1. Анализ расходов и выявление аномальных транзакций

Генерируем случайные транзакции с разными суммами:

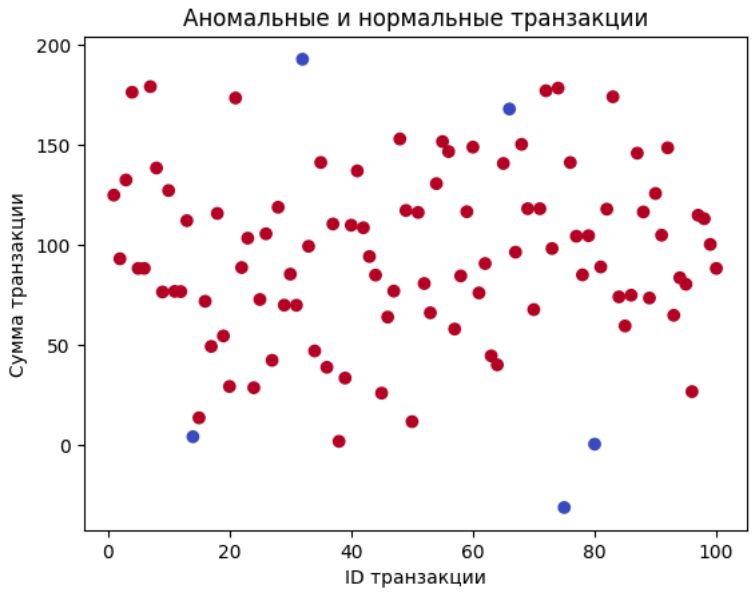
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.ensemble import IsolationForest

# Генерация случайных данных о транзакциях
np.random.seed(42)
data = pd.DataFrame({
    'Transaction_ID': np.arange(1, 101),
    'Amount': np.random.normal(loc=100, scale=50, size=100)  # среднее = 100, стандартное отклонение = 50
})

# Используем IsolationForest для обнаружения аномалий
model = IsolationForest(contamination=0.05)  # предположим, что 5% транзакций аномальные
model.fit(data[['Amount']])
data['Anomaly'] = model.predict(data[['Amount']])

# Визуализация аномальных и нормальных транзакций
plt.scatter(data['Transaction_ID'], data['Amount'], c=data['Anomaly'], cmap='coolwarm')
plt.title('Аномальные и нормальные транзакции')
plt.xlabel('ID транзакции')
plt.ylabel('Сумма транзакции')
plt.show()

# Вывод аномальных транзакций
anomalies = data[data['Anomaly'] == -1]
print("Аномальные транзакции:")
print(anomalies)
```




```
Аномальные транзакции:
    Transaction_ID      Amount  Anomaly
13              14    4.335988       -1
31              32  192.613909       -1
65              66  167.812001       -1
74              75  -30.987255       -1
79              80    0.621554       -1
```

Данный код генерирует синтетические данные о банковских транзакциях и использует алгоритм Isolation Forest для обнаружения аномальных операций. Модель помечает 5% транзакций как аномалии, после чего результаты визуализируются на точечном графике, где аномальные транзакции выделяются цветом. В завершение программа выводит список всех обнаруженных аномальных транзакций с их идентификаторами и суммами.

Генерация данных: **Текст, выделенный полужирным шрифтом** Здесь сгенерированы случайные транзакции с использованием нормального распределения (средняя сумма 100, стандартное отклонение 50). IsolationForest помогает обнаруживать аномальные транзакции, которые значительно отличаются от остальных.

Библиотеки:

pandas -

numpy -

2. Анализ отчетов о прибылях и убытках для выявления трендов

Генерация случайных финансовых отчетов компании по годам:

Решение:

```
# Генерация данных по годам
np.random.seed(42) #
years = np.arange(2010, 2021)
data = pd.DataFrame({
    'Year': years,
    'Revenue': np.random.uniform(500, 1500, len(years)),  # случайная выручка в диапазоне от 500 до 1500 млн
    'Expenses': np.random.uniform(300, 1200, len(years)),  # случайные расходы в диапазоне от 300 до 1200 млн
})

# Расчет прибыли
data['Profit'] = data['Revenue'] - data['Expenses']

# Визуализация трендов
plt.plot(data['Year'], data['Revenue'], label='Выручка', color='green', marker='o')
plt.plot(data['Year'], data['Expenses'], label='Расходы', color='red', marker='o')
plt.plot(data['Year'], data['Profit'], label='Прибыль', color='blue', marker='o')

plt.title('Тренды выручки, расходов и прибыли компании')
plt.xlabel('Год')
plt.ylabel('Сумма (в млн)')
plt.legend()
plt.grid(True)
plt.show()
```

Этот код анализирует финансовые показатели компании за 10 лет. Он генерирует случайные данные о выручке и расходах с 2010 по 2020 год, затем рассчитывает прибыль как разницу между ними. На линейном графике визуализируются тренды всех трех показателей: выручка отображается зеленой линией, расходы - красной, а прибыль - синей. График показывает, как изменялись финансовые показатели компании за десятилетний период.

Генерация данных:

Используются случайные значения выручки и расходов за несколько лет, что позволяет увидеть, как эти показатели изменялись с течением времени и как это влияло на прибыль компании.

3. Анализ долговых обязательств компании

Генерация случайных данных о долговых обязательствах:

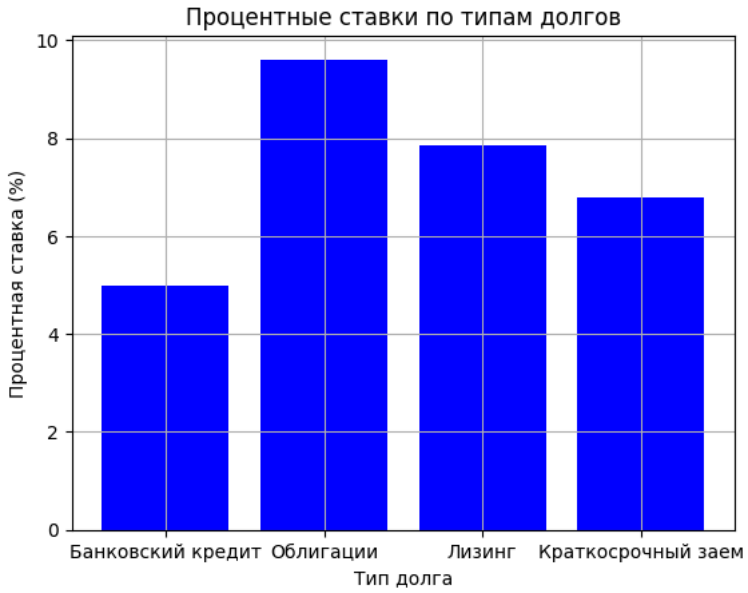
Решение:

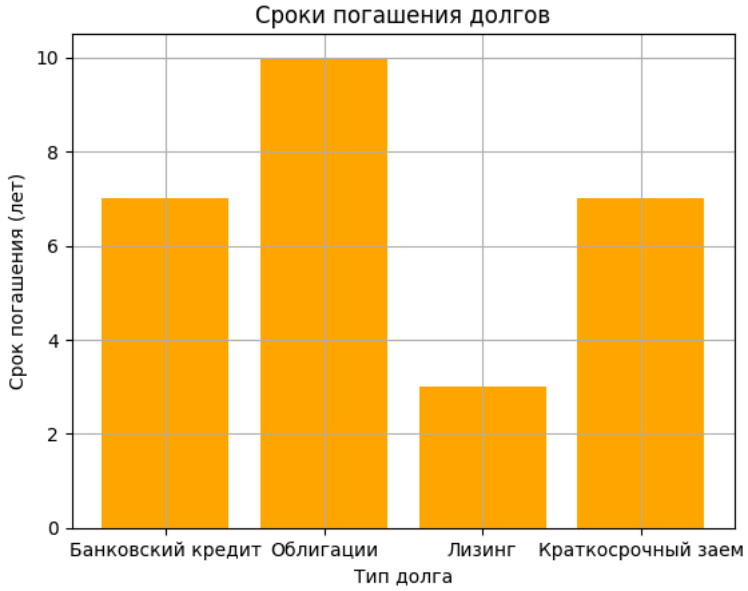
```
# Генерация данных о долговых обязательствах
np.random.seed(42)
debt_types = ['Банковский кредит', 'Облигации', 'Лизинг', 'Краткосрочный заем']
data = pd.DataFrame({
    'Debt_Type': debt_types,
    'Interest_Rate': np.random.uniform(2, 10, len(debt_types)),  # процентные ставки от 2% до 10%
    'Repayment_Term': np.random.randint(1, 15, len(debt_types))  # срок погашения от 1 до 15 лет
})

# Визуализация процентных ставок
plt.bar(data['Debt_Type'], data['Interest_Rate'], color='blue')
plt.title('Процентные ставки по типам долгов')
plt.xlabel('Тип долга')
plt.ylabel('Процентная ставка (%)')
plt.grid(True)
plt.show()

# Визуализация сроков погашения
plt.bar(data['Debt_Type'], data['Repayment_Term'], color='orange')
plt.title('Сроки погашения долгов')
plt.xlabel('Тип долга')
plt.ylabel('Срок погашения (лет)')
plt.grid(True)
plt.show()

# Средняя процентная ставка
average_interest_rate = data['Interest_Rate'].mean()
print(f"Средняя процентная ставка: {average_interest_rate:.2f}%")
```






```
Средняя процентная ставка: 7.31%
```

Этот код анализирует долговые обязательства компании по четырем типам заимствований. Создаются данные о процентных ставках и сроках погашения для банковских кредитов, облигаций, лизинга и краткосрочных займов. На двух столбчатых диаграммах визуализируются сравнительные характеристики по процентным ставкам и срокам погашения для каждого типа долга. В завершение рассчитывается и выводится средняя процентная ставка по всем видам долговых обязательств компании.

Генерация данных:

Для каждого типа долга генерируются случайные процентные ставки и сроки погашения, что позволяет провести аудит финансовых обязательств компании.

4. Анализ дебиторской задолженности

Генерация случайных данных по дебиторской задолженности:

Решение:

```
# Генерация данных по дебиторской задолженности
np.random.seed(42)
customers = ['Клиент А', 'Клиент Б', 'Клиент В', 'Клиент Г']
data = pd.DataFrame({
    'Customer': customers,
    'Total_Receivable': np.random.uniform(100, 500, len(customers)),  # общая задолженность в тыс.
    'Overdue_Amount': np.random.uniform(10, 100, len(customers))  # просроченная задолженность
})

# Расчет процента просроченной задолженности
data['Overdue_Percentage'] = (data['Overdue_Amount'] / data['Total_Receivable']) * 100

# Визуализация просроченной задолженности
plt.bar(data['Customer'], data['Overdue_Percentage'], color='red')
plt.title('Процент просроченной дебиторской задолженности по клиентам')
plt.xlabel('Клиент')
plt.ylabel('Процент просрочки (%)')
plt.grid(True)
plt.show()

# Средний процент просроченной задолженности
average_overdue = data['Overdue_Percentage'].mean()
print(f"Средний процент просроченной задолженности: {average_overdue:.2f}%")
```


```
Средний процент просроченной задолженности: 11.10%
```

Этот код анализирует дебиторскую задолженность по четырем клиентам компании. Генерируются данные об общей сумме задолженности и просроченной части для каждого клиента, затем рассчитывается процент просроченной задолженности. На столбчатой диаграмме визуализируется доля просрочки по каждому клиенту, что позволяет оценить риски неплатежей. В заключение вычисляется и выводится средний процент просроченной задолженности по всем клиентам, что дает общую картину состояния расчетов с дебиторами.

Генерация данных:

Генерируются случайные суммы общей дебиторской задолженности и просроченной задолженности для нескольких клиентов, что позволяет проанализировать кредитные риски компании.

5. Оценка эффективности затрат и прибыли по подразделениям компании

Генерация случайных данных по эффективности подразделений:

Решение:

```
# Генерация данных по подразделениям
np.random.seed(42)
departments = ['Продажи', 'Маркетинг', 'HR', 'IT', 'Производство']
data = pd.DataFrame({
    'Department': departments,
    'Costs': np.random.uniform(100, 500, len(departments)),  # затраты в тыс.
    'Profit': np.random.uniform(150, 600, len(departments))  # прибыль в тыс.
})

# Вычисление отношения прибыли к затратам
data['Profit_to_Cost_Ratio'] = data['Profit'] / data['Costs']

# Визуализация эффективности подразделений
plt.bar(data['Department'], data['Profit_to_Cost_Ratio'], color='green')
plt.title('Эффективность подразделений (отношение прибыли к затратам)')
plt.xlabel('Подразделение')
plt.ylabel('Отношение прибыли к затратам')
plt.grid(True)
plt.show()

# Подразделение с максимальной эффективностью
most_efficient_department = data.loc[data['Profit_to_Cost_Ratio'].idxmax()]
print(f"Наиболее эффективное подразделение: {most_efficient_department['Department']}")
```


```
Наиболее эффективное подразделение: Производство
```

Этот код анализирует эффективность различных подразделений компании. Создаются данные о затратах и прибыли для пяти отделов: продажи, маркетинг, HR, IT и производство. Рассчитывается коэффициент эффективности как отношение прибыли к затратам. На столбчатой диаграмме визуализируется эффективность каждого подразделения, что позволяет сравнить их результативность. В завершение определяется и выводится наиболее эффективное подразделение с максимальным значением коэффициента прибыли к затратам.

Генерация данных:

Случайные данные по затратам и прибыли для различных подразделений компании позволяют провести аудит их эффективности.

Заключение

Во всех примерах использованы случайные данные, сгенерированные с помощью библиотеки NumPy. Это позволяет тренироваться на данных, максимально приближенных к реальным сценариям в сфере финансового аудита, анализировать аномалии, тренды и риски в деятельности компаний.