledge, leveraging Python programming and Excel for a hands-on learning experience. Below are the recommended prerequisites and technical skills required for participants:

For Students:

Python Proficiency: Students are expected to have completed at least one semester of a Python programming course or a lower-division business data analytics class with a substantial Python component. This foundational knowledge should include understanding of Python syntax, basic programming constructs (loops, conditionals, functions), and familiarity with the Python data science stack, including libraries such as pandas, numpy, and matplotlib.

Excel Skills: Basic to intermediate proficiency in Microsoft Excel or similar spreadsheet software is required. Students should be comfortable with data manipulation techniques, formulas, pivot tables, and chart creation.

Statistical Fundamentals: A foundational understanding of statistics is beneficial, including concepts such as means, medians, standard deviation, and basic principles of statistical inference. This knowledge will aid in the analysis of data and the identification of anomalies indicative of fraud.

For Educators:

Technical Knowledge: Educators should have a proficient understanding of both Python and Excel, capable of guiding students through the analysis processes and troubleshooting common issues. Familiarity with Python's data science libraries and advanced Excel functions is essential.

Curriculum Integration: Instructors should assess how this case study fits within the broader curriculum, ensuring that students have received or are concurrently receiving instruction in the necessary programming and statistical concepts.

Resource Availability: Access to computers with Python installed (either directly or via an Anaconda distribution) and Excel is required. Institutions should ensure that all students have equitable access to these resources, considering the possibility of using cloud-based environments like Google Colab for Python programming to minimize technical barriers.

Institutional Considerations:

Academic institutions planning to incorporate this case study into their curriculum must evaluate the availability of necessary technical resources, including software access and computer lab facilities. Institutions should also consider the development of bridging courses or workshops for students who lack the required background, ensuring that all interested students have the opportunity to participate.

In summary, the successful integration of this case study into analytics education requires careful planning and preparation, both in terms of curriculum design and technical resource allocation. By establishing clear prerequisites and ensuring that students and educators are

adequately prepared, institutions can maximize the educational value of this case study and equip students with valuable skills for detecting and preventing fraud.

III. Case Learning Objectives and Implementation Guidance

This analytics case study involves leveraging Python or MS Excel to conduct a detailed analysis of an invoice dataset to identify potentially fraudulent activities. It is designed to simulate a real-world scenario where data analysis skills are crucial for uncovering fraudulent activities. It will test students critical thinking skills, proficiency in using Python or Excel, and their ability to properly communicate complex findings in an effective manner.

Prior Literature

The growth of analytics in the accounting industry has revolutionized how professionals approach various aspects of financial management. Alongside traditional practices, the integration of analytics tools has empowered accountants to delve deeper into data analysis, enabling them to extract valuable insights, predict risks, and enhance decision-making processes.

Additionally, the incorporation of fraud examinations within this analytical framework further strengthens the industry's ability to detect and prevent fraudulent activities effectively. Fraud examiners play a crucial role in investigating specific allegations of fraud, determining its occurrence, identifying perpetrators, and providing detailed reports that can be utilized in court proceedings. The link between analytics and fraud investigations emphasizes the need for a more comprehensive approach that ensures not only financial accuracy but also integrity and security within the financial operations.

This has led to a demand for data analytics education in academia. The American Institute of Certified Public Accountant (AICPA) placed an emphasis on the significance of data analytics in education, highlighting its growing importance in preparing students for their careers. In May 2021, the AICPA released a new certificate on Data Analytics Core Concepts to assure that practicing CPAs have the skills that are necessary due to the growing importance that data analytics play in business. The Association for the Advancement of Collegiate Schools of Business (AACSB) includes data analytics in their accreditation process.

In their case study, “Data Analytics for Undergraduate Tax Students: An Alteryx Case Study for MACRS Depreciation”, Cheng, Rhoades-Catanach and Watson (2023) focused their case on an undergraduate classroom and utilized the software Alteryx in a tax simulation scenario. Their results showed a high increase in the overall knowledge of the need for analytics as well as utilizing the Alteryx software itself.

Richardson and Watson (2022) discussed the need for faculty to teach accounting students to address accounting questions utilizing data analytics. They expressed that accounting questions fall into five broad categories, with the newest one being adaptive and autonomous analytics. Their paper goes on to explain how each type of accounting questions can be matched with an analytics technique.

In their October 2023 article in the Journal of Emerging Technologies in Accounting, Bierstaker, Lombardi and Wang use a case study that emphasizes Python programming and

visualizations into the case analysis of a real grocery store chain. The case was designed to provide students with a large data set with which to learn and practice manipulating big data. The fictitious dataset allowed the students to explore data analytic techniques utilizing Python.

Learning Objectives

Upon completion of the Fraud Detection Analysis Case Study, students will be able to: (1) master data extraction techniques, (2) enhance their data analysis capabilities, (3) foster their critical thinking skills, (4) learn to integrate Python with traditional accounting practices, and (5) prepare for a future career in the accounting profession.

Evidence of Efficacy

In order to evaluate the effectiveness of the FinForensix Generator Case in achieving the intended learning objectives, we surveyed 30 students in an undergraduate analytics course and 21 students in a graduate level analytics course. The survey contained a series of statements that the students could agree or disagree with on a 5-point Likert scale (1=strongly disagree, 2=disagree, 3=Neither agree nor disagree, 4=agree, and 5=strongly agree). The questions related directly to the learning objectives. There was a pretest and a posttest given.
Learning Objectives:

1. L_0: Master Data Extraction Techniques
2. L_1: Enhance Data Analysis Capabilities
3. L_2: Foster Critical Thinking through Case Studies
4. L_3: Integrate Python with Traditional Accounting Practices
5. L_4: Prepare for the Future of Accounting Profession

Questions Mapped to Objectives:

- L_0: Master Data Extraction Techniques
    - Question 1: I am confident in my ability to use Python scripts to automatically extract data from SEC filings and other financial documents.
    - Question 2: I believe I can enhance financial analysis by automating data extraction using Python, including retrieving data from sources like the stock market.
- L_1: Enhance Data Analysis Capabilities
    - Question 3: I feel proficient in applying Python for conducting complex data analyses, such as regression analysis, sentiment analysis, and identifying trends in financial data.
    - Question 4: I am comfortable using Python to enhance my analytical skills and apply these to various datasets to uncover insights.
- L_2: Foster Critical Thinking through Case Studies
    - Question 5: I am capable of critically engaging with case studies that apply Bloom's Taxonomy in the accounting context, enabling me to solve complex financial scenarios.
    - Question 6: I believe I can apply, analyze, evaluate, and create solutions to complex financial scenarios after engaging with relevant case studies.
- L_3: Integrate Python with Traditional Accounting Practices

    - Question 7: I feel confident in my ability to integrate Python with traditional accounting practices to enhance the analytical aspects of accounting and finance.
    - Question 8: I believe that the use of Python can significantly improve traditional accounting practices by incorporating modern computational methods.
- L_4: Prepare for the Future of Accounting Profession
    - Question 9: I am prepared for the future of the accounting profession, where proficiency in tools like Python and an understanding of advanced data analysis techniques are crucial.
    - Question 10: I feel equipped with the skills and knowledge necessary to thrive in a rapidly evolving professional landscape that values analytical acumen and computational proficiency.

We mapped the learning objectives to each of the questions with the following reasoning and results:

Master Data Extraction Techniques (L_0)

Question 1 & 2 primarily assess confidence in using Python for data extraction from financial documents and enhancing financial analysis. The results indicate a general tendency towards disagreement or neutrality, suggesting a lack of confidence among students in their ability to utilize Python for data extraction and analysis purposes.

Enhance Data Analysis Capabilities (L_1)

Question 3 & 4 focus on proficiency in applying Python for complex data analyses and comfort in using Python to enhance analytical skills. The responses leaning towards disagreement or neutrality in these questions highlight a potential area for improvement in students' perceptions of their data analysis skills with Python.

Foster Critical Thinking through Case Studies (L_2)

Question 5 & 6 are about the capability to engage critically with case studies and apply analytical skills to solve complex financial scenarios. The relatively lower agreement levels suggest students may feel less prepared to tackle complex case studies effectively, indicating a need for further development in critical thinking and application skills.

Integrate Python with Traditional Accounting Practices (L_3)

Question 7 & 8 examine confidence in integrating Python with traditional accounting practices and beliefs about the improvement Python can bring. The results show a mixed response but tend towards more agreement, especially for Question 8, suggesting a recognition of Python's potential benefits in modernizing accounting practices despite some reservations about personal ability.

Prepare for the Future of Accounting Profession (L_4)

Question 9 & 10 investigate preparedness for a future where Python and advanced data analysis are crucial and feeling equipped with necessary skills. These questions have relatively higher agreement levels, indicating a more positive outlook among students about their readiness for future professional landscapes, despite the overall trend of neutral or negative responses in earlier sections.

Table 1
Mean Response by Course, by Question and Pretest/Posttest

	Q 1	Q 2	Q 3	Q 4	Q 5	Q 6	Q 7	Q 8	Q 9	Q 10
Undergrad Pretest	3.17	2.86	2.97	3.10	2.66	2.24	2.86	3.69	3.48	3.45
Undergrad Posttest	3.70	3.47	3.10	3.67	3.63	3.33	3.67	3.87	3.80	3.83
Graduate Pretest	2.43	2.57	2.19	2.81	2.24	2.00	2.76	3.43	3.33	3.48
Graduate Posttest	3.48	3.29	3.38	3.52	3.05	2.90	3.43	3.62	3.57	3.81

Graph 1

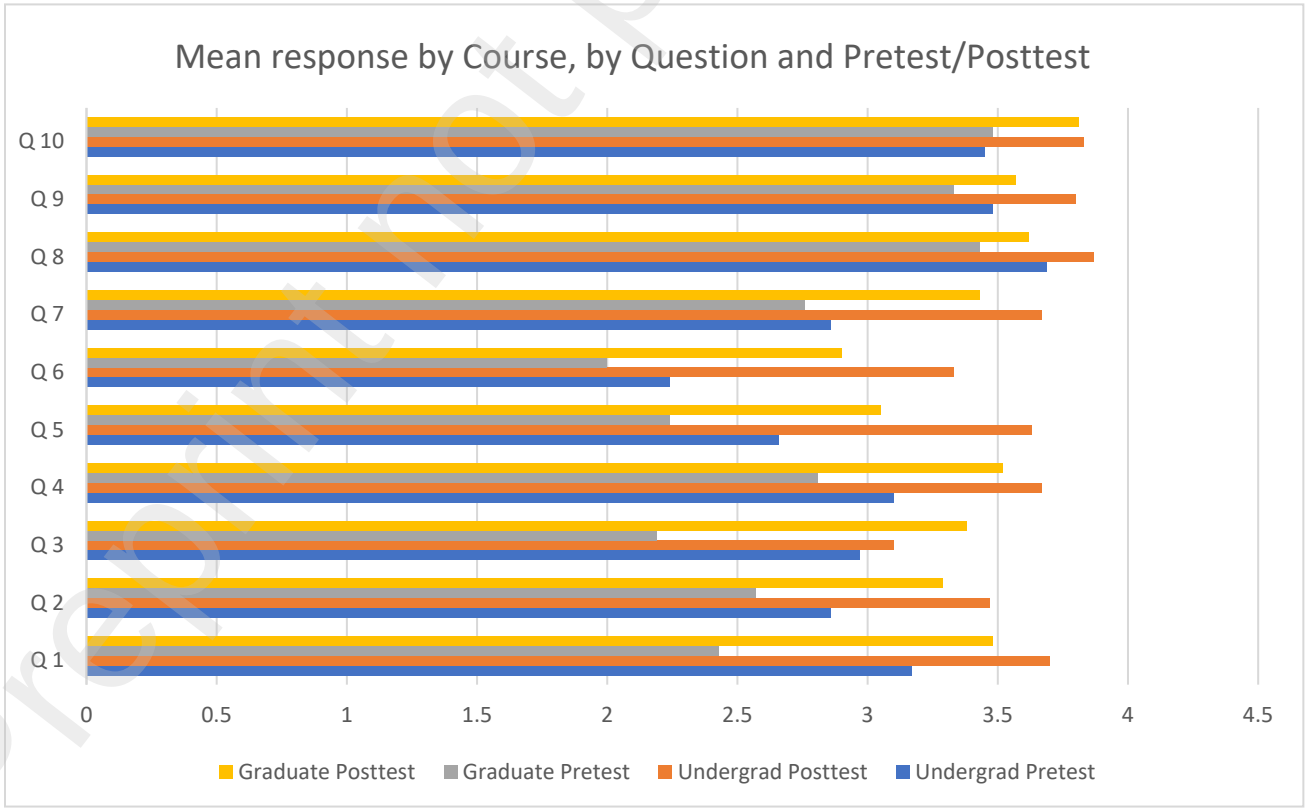


Table 2
Mean response by Course, by Learning Objective and Pretest/Posttest

	L0	L1	L2	L3	L4
Undergrad Pretest	3.01	3.04	2.45	3.28	3.46
Undergrad Posttest	3.58	3.38	3.48	3.77	3.82
Graduate Pretest	2.50	2.50	2.12	3.10	3.40
Graduate Posttest	3.38	3.45	2.97	3.53	3.69

Graph 2

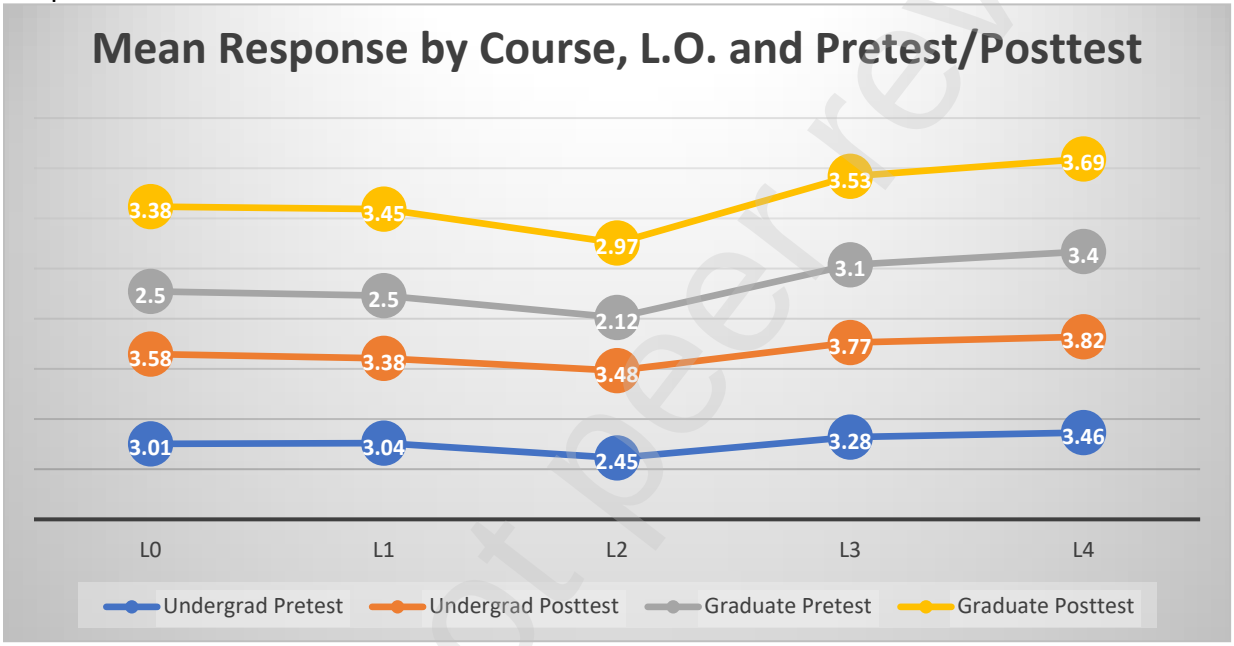


Table 2
Overall Mean Response Pretest

Learning Objective	Mean	Median	Min	Max	Std Dev.	T-Statistic	P-Value
L_0 Master Data Extraction Techniques - n = 51	3.12	3.2	2.5	3.58	0.48	3.6811165	0.00602527
L_1 Enhance Data Analysis Capabilities - n = 51	3.09	3.21	2.5	3.45	0.44	3.381144	0.03504787
L_2 Foster Critical Thinking through Case Studies - n = 51	2.76	2.71	2.12	3.48	0.6	2.42755	0.07607384
L_3 Integrate Python with Traditional	3.42	3.4	3.1	3.77	0.29	5.22174	0.00002664

Accounting Practices - n= 51							
L_4 Prepare for the Future of Accounting Profession - n = 51	3.59	3.58	3.4	3.82	0.19	6.0918655	0.00000052

Implementation Guidance

The case is designed to allow instructors to tailor its implementation. The nature of the case lends itself well to either individual work or to a small-group assignment. We broke the assignment into two groups, one utilizing MS Excel and the other Python. We mentioned to the class that there are suspected fraudulent activities within the dataset and that the fraud is being orchestrated by certain individuals and prefaced the case with the following common fraud indicators:

1. **Mismatched Dates**: Payment dates that occur before the invoice date or very shortly after could be suspicious, as they do not follow typical billing cycles.
2. **Rounded Amounts**: Frauds often use rounded numbers. Regular transactions often involve cents, so exact amounts like $300.00 could be scrutinized.
3. **Frequency of Transactions**: If the same customer has multiple transactions in a short period, especially with significant amounts, this might be indicative of fraud.
4. **Duplicate Invoice Numbers**: Each invoice number should be unique. If duplicates are found, it could indicate fraudulent activity.
5. **Large Transactions**: Significantly higher invoice amounts compared to the average could be a sign of fraud, especially if not consistent with the customer's usual transaction pattern.
6. **Unusual Customer Behavior**: Transactions that don't fit the typical pattern of a customer, such as a sudden change in the frequency or dollar amount of invoices.
7. **Anomalies in Payment Patterns**: Payments that always occur at the edge of the payment term, or inconsistencies in payment delays.
8. **Outliers in Amounts**: Invoice amounts that are consistently high or low outliers compared to the typical distribution of amounts could be suspicious.

The dataset included invoices in a MS Excel spreadsheet. The transactions had the following attributes: Invoice ID, Invoice number, Invoice Date, Invoice Due Date, Invoice Amount, Payment Date, Customer Name, and Customer Gender.

Sample data:

InvoiceID	InvoiceNumber	InvoiceDate	InvoiceDueDate	InvoiceAmount	PaymentDate	CustomerName	CustomerGender
97542	2BF635TXHH	1/1/2024	1/31/2024	436.59	1/5/2024	Gregory Y Rogers	M
22864	AW3H49I3AR	1/1/2024	1/31/2024	318.59	1/1/2024	Hannah U Powell	F
70545	0KXAU0DBW2	12/31/2023	1/30/2024	252.19	2/3/2024	Sean S Howard	M
371452	0M7J9SFU7F	12/31/2023	1/30/2024	383.37	1/30/2024	Hannah U Powell	F
378911	0M7J9SFU7F	12/31/2023	1/30/2024	383.37	1/30/2024	Hannah U Powell	F
63661	171PY405M6	12/31/2023	1/30/2024	577.38	2/1/2024	Kelly N Green	F
249491	1ESKAFSBPQ	12/31/2023	1/30/2024	385.81	1/30/2024	Teresa J Cox	F
99222	1Q11T63PUU	12/31/2023	1/30/2024	204.78	2/3/2024	Tiffany O Bailey	F
299581	24G8CMI7OW	12/31/2023	1/30/2024	282.25	2/4/2024	Frank Q Wright	M
375159	24RXTDZQVU	12/31/2023	1/30/2024	346.21	2/4/2024	Ryan H Rodriguez	M
95681	2BF635TXHH	12/31/2023	1/30/2024	436.59	2/1/2024	Gregory Y Rogers	M
314210	2D1SLGNOQP	12/31/2023	1/30/2024	526.42	1/31/2024	Sara S Scott	F
59950	2I1LCIV650	12/31/2023	1/30/2024	327.96	1/30/2024	Henry O Gomez	M

The instructions given to the students are given in a nine-step process. They are instructed to complete each task using their assigned tool (MS Excel or Python) and that their analysis should aim to uncover irregularities and potential fraud within the dataset. The nine-steps are:

1. **Data Preparation and Cleaning**
    - Convert all date columns to the correct datetime format to facilitate analysis.
    - Identify and highlight any duplicate invoice numbers, as they may indicate fraudulent entries or clerical errors.
2. **Exploratory Data Analysis**
    - Calculate the frequency of transactions for each customer to identify unusually high activity levels.
    - Find transactions where payment dates precede invoice dates, which could indicate post-dated invoicing or data entry errors.
3. **Fraud Indicators Analysis**
    - Round invoice amounts may be easier for fraudsters to fabricate. Identify all transactions with round-number invoice amounts.
    - Detect transactions significantly above the average invoice amount, as large transactions can be more susceptible to fraud.
4. **Payment Timing Anomalies**
    - Highlight transactions with payment dates suspiciously close to the invoice dates, especially those paid within one day, as they may indicate pre-arranged payments.
    - Identify late payments, specifically those made more than five days after the invoice due date, which could suggest anomalies in payment behavior.
5. **Statistical Analysis for Anomaly Detection**
    - Employ the Interquartile Range (IQR) method to find outliers in the invoice amounts, which could be indicative of fraudulent transactions.
6. **Detailed Analysis of Suspected Fraud**
    - Investigate transactions associated with two individuals suspected of fraud. Analyze their transactions for any of the above irregularities or any patterns that differ significantly from the norm.
7. **Investigation of Payment Patterns**

    - For the suspected individuals, check for multiple transactions on the same date and transactions with round numbers, as these can be potential flags for fraud.
8. **Aggregate and Comparative Analysis**
    - Compile the total sales per customer and identify the top 10 customers by sales volume, sorting from largest to smallest. This can help in identifying any anomalies in sales patterns.
    - Calculate the average invoice amount by gender to determine if there are significant discrepancies that could indicate targeted fraudulent schemes.
9. **Advanced Data Insights**
    - Create a pivot table summarizing invoice amounts by gender to visualize potential patterns or inconsistencies in billing or payment practices.
    - Analyze yearly sales trends to identify any abnormal spikes or drops that may warrant further investigation.

Deliverables for the case are:

- **Python Group:** Submit a Jupyter Notebook containing the Python code for each analysis step. Include comments to explain the rationale behind your methods and your interpretation of the results.
- **Excel Group:** Submit an Excel workbook with separate sheets for each analysis step, utilizing formulas, pivot tables, conditional formatting, and charts as appropriate. Include a summary sheet explaining your methodologies and findings.

The evaluation criteria for the instructor include:

- Completeness of the analysis for each task.
- Accuracy in identifying and explaining potential indicators of fraud.
- Creativity in applying data analysis techniques to uncover insights.
- Clarity and thoroughness in documenting the analysis process and findings.

References

AICPA (May 2021). *AICPA Releases new certificate on data analytics core concepts*. https://www.aicpa-cima.com/news/article/aicpas-releases-new-certificate-on-data-analytics-core-concepts

Bierstaker, J., Lombardi, D., & Wang, W. (2023). WILDCAT grocery stores: a case study on information systems and data analytics. Journal of Emerging Technologies in Accounting.

Cheng, C., Rhoades-Catanach, S. & Watson, L. (2023). *Data analytics for undergraduate tax students: An Alteryx case study for MACRS depreciation*. Issues in Accounting Education, Vol. 38 (4)

Richardson, V. and Watson, M. (2022). *Teach students to address accounting questions using data analytics*. Journal of Accountancy. https://www.journalofaccountancy.com/newsletters/extra-credit/teach-students-to-address-accounting-questions-using-data-analytics.html

Appendix A

```
# FinForensix Generator:

# Import necessary libraries
from google.colab import drive
import pandas as pd
import numpy as np
import random
from datetime import datetime, timedelta

# Function to generate names and genders
def generate_name_gender(name_list):
    male_names = {
        'Stephen', 'Jeremy', 'Frank', 'Gregory', 'Henry', 'James', 'Jason',
        'Jeffrey', 'Jesse', 'John', 'Jordan', 'Paul', 'Robert', 'Russell', 'Ryan',
        'Samuel', 'Scott', 'Sean', 'Steven', 'Thomas', 'Vincent', 'Walter', 'Wayne',
'William'
    }
    name_gender = {}
    for name in name_list:
        first_name = name.split()[0]
        if first_name in male_names:
            name_gender[name] = 'M'
        else:
            name_gender[name] = 'F'
    return name_gender

# Function to generate a unique invoice number
def generate_unique_invoice_number(existing_numbers, length=10):
    chars = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789'
    while True:
        invoice_number = ''.join(random.choices(chars, k=length))
        if invoice_number not in existing_numbers:
            return invoice_number


def benford_law_numbers(count, min_amount, max_amount):
    # Benford's Law probabilities for the first digit
    digits = np.arange(1, 10)
    probabilities = np.log10(1 + 1/digits)

    first_digits = np.random.choice(digits, size=count, p=probabilities)
    other_digits = np.random.uniform(low=0, high=1, size=count) * (max_amount -
min_amount) + min_amount

    amounts = first_digits * 10**(np.floor(np.log10(other_digits)))
    amounts = np.round(amounts, 2)  # Round to two decimal places
    return amounts
```

```
def generate_invoice_data(num_invoices, start_date, end_date, min_amount, max_amount,
name_list):
    name_gender = generate_name_gender(name_list)
    invoices = []
    start_date = datetime.strptime(start_date, '%Y-%m-%d')
    end_date = datetime.strptime(end_date, '%Y-%m-%d')
    date_range = (end_date - start_date).days
    existing_invoice_numbers = set()

    # Generate initial Benford's Law compliant amounts
    benford_amounts = benford_law_numbers(num_invoices, min_amount, max_amount)

    fraudsters = {"Gregory Y Rogers": [], "Hannah U Powell": []}

    for i in range(num_invoices):
        invoice_date = start_date + timedelta(days=random.randint(0, date_range))
        due_date = invoice_date + timedelta(days=30)
        payment_date_variation = random.randint(-2, 5)
        payment_date = due_date + timedelta(days=payment_date_variation)
        customer = random.choice(list(name_gender.items()))

        invoice_number = generate_unique_invoice_number(existing_invoice_numbers)
        existing_invoice_numbers.add(invoice_number)

        is_fraudulent = customer[0] in fraudsters.keys()

        # Generate invoice amount
        if is_fraudulent:
            amount = round(random.uniform(min_amount, max_amount), 2)
        else:
            # Add random cents value between $0.01 and $99.99 to Benford's Law
amounts
            random_cents = round(random.uniform(0.01, 99.99), 2)
            amount = round(benford_amounts[i] + random_cents, 2)
            # Make sure the amount does not exceed the max_amount
            if amount > max_amount:
                amount = round(max_amount - random.uniform(0.01, 0.99), 2)

        invoice = {
            'InvoiceID': i + 1,
            'InvoiceNumber': invoice_number,
            'InvoiceDate': invoice_date.strftime('%Y-%m-%d'),
            'InvoiceDueDate': due_date.strftime('%Y-%m-%d'),
            'InvoiceAmount': amount,
            'PaymentDate': payment_date.strftime('%Y-%m-%d'),
            'CustomerName': customer[0],
            'CustomerGender': customer[1]
        }
        invoices.append(invoice)

        # Store original fraudulent invoices to duplicate later
```

```
            if is_fraudulent:
                fraudsters[customer[0]].append(invoice)
                if len(fraudsters[customer[0]]) == 5:  # Limit to 5 per fraudster
                    break

        # Add duplicates for the collected fraudulent invoices
        for fraudster, fraud_invoices in fraudsters.items():
            for fraud_invoice in fraud_invoices:
                duplicate_invoice = fraud_invoice.copy()
                duplicate_invoice['InvoiceID'] = len(invoices) + 1
                invoices.append(duplicate_invoice)

        return invoices


# Parameters for invoice generation
num_invoices_to_generate = 500000  # Adjusted for demonstration purposes
start_date = '2015-01-01'
end_date = '2024-06-30'
min_invoice_amount = 100
max_invoice_amount = 1000

# Name list
name_list = [
    "Alexis N Thompson", "Amanda M Turner", "Amy A Powell", "Amy F White",
    "Anna C Walker", "Anthony T Foster", "Arthur D Jenkins", "Ashley A Walker",
    "Ashley D Gutierrez", "Ashley I Russell", "Barbara O Howard", "Betty H Walker",
    "Betty V Hill", "Bobby T Sanders", "Bruce B Turner", "Bruce J Adams",
    "Carol H Watson", "Catherine G Hill", "Charles L Gonzalez", "Christina X Reyes",
    "Christine F Stewart", "Danielle V Butler", "Debra P Watson", "Debra R Johnson",
    "Diane P Harris", "Donna I Morales", "Donna N Bell", "Douglas P Cox",
    "Emma B Allen", "Emma L Stewart", "Frances S Smith", "Frank Q Wright",
    "Gary L Hall", "Gregory Y Rogers", "Hannah U Powell", "Heather Z Cooper",
    "Henry L Sanchez", "Henry O Gomez", "Henry R Reyes", "Jacqueline D Walker",
    "Jacqueline G Hughes", "James W Adams", "Janet V Taylor", "Jason L Adams",
    "Jason L Price", "Jean A Hall", "Jeffrey F Sanders", "Jennifer Y Sullivan",
    "Jennifer Z Morales", "Jeremy J Gonzalez", "Jeremy R Johnson", "Jerry C Evans",
    "Jesse B Hughes", "John C Barnes", "Jordan B Garcia", "Judith V Edwards",
    "Julia C Watson", "Kathleen H King", "Kelly N Green", "Kyle I Ortiz",
    "Kyle W Bailey", "Kyle W Sullivan", "Lauren P Carter", "Lawrence V Morgan",
    "Linda R Taylor", "Lisa J Roberts", "Lisa T King", "Lori P Smith",
    "Lori X Anderson", "Maria H Lewis", "Melissa H Morales", "Noah Z Powell",
    "Patricia P Thompson", "Paul H Anderson", "Robert R Myers", "Robert S Reed",
    "Russell Y Fisher", "Ruth G Johnson", "Ruth S Garcia", "Ryan H Rodriguez",
    "Samantha N Gray", "Samuel Y Morgan", "Sandra K Flores", "Sara S Scott",
    "Scott X Ross", "Sean L Kelly", "Sean S Howard", "Sean X Wilson",
    "Shirley B Smith", "Stephen S King", "Steven Q Young", "Teresa F Rogers",
    "Teresa J Cox", "Thomas R Williams", "Tiffany O Bailey", "Vincent J Barnes",
    "Virginia A Brooks", "Walter M Clark", "Wayne S Cook", "William U Barnes"
]
```

```
# Generate the invoices, including fraudulent transactions
invoices = generate_invoice_data(num_invoices_to_generate, start_date, end_date,
min_invoice_amount, max_invoice_amount, name_list)

# Convert to DataFrame
df_invoices = pd.DataFrame(invoices)

drive.mount('/content/drive')

file_path = '/content/drive/My Drive/Invoices.csv'
df_invoices.to_csv(file_path, index=False)

print(f"Invoice data saved to {file_path}")
```

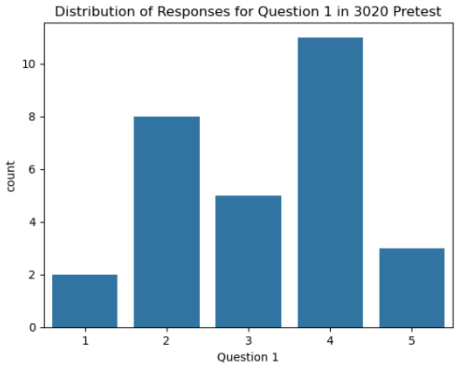
Appendix B - Python For Teachers:

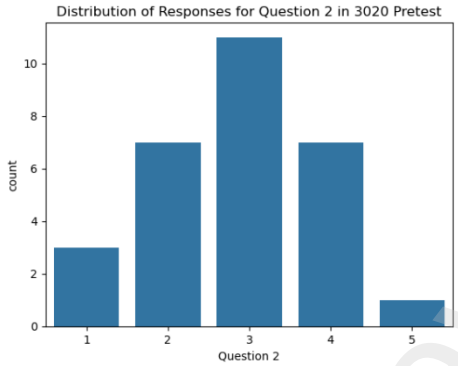
```
from google.colab import drive
import pandas as pd

# Mount Google Drive
drive.mount('/content/drive')

# Full path to the dataset
file_path = '/content/drive/My Drive/UtahTech/ISA3020/Invoices.csv'

# Setup stage: Load the dataset into a DataFrame
df = pd.read_csv(file_path)

# Display the data types of the DataFrame columns
print(df.dtypes)

# 2. Convert Dates from Strings to Datetime
df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'])
df['InvoiceDueDate'] = pd.to_datetime(df['InvoiceDueDate'])
df['PaymentDate'] = pd.to_datetime(df['PaymentDate'])
# Display the data types of the DataFrame columns
print(df.dtypes)

# 3. Please check for duplicate invoice numbers in the dataset?
# First, sort the DataFrame by 'InvoiceNumber'
sorted_df = df.sort_values(by='InvoiceNumber')

# Then, filter to find duplicates in 'InvoiceNumber' and ensure they are listed
together
duplicates = sorted_df[sorted_df.duplicated(['InvoiceNumber'], keep=False)]
duplicates

# 4. Please list the frequency of transactions for each customer?
df['CustomerName'].value_counts()

# 5. Please list payment dates that occur before the invoice date?
df[df['PaymentDate'] < df['InvoiceDate']]

# 6. Please list rounded invoice amounts, which could be indicative of fraud?
df[df['InvoiceAmount'] % 1 == 0]

# 7. Please identify large transactions that are significantly higher than the
average?
average_amount = df['InvoiceAmount'].mean()
df[df['InvoiceAmount'] > 1.2 * average_amount]

# 8. Find transactions where the payment date is suspiciously close to the invoice
date?
from datetime import timedelta
df[df['PaymentDate'] - df['InvoiceDate'] <= timedelta(days=1)]
```

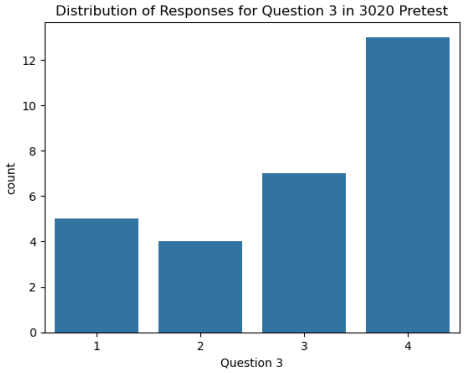
```
# 9. Locate anomalies in payment patterns, such as payments always at the edge of the
payment term.
df[df['PaymentDate'] - df['InvoiceDueDate'] >= timedelta(days=5)]

# 10. Identify outliers in the amounts column.  Employ the Interquartile Range (IQR)
method
df['InvoiceAmount'].quantile([0.25, 0.75])

# 11. Pinpoint the exact fraudulent transactions for person 1.
df[df['CustomerName'] == 'Gregory Y Rogers']

# 12. Find if person 2 has multiple transactions on the same date?
 # Filter for transactions by Hannah U Powell
filtered_df = df[df['CustomerName'] == 'Hannah U Powell']

# Group by 'InvoiceDate' to find multiple transactions on the same date
grouped = filtered_df.groupby('InvoiceDate').size()

# To identify dates with multiple transactions
multiple_transactions = grouped[grouped > 1]

# Displaying the result
print(multiple_transactions)

# 13. Find those instances where the payment date is significantly after the due
date. Up to 4 days.
df[df['PaymentDate'] > df['InvoiceDueDate'] + timedelta(days=4)]

# 14. Find if there are any transactions where the invoice amount is a round number
for Person 2.
df[(df['CustomerName'] == 'Hannah U Powell') & (df['InvoiceAmount'] % 1 == 0)]

# 15. List the total sales per customer for the top 10 customers
# and sort them from largest to smallest
total_sales_per_customer_sorted =
df.groupby('CustomerName')['InvoiceAmount'].sum().sort_values(ascending=False).head(1
0)

# Display the top 10 customers with the largest total sales
print(total_sales_per_customer_sorted)

# 16. Please list the average invoice amount for each gender.
# Calculate the average Invoice Amount for each gender
average_invoice_amount_by_gender =
df.groupby('CustomerGender')['InvoiceAmount'].mean()

# Display the average Invoice Amount for each gender
print(average_invoice_amount_by_gender)

# 17. Create a Pivot Table for Invoice Amounts by Gender
```

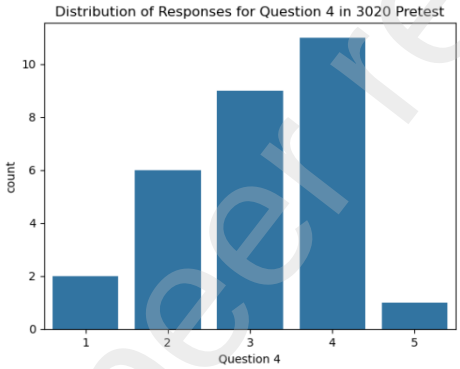
```
pivot_table_gender = df.pivot_table(values='InvoiceAmount', index='CustomerGender',
aggfunc='sum')
# Set display format for floating numbers to currency style
pd.options.display.float_format = '${:,.2f}'.format

pivot_table_gender

# 18. Analyze Yearly Sales Trends
df['InvoiceYear'] = df['InvoiceDate'].dt.year
yearly_sales = df.groupby('InvoiceYear')['InvoiceAmount'].sum()
print(f"Yearly Sales Trends:\n{yearly_sales}")

pip install nbconvert # 19. install the nbconvert library

# 20. Mount Google Drive if not already mounted
from google.colab import drive
drive.mount('/content/drive', force_remount=True)

# Full path to the notebook
notebook_path = '/content/drive/My Drive/Invoice_Basic_Tasks.ipynb'

# 21. Convert the notebook to an HTML file.
!jupyter nbconvert --to html "{notebook_path}"
```

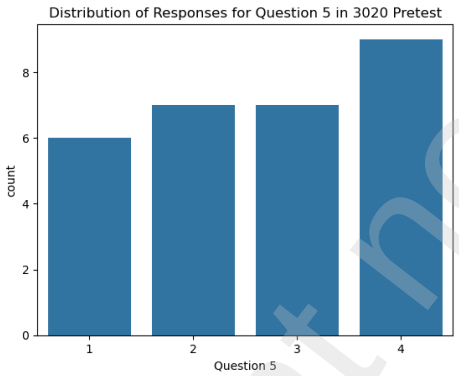
Appendix C
Further Statistical Results of Efficacy

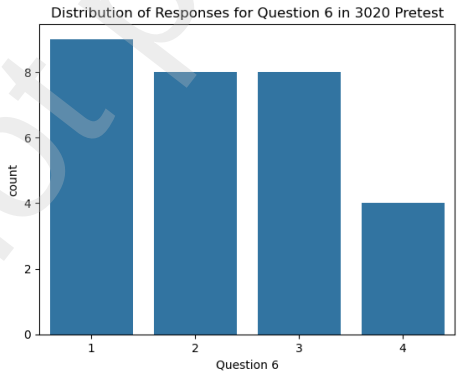


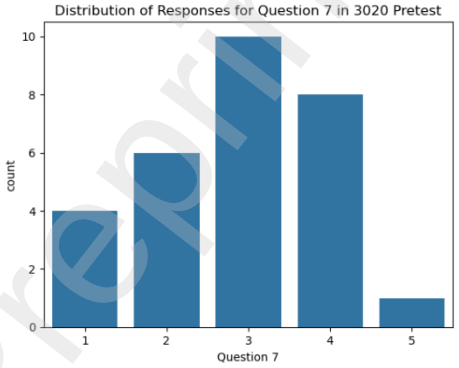


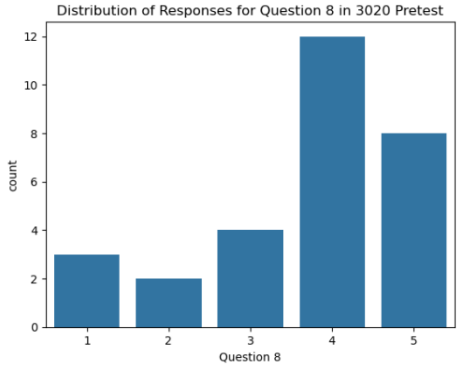


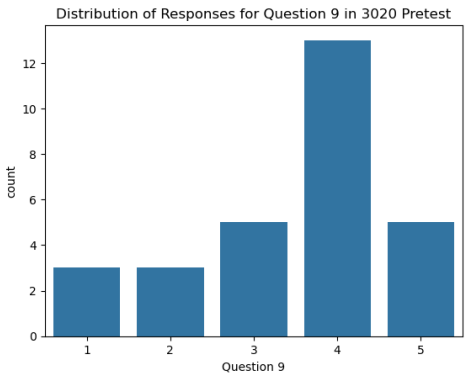


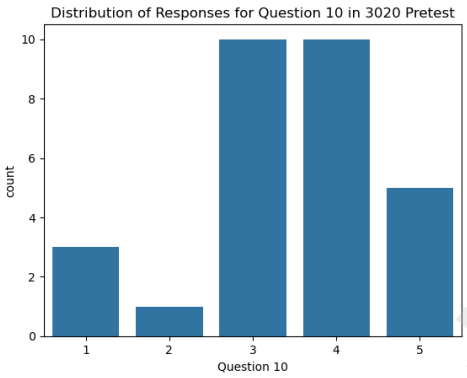


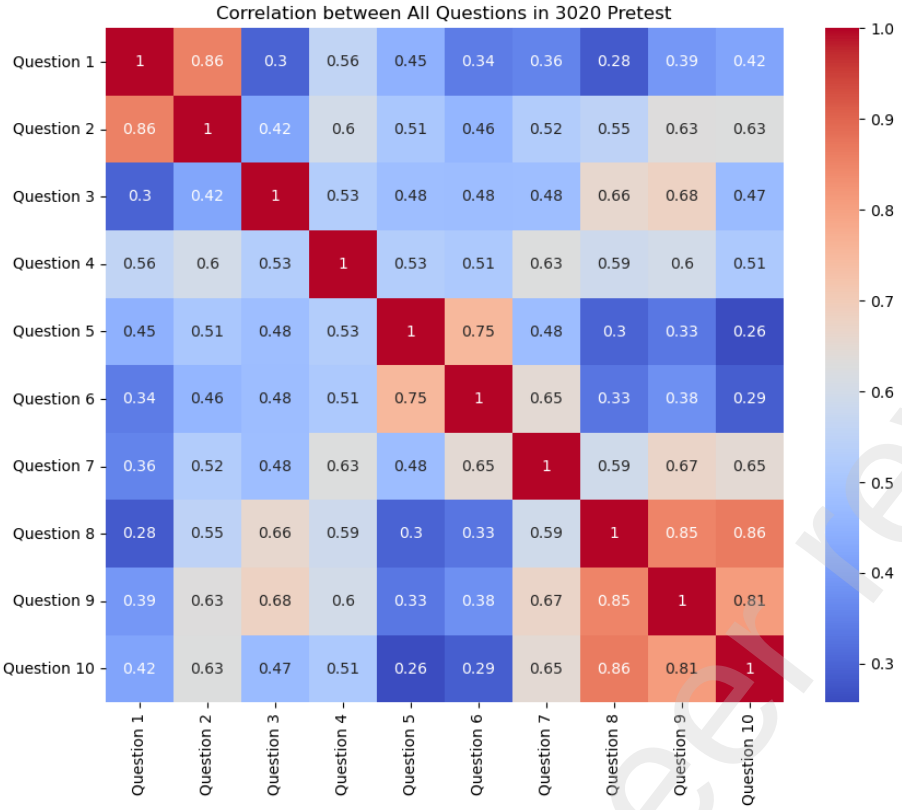


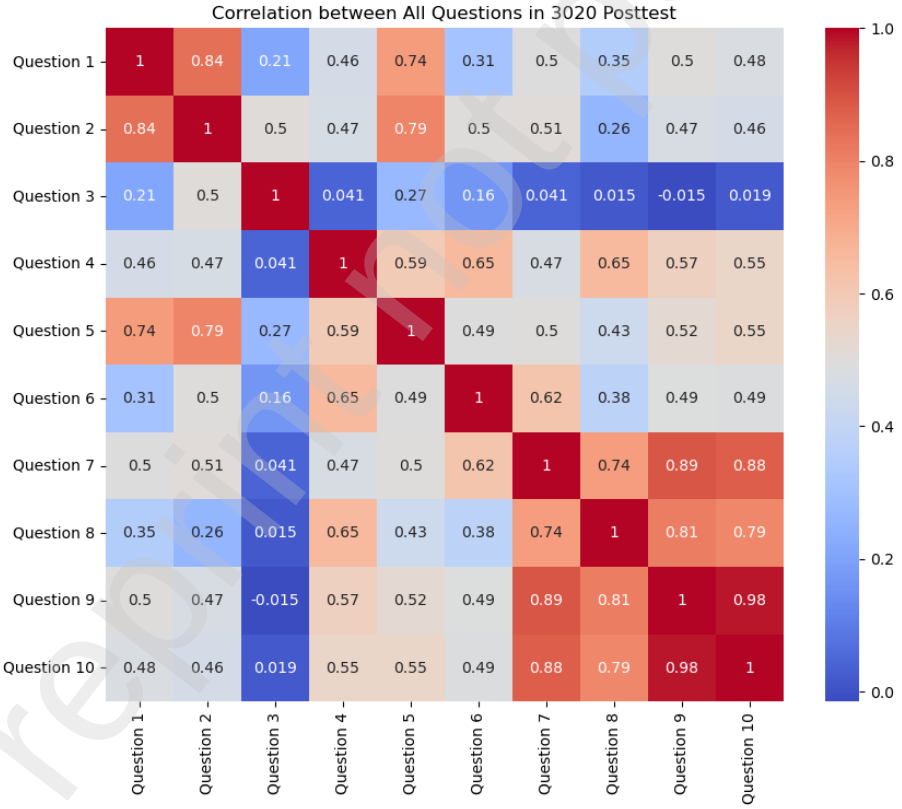


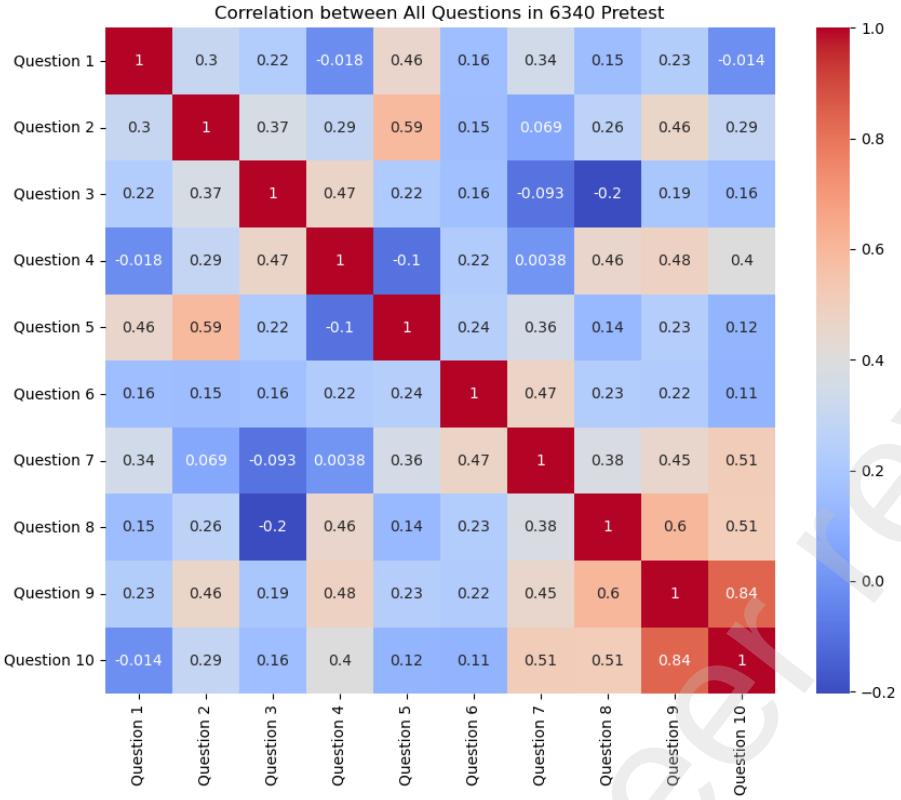


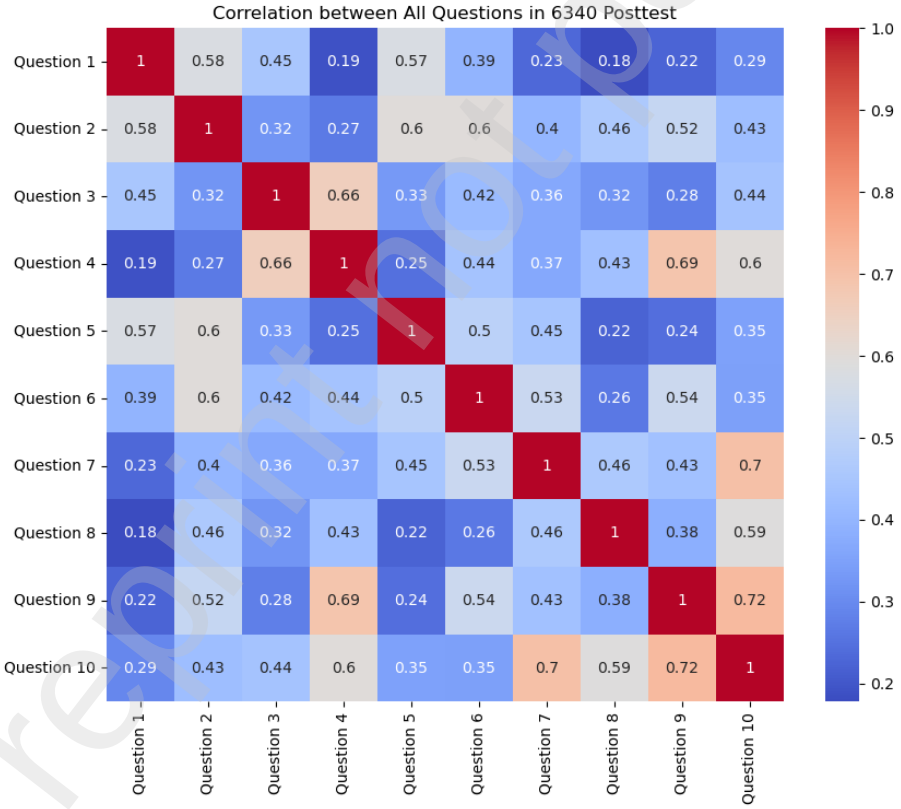


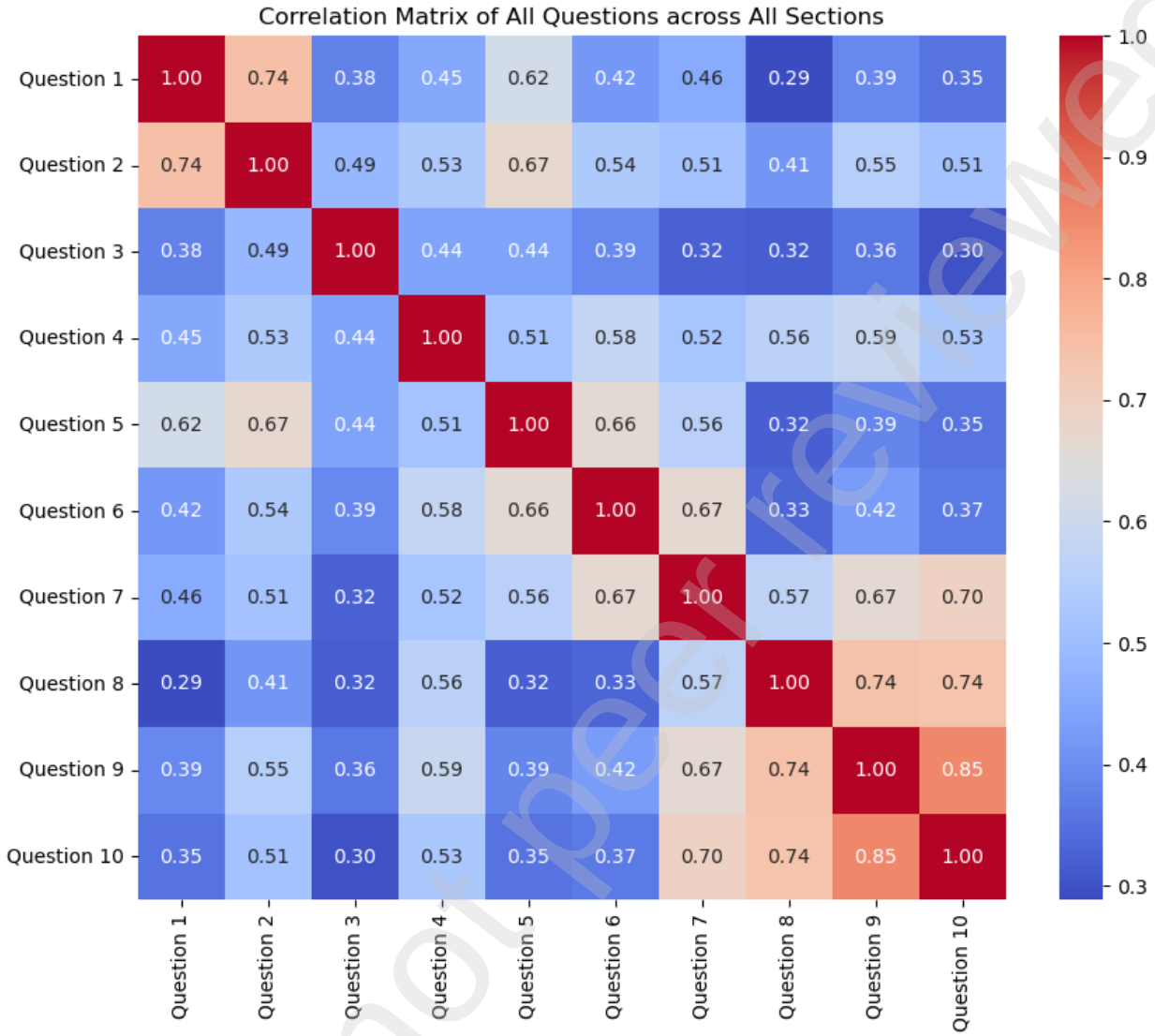












Appendix D

Teaching Notes Outline: From Theory to Practice

I. Introduction

Background Information

The rapidly evolving landscape of financial transactions, marked by increasing complexity and volume, demands robust mechanisms for fraud detection. Traditional methods often fall short due to their limited scope and inability to adapt to the sophisticated techniques employed by fraudsters. This gap underscores the critical need for integrating data analytics into accounting education, preparing the next generation of accounting professionals to effectively combat fraud.

In the academic paper "From Theory to Practice: Using Python and Excel to Uncover Financial Fraud in Education," we advocate for a dual approach that harnesses both Python and Excel as essential tools in the educational toolkit for accounting students. This integration aims to cultivate "bilingual" professionals who are fluent not only in accounting principles but also in the analytical capabilities provided by modern data tools.

By incorporating Python, students can leverage its powerful data manipulation capabilities to simulate real-world scenarios, enhancing their ability to parse, analyze, and derive meaningful insights from complex datasets. Excel complements this by offering a familiar interface for further data analysis and visualization, making the transition from theoretical knowledge to practical application seamless.

It is important to note that the FinForensix Generator tool is used by educators to create the synthetic dataset for the case study; it is not part of the materials directly handled by the students. This tool allows the professor to generate a realistic, yet controlled environment where both legitimate and fraudulent transactions are simulated. The students then engage with this dataset, applying their analytical skills to detect and understand the characteristics of financial fraud.

This hands-on approach not only boosts students' data analytic skills but also enhances their critical thinking and problem-solving abilities. Moreover, it prepares them to communicate their findings effectively and efficiently, a crucial skill in professional settings.

Ultimately, the goal of integrating data analytics into accounting education extends beyond merely teaching technical skills; it is about fostering a mindset that blends analytical problem-solving with financial expertise. This educational strategy ensures that future accountants are not only proficient in using advanced tools but are also adept at navigating and resolving complex financial issues that characterize the modern economic landscape.

II. Case Introduction

Overview of the Case Study

The case study, titled "From Theory to Practice: Using Python and Excel to Uncover Financial Fraud in Education," is designed to immerse accounting students in the practical aspects of fraud detection through a carefully crafted dataset of financial transactions. This dataset, prepared using the FinForensix Generator by the instructor, contains a mix of legitimate and fraudulent transactions that mimic real-world financial environments. It's important to note that while the FinForensix Generator plays a crucial role in creating these data scenarios, it remains a background tool used by educators to set the stage for student engagement with the actual data.

Educational Value

Students engage directly with the dataset, applying their analytical skills to uncover signs of fraudulent activities. This process is integral to their learning, as it not only allows them to apply their theoretical knowledge in a practical setting but also challenges them to use critical thinking and investigative skills in a simulated professional context. The case study approach ensures that students not only learn how to use tools like Python and Excel for data analysis but also understand the nuances and complexities involved in financial fraud detection.

Customizability and Realism

The case study is designed with flexibility in mind, allowing instructors to adjust the complexity of the data according to the course level. This customization ensures that students at different stages of their educational journey can be adequately challenged and engaged. The realism of the dataset is enhanced by features such as varying transaction dates, customer details, and transaction amounts, along with the incorporation of common fraud indicators like duplicate invoices and unusually large transactions. These elements provide students with a realistic and immersive experience that closely replicates real-world accounting challenges.

Integration with Technological Tools

Students analyze the dataset using either Python or Excel, based on the course requirements or their personal learning preferences. This dual-tool approach not only familiarizes them with popular data analysis software but also provides them with the flexibility to approach problem-solving from different technological perspectives. Whether they are scripting in Python to automate data analysis tasks or using Excel to visually interpret data through charts and pivot tables, students gain valuable hands-on experience that is directly applicable to modern accounting and auditing roles.

In summary, the case study serves as a core component of the educational experience, offering students a dynamic and interactive environment to hone their skills. The focus is on engaging students with realistic data analysis tasks that prepare them for the complexities of the financial world, ensuring they are not only proficient in technical skills but also adept at critical analysis and problem-solving in their professional careers.

III. Case Requirements

Technical Skills and Prerequisites

For this case study, students are required to have specific technical skills in Python and Excel, alongside a fundamental understanding of statistical concepts. Below are the detailed prerequisites:

- **Python Skills**: Students should be adept at basic Python programming, including familiarity with data structures (like lists and dictionaries), loops, conditionals, and functions. Proficiency with Python's data science libraries such as Pandas for data manipulation, NumPy for numerical data, and Matplotlib for data visualization is essential. Prior completion of a Python programming course or a related business data analytics course is recommended.
- **Google Colab Knowledge**: Students must be proficient in using Google Colab, which will be the primary platform for Python scripting in this case study. They should know how to create, share, and manage notebooks within Google Colab.
- **Excel Skills**: Students need to be proficient in Excel for data analysis tasks such as using formulas, creating pivot tables, and generating charts. This knowledge will be crucial for part of the data analysis process that complements the Python analysis.
- **Statistical Knowledge**: A basic understanding of statistics is required, including the ability to compute measures like mean, median, standard deviation, and recognize outliers. Familiarity with statistical inference will aid in interpreting data and hypothesizing about observed patterns.

Setup and Implementation

To facilitate an effective learning experience, here are the setup and implementation guidelines tailored specifically for using Google Colab and Excel:

- **Setting Up Google Colab**:
    1. **Accessing Colab**: Direct students to sign in to their Google accounts and access Google Colab online. This ensures that all students are working within the same environment and eliminates compatibility issues.

2. **Loading Data**: Instruct students on how to upload the CSV file provided by the FinForensix Generator into Google Drive and how to load this data into a Colab notebook using the google.colab module. This step is crucial for accessing and manipulating data within the notebook.
3. **Utilizing Resources**: Encourage the use of Colab's integrated features such as code snippets and its extensive library support to enhance their data analysis capabilities.

- **Setting Up Excel**:
    1. **Software Installation**: Ensure that students have access to Microsoft Excel. If not available, recommend Google Sheets as a free alternative with similar functionalities.
    2. **Data Handling**: Provide guidance on importing the CSV file into Excel or Google Sheets and demonstrate how to effectively utilize tools like filters, pivot tables, and chart functions for data analysis.

Ensuring Accessibility

- **Resource Accessibility**: Ensure all students have access to Google accounts and stable internet connections for accessing Google Colab and Google Drive.
- **Remote Learning Accommodations**: Make sure that digital resources such as video tutorials on using Google Colab, installation guides for necessary software, and links to online help forums are readily available.
- **Support Systems**: Set up a structured support system where students can easily request help. This could include scheduled virtual office hours, an online Q&A forum, or dedicated support sessions for technical issues.

IV. Learning Objectives

This case study is meticulously crafted to develop a broad spectrum of analytical skills among accounting students. These objectives are tailored to meet the technical demands of modern accounting and enhance students' capabilities to handle complex financial analyses in their professional futures. Here's an outline of the learning goals set for participants in this case study:

Detailed Goals

Enhanced Data Analysis Capabilities:

- **Statistical Techniques:**

    - **Benford's Law for Fraud Detection:** Students will delve into Benford's Law, learning to apply this statistical technique to large sets of numerical data to identify irregularities and potential fraud. They will explore how the law's prediction of frequency distribution of first digits in natural datasets can be used to detect anomalies in financial records.
    - **Descriptive Statistics:** The curriculum covers the use of descriptive statistical measures such as mean, median, mode, variance, and standard deviation. Students will learn how these metrics can be utilized to understand the distribution, central tendency, and variability of data, helping identify outliers and unusual patterns that might suggest fraudulent activities.
- **Real-World Data Application:**
    - **Identifying Anomalies:** Participants will use their statistical knowledge to recognize patterns and anomalies in transaction data. They will learn how to detect deviations from expected patterns that could suggest fraudulent activities, using statistical tools to distinguish between normal variations and suspicious discrepancies.
    - **Data Validation and Cleaning:** Accurate data analysis begins with clean data. Students will master techniques for inspecting, cleaning, transforming, and validating datasets to ensure the integrity and accuracy of their analyses. This includes handling missing values, correcting data entry errors, and ensuring consistency across data sources, which are crucial skills for any data analyst.

Develop Proficiency in Using Analytical Software Tools:

- **Software Skills Enhancement**: Gain proficiency in using Python with libraries such as Pandas for data manipulation and NumPy for numerical calculations, along with mastering advanced Excel functionalities like pivot tables, conditional formatting, and basic data visualization tools.
- **Versatility Across Platforms**: Develop the ability to switch seamlessly between Python and Excel, leveraging the strengths of each platform to enhance data analysis and reporting efficiency.

Integrate Software Tools with Traditional Accounting Practices:

- **Technology-Enhanced Traditional Methods**: Demonstrate how traditional accounting methods can be enhanced with digital tools, increasing the efficiency and accuracy of financial practices.
- **Automation for Accuracy**: Employ software to automate the generation and analysis of financial transactions, improving the precision of financial reports and facilitating more robust fraud detection.

Foster Critical Thinking and Problem-Solving Skills:

- **Complex Problem Solving**: Engage in exercises that simulate real-world fraud detection scenarios to enhance critical thinking and analytical skills.
- **Strategic Decision Making**: Learn to analyze complex data sets and derive actionable insights, making informed decisions to tackle financial challenges effectively.

Prepare for Future Careers in Accounting:

- **Career Readiness**: Equip students with contemporary skills necessary for roles in forensic accounting and financial analysis as these fields increasingly incorporate data analytics.
- **Ethical Standards and Professional Responsibility**: Highlight the importance of ethical considerations and professional integrity in managing and analyzing financial data.

V. Implementation Guidance

- **Step-by-Step Instructions**:

For instructors:

1. Open a browser of your choice & into your google account.
2. Navigate to https://colab.research.google.com/ and from the menu choose file and new notebook.
3. Copy and paste the code from appendix A into a window in this new notebook. You may optionally adjust the begin date, end date, minimum amount for the invoice as well as the maximum amount for the invoice in the programming code. If you leave the code unchanged, the beginning date will be January 1st, 2015, and the ending date will be June 30th, 2024. The minimum amount is defaulted to $100, and the maximum amount is defaulted to $1000.
4. The invoices file will be saved to the Google Drive at the root or outermost directory. If you want to save it elsewhere, modify the directory path in the code which is this line: `file_path = '/content/drive/My Drive/Invoices.csv'`
5. When you execute this code, you will be directed to allow the Python program access to your Google Drive. Click on connect to Google Drive.
6. In the pop-up window, click on your Google account.
7. Click continue in the next pop-up window.
8. In the next pop-up window, scroll to the bottom and click continue again.
9. The Python will build the invoices data file and save it to the Google Drive.
10. Download the csv file and disseminate it to the students.

For Students:

1. Acquire the Invoices.csv from your professor.
2. Navigate to https://www.google.com/drive/ and sign in to you google account.
3. In the upper left corner, click the + New button and scroll to the file upload option
4. Locate the Invoices.csv file on your local computer, highlight it and click open.
5. Open a new tab on the browser and navigate to https://colab.research.google.com/ and from the menu, choose file and new notebook.
6. Rename the notebook by clicking in the upper left corner where it says untitled.ipynb and renaming it to Invoice_Basic_Tasks.ipynb.
7. From Appendix B in the paper, copy and paste this code to the first window:

```
from google.colab import drive
import pandas as pd

# Mount Google Drive
drive.mount('/content/drive')

# Full path to the dataset
file_path = '/content/drive/My Drive/Invoices.csv'

# Setup stage: Load the dataset into a DataFrame
df = pd.read_csv(file_path)

# Display the data types of the DataFrame columns
print(df.dtypes)
```

8. When you execute this code, you will be directed to allow the Python program access to your Google Drive. Click on connect to Google Drive.
9. In the pop-up window, click on your Google account.
10. Click continue in the next pop-up window.
11. In the next pop-up window, scroll to the bottom and click continue again.
12. From this first window, the output should look like this:

```
Mounted at /content/drive
InvoiceID            int64
InvoiceNumber       object
InvoiceDate         object
InvoiceDueDate      object
InvoiceAmount      float64
PaymentDate         object
CustomerName        object
CustomerGender      object
dtype: object
```

13. Copy and paste the code from Appendix B to the next window which is going to change the date columns to a date datatype and display the new datatypes.

```
# 1. Convert Dates from Strings to Datetime
```

```
df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'])
df['InvoiceDueDate'] = pd.to_datetime(df['InvoiceDueDate'])
df['PaymentDate'] = pd.to_datetime(df['PaymentDate'])
# Display the data types of the DataFrame columns
print(df.dtypes)
```

14. Copy and paste the code from Appendix B to a separate window for every task from this point on. Please check for duplicate invoice numbers in the dataset.

```
# 3. Please check for duplicate invoice numbers in the dataset.
df[df.duplicated(['InvoiceNumber'], keep=False)]
```

15. Please list the frequency of transactions for each customer.

```
# 4. Please list the frequency of transactions for each customer.
df['CustomerName'].value_counts()
```

16. Please list payment dates that occur before the invoice date.

```
# 5. Please list payment dates that occur before the invoice
date.
df[df['PaymentDate'] < df['InvoiceDate']]
```

17. Please list rounded invoice amounts, which could be indicative of fraud.

```
# 6. Please list rounded invoice amounts, which could be
indicative of fraud.
df[df['InvoiceAmount'] % 1 == 0]
```

18. Please identify large transactions that are significantly higher than the average.

```
# 7. Please identify large transactions that are significantly
higher than the average.
average_amount = df['InvoiceAmount'].mean()
df[df['InvoiceAmount'] > 1.2 * average_amount]
```

19. Please find transactions where the payment date is suspiciously close to the invoice date.

```
# 8. Find transactions where the payment date is suspiciously
close to the invoice date?
from datetime import timedelta
df[df['PaymentDate'] - df['InvoiceDate'] <= timedelta(days=1)]
```

20. Please Locate anomalies in payment patterns, such as payments always at the edge of the payment term.

```
# 9. Locate anomalies in payment patterns, such as payments
always at the edge of the payment term.
df[df['PaymentDate'] - df['InvoiceDueDate'] >= timedelta(days=5)]
```

21. Please identify outliers in the amounts column. Employ the Interquartile Range (IQR) method .

```
# 10. Identify outliers in the amounts column.  Employ the
Interquartile Range (IQR) method
df['InvoiceAmount'].quantile([0.25, 0.75])
```

22. Please pinpoint the exact fraudulent transactions for person 1.

```
# 11. Pinpoint the exact fraudulent transactions for person 1.
df[df['CustomerName'] == 'Gregory Y Rogers']
```

23. Please find if person 2 has multiple transactions on the same date.

```
# 12. Find if person 2 has multiple transactions on the same
date.
 # Filter for transactions by Hannah U Powell
filtered_df = df[df['CustomerName'] == 'Hannah U Powell']
```

24. Please group by 'InvoiceDate' to find multiple transactions on the same date.

```
# Group by 'InvoiceDate' to find multiple transactions on the
same date.
grouped = filtered_df.groupby('InvoiceDate').size()
```

25. Please identify dates with multiple transactions.

```
# To identify dates with multiple transactions
multiple_transactions = grouped[grouped > 1]
```

```
# Displaying the result
print(multiple_transactions)
```

26. Please find those instances where the payment date is significantly after the due date; up to 4 days.

```
# 13. Find those instances where the payment date is
significantly after the due date; up to 4 days.
df[df['PaymentDate'] > df['InvoiceDueDate'] + timedelta(days=4)]
```

27. Please find if there are any transactions where the invoice amount is a round number for Person 2.

```
# 14. Find if there are any transactions where the invoice amount
is a round number for Person 2.
df[(df['CustomerName'] == 'Hannah U Powell') &
(df['InvoiceAmount'] % 1 == 0)]
```

28. Please list the total sales per customer for the top 10 customers and sort them from largest to smallest.

```
# 15. List the total sales per customer for the top 10 customers
# and sort them from largest to smallest.
total_sales_per_customer_sorted =
df.groupby('CustomerName')['InvoiceAmount'].sum().sort_values(asc
ending=False).head(10)

# Display the top 10 customers with the largest total sales
print(total_sales_per_customer_sorted)
```

29. Please list the average invoice amount for each gender.

```
# 16. Please list the average invoice amount for each gender.
# Calculate the average Invoice Amount for each gender.
average_invoice_amount_by_gender =
df.groupby('CustomerGender')['InvoiceAmount'].mean()

# Display the average Invoice Amount for each gender
print(average_invoice_amount_by_gender)
```

30. Please create a pivot table for invoice amounts by gender.

```
# 17. Create a Pivot Table for Invoice Amounts by Gender.
pivot_table_gender = df.pivot_table(values='InvoiceAmount',
index='CustomerGender', aggfunc='sum')
# Set display format for floating numbers to currency style
pd.options.display.float_format = '${:,.2f}'.format
pivot_table_gender
```

31. Please analyze yearly sales trends.

```
# 18. Analyze Yearly Sales Trends
df['InvoiceYear'] = df['InvoiceDate'].dt.year
yearly_sales = df.groupby('InvoiceYear')['InvoiceAmount'].sum()
print(f"Yearly Sales Trends:\n{yearly_sales}")
```

32. Please install the nbconvert library in order to convert the notebook to an HTML file suitable for submission.

```
pip install nbconvert # 19. install the nbconvert library
```

33. Please run the appropriate code to mount the google drive in preparation for converting the notebook file to an HTML file.

```
# 20. Mount Google Drive if not already mounted
from google.colab import drive
drive.mount('/content/drive', force_remount=True)

# Full path to the notebook
notebook_path = '/content/drive/My
Drive/Invoice_Basic_Tasks.ipynb'
```

34. Convert the Jupyter notebook file to HTML.

```
# 21. Convert the notebook to an HTML file.
!jupyter nbconvert --to html "{notebook_path}"
```

Discussion Points

- Ethical Considerations in Fraud Detection: Engage students in discussing the ethical implications of data handling and fraud detection. What responsibilities do accountants

have when handling sensitive data? How should they act upon discovering fraudulent activities?

- Analytical Techniques vs. Practical Outcomes: Discuss the balance between using advanced analytical techniques and the practical outcomes these techniques yield. Are more complex methods always better?
- Limitations of Tools: Explore the limitations of Python and Excel in detecting fraud. What are the inherent limitations of these tools, and how can they affect the analysis?
- Real-world Application: How can the skills learned in this case study translate to real-world scenarios? Discuss instances where these skills could directly impact an accountant's professional duties.
- Future of Accounting with AI and Machine Learning: Consider the future role of AI and machine learning in accounting. How might these technologies change traditional practices in fraud detection and financial reporting?

Potential Challenges and Solutions

- Complexity of Data Analysis Tools: Some students may struggle with the complexity of Python or the detailed functions in Excel. To mitigate this, provide supplementary tutorial sessions or resources that can assist in learning these tools.
- Data Overload: The extensive data provided might overwhelm some students. Encourage students to focus on one section at a time and use collaborative methods like study groups to enhance understanding.
- Technical Issues with Software: Issues such as software malfunctions or errors in code can impede progress. Ensure that there are IT support systems in place to help students troubleshoot these problems quickly.
- Interpretation of Results: Students might find it challenging to interpret analytical results correctly. To address this, conduct workshops on data interpretation and encourage peer review of findings to foster a deeper understanding and accuracy in analysis.
- Engagement with the Case Study: Maintaining engagement throughout the case study can be challenging. Incorporate interactive elements such as quizzes, group discussions, and competitive tasks to keep students actively involved.
- Potential Challenges and Solutions: Address common issues that might arise during the case study and propose solutions.

VI. Assessment of Learning Outcomes

Overview

The assessment for the "From Theory to Practice: Using Python and Excel to Uncover Financial Fraud in Education" case study is designed to evaluate students' understanding and application of analytical techniques in fraud detection. The evaluation will focus on their ability to utilize Python and Excel to analyze data, identify potentially fraudulent activities, and communicate their findings effectively.

Assessment Components

1. Technical Skills Evaluation
    - Python Group: Students will submit a Jupyter Notebook containing the Python code for each analysis step. The code should be well-commented to explain the rationale behind the methods used and the interpretation of the results.
    - Excel Group: Students will submit an Excel workbook with separate sheets for each analysis step, utilizing formulas, pivot tables, conditional formatting, and charts as appropriate. A summary sheet should explain the methodologies and findings.
2. Analytical Report Writing
    - All students are required to compile a comprehensive report detailing the analysis process, findings, and implications of the fraud detection results. The report should demonstrate a clear understanding of the tools used and the relevance of the findings in a real-world context.
3. Presentation of Findings
    - Students will present their findings in a class presentation, highlighting key insights, challenges encountered during the analysis, and recommendations based on the data. This will assess their communication skills and ability to engage an audience with complex information.

Evaluation Criteria

- Completeness: All tasks must be completed with detailed analysis provided for each step. Missing components or incomplete analyses will impact the overall score.
- Accuracy: The ability to correctly identify and explain potential indicators of fraud based on the data analyzed. This includes proper use of statistical methods and correct interpretation of data patterns.
- Creativity: Application of innovative analytical techniques and thoughtful approaches in tackling the fraud detection challenges. Creativity in problem-solving and data presentation will be highly regarded.

- Clarity and Thoroughness: The documentation of the analysis process and findings must be clear, well-organized, and comprehensive. Reports and presentations should be easily understandable, with technical terms adequately explained.

Assessment Tools - The following rubric is designed to assess the student's performance in the fraud detection case study. Each category has specific criteria and is graded on a scale from 1 to 5, where 1 indicates "Needs Improvement" and 5 indicates "Exemplary."

1. Technical Proficiency (30%)

- **1 (Poor)**: Frequent errors in code or formulae; the submitted work does not run or produce results.
- **2 (Fair)**: Occasional errors in code or formulae that affect some results; basic understanding of Python or Excel.
- **3 (Good)**: Minor errors in code or formulae; demonstrates a functional understanding of Python or Excel with correct results.
- **4 (Very Good)**: Code or formulae are mostly correct with some advanced features used; demonstrates a strong proficiency in Python or Excel.
- **5 (Exemplary)**: Code or formulae are correct and efficient; extensive use of advanced features in Python or Excel demonstrating high technical proficiency.

2. Analysis and Interpretation (30%)

- **1 (Poor)**: Analysis is superficial or incorrect; demonstrates a lack of understanding of the data and its implications.
- **2 (Fair)**: Basic analysis performed; some correct conclusions drawn but lacks depth.
- **3 (Good)**: Competent analysis with logical conclusions; good understanding of the data's implications with minor inaccuracies.
- **4 (Very Good)**: Thorough and accurate analysis; conclusions are well-supported by the data.
- **5 (Exemplary)**: Exceptional analysis that is both deep and broad; insightful conclusions that are fully supported by detailed evidence from the data.

3. Identification of Fraud Indicators (20%)

- **1 (Poor)**: Fails to identify key fraud indicators; misses critical data points.
- **2 (Fair)**: Identifies only the most obvious fraud indicators; some critical data points overlooked.

- **3 (Good)**: Correctly identifies many fraud indicators; occasionally misses some subtler indicators.
- **4 (Very Good)**: Accurately identifies most fraud indicators including some subtler ones; very thorough.
- **5 (Exemplary)**: Excellently identifies all relevant fraud indicators; includes detailed explanations of how each indicator points to potential fraud.

4. Creativity and Problem-Solving (10%)

- **1 (Poor)**: Little to no original thought; relies heavily on basic or provided methods.
- **2 (Fair)**: Some attempts at originality or creative approaches; basic but functional solutions.
- **3 (Good)**: Demonstrates creative problem-solving skills; good use of innovative approaches to enhance the analysis.
- **4 (Very Good)**: Shows advanced problem-solving skills; creative and effective use of tools and methods.
- **5 (Exemplary)**: Outstanding creativity; introduces novel and highly effective methods and solutions.

5. Communication and Presentation (10%)

- **1 (Poor)**: Inadequate documentation and explanation of methods and findings; hard to follow.
- **2 (Fair)**: Sufficient documentation; explanations lack clarity or detail but communicate basic ideas.
- **3 (Good)**: Clear documentation and explanations; well-organized presentation of findings.
- **4 (Very Good)**: Very clear and detailed documentation; effectively communicates complex ideas in an understandable manner.
- **5 (Exemplary)**: Exceptional communication; excellent documentation and presentation that enhances understanding of the analysis.

Implementation

Students will be evaluated based on their submitted work and presentations. Each rubric category should be scored independently, and feedback should be provided to guide improvements. Final grades are calculated by weighing each category according to the

percentages provided. This comprehensive approach ensures a balanced assessment of technical skills, analytical depth, problem-solving ability, creativity, and communication.

Submission Deadlines

- Project Submissions: All projects (Python notebooks, Excel workbooks, and analytical reports) must be submitted by the end of the semester, two weeks before the final exam period.

This structured assessment approach aims to rigorously evaluate the technical proficiency, analytical thinking, and communication skills of students, preparing them for complex real-world challenges in accounting and fraud detection.

Evaluation Criteria: Define how students' work will be assessed, including clarity of analysis, accuracy in identifying fraud indicators, and the ability to use analytical tools effectively.

- Feedback Mechanisms: Describe how feedback will be provided to students, emphasizing areas for improvement and acknowledging analytical successes.

VII. Additional Resources

Supplementary Materials

1. Books and Articles:
    - *Forensic and Investigative Accounting* by D. Larry Crumbley, Lester E. Heitger, G. Stevenson Smith: This textbook offers a comprehensive view on the forensic accounting field, covering theories and practical approaches in fraud detection.
    - *Data Science for Business* by Foster Provost and Tom Fawcett: A key resource for understanding how data science techniques, including those used in fraud detection, can be applied to business problems.
    - *Benford's Law: Applications for Forensic Accounting, Auditing, and Fraud Detection* by Mark J. Nigrini: A detailed exploration of Benford's Law and its application in real-world scenarios.
2. **Online Resources:**
    - AICPA's Digital Mindset Pack: A series of e-learning modules, including a course on data analytics and fraud detection.

- Coursera and edX offer courses such as "Data Analytics in Accountancy" and "Forensic Accounting and Fraud Examination", which provide practical skills and insights.

3. **Software Tutorials:**
    - Python Programming: Tutorials on Python libraries such as Pandas, NumPy, and Matplotlib from websites like Real Python and Python.org.
    - Excel Training: Advanced courses from platforms like LinkedIn Learning or Microsoft Learn, focusing on pivot tables, advanced formulas, and data analysis techniques.

Professional Development

The skills developed through the "From Theory to Practice" case study have direct applications in various professional accounting roles. Here's how they align with industry needs, particularly in forensic accounting and fraud examination:

1. **Forensic Accounting:**
    - **Data Handling and Analysis:** The ability to manipulate and analyze large datasets is crucial. Forensic accountants often deal with complex data sets to identify unusual patterns that may indicate fraud.
    - **Technological Proficiency:** Proficiency in Python and Excel allows forensic accountants to automate tasks and focus on higher-level analysis and decision-making. These tools are essential for tasks like tracing illicit funds and uncovering hidden assets.
2. **Fraud Examination:**
    - **Fraud Detection Techniques:** Techniques such as data validation, anomaly detection, and application of statistical methods (like Benford's Law) are directly applicable to fraud examination. Students can apply these methods to detect and investigate fraud more efficiently.
    - **Critical Thinking and Problem Solving:** The ability to approach problems systematically and think critically is essential for fraud examiners who must often sift through misleading or incomplete information to determine the reality of a financial situation.
3. **Career Pathways:**
    - The integration of analytics into accounting education prepares students for roles such as data analysts, forensic accountants, compliance officers, and

auditors within various organizations, including public accounting firms, corporations, and government agencies.

- Certifications like Certified Fraud Examiner (CFE) or Certified Forensic Accountant (CR.FA) can be pursued to further enhance credibility and career prospects in these areas.

Connecting Education to Professional Practices

In professional environments, the ability to integrate traditional accounting practices with modern technological tools is highly valued. This case study not only equips students with these skills but also emphasizes the importance of ethical considerations and professional integrity in managing and analyzing financial data. As they enter the workforce, they will find that these competencies not only make them effective in their roles but also pivotal in driving the adoption of analytics in their organizations.

VIII. Conclusion

As we reflect on the insights and experiences facilitated by this case, it is evident that the integration of data analytics into accounting education is not merely an enhancement but a necessity. This educational approach not only equips students with the requisite technical skills but also cultivates a mindset that blends analytical rigor with financial acuity, preparing them for the complexities of the modern financial landscape.

Summary of Key Learning Points:

- **Technical Proficiency**: Students gain hands-on experience with Python and Excel, tools that are critical in the analysis and interpretation of financial data.
- **Analytical Thinking**: The case study challenges students to think critically and analytically, skills that are indispensable in detecting and investigating fraud.
- **Real-World Application**: By simulating real-world scenarios, the case study provides a practical context in which students can apply their knowledge, enhancing their understanding of both the tools and the tactics used in fraud detection.
- **Ethical Considerations**: The case study underscores the importance of ethical behavior and professional responsibility, particularly in handling sensitive financial information.

Encouragement for Further Exploration:

- **Continued Learning**: Students are encouraged to further their education in data analytics and forensic accounting. Advanced courses, certifications, and workshops can provide deeper insights and more sophisticated techniques.
- **Professional Development**: As the accounting profession continues to evolve, staying abreast of technological advances and regulatory changes is crucial. Engaging in

continuous professional development through seminars, conferences, and professional bodies will keep students at the forefront of the field.

- **Research and Innovation**: Students are motivated to contribute to research in accounting analytics, possibly exploring new methods of data analysis or developing tools that can enhance fraud detection and financial transparency.

Looking Forward:

This case not only prepares students for immediate challenges in financial roles but also sets a foundation for lifelong learning and adaptation in a rapidly changing field. As educators and professionals, our goal is to continuously refine and adapt our teaching strategies to ensure that they meet the current and future needs of the accounting profession.

By emphasizing both the technical and ethical dimensions of accounting, we help forge a generation of accountants who are not only proficient in using advanced tools but are also committed to upholding the highest standards of integrity and accountability. This commitment will ultimately enrich their careers and contribute to the broader goal of ensuring transparency and trust in financial systems worldwide.