Лабораторная работа 4.3 Интеграция данных из нескольких источников. Обработка и согласование данных из разных источников

Цель работы: получить практические навыки интеграции, обработки и согласования данных из различных источников.

Задачи:

- Изучить методы чтения данных из разных источников.
- Освоить техники обработки и очистки данных.
- Научиться согласовывать данные из разных источников.
- Реализовать сохранение обработанных данных.

Варианты индивидуальных заданий

№	Тема	Источники данных	Задача
1	Продажи	• CSV файл с транзакциями • Excel файл со справочником товаров • MySQL база с данными о клиентах	Объединить данные, рассчитать аналитику по продажам
2	Данные о сотрудниках	• PostgreSQL база с личными данными • Excel файл с данными об окладах • CSV файл с данными о премиях	Создать единый отчет по заработной плате
3	Складские остатки	• MySQL база данных основного склада • PostgreSQL база данных розничных магазинов • Excel файл с поставками	Создать единую систему учета остатков, выявить расхождения
4	Финансовые данные	• CSV файлы банковских выписок • Excel файл бюджета • PostgreSQL база данных с транзакциями	Сформировать финансовый отчет, провести сверку движения средств
5	Клиентские заказы	• MySQL база данных интернет-магазина • CSV файл с данными службы доставки • Excel файл с отзывами клиентов	Создать комплексный отчет по качеству обслуживания
6	Маркетинговые данные	• PostgreSQL база с данными рекламных кампаний • CSV файлы с данными из социальных сетей • Excel файл с бюджетами на рекламу	Проанализировать эффективность маркетинговых каналов
7	HR-данные	• MySQL база данных персонала • Excel файл с данными об обучении • CSV файл с результатами аттестаций	Создать комплексную оценку развития персонала
8	Производственные данные	• PostgreSQL база данных производственных заказов • CSV файлы с данными контроля качества • Excel файл с нормативами производства	Проанализировать эффективность производства

9	Логистические данные	• MySQL база данных перевозок • Excel файл с затратами на топливо • CSV файл с маршрутными листами	Оптимизировать логистические затраты
10	Техническая поддержка	• PostgreSQL база обращений клиентов • Excel файл с регламентами обслуживания • CSV файл с оценками качества поддержки	Проанализировать качество технической поддержки
11	Закупки	• MySQL база данных поставщиков • Excel файл с ценовыми предложениями • CSV файл с историей закупок	Оптимизировать закупочные процессы
12	Оборудование	• PostgreSQL база данных инвентаризации • Excel файл с графиком обслуживания • CSV файл с историей ремонтов	Создать систему учета технического обслуживания
13	Лояльность клиентов	• MySQL база данных бонусной программы • CSV файл с историей покупок • Excel файл с акциями и спецпредложениями	Проанализировать эффективность программы лояльности
14	Учебные данные	• PostgreSQL база данных студентов • Excel файл с оценками • CSV файл с посещаемостью	Создать комплексный анализ успеваемости
15	Недвижимость	• MySQL база данных объектов • Excel файл с оценкой состояния • CSV файл с историей арендных платежей	Оценить эффективность управления недвижимостью
16	Электронная коммерция	• PostgreSQL база данных товаров • CSV файл с отзывами покупателей • Excel файл с данными о возвратах	Проанализировать качество товарного ассортимента
17	Страховые данные	• MySQL база данных полисов • Excel файл со страховыми случаями • CSV файл с оценкой рисков	Проанализировать страховой портфель
18	Call-центр	• PostgreSQL база данных звонков • Excel файл с оценками операторов • CSV файл с тематикой обращений	Оценить эффективность работы call-центра
19	Проекты	• MySQL база данных проектов • Excel файл с ресурсами и бюджетами • CSV файл с отчетами о выполнении	Создать систему мониторинга проектов
20	Качество продукции	• PostgreSQL база данных контроля качества • Excel файл с нормативами • CSV файл с рекламациями	Проанализировать систему контроля качества

Требования к выполнению работы

1. Подготовка данных:
   - Создание тестовых наборов данных.
   - Настройка подключений к БД.
   - Подготовка скриптов очистки.
2. Разработка программы:
   - Модульная структура.
   - Обработка ошибок.
   - Логирование действий.
3. Оформление результатов:
   - Графики анализа данных.
   - Статистика обработки.
   - Выводы по качеству данных.

Содержание отчета

1. Титульный лист.
2. Цель и задачи работы.
3. Описание входных данных.
4. Листинг программы.
5. Результаты обработки.
6. Анализ результатов.
7. Выводы.

Генерации синтетических данных для варианта 20
"Интеграция данных о качестве продукции"

```
!pip install Faker
import pandas as pd
import numpy as np
from datetime import datetime, timedelta
import random
from faker import Faker
import os
from google.colab import files

# Установка seed для воспроизводимости
np.random.seed(42)
fake = Faker('ru_RU')

# Создание базовых данных о продукции
products = [
    'Смартфон Model X', 'Ноутбук Pro Y', 'Планшет Z',
    'Наушники Alpha', 'Смарт-часы Beta', 'Камера Ultra'
]

defect_types = [
    'Дефект экрана', 'Проблема с батареей', 'Неисправность динамика',
    'Проблема с зарядкой', 'Программный сбой', 'Механическое повреждение'
]

# Генерация данных для PostgreSQL базы контроля качества
def generate_quality_control_data(start_date, num_records):
    data = []
    current_date = start_date

    for _ in range(num_records):
        product = random.choice(products)
        passed = random.random() > 0.15  # 85% проходят контроль качества

        record = {
            'inspection_id': f'QC{_+1:05d}',
            'date': current_date.strftime('%Y-%m-%d'),
            'product': product,
            'batch_number': f'B{random.randint(1000, 9999)}',
            'inspector': fake.name(),
            'quality_score': random.randint(75, 100) if passed else random.randint(50, 74),
            'passed_inspection': passed,
            'notes': 'Проверка пройдена' if passed else random.choice(defect_types)
```

```
        }
        data.append(record)
        current_date += timedelta(hours=random.randint(1, 8))

    return pd.DataFrame(data)

# Генерация данных о нормативах для Excel
def generate_quality_standards():
    data = []
    for product in products:
        record = {
            'product': product,
            'min_quality_score': random.randint(75, 85),
            'optimal_quality_score': random.randint(90, 98),
            'max_defect_rate': round(random.uniform(0.01, 0.05), 3),
            'inspection_frequency_hours': random.choice([4, 8, 12]),
            'sample_size_percent': random.randint(5, 15),
            'shelf_life_days': random.randint(365, 730),
            'storage_temp_min': random.randint(10, 15),
            'storage_temp_max': random.randint(25, 30),
            'humidity_percent': random.randint(40, 60)
        }
        data.append(record)
    return pd.DataFrame(data)

# Генерация данных о рекламациях для CSV
def generate_complaints_data(start_date, num_records):
    data = []
    current_date = start_date

    for _ in range(num_records):
        product = random.choice(products)
        defect = random.choice(defect_types)

        record = {
            'complaint_id': f'C{_+1:05d}',
            'date_received': current_date.strftime('%Y-%m-%d'),
            'product': product,
            'customer_id': f'CUS{random.randint(10000, 99999)}',
            'purchase_date': (current_date - timedelta(days=random.randint(1, 90))).strftime('%Y-
%m-%d'),
            'defect_type': defect,
            'severity': random.choice(['Низкая', 'Средняя', 'Высокая']),
            'description': f'Обнаружен {defect.lower()} после {random.randint(1, 30)} дней
использования',
```

```
            'status': random.choice(['Новая', 'В обработке', 'Решена', 'Отклонена']),
            'compensation_amount': random.choice([0, 1000, 2000, 5000, 10000])
        }
        data.append(record)
        current_date += timedelta(days=random.randint(0, 2))

    return pd.DataFrame(data)

# Генерация данных
start_date = datetime(2023, 1, 1)
quality_control_df = generate_quality_control_data(start_date, 1000)
standards_df = generate_quality_standards()
complaints_df = generate_complaints_data(start_date, 200)

# Сохранение данных в файлы
quality_control_df.to_csv('quality_control_data.csv', index=False, encoding='utf-8')
standards_df.to_excel('quality_standards.xlsx', index=False)
complaints_df.to_csv('customer_complaints.csv', index=False, encoding='utf-8')

# Вывод информации о созданных файлах
print("Созданные файлы:")
print(f"1. quality_control_data.csv - {len(quality_control_df)} записей")
print(f"2. quality_standards.xlsx - {len(standards_df)} записей")
print(f"3. customer_complaints.csv - {len(complaints_df)} записей")

# Вывод примера данных
print("
Пример данных из каждого файла:")
print("
Контроль качества (первые 3 записи):")
print(quality_control_df.head(3))
print("
Нормативы качества (первые 3 записи):")
print(standards_df.head(3))
print("\
Рекламации (первые 3 записи):")
print(complaints_df.head(3))

# Скачивание файлов
files.download('quality_control_data.csv')
files.download('quality_standards.xlsx')
files.download('customer_complaints.csv')
```