

Департамент образования и науки города Москвы  
Государственное автономное образовательное учреждение высшего  
образования города Москвы  
«Московский городской педагогический университет»  
Институт цифрового образования  
Департамент информатики, управления и технологий

**ДИСЦИПЛИНА:**

Проектный практикум по разработке ETL-решений

**Лабораторная работа 5.1**

**Проектирование объектной модели данных. Проектирование  
сквозного конвейера ETL**

Выполнила: Сергеева А. И., группа: АДЭУ-211

Преподаватель: Босенко Т.М.

Москва

2025

**Цель работы:** реализация конвейера ETL для решения бизнес-кейса «Umbrella» с использованием Apache Airflow и базового конвейера ETL на основе предоставленных данных Kaggle API.

### Задачи:

- Запустить контейнер с кейсом «Umbrella».
- Изучить и описать основные элементы интерфейса Apache Airflow (DAGs, Tasks, Operators, Variables, Connections и т. д.).
- Спроектировать верхнеуровневую архитектуру аналитического решения в draw.io.
- Разработать конвейер ETL на основе предоставленного шаблона Basic pipeline ETL.rar.

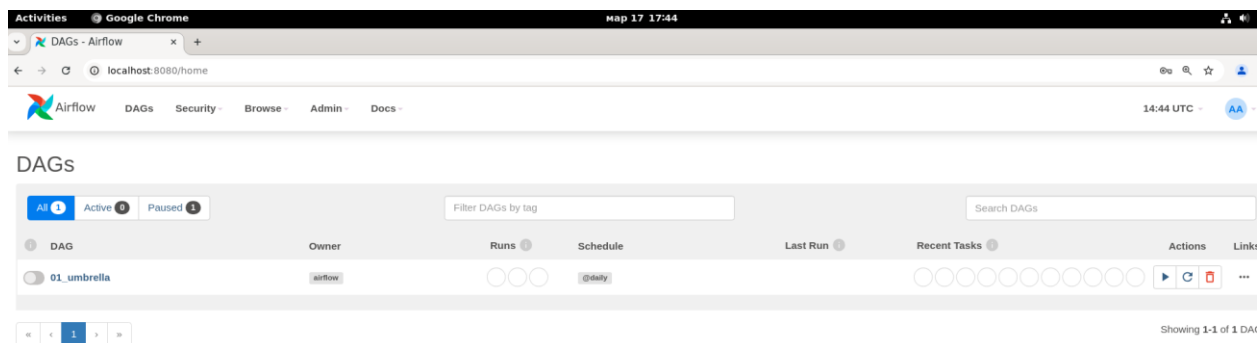
### Ход работы:

Для работы был запущен контейнер с Airflow на рисунке 1.

```
dev@dev-vm:~/Downloads/lab_etl/data_for_labs/lab_airflow/workshop-on-ETL-main/business_case_umbrella$ sudo docker compose up -d
[sudo] password for dev:
WARN[0000] /home/dev/Downloads/lab_etl/data_for_labs/lab_airflow/workshop-on-ETL-main/business_case_umbrella/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please re
it to avoid potential confusion
[+] Running 35/35
  ✓ postgres Pulled
  ✓ init Pulled
  ✓ webserver Pulled
  ✓ scheduler Pulled
[+] Running 6/6
  ✓ Network business_case_umbrella default      Created
  ✓ Volume "business_case_umbrella_logs"        Created
  ✓ Container business_case_umbrella-postgres-1 Started
  ✓ Container business_case_umbrella-scheduler-1 Started
  ✓ Container business_case_umbrella-init-1      Started
  ✓ Container business_case_umbrella-webserver-1 Started
dev@dev-vm:~/Downloads/lab_etl/data_for_labs/lab_airflow/workshop-on-ETL-main/business_case_umbrella$
```

Рисунок 1 – Запуск контейнера

Для начала необходимо было ознакомиться с основными элементами интерфейса Apache Airflow. Для перехода в интерфейс необходимо было перейти на адресу <http://localhost:8080/> на рисунке 2. Пароль и логин admin.



## Рисунок 2 – Запуск Airflow

Первый основной элемент - DAGs (Directed Acyclic Graphs). Это главная страница, на которой отображаются все доступные DAG (Directed Acyclic Graphs). DAG — это набор задач, которые выполняются в определенном порядке. На данной вкладке есть список всех доступных DAG с их статусами (например, "Running", "Failed" и т. д.). Их можно также с помощью фильтра фильтровать по тегу, статусу, владельцу. Кнопка "Trigger Dag"(изображена стрелкой справа) позволяет вручную запустить выполнение DAG. Кнопка "Refresh"(правее стрелки) обновляет список DAG. На данный момент на экране можно увидеть DAG с именем 01\_umbrella. Всего 1 DAG и пока не запускался.

Graph View - графическое представление DAG, где задачи отображаются в виде блоков, а зависимости между ними — в виде стрелок. Оно состоит из задач, которые представлены в виде прямоугольника (цвет прямоугольника указывает на статус задачи (например, зеленый — успешно выполнена, красный — ошибка)). Стрелки между задачами показывают порядок выполнения. Вверху страницы есть легенда, которая объясняет, что означают цвета и статусы. Для DAG 01\_umbrella задачами являются fetch\_weather\_forecast, fetch\_sales\_data, clean\_forecast\_data и т. д., соединённые стрелками, как видно на рисунке 3.

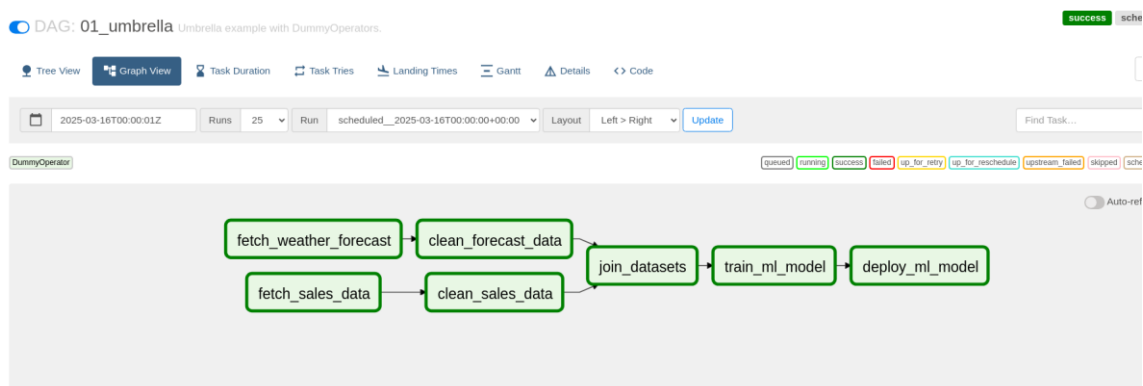


Рисунок 3 – Графическое представление DAG

Tree View - древовидное представление выполнения DAG за разные периоды времени. Каждый кружок представляет собой задачу, а также есть возможность фильтровать задачи по статусу или дате, что продемонстрировано на рисунке 4.

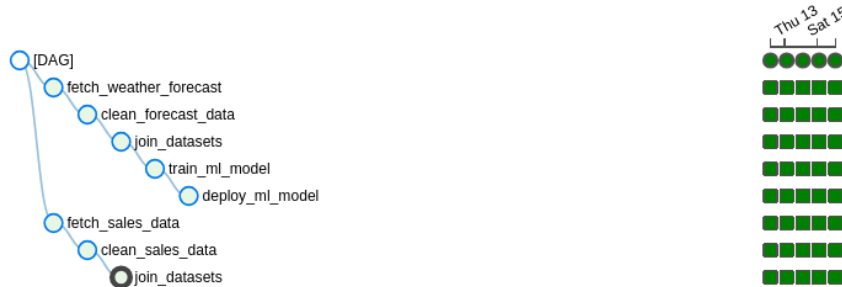


Рисунок 4 – Древовидное представление DAG

Task Duration - детальная информация о выполнении конкретной задачи. Далее идёт Code, в котором можно просмотреть исходный код DAG на рисунке 5.

```
DAG: 01_umbrella Umbrella example with DummyOperators.

Tree View Graph View Task Duration Task Tries Landing Times Gantt Details <> Code

1 """DAG demonstrating the umbrella use case with dummy operators."""
2
3 import airflow.utils.dates
4 from airflow import DAG
5 from airflow.operators.dummy import DummyOperator
6
7 dag = DAG(
8     dag_id="01_umbrella",
9     description="Umbrella example with DummyOperators.",
10    start_date=airflow.utils.dates.days_ago(5),
11    schedule_interval="@daily",
12)
13
14 fetch_weather_forecast = DummyOperator(task_id="fetch_weather_forecast", dag=dag)
15 fetch_sales_data = DummyOperator(task_id="fetch_sales_data", dag=dag)
16 clean_forecast_data = DummyOperator(task_id="clean_forecast_data", dag=dag)
17 clean_sales_data = DummyOperator(task_id="clean_sales_data", dag=dag)
18 join_datasets = DummyOperator(task_id="join_datasets", dag=dag)
19 train_ml_model = DummyOperator(task_id="train_ml_model", dag=dag)
20 deploy_ml_model = DummyOperator(task_id="deploy_ml_model", dag=dag)
21
22 # Set dependencies between all tasks
23 fetch_weather_forecast >> clean_forecast_data
24 fetch_sales_data >> clean_sales_data
25 [clean_forecast_data, clean_sales_data] >> join_datasets
26 join_datasets >> train_ml_model >> deploy_ml_model
```

Рисунок 5 – Исходный код DAG

Task Tries — это механизм, который позволяет задаче повторно выполняться в случае сбоя. Landing Times — это метрика, которая показывает, когда задача фактически завершила выполнение. А также есть диаграмма

Ганта, по которой также можно отслеживать выполнение DAG и представлена на рисунке 6.



Рисунок 6 – Диаграмма Ганта

Раздел Admin позволяет управлять настройками Airflow. Элементы:

- **Connections:** Настройка подключений к внешним системам (например, базам данных, API).
- **Variables:** Управление переменными, которые могут использоваться в DAG.
- **Pools:** Управление пулами ресурсов для выполнения задач.
- **Users:** Управление пользователями и их правами (если включен RBAC — Role-Based Access Control).

Browse - раздел для просмотра различных данных и метаданных. Docs - встроенная документация Airflow.

Сам DAG состоит из нескольких пустых операторов (DummyOperator) и иллюстрирует процесс ETL. Данные извлекаются, очищаются, объединяются, происходит обучение модели и ее развёртывание.

Данные могут браться из API погодных данных (например, WeatherAPI, который предоставляет доступ к информации о текущей погоде, прогнозах, исторических данных), данные из CRM систем о продажах. Для хранения данных был выбран PostgreSQL, он отлично подходит для обработки транзакционных задач, таких как оплаты и обработка заказов, может хранить JSONB, что позволяет хранить полуструктурированные данные и использовать их в продажах (например, детальную информацию о покупке). MLflow Model Registry позволяет хранить модели машинного обучения. При

обработке данных используется Python и Airflow для управления задачами обработки данных. Для бизнес-слоя в рамках визуализации подойдёт Apache Superset, это открытое решение. Также это может быть рекомендательная система или рассылки при изменении погоды. Готовую архитектуру можно увидеть на рисунке 7.

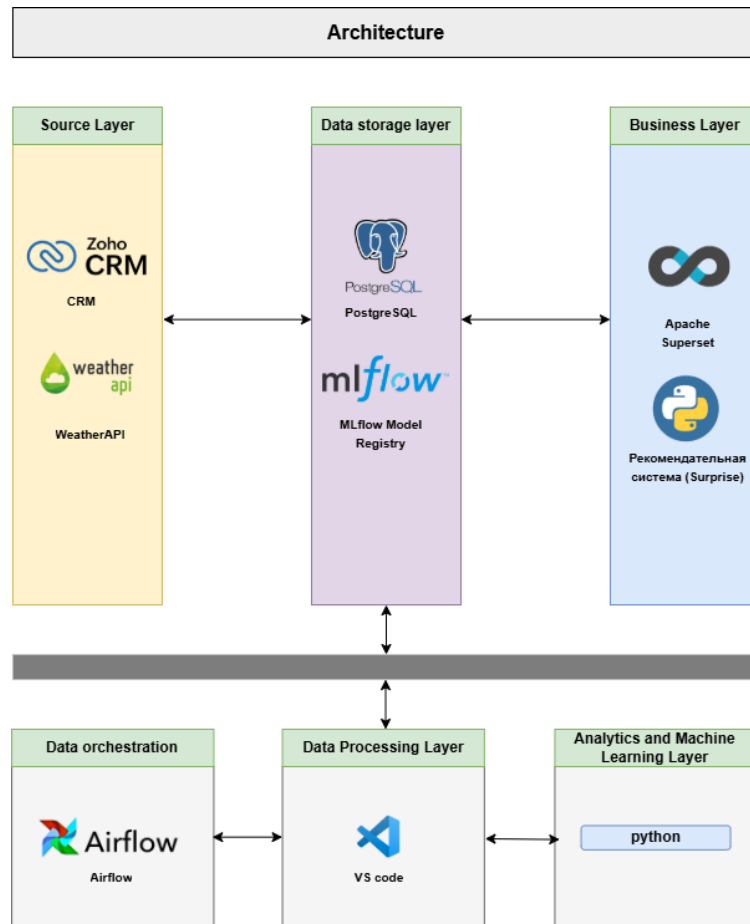


Рисунок 7 – Архитектура для бизнеса по продаже зонтов

Следующим этапом необходимо было обработать данные из 3 лабораторной работы и выгрузить их в SQLite. В рамках 12 варианта прошлой работы были даны данные об оборудовании, а именно об инвентаризации, истории ремонтов и обслуживании. Нужная таблица находится в базе MySQL. Было выполнено подключение к базе данных и выгрузка на рисунке 8.

```

# Подключаем нужные библиотеки
from sqlalchemy import create_engine
import pandas as pd

# Создаём соединение с базой данных
engine = create_engine('mysql+mysqlconnector://mgpu_ico_etl_12:ta1lo3z4@95.131.149.21/mgpu_ico_etl_12')

# Выгружаем таблицу в DataFrame
df = pd.read_sql_table('techni_obs1', con=engine)

df.head()

```

|   | id | equipment_name   | equipment_type | purchase_date | warranty_expiration | Maintenance_Date | Service_Type             | Maintenance Notes                                 | Repair_Date | Repair_Type      | Cost     | Notes   |
|---|----|------------------|----------------|---------------|---------------------|------------------|--------------------------|---|-------------|------------------|----------|---|
| 0 | 1  | Administration D | ПК             | 2022-05-25    | 2023-10-24          | 2022-07-23       | Регулярное обслуживание  | Plan first charge kitchen building young indiv... | 2024-10-03  | Замена детали    | 5636.45  | No challenge loss I order strong city debate w... |
| 1 | 2  | Lead B           | ПК             | 2022-12-12    | 2025-12-06          | 2023-02-02       | Чистка систем охлаждения | Price name share occur while.                     | 2023-09-18  | Системная ошибка | 10269.31 | Can trouble strong college suffer experience p... |
| 2 | 3  | Small B          | ПК             | 2022-06-13    | 2023-09-24          | 2022-07-29       | Обновление ПО            | Term color director cause available brother.      | 2023-12-17  | Системная ошибка | 10959.87 | Until new return reflect process generation.      |
| 3 | 4  | President A      | Сервер         | 2023-03-09    | 2026-02-16          | 2023-04-15       | Чистка систем охлаждения | Pattern our everybody news center daughter wor... | 2025-05-15  | Системная ошибка | 18746.02 | Lay use current answer sure development.          |
|   |    |                  |                |               |                     |                  | Чистка систем            | Difficult Mr yard ago                             |             | Системная        |          | Fill hard tax bit few                             |

Рисунок 8 – Подключение к MySQL и выгрузка в DataFrame

Так как над данными в 3 лабораторной уже была выполнена предобработка, то сразу был выполнен переход к фильтрации данных, выбирались только записи для ПК и сервера, а также цена была преобразована в тип данных с плавающей запятой на рисунке 9.

```

[3] import sqlite3
import csv

```

Далее была отобрана информация только по ПК и серверу.

```

[4] # Типы оборудования
equipment_types = ['ПК', 'Сервер']

# Пустой список для хранения отфильтрованных данных
equipment_data = []

[5] # Проходим по строкам DataFrame
for index, row in df.iterrows():
    if row['equipment_type'] in equipment_types: # Фильтрация по типу оборудования
        # Преобразование данных (стоимость во float)
        row['Cost'] = float(row['Cost']) # Преобразуем Cost в float
        equipment_data.append(row.tolist()) # Добавляем строку в список

# Выводим число значений и первые записи
print(len(equipment_data))
print(equipment_data[0:2])

```

255

[[1, 'Administration D', 'ПК', '2022-05-25', '2023-10-24', '2022-07-23', 'Регулярное обслуживание', 'Plan first charge kitchen building young individual

Всего 510 записей.

Рисунок 9 – Фильтрация данных

Далее было выполнено подключение к SQLite, как показано на рисунке 10. Был автоматически создан файл session.db, ведь все данные базы хранятся в одном файле, все изменения будут фиксироваться в данном файле и при закрытии соединения он обновляется.

```

✓ [10] import sqlite3
0
ЕК.

# Подключение к базе данных
conn = sqlite3.connect('session.db')
cursor = conn.cursor()

✓ [12] # Удаление таблицы, если она существует
0
ЕК.
try:
    cursor.execute('DROP TABLE IF EXISTS equipment')
except Exception as e:
    print(str(e))

```

Рисунок 10 – Подключение к базе SQLite и удаление существующей таблицы

Перед загрузкой данных необходимо было создать таблицу на рисунке 11.

```

13 try:
    cursor.execute('''
        CREATE TABLE equipment (
            id INTEGER PRIMARY KEY,
            equipment_name TEXT NOT NULL,
            equipment_type TEXT NOT NULL,
            purchase_date TEXT,
            warranty_expiration TEXT,
            maintenance_date TEXT,
            service_type TEXT,
            maintenance_notes TEXT,
            repair_date TEXT,
            repair_type TEXT,
            cost REAL,
            notes TEXT
        )
    ''')
    print("Table created successfully")
except Exception as e:
    print(str(e))
    print('Table Creation Failed!!!!')

```

Рисунок 11 – Создание таблицы

Как только таблица успешно создавалась, был выполнен отбор нужных столбцов и только потом вставка данных на рисунке 12.

сь выбираются необходимые столбцы для вставки данных, в данном случае нужны все столбцы.

```

equipment_sql_data = [
    (row[0], row[1], row[2], row[3], row[4], row[5], row[6], row[7], row[8], row[9], row[10], row[11])
    for row in equipment_data
]

```

авка данных.

```

try:
    cursor.executemany('''
        INSERT INTO equipment (
            id, equipment_name, equipment_type, purchase_date, warranty_expiration,
            maintenance_date, service_type, maintenance_notes, repair_date, repair_type, cost, notes
        ) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
    ''', equipment_sql_data)
    conn.commit()
    print('Data Inserted Successfully')
except Exception as e:
    print(str(e))
    print('Data Insertion Failed')

```

Data Inserted Successfully

Рисунок 12 – Выбор всех столбцов и загрузка данных в базу SQLite



Для проверки корректной загрузки данных был выполнен их просмотр, что можно увидеть на рисунке 13.

```
# Чтение данных
rows = cursor.execute('SELECT * FROM equipment')
for row in rows:
    print(row)

(343, 'Упрощенный D', 'Сервер', '2024-01-10', '2021-01-06', '2024-03-23', 'Регулярное обслуживание', 'pet dinner only would parent fear.', '2024-11-2',
(348, 'Machine C', 'Сервер', '2023-08-29', '2026-02-15', '2023-10-27', 'Обновление ПО', 'People individual opportunity west gun about everyone.', '202
(349, 'Recent D', 'Сервер', '2024-06-09', '2026-09-11', '2024-08-13', 'Регулярное обслуживание', 'Left quality size above enter.', '2025-10-16', 'Заме
(352, 'Note A', 'ПК', '2024-06-17', '2026-10-18', '2024-08-12', 'Регулярное обслуживание', 'Key want debate wall act sister gun either system.', '2025
(353, 'Fill A', 'Сервер', '2023-06-14', '2025-03-14', '2023-07-19', 'Чистка систем охлаждения', 'Player out build guy management group.', '2024-08-09'
(356, 'Structure A', 'Сервер', '2024-05-23', '2025-06-08', '2024-07-06', 'Чистка систем охлаждения', 'Method amount fact price candidate subject green
(358, 'Able C', 'ПК', '2022-09-01', '2024-08-20', '2022-11-17', 'Регулярное обслуживание', 'Culture water nothing to future marriage project myself st
(359, 'Ever B', 'ПК', '2024-11-10', '2027-07-12', '2025-01-18', 'Чистка систем охлаждения', 'Fast difference they kid wife article.', '2027-03-15', 'C
(360, 'Kitchen A', 'ПК', '2023-01-25', '2024-05-09', '2023-03-29', 'Регулярное обслуживание', 'Yeah arrive picture yourself four loss benefit mouth ag
(362, 'Republican B', 'Сервер', '2024-10-06', '2026-05-05', '2024-12-04', 'Обновление ПО', 'Must field over impact dream lawyer.', '2026-05-29', 'Сист
(363, 'Discover A', 'Сервер', '2024-06-23', '2025-07-11', '2024-08-27', 'Регулярное обслуживание', 'Personal question rest fly authority difference th
(364, 'Under C', 'ПК', '2025-02-01', '2027-04-24', '2025-03-18', 'Обновление ПО', 'Course day manager investment bad public early.', '2027-01-11', '3a
(365, 'Stage C', 'Сервер', '2024-07-28', '2026-09-04', '2024-08-28', 'Чистка систем охлаждения', 'Moment smile boy guess in marriage hour would happy
(368, 'Performance D', 'ПК', '2024-09-21', '2026-06-28', '2024-12-06', 'Чистка систем охлаждения', 'Weight effect lay husband me poor high beyond woul
(371, 'Assume C', 'Сервер', '2024-10-24', '2027-05-11', '2025-01-13', 'Регулярное обслуживание', 'Mission market year item sure throughout.', '2027-01
(372, 'Traditional B', 'Сервер', '2023-03-21', '2025-07-27', '2023-04-29', 'Обновление ПО', 'Move kitchen health become look option only.', '2025-07-2
(373, 'All A', 'ПК', '2022-12-20', '2024-12-22', '2023-02-11', 'Регулярное обслуживание', 'Identify ever group Republican thousand sometimes white wor
(374, 'Bill B', 'ПК', '2024-08-09', '2026-12-11', '2024-10-27', 'Обновление ПО', 'Able yourself perform present mission can mean new.', '2025-11-08',
(376, 'Half B', 'Сервер', '2023-02-28', '2025-07-02', '2023-05-08', 'Чистка систем охлаждения', 'Mean minute Republican film.', '2024-08-17', 'Системн
(379, 'Carry A', 'ПК', '2022-04-08', '2023-10-15', '2022-06-03', 'Обновление ПО', 'Reach head officer free simple cultural wish him.', '2023-12-05', 'C
```

Рисунок 13 – Просмотр данных из базы SQLite

После успешной загрузки данных их можно выгрузить в формате CSV на рисунке 14. При этом метод `writerow` берёт только вставку в файл 1 записи с названиями столбцов, `writerows` позволяет вставлять несколько данных, в данном случае все строки данных. В конце соединение с базой закрывается, чтобы всё записать в файл `session.db`.

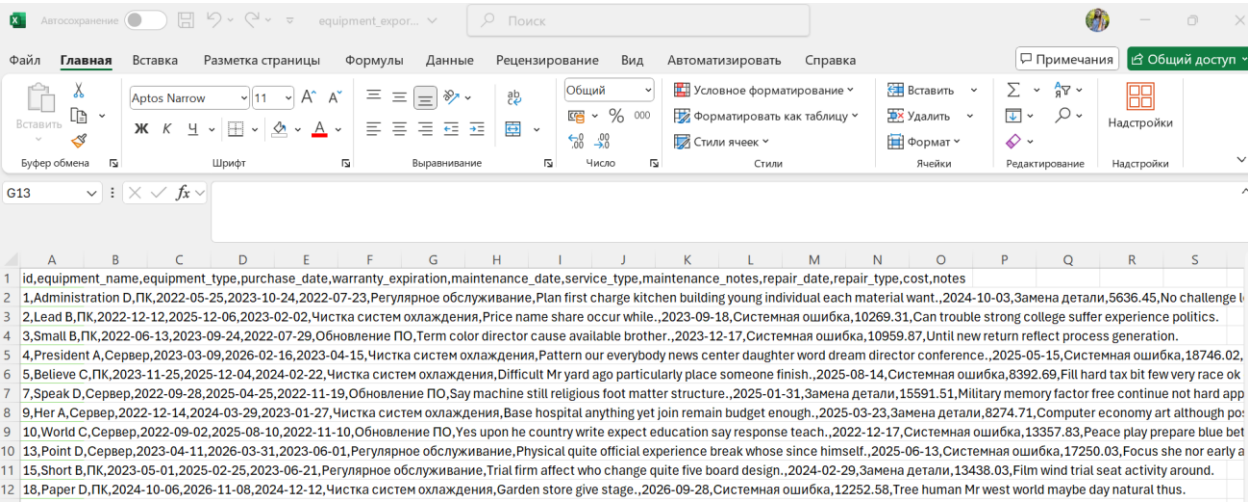
```
# Чтение данных
rows = cursor.execute('SELECT * FROM equipment')

# Запись в CSV
with open('equipment_export.csv', 'w', newline='', encoding='utf-8') as csvfile:
    csv_writer = csv.writer(csvfile)
    # Заголовки
    csv_writer.writerow([
        'id', 'equipment_name', 'equipment_type', 'purchase_date', 'warranty_expiration',
        'maintenance_date', 'service_type', 'maintenance_notes', 'repair_date', 'repair_type', 'cost', 'notes'
    ])
    # Данные
    csv_writer.writerows(rows)

conn.close()
```

Рисунок 14 – Экспорт данных в CSV-файл

Далее были выполнены проверки CSV-файла и session.db на то, что представленные выше операции сработали успешно, на рисунках 15–16.



|    | A  | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S |
|----|--|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1  | id,equipment_name,equipment_type,purchase_date,warranty_expiration,maintenance_date,service_type,maintenance_notes,repair_date,repair_type,cost,notes  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 2  | 1,Administration D,ПК,2022-05-25,2023-10-24,2022-07-23,Регулярное обслуживание,Plan first charge kitchen building young individual each material want.,2024-10-03,Замена детали,5636.45,No challenge l   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 3  | 2,Lead B,ПК,2022-12-12,2025-12-06,2023-02-02,Чистка систем охлаждения,Price name share occur while.,2023-09-18,Системная ошибка,10269.31,Can trouble strong college suffer experience politics.          |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 4  | 3,Small B,ПК,2022-06-13,2023-09-24,2022-07-29,Обновление ПО,Term color director cause available brother.,2023-12-17,Системная ошибка,10959.87,Until new return reflect process generation.               |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 5  | 4,President A,Сервер,2023-03-09,2026-02-16,2023-04-15,Чистка систем охлаждения,Pattern our everybody news center daughter word dream director conference.,2025-05-15,Системная ошибка,18746.02,          |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 6  | 5,Believe C,ПК,2023-11-25,2025-12-04,2024-02-22,Чистка систем охлаждения,Difficult Mr yard ago particularly place someone finish.,2025-08-14,Системная ошибка,8392.69,Fill hard tax bit few very race ok |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 7  | 7,Speak D,Сервер,2022-09-28,2025-04-25,2022-11-19,Обновление ПО,Say machine still religious foot matter structure.,2025-01-31,Замена детали,15591.51,Military memory factor free continue not hard app   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 8  | 9,Her A,Сервер,2022-12-14,2024-03-29,2023-01-27,Чистка систем охлаждения,Base hospital anything yet join remain budget enough.,2025-03-23,Замена детали,8274.71,Computer economy art although po         |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 9  | 10,World C,Сервер,2022-09-02,2025-08-10,2022-11-10,Обновление ПО,Yes upon he country write expect education say response teach.,2022-12-17,Системная ошибка,13357.83,Peace play prepare blue bet         |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 10 | 13,Point D,Сервер,2023-04-11,2026-03-31,2023-06-01,Регулярное обслуживание,Physical quite official experience break whose since himself.,2025-06-13,Системная ошибка,17250.03,Focus she nor early a      |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 11 | 15,Short B,ПК,2023-05-01,2025-02-25,2023-06-21,Регулярное обслуживание,Trial firm affect who change quite five board design.,2024-02-29,Замена детали,13438.03,Film wind trial seat activity around.     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 12 | 18,Paper D,ПК,2024-10-06,2026-11-08,2024-12-12,Чистка систем охлаждения,Garden store give stage.,2026-09-28,Системная ошибка,12252.58,Tree human Mr west world maybe day natural thus.                   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

Рисунок 15 – Выгруженный файл



```
id INTEGER PRIMARY KEY,
equipment_name TEXT NOT NULL,
equipment_type TEXT NOT NULL,
purchase_date TEXT,
warranty_expiration TEXT,
maintenance_date TEXT,
service_type TEXT,
maintenance_notes TEXT,
repair_date TEXT,
repair_type TEXT,
cost REAL,
notes TEXT
```

Рисунок 16 – Файл session.db

Выводы:

В рамках работы были изучены основные элементы Apache Airflow. Разработана архитектура аналитического решения, включающая слои источников данных, их обработки, хранения и доступа для бизнес-пользователей, а также при просмотре дага была выявлена функция, отвечающая за машинное обучение, поэтому был добавлен и этот слой и зафиксирован Airflow в рамках отдельного слоя. Данная архитектура демонстрирует, что данные берутся о погоде и продажах, анализируются и обучаются модели на Python, хранятся не только сами данные в базе, но и

версионность моделей машинного обучения. С помощью специальной библиотека настраивается рекомендательная система для пользователей, а также есть блок с визуализацией. Так, если по прогнозу погоды приближаются дожди, то пользователям в рекомендациях магазина будут высвечиваться зонтики. Разработан конвейер ETL для обработки данных из Kaggle API, а также выполнена работа с базой данных SQLite. Это встраиваемая база данных, которая хранит все данные (таблицы, индексы, записи) в одном файле на диске. Это удобно, так как нет необходимости вручную создавать файл или настраивать сервер базы данных. Цель и задачи были выполнены.