**Техническая документация по выполненной работе**

**ЛЦТ, задача 02 «Сервис прогнозирования и проведения закупок»**

Автор: Жиряков А. Р., Команда: CP Zoro, Роль: Backend-разработка

В рамках выполнения задачи были выполнены скрипты на языке Python 3.11, выполняющие роль backend-обработки запросов пользователя. В документе приложены несколько листингов программных файлов и их текстовое описание функционала. В качестве базы данных используется PostgreSQL.

Список скриптов:

* Преобразование\_в\_csv.py
* Преобразование\_csv\_в\_удобный\_формат(обороты).py
* Преобразование\_csv\_в\_удобный\_формат(остатки).py
* Перенос\_в\_БД.py
* Прогноз.py

**Скрипт Преобразование\_в\_csv.py**

Описание:

Данный программный код исполняет функцию преобразования таблиц формата MS Excel в вид csv таблиц для удобства обработки данных сервером. Используется библиотека pandas в качестве обработчика. Используются разные кодировки в силу различного содержимого файлов.

Листинг:

|  |
| --- |
| import pandas as pd  #преобразование данных в csv - ведомости остатков  data\_xlsx = pd.read\_excel('Ведомость остатков на 31.03.2022г. (сч. 105).xlsx', 'Лист\_1', index\_col=None)  data\_xlsx.to\_csv('105\_31.03.2022.csv', encoding='utf-8-sig')  data\_xlsx = pd.read\_excel('Ведомость остатков на 31.03.2022г. (сч. 101).xlsx', 'Лист\_1', index\_col=None)  data\_xlsx.to\_csv('101\_31.03.2022.csv', encoding='utf-8-sig')  data\_xlsx = pd.read\_excel('Ведомость остатков на 31.03.2022г. (сч. 21).xlsx', 'Лист\_1', index\_col=None)  data\_xlsx.to\_csv('21\_31.03.2022.csv', encoding='utf-8-sig')  data\_xlsx = pd.read\_excel('Ведомость остатков на 30.06.2022г. (сч. 105).xlsx', 'Лист\_1', index\_col=None)  data\_xlsx.to\_csv('105\_30.06.2022.csv', encoding='utf-8-sig')  data\_xlsx = pd.read\_excel('Ведомость остатков на 30.06.2022г. (сч. 101).xlsx', 'Лист\_1', index\_col=None)  data\_xlsx.to\_csv('101\_30.06.2022.csv', encoding='utf-8-sig')  data\_xlsx = pd.read\_excel('Ведомость остатков на 30.06.2022г. (сч. 21).xlsx', 'Лист\_1', index\_col=None)  data\_xlsx.to\_csv('21\_30.06.2022.csv', encoding='utf-8-sig')  data\_xlsx = pd.read\_excel('Ведомость остатков на 30.09.2022г. (сч. 105).xlsx', 'Лист\_1', index\_col=None)  data\_xlsx.to\_csv('105\_30.09.2022.csv', encoding='utf-8-sig')  data\_xlsx = pd.read\_excel('Ведомость остатков на 30.09.2022г. (сч. 101).xlsx', 'Лист\_1', index\_col=None)  data\_xlsx.to\_csv('101\_30.09.2022.csv', encoding='utf-8-sig')  data\_xlsx = pd.read\_excel('Ведомость остатков на 30.09.2022г. (сч. 21).xlsx', 'Лист\_1', index\_col=None)  data\_xlsx.to\_csv('21\_30.09.2022.csv', encoding='utf-8-sig')  data\_xlsx = pd.read\_excel('Ведомость остатков на 31.12.2022г. (сч. 105).xlsx', 'Лист\_1', index\_col=None)  data\_xlsx.to\_csv('105\_31.12.2022.csv', encoding='utf-8-sig')  data\_xlsx = pd.read\_excel('Ведомость остатков на 31.12.2022г. (сч. 101).xlsx', 'Лист\_1', index\_col=None)  data\_xlsx.to\_csv('101\_31.12.2022.csv', encoding='utf-8-sig')  data\_xlsx = pd.read\_excel('Ведомость остатков на 31.12.2022г. (сч. 21).xlsx', 'Лист\_1', index\_col=None)  data\_xlsx.to\_csv('21\_31.12.2022.csv', encoding='utf-8-sig')  #преобразование данных в csv - оборотно-счётные ведомости  data\_xlsx = pd.read\_excel('Оборотная ведомость по сч. 105 за 1 кв. 2022г..xlsx', 'Лист\_1', index\_col=None)  data\_xlsx.to\_csv('105\_1\_2022.csv', encoding='utf-32')  data\_xlsx = pd.read\_excel('Оборотная ведомость по сч. 105 за 2 кв. 2022г..xlsx', 'Лист\_1', index\_col=None)  data\_xlsx.to\_csv('105\_2\_2022.csv', encoding='utf-32')  data\_xlsx = pd.read\_excel('Оборотная ведомость по сч. 105 за 3 кв. 2022г..xlsx', 'Лист\_1', index\_col=None)  data\_xlsx.to\_csv('105\_3\_2022.csv', encoding='utf-32')  data\_xlsx = pd.read\_excel('Оборотная ведомость по сч. 105 за 4 кв. 2022г..xlsx', 'Лист\_1', index\_col=None)  data\_xlsx.to\_csv('105\_4\_2022.csv', encoding='utf-32')  data\_xlsx = pd.read\_excel('Оборотно-сальдовая ведомость по сч. 101 за 1 кв. 2022г..xlsx', 'Лист\_1', index\_col=None)  data\_xlsx.to\_csv('101\_1\_2022.csv', encoding='utf-32')  data\_xlsx = pd.read\_excel('Оборотно-сальдовая ведомость по сч. 101 за 2 кв. 2022г..xlsx', 'Лист\_1', index\_col=None)  data\_xlsx.to\_csv('101\_2\_2022.csv', encoding='utf-32')  data\_xlsx = pd.read\_excel('Оборотно-сальдовая ведомость по сч. 101 за 3 кв. 2022г..xlsx', 'Лист\_1', index\_col=None)  data\_xlsx.to\_csv('101\_3\_2022.csv', encoding='utf-32')  data\_xlsx = pd.read\_excel('Оборотно-сальдовая ведомость по сч. 101 за 4 кв. 2022г..xlsx', 'Лист\_1', index\_col=None)  data\_xlsx.to\_csv('101\_4\_2022.csv', encoding='utf-32')  data\_xlsx = pd.read\_excel('Оборотно-сальдовая ведомость по сч. 21 за 1 кв. 2022г..xlsx', 'Лист\_1', index\_col=None)  data\_xlsx.to\_csv('21\_1\_2022.csv', encoding='utf-32')  data\_xlsx = pd.read\_excel('Оборотно-сальдовая ведомость по сч. 21 за 2 кв. 2022г..xlsx', 'Лист\_1', index\_col=None)  data\_xlsx.to\_csv('21\_2\_2022.csv', encoding='utf-32')  data\_xlsx = pd.read\_excel('Оборотно-сальдовая ведомость по сч. 21 за 3 кв. 2022г..xlsx', 'Лист\_1', index\_col=None)  data\_xlsx.to\_csv('21\_3\_2022.csv', encoding='utf-32')  data\_xlsx = pd.read\_excel('Оборотно-сальдовая ведомость по сч. 21 за 4 кв. 2022г..xlsx', 'Лист\_1', index\_col=None)  data\_xlsx.to\_csv('21\_4\_2022.csv', encoding='utf-32') |

**Скрипт Преобразование\_csv\_в\_удобный\_формат(обороты).py**

Описание:

В данном программном файле содержится скрипт, исполняющий преобразование Excel-таблиц оборотных и оборотно-сальдовых ведомостей в csv-файлы.

**Основные шаги и функции**

1. **Импорт библиотек:**
   * csv, os, pandas и re используются для работы с файлами CSV, обработки данных и регулярных выражений.
2. **Функция obrabotka(data, schN, qua):**
   * Добавляет столбцы с кварталом, номером счёта и классификатором.
   * В зависимости от значения schN (номер счёта), выполняет различные преобразования данных.
   * Удаляет ненужные столбцы, заменяет пустые строки на NaN, удаляет строки с NaN в ключевых столбцах.
   * Упорядочивает столбцы для счёта 105.
   * Переиндексирует данные, начиная с 1.
3. **Чтение и обработка данных:**
   * Чтение CSV файлов с оборотных ведомостей для каждого счёта (21, 101, 105) за разные кварталы в 2022 году.
   * Обработка данных с помощью функции obrabotka.
4. **Объединение и сохранение данных:**
   * Объединение обработанных данных для каждого счёта в один DataFrame и сохранение в новые CSV файлы.
   * Объединение данных по всем счетам в один общий DataFrame и сохранение его в CSV файл.

Листинг:

|  |
| --- |
| import csv  import os  import pandas as pd  import re  #функция обработки данных для исследования  def obrabotka(data, schN, qua):    #добавление столбца классификатора  data['Классификатор'] = None  #регулярное выражение для поиска классификаторов  classifier\_pattern = re.compile(fr'^({schN}\.\d+)')  #текущий классификатор  current\_classifier = None    #удаление ненужных столбцов  if 'Unnamed: 0' in data.columns:  data.drop(columns=['Unnamed: 0'], inplace=True)    if 'Unnamed: 0.1' in data.columns:  data.drop(columns=['Unnamed: 0.1'], inplace=True)    #замена пустых строк на NaN  data.replace('', pd.NA, inplace=True)  #добавление столбцов с кварталом и номером счёта  data['Квартал'] = qua  data['Номер счёта'] = schN    if schN == 105:  #проход по строкам DataFrame  for index, row in data.iterrows():  unnamed\_value = str(row['Инвентарный / номенклатурный номер']).strip()  match = classifier\_pattern.search(unnamed\_value)  if match:  #оставляем только код, убирая текстовое описание  current\_classifier = match.group(1)  elif pd.notna(row['Код справочника']) and pd.notna(row['Наименование нефинансового актива']):  data.at[index, 'Классификатор'] = current\_classifier  #удаление строк, в которых 'Основные средства' является NaN  data = data.dropna(subset=['Код справочника', 'Наименование нефинансового актива']).copy()  #удаление ненужных столбцов  if 'Инвентарный / номенклатурный номер' in data.columns:  data.drop(columns=['Инвентарный / номенклатурный номер'], inplace=True)    data.rename(columns = {'Наименование нефинансового актива':'Основные средства'}, inplace = True)  # Список столбцов в нужном порядке  columns\_titles = [  'Основное средство', 'Код справочника', 'Единица измерения',  'Остаток на 1 января 2022 г. дебет', 'Сумма на 1 января 2022 г. дебет',  'Оборот за 1 квартал 2022 г. дебет', 'Суммарный оборот за 1кв. дебет',  'Оборот за 1 квартал 2022 г. кредит', 'Суммарный оборот за 1кв. кредит',  'Остаток на 31 марта 2022 г. дебет ', 'Суммарный остаток на 31 марта 2022 г. дебет ',  'Классификатор', 'Квартал', 'Номер счёта', 'Остаток на 1 апреля 2022 г. Дебет',  'Сумма на 1 апреля 2022 г. дебет', 'Оборот за 2 квартал 2022 г. дебет',  'Суммарный оборот за 2кв. дебет', 'Оборот за 2 квартал 2022 г. кредит',  'Суммарный оборот за 2кв. кредит', 'Остаток на 30 июня 2022 г. дебет ',  'Суммарный остаток на 30 июня 2022 г. дебет ', 'Остаток на 1 июля 2022 г. дебет',  'Сумма на 1 июля 2022 г. дебет', 'Оборот за 3 квартал 2022 г. дебет',  'Суммарный оборот за 3кв. дебет', 'Оборот за 3 квартал 2022 г. кредит',  'Суммарный оборот за 3кв. кредит', 'Остаток на 30 сентября 2022 г. дебет ',  'Суммарный остаток на 30 сентября 2022 г. дебет ', 'Остаток на 1 октября 2022 г. дебет',  'Сумма на 1 октября 2022 г. дебет', 'Оборот за 4 квартал 2022 г. дебет',  'Суммарный оборот за 4кв. дебет', 'Оборот за 4 квартал 2022 г. кредит',  'Суммарный оборот за 4кв. кредит', 'Остаток на 31 декабря 2022 г. дебет ',  'Суммарный остаток на 31 декабря 2022 г. дебет'  ]  # Переупорядочиваем столбцы  data = data.reindex(columns=columns\_titles)      elif schN == 101 or schN == 21:  #проход по строкам DataFrame  for index, row in data.iterrows():  unnamed\_value = str(row['Основные средства']).strip()  match = classifier\_pattern.search(unnamed\_value)  if match:  #оставляем только код, убирая текстовое описание  current\_classifier = match.group(1)  elif pd.notna(row['Основные средства']) and pd.notna(row['Количество(начало периода)']) and pd.notna(row['Количество(конец периода)']):  data.at[index, 'Классификатор'] = current\_classifier  #удаление строк, в которых 'Основные средства' является NaN  data = data.dropna(subset=['Основные средства', 'Сальдо на начало периода, дебет',  'Продукт, в интересах которого осуществляется закупка, код']).copy()    #удаление пустых строк  data = data.dropna(how='all')    #переиндексация, начиная с 1  data.reset\_index(drop=True, inplace=True)      return data  #чтение CSV (ведомость остатков сч.21)  #обработка данных на основе оборотных ведомостей 2022 год  data\_21\_1 = pd.read\_csv("21\_1\_2022.csv", encoding="utf-32")  data\_21\_2 = pd.read\_csv("21\_2\_2022.csv", encoding="utf-32")  data\_21\_3 = pd.read\_csv("21\_3\_2022.csv", encoding="utf-32")  data\_21\_4 = pd.read\_csv("21\_4\_2022.csv", encoding="utf-32")  #обработка данных сч.No Q  data\_21\_1 = obrabotka(data\_21\_1, 21, 1)  data\_21\_2 = obrabotka(data\_21\_2, 21, 2)  data\_21\_3 = obrabotka(data\_21\_3, 21, 3)  data\_21\_4 = obrabotka(data\_21\_4, 21, 4)  #чтение CSV (ведомость остатков сч.101)  #обработка данных на основе оборотных ведомостей 2022 год  data\_101\_1 = pd.read\_csv("101\_1\_2022.csv", encoding="utf-32")  data\_101\_2 = pd.read\_csv("101\_2\_2022.csv", encoding="utf-32")  data\_101\_3 = pd.read\_csv("101\_3\_2022.csv", encoding="utf-32")  data\_101\_4 = pd.read\_csv("101\_4\_2022.csv", encoding="utf-32")  #обработка данных сч.No Q  data\_101\_1 = obrabotka(data\_101\_1, 101, 1)  data\_101\_2 = obrabotka(data\_101\_2, 101, 2)  data\_101\_3 = obrabotka(data\_101\_3, 101, 3)  data\_101\_4 = obrabotka(data\_101\_4, 101, 4)  #чтение CSV (ведомость остатков сч.105)  #обработка данных на основе оборотных ведомостей 2022 год  data\_105\_1 = pd.read\_csv("105\_1\_2022.csv", encoding="utf-32")  data\_105\_2 = pd.read\_csv("105\_2\_2022.csv", encoding="utf-32")  data\_105\_3 = pd.read\_csv("105\_3\_2022.csv", encoding="utf-32")  data\_105\_4 = pd.read\_csv("105\_4\_2022.csv", encoding="utf-32")  #обработка данных сч.No Q  data\_105\_1 = obrabotka(data\_105\_1, 105, 1)  data\_105\_2 = obrabotka(data\_105\_2, 105, 2)  data\_105\_3 = obrabotka(data\_105\_3, 105, 3)  data\_105\_4 = obrabotka(data\_105\_4, 105, 4)  #ведомость остатков по счёту 21  data21\_q14 = pd.concat([data\_21\_1, data\_21\_2,  data\_21\_3, data\_21\_4])  data21\_q14.to\_csv('Оборотная ведомость сч. 21 за 2022г. .csv', index=False, encoding="utf-32")  #ведомость остатков по счёту 101  data101\_q14 = pd.concat([data\_101\_1, data\_101\_2,  data\_101\_3, data\_101\_4])  data101\_q14.to\_csv('Оборотная ведомость сч. 101 за 2022г. .csv', index=False, encoding="utf-32")  #ведомость остатков по счёту 105  data105\_q14 = pd.concat([data\_105\_1, data\_105\_2,  data\_105\_3, data\_105\_4])  data105\_q14.to\_csv('Оборотная ведомость сч. 105 за 2022г. .csv', index=False, encoding="utf-32")  #общая оборотная ведомость по всем счетам  dataedinya = pd.concat([data21\_q14, data101\_q14, data105\_q14])  dataedinya.to\_csv('Оборотная ведомость по сч. 21, 101, 105 за 2022г. .csv', index=False, encoding="utf-32") |

**Скрипт Преобразование\_csv\_в\_удобный\_формат(остатки).py**

Основные шаги и функции

1. Импорт библиотек:
   * csv, os, pandas и re используются для работы с файлами CSV, обработки данных и регулярных выражений.
2. Функция obrabotka(data, schN, date):
   * Добавляет столбцы с датой и номером счёта.
   * В зависимости от значения schN (номер счёта), выполняет различные преобразования данных.
   * Преобразует столбцы в числовой формат, удаляет ненужные столбцы, добавляет и заполняет столбец классификатора.
   * Удаляет строки, не содержащие данных о товарах, и переименовывает столбцы.
3. Чтение и обработка данных:
   * Чтение CSV файлов с ведомостями остатков для каждого счёта (21, 101, 105) за разные даты в 2022 году.
   * Обработка данных с помощью функции obrabotka.
4. Объединение и сохранение данных:
   * Объединение обработанных данных для каждого счёта в один DataFrame и сохранение в новые CSV файлы.
   * Объединение данных по всем счетам в один общий DataFrame и сохранение его в CSV файл.

Листинг:

|  |
| --- |
| import csv  import os  import pandas as pd  import re  #функция обработки данных для исследования  def obrabotka(data, schN, date):    #добавление столбцов с датой и номером счёта  data['Номер счёта'] = schN  data['Дата'] = date  if schN == 105:  #преобразование в числовой формат  data.loc[:, 'Цена'] = pd.to\_numeric(data['Цена'], errors='coerce')  data.loc[:, 'Количество'] = pd.to\_numeric(data['Количество'], errors='coerce')  data.loc[:, 'Сумма'] = pd.to\_numeric(data['Сумма'], errors='coerce')    #удаление ненужных столбцов  data.drop(columns=['Unnamed: 0'], inplace=True)    #переименование столбца с номером строки  data.rename(columns={'Unnamed: 0': 'ID'}, inplace=True)  #добавление столбца классификатора  data['Классификатор'] = None  #регулярное выражение для поиска классификаторов в формате 105.№  classifier\_pattern = re.compile(r'^\d{3}\.\d+$')  #текущий классификатор  current\_classifier = None  #проход по строкам DataFrame  for index, row in data.iterrows():  mol\_value = str(row['МОЛ']).strip()  if classifier\_pattern.match(mol\_value):  current\_classifier = mol\_value  elif pd.notna(row['Цена']) and pd.notna(row['Количество']) and pd.notna(row['Сумма']):  data.at[index, 'Классификатор'] = current\_classifier  #удаление строк, которые не содержат данные о товарах  data = data.dropna(subset=['Цена', 'Количество', 'Сумма'], how='all').reset\_index(drop=True)  data = data.dropna(subset=['Цена']).copy() #удалить NaN-значные строки  data.rename(columns = {'МОЛ':'Основные средства'}, inplace = True)    elif schN == 101:    # Преобразование в числовой формат  columns\_to\_numeric = [  'Инвентарный номер', 'Амортизационная группа', 'Срок полезного использования',  'Мес. норма износа, %', 'Износ, %', 'Балансовая стоимость', 'Количество',  'Сумма амортизации', 'Остаточная стоимость'  ]    for col in columns\_to\_numeric:  data[col] = pd.to\_numeric(data[col], errors='coerce')    # Добавление столбца классификатора  data['Классификатор'] = None  # Регулярное выражение для поиска классификаторов в формате 101.№  classifier\_pattern = re.compile(r'^(101\.\d+)')  # Текущий классификатор  current\_classifier = None  # Проход по строкам DataFrame  for index, row in data.iterrows():  osn\_sredstvo\_value = str(row['Unnamed: 0']).strip()  match = classifier\_pattern.search(osn\_sredstvo\_value)  if match:  # Оставляем только код, убирая текстовое описание  current\_classifier = match.group(1)  elif pd.notna(row['Инвентарный номер']) and pd.notna(row['Балансовая стоимость']) and pd.notna(row['Количество']):  data.at[index, 'Классификатор'] = current\_classifier  # Удаление ненужных столбцов  if 'Unnamed: 0' in data.columns:  data.drop(columns=['Unnamed: 0'], inplace=True)    if 'Unnamed: 0.1' in data.columns:  data.drop(columns=['Unnamed: 0.1'], inplace=True)  # Удаление строк, которые не содержат данные о товарах  data = data.dropna(subset=[  'Основное средство', 'Инвентарный номер', 'ОКОФ', 'Амортизационная группа',  'Способ начисления амортизации', 'Дата принятия к учету', 'Состояние',  'Срок полезного использования', 'Мес. норма износа, %', 'Износ, %',  'Балансовая стоимость', 'Количество', 'Сумма амортизации', 'Остаточная стоимость'  ], how='all').reset\_index(drop=True)    # Удаление строк, в которых 'Инвентарный номер' является NaN  data = data.dropna(subset=['Инвентарный номер']).copy()    data.rename(columns = {'Основное средство':'Основные средства'}, inplace = True)    elif schN == 21:  # Преобразование в числовой формат  columns\_to\_numeric = [  'Инвентарный номер', 'Амортизационная группа', 'Срок полезного использования',  'Мес. норма износа, %', 'Износ, %', 'Балансовая стоимость', 'Количество',  'Сумма амортизации', 'Остаточная стоимость'  ]    for col in columns\_to\_numeric:  data[col] = pd.to\_numeric(data[col], errors='coerce')    # Добавление столбца классификатора  data['Классификатор'] = None  # Регулярное выражение для поиска классификаторов в формате 21.№  classifier\_pattern = re.compile(r'^(21\.\d+)')  # Текущий классификатор  current\_classifier = None  # Проход по строкам DataFrame  for index, row in data.iterrows():  osn\_sredstvo\_value = str(row['Unnamed: 0']).strip()  match = classifier\_pattern.search(osn\_sredstvo\_value)  if match:  # Оставляем только код, убирая текстовое описание  current\_classifier = match.group(1)  elif pd.notna(row['Дата принятия к учету']) and pd.notna(row['Балансовая стоимость']) and pd.notna(row['Количество']):  data.at[index, 'Классификатор'] = current\_classifier  # Удаление ненужных столбцов  if 'Unnamed: 0' in data.columns:  data.drop(columns=['Unnamed: 0'], inplace=True)    if 'Unnamed: 0.1' in data.columns:  data.drop(columns=['Unnamed: 0.1'], inplace=True)  # Удаление строк, которые не содержат данные о товарах  data = data.dropna(subset=[  'Основное средство', 'Инвентарный номер', 'ОКОФ', 'Амортизационная группа',  'Способ начисления амортизации', 'Дата принятия к учету', 'Состояние',  'Срок полезного использования', 'Мес. норма износа, %', 'Износ, %',  'Балансовая стоимость', 'Количество', 'Сумма амортизации', 'Остаточная стоимость'  ], how='all').reset\_index(drop=True)    # Удаление строк, в которых 'Дата принятия к учету' является NaN  data = data.dropna(subset=['Дата принятия к учету']).copy()    data.rename(columns = {'Основное средство':'Основные средства'}, inplace = True)  #переиндексация, начиная с 1  data.reset\_index(drop=True, inplace=True)    return data  #чтение CSV (ведомость остатков сч.21)  #обработка данных на основе ведомости остатков  #на 31.03, 30.06, 30.09, 31.12 2022 года.  data\_21\_31032022 = pd.read\_csv("21\_31.03.2022.csv", encoding="utf-8")  data\_21\_30062022 = pd.read\_csv("21\_30.06.2022.csv", encoding="utf-8")  data\_21\_30092022 = pd.read\_csv("21\_30.09.2022.csv", encoding="utf-8")  data\_21\_31122022 = pd.read\_csv("21\_31.12.2022.csv", encoding="utf-8")  #обработка данных сч.No Дата  data\_21\_31032022 = obrabotka(data\_21\_31032022, 21, "31.03.2022")  data\_21\_30062022 = obrabotka(data\_21\_30062022, 21, "30.06.2022")  data\_21\_30092022 = obrabotka(data\_21\_30092022, 21, "30.09.2022")  data\_21\_31122022 = obrabotka(data\_21\_31122022, 21, "31.12.2022")  #чтение CSV (ведомость остатков сч.101)  #обработка данных на основе ведомости остатков  #на 31.03, 30.06, 30.09, 31.12 2022 года.  data\_101\_31032022 = pd.read\_csv("101\_31.03.2022.csv", encoding="utf-8")  data\_101\_30062022 = pd.read\_csv("101\_30.06.2022.csv", encoding="utf-8")  data\_101\_30092022 = pd.read\_csv("101\_30.09.2022.csv", encoding="utf-8")  data\_101\_31122022 = pd.read\_csv("101\_31.12.2022.csv", encoding="utf-8")  #обработка данных сч.No Дата  data\_101\_31032022 = obrabotka(data\_101\_31032022, 101, "31.03.2022")  data\_101\_30062022 = obrabotka(data\_101\_30062022, 101, "30.06.2022")  data\_101\_30092022 = obrabotka(data\_101\_30092022, 101, "30.09.2022")  data\_101\_31122022 = obrabotka(data\_101\_31122022, 101, "31.12.2022")  #чтение CSV (ведомость остатков сч.105)  #обработка данных на основе ведомости остатков  #на 31.03, 30.06, 30.09, 31.12 2022 года.  data\_105\_31032022 = pd.read\_csv("105\_31.03.2022.csv", encoding="utf-8")  data\_105\_30062022 = pd.read\_csv("105\_30.06.2022.csv", encoding="utf-8")  data\_105\_30092022 = pd.read\_csv("105\_30.09.2022.csv", encoding="utf-8")  data\_105\_31122022 = pd.read\_csv("105\_31.12.2022.csv", encoding="utf-8")  #обработка данных сч.No Дата  data\_105\_31032022 = obrabotka(data\_105\_31032022, 105, "31.03.2022")  data\_105\_30062022 = obrabotka(data\_105\_30062022, 105, "30.06.2022")  data\_105\_30092022 = obrabotka(data\_105\_30092022, 105, "30.09.2022")  data\_105\_31122022 = obrabotka(data\_105\_31122022, 105, "31.12.2022")  #ведомость остатков по счёту 21  data21 = pd.concat([data\_21\_31032022, data\_21\_30062022,  data\_21\_30092022, data\_21\_31122022])  data21.to\_csv('Ведомость остатков сч. 21 за 2022г. .csv', index=False, encoding="utf-8-sig")  #ведомость остатков по счёту 101  data101 = pd.concat([data\_101\_31032022, data\_101\_30062022,  data\_101\_30092022, data\_101\_31122022])  data101.to\_csv('Ведомость остатков сч. 101 за 2022г. .csv', index=False, encoding="utf-8-sig")  #ведомость остатков по счёту 105  data105 = pd.concat([data\_105\_31032022, data\_105\_30062022,  data\_105\_30092022, data\_105\_31122022])  data105.to\_csv('Ведомость остатков сч. 105 за 2022г. .csv', index=False, encoding="utf-8-sig")  #общая ведомость остатков по всем счетам  dataedinya = pd.concat([data21, data101, data105])  dataedinya.to\_csv('Ведомость остатков по сч. 21, 101, 105 за 2022г. .csv', index=False, encoding="utf-8-sig") |

**Скрипт Перенос\_в\_БД.py**

Описание:

Этот скрипт предназначен для автоматизированного создания таблиц в базе данных PostgreSQL и загрузки данных из CSV файлов в эти таблицы. Скрипт использует библиотеки psycopg2 для работы с PostgreSQL и pandas для чтения CSV файлов. Конфигурация базы данных хранится в отдельном файле config.py.

Основные шаги выполнения скрипта

1. Импорт библиотек и конфигурации: импортируются необходимые библиотеки и конфигурация базы данных.
2. Определение CSV файлов: задаются пути к CSV файлам, имена таблиц и кодировки.
3. Функции для работы с PostgreSQL:
   * get\_pg\_type(dtype): определяет тип данных PostgreSQL на основе типа данных pandas.
   * get\_db\_connection(): Устанавливает соединение с базой данных.
   * get\_table\_columns(conn, table\_name): Получает список столбцов таблицы из базы данных.
4. Основная функция main():
   * Устанавливает соединение с базой данных и создает курсор.
   * Для каждого CSV файла читает его структуру и создает соответствующую таблицу в базе данных.
   * Загружает данные из CSV файла в созданную таблицу.
   * Выводит список столбцов созданной таблицы.
5. Запуск скрипта: запускает выполнение основной функции при запуске скрипта.

Листинг:

|  |
| --- |
| # Импорт необходимых библиотек  import psycopg2 # Библиотека для работы с PostgreSQL  import pandas as pd # Библиотека для работы с данными, в том числе CSV файлами  from config import DATABASE # Импорт настроек базы данных из файла config.py  # Список CSV файлов с соответствующими названиями таблиц и кодировками  csv\_files = [  ('Оборотная ведомость сч. 21 за 2022г. .csv', 'oboroty\_21\_2022', 'utf-32'),  ('Оборотная ведомость сч. 101 за 2022г. .csv', 'oboroty\_101\_2022', 'utf-32'),  ('Оборотная ведомость сч. 105 за 2022г. .csv', 'oboroty\_105\_2022', 'utf-32'),  ('Ведомость остатков сч. 21 за 2022г. .csv', 'ostatki\_21\_2022', 'utf-8-sig'),  ('Ведомость остатков сч. 101 за 2022г. .csv', 'ostatki\_101\_2022', 'utf-8-sig'),  ('Ведомость остатков сч. 105 за 2022г. .csv', 'ostatki\_105\_2022', 'utf-8-sig')  ]  # Функция для определения типа данных PostgreSQL на основе типа данных pandas  def get\_pg\_type(dtype):  if dtype == 'object':  return 'TEXT'  elif dtype == 'int64':  return 'NUMERIC'  elif dtype == 'float64':  return 'NUMERIC'  else:  return 'TEXT'  # Функция для установления соединения с базой данных PostgreSQL  def get\_db\_connection():  conn = psycopg2.connect(  database=DATABASE['dbname'],  user=DATABASE['user'],  password=DATABASE['password'],  host=DATABASE['host'],  port=DATABASE['port']  )  return conn  # Функция для получения списка столбцов таблицы из базы данных  def get\_table\_columns(conn, table\_name):  with conn.cursor() as cur:  cur.execute(f"SELECT column\_name FROM information\_schema.columns WHERE table\_name = '{table\_name}';")  columns = cur.fetchall()  return [column[0] for column in columns]  # Основная функция, выполняющая всю логику скрипта  def main():  conn = None  cur = None  try:  print("Подключение к базе данных...")  conn = get\_db\_connection() # Подключаемся к базе данных  cur = conn.cursor() # Создаем курсор для выполнения операций  for csv\_file, table\_name, encoding in csv\_files:  try:  print(f"Чтение файла {csv\_file} с кодировкой {encoding}...")  df = pd.read\_csv(csv\_file, nrows=1, encoding=encoding) # Читаем CSV файл  # Получаем названия и типы столбцов  column\_names = df.columns.tolist()  column\_types = [get\_pg\_type(str(df[col].dtype)) for col in df.columns]  # Формируем запрос для создания таблицы  create\_table\_query = f"CREATE TABLE {table\_name} ("  for i, (col\_name, col\_type) in enumerate(zip(column\_names, column\_types)):  create\_table\_query += f'"{col\_name}" {col\_type}'  if i < len(column\_names) - 1:  create\_table\_query += ", "  create\_table\_query += ");"  print(f"Создание таблицы {table\_name}...")  cur.execute(create\_table\_query) # Создаем таблицу  conn.commit() # Применяем изменения  print(f"Загрузка данных в таблицу {table\_name}...")  copy\_query = f"COPY {table\_name} FROM STDIN WITH (FORMAT CSV, HEADER true, DELIMITER ',');"  with open(csv\_file, 'r', encoding=encoding) as f:  cur.copy\_expert(sql=copy\_query, file=f) # Загружаем данные из CSV в таблицу  conn.commit() # Применяем изменения  print(f"Таблица {table\_name} создана и данные успешно загружены из CSV в PostgreSQL!")  # Вывод названий столбцов для созданной таблицы  columns = get\_table\_columns(conn, table\_name)  print(f"Таблица {table\_name} имеет столбцы: {columns}")  except Exception as e:  print(f"Ошибка при обработке файла {csv\_file}: {e}")  except psycopg2.Error as e:  print("Ошибка при подключении к PostgreSQL:", e.pgcode, e.pgerror, e.diag.message\_primary)  finally:  if cur:  cur.close() # Закрываем курсор  if conn:  conn.close() # Закрываем соединение с базой данных  # Запуск основной функции  if \_\_name\_\_ == '\_\_main\_\_':  main() |

**Скрипт Прогноз.py**

Описание:

Основные шаги выполнения скрипта:

 **Импорт библиотек:**

* pandas для работы с данными.
* matplotlib.pyplot для построения графиков.
* numpy для вычислений.

 **Функция read\_csv\_with\_encodings:**

* Пробует читать CSV файл с различными кодировками, возвращает DataFrame, если чтение успешно.

 **Функция search\_product\_in\_files:**

* Ищет указанный товар в каждом из предоставленных CSV файлов.
* Сохраняет данные по остаткам товара, если они найдены.

 **Функция calculate\_consumption:**

* Рассчитывает потребление товара за каждый квартал на основе данных по остаткам.

 **Функция forecast\_end\_date:**

* Прогнозирует дату окончания товара, используя среднюю скорость потребления.

 **Функция plot\_product\_data:**

* Строит график остатков товара по времени.

 **Основной блок кода:**

* Запрашивает у пользователя имя товара.
* Читает данные из CSV файлов и ищет товар.
* Если данные найдены, строит график остатков, рассчитывает потребление за кварталы и прогнозирует дату окончания товара.

Листинг:

|  |
| --- |
| import pandas as pd  import matplotlib.pyplot as plt  import numpy as np  # Функция для чтения CSV файла с различными кодировками  # Пробует читать файл с каждой кодировкой по очереди, пока не найдет подходящую  def read\_csv\_with\_encodings(file\_path, encodings):  for encoding in encodings:  try:  df = pd.read\_csv(file\_path, encoding=encoding)  return df  except Exception as e:  print(f"Ошибка чтения {file\_path} с кодировкой {encoding}: {e}")  return None  # Функция для поиска товара в файлах и вывода информации  # Ищет товар в каждом из CSV файлов и собирает данные по остаткам  def search\_product\_in\_files(product\_name, csv\_files):  product\_data = [] # Список для хранения данных по остаткам товара  columns\_of\_interest = [  'Остаточная стоимость', 'Сальдо на конец периода, дебет',  'Остаток на 31 марта 2022 г. дебет', 'Остаток на 30 июня 2022 г. дебет',  'Остаток на 30 сентября 2022 г. дебет', 'Остаток на 31 декабря 2022 г. дебет',  'Количество(конец периода)', 'Суммарный остаток на 31 декабря 2022 г. дебет',  'Количество'  ]  for file\_name, encodings in csv\_files:  print(f"Чтение: {file\_name}")  df = read\_csv\_with\_encodings(file\_name, encodings)    if df is not None:  # Проверяем наличие нужных столбцов в DataFrame и ищем товар  if 'Основные средства' in df.columns:  product\_info = df[df['Основные средства'].str.contains(product\_name, na=False)]  elif 'Наименование нефинансового актива' in df.columns:  product\_info = df[df['Наименование нефинансового актива'].str.contains(product\_name, na=False)]  elif 'МОЛ' in df.columns:  product\_info = df[df['МОЛ'].str.contains(product\_name, na=False)]  else:  print(f"Товар не найден в файле {file\_name}")  continue    if not product\_info.empty:  print(product\_info)  # Сохраняем данные по остаткам товара  if 'Дата' in df.columns:  for \_, row in product\_info.iterrows():  date = pd.to\_datetime(row['Дата'])  value = None  for column in columns\_of\_interest:  if column in row and not pd.isna(row[column]):  value = row[column]  break  if value is not None:  product\_data.append((date, value))  else:  print(f"Нет данных по остаткам для строки: {row}")  else:  print(f"Товар '{product\_name}' не найден в файле {file\_name}")  else:  print(f"Ошибка чтения {file\_name}")  print(f"Собранные данные по товару: {product\_data}")  return product\_data  # Функция для расчета потребления за каждый квартал  # Определяет разницу между начальным и конечным остатками за каждый квартал  def calculate\_consumption(product\_data):  quarterly\_consumption = []    # Сортируем данные по дате  product\_data.sort(key=lambda x: x[0])    # Рассчитываем потребление за каждый квартал  for i in range(len(product\_data) - 1):  start\_date, start\_value = product\_data[i]  end\_date, end\_value = product\_data[i + 1]    # Рассчитываем потребление как разницу между конечным и начальным остатками  consumption = start\_value - end\_value  quarterly\_consumption.append((start\_date, consumption))    return quarterly\_consumption  # Функция для прогнозирования времени до окончания товара  # Использует среднюю скорость потребления для прогнозирования даты окончания  def forecast\_end\_date(product\_data):  # Посчитаем среднее время для потребления одной единицы товара  consumption\_rates = []  for i in range(len(product\_data) - 1):  start\_date, start\_value = product\_data[i]  end\_date, end\_value = product\_data[i + 1]    # Рассчитываем потребление за период  consumption = start\_value - end\_value  if consumption > 0:  rate = (end\_date - start\_date) / consumption  consumption\_rates.append(rate.days)    # Среднее время потребления одной единицы товара  average\_consumption\_rate = np.mean(consumption\_rates)    if np.isnan(average\_consumption\_rate) or average\_consumption\_rate <= 0:  print("Недостаточно данных для прогноза или неверные данные.")  return None    # Прогнозируем время до окончания товара  last\_date, last\_value = product\_data[-1]  future\_date = last\_date + pd.DateOffset(days=int(last\_value \* average\_consumption\_rate))    return future\_date  # Функция для построения графика остатков  # Строит график изменения остатков товара по времени  def plot\_product\_data(product\_data):  if not product\_data:  print("Нет данных для построения графика.")  return  product\_data.sort(key=lambda x: x[0]) # Сортируем по дате  dates, values = zip(\*product\_data)  plt.figure(figsize=(10, 5))  plt.plot(dates, values, marker='o')  plt.xlabel('Дата')  plt.ylabel('Остаток')  plt.title('График остатков товара')  plt.grid(True)  plt.show()  # Основной блок кода для выполнения программы  if \_\_name\_\_ == "\_\_main\_\_":  product\_name = input("Введите имя товара: ")  csv\_files = [  ('Оборотная ведомость сч. 21 за 2022г. .csv', ['utf-32', 'utf-8-sig', 'utf-16', 'latin1']),  ('Оборотная ведомость сч. 101 за 2022г. .csv', ['utf-32', 'utf-8-sig', 'utf-16', 'latin1']),  ('Оборотная ведомость сч. 105 за 2022г. .csv', ['utf-32', 'utf-8-sig', 'utf-16', 'latin1']),  ('Ведомость остатков сч. 21 за 2022г. .csv', ['utf-8-sig', 'utf-32', 'utf-16', 'latin1']),  ('Ведомость остатков сч. 101 за 2022г. .csv', ['utf-8-sig', 'utf-32', 'utf-16', 'latin1']),  ('Ведомость остатков сч. 105 за 2022г. .csv', ['utf-8-sig', 'utf-32', 'utf-16', 'latin1'])  ]    # Поиск товара в файлах и сбор данных  product\_data = search\_product\_in\_files(product\_name, csv\_files)  if product\_data:  # Построение графика остатков товара  plot\_product\_data(product\_data)    # Рассчет потребления за кварталы  quarterly\_consumption = calculate\_consumption(product\_data)    if quarterly\_consumption:  # Вывод потребления за каждый квартал  print("Потребление за каждый квартал:")  for date, consumption in quarterly\_consumption:  print(f"{date.date()}: {consumption}")    # Прогнозирование времени до окончания товара  end\_date = forecast\_end\_date(product\_data)    if end\_date:  print(f"Товар '{product\_name}' закончится к {end\_date.date()}.")  else:  print("Не удалось спрогнозировать дату окончания товара.")  else:  print("Недостаточно данных для расчета потребления.")  else:  print("Нет данных для построения графика и прогнозирования.") |

Пример выполнения (Товар 'Бумага для офисной техники IQ Allround А3'):

Изображение выглядит как линия, График, диаграмма, текст

Автоматически созданное описание

Рисунок 1 ­– График остатков товара

Потребление за каждый квартал:

2022-03-31: 15.0

2022-06-30: 25.0

2022-09-30: 8.0

Товар 'Бумага для офисной техники IQ Allround А3' закончится к 2023-12-12.