

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: Шведова С.С., группа: АДЭУ-211

Преподаватель: Босенко Т.М.

Москва

2025

Задание 4.1. Бизнес кейс «Umbrella»

4.1.1. Развернуть «Конфигурация репозиторий VM» в VirtualBox.

4.1.2. Клонировать на ПК задание Бизнес кейс Umbrella в домашний каталог VM.

```
git clone https://github.com/BosenkoTM/workshop-on-ETL.git
```

4.1.3. Запустить контейнер с кейсом, изучить и описать основные элементы интерфейса Apache Airflow.

4.1.4. Спроектировать верхнеуровневую архитектуру аналитического решения задания Бизнес кейс Umbrella в draw.io. Необходимо использовать:

- Source Layer - слой источников данных.
- Storage Layer - слой хранения данных.
- Business Layer - слой для доступа к данным бизнес пользователей.

4.1.5. Результаты работы представить в виде файла ФИО.pdf, выгрузить в учебный портал moodle.

Задание 4.2. Basic pipeline ETL

4.2.1. Построить конвейер данных на основании Basic pipeline ETL.rar

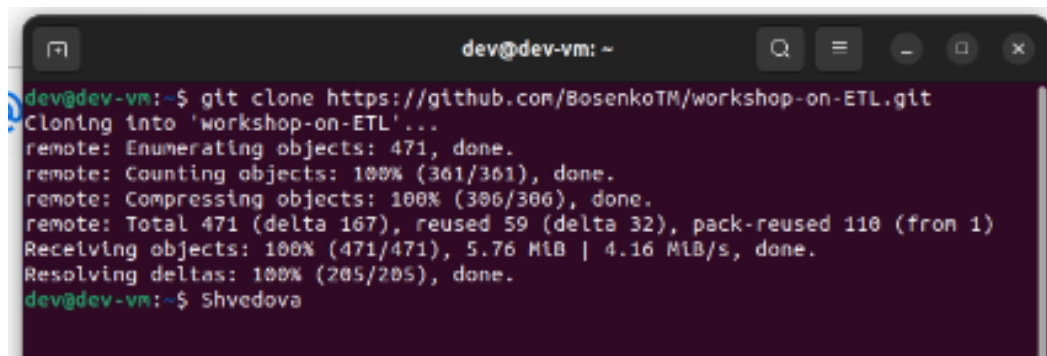
Набор данных использовать из Практическая работа 3. Работа с API.

Тестовые наборы данных Kaggle API.

4.2.2. Результаты работы представить в виде файла ФИО.ipynb, выгрузить в учебный портал moodle.

Задание 4.1.

На рисунке 1 показано клонирование репозитория



```
dev@dev-vm: ~  
dev@dev-vm:~$ git clone https://github.com/BosenkoTM/workshop-on-ETL.git  
Cloning into 'workshop-on-ETL'...  
remote: Enumerating objects: 471, done.  
remote: Counting objects: 100% (361/361), done.  
remote: Compressing objects: 100% (306/306), done.  
remote: Total 471 (delta 167), reused 59 (delta 32), pack-reused 110 (from 1)  
Receiving objects: 100% (471/471), 5.76 MiB | 4.16 MiB/s, done.  
Resolving deltas: 100% (205/205), done.  
dev@dev-vm:~$ Shvedova
```

Рисунок 1. Клонирование репозитория

На рисунке 2 показан запуск контейнера с кейсом

```
dev@dev-vm:~/workshop-on-ETL/business_case_umbrella$ sudo docker compose up -d
WARN[0000] /home/dev/workshop-on-ETL/business_case_umbrella/docker-compose.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion
[+] Running 5/5
 ✓ Network business_case_umbrella_default          Created           0.1s
 ✓ Container business_case_umbrella-postgres-1     Started           0.5s
 ✓ Container business_case_umbrella-scheduler-1    Started           1.3s
 ✓ Container business_case_umbrella-init-1         Started           1.1s
 ✓ Container business_case_umbrella-webserver-1    Started           1.4s
dev@dev-vm:~/workshop-on-ETL/business_case_umbrella$
```

Рисунок 2. Запуск контейнера с кейсом

На рисунке 3 показан запуск airflow

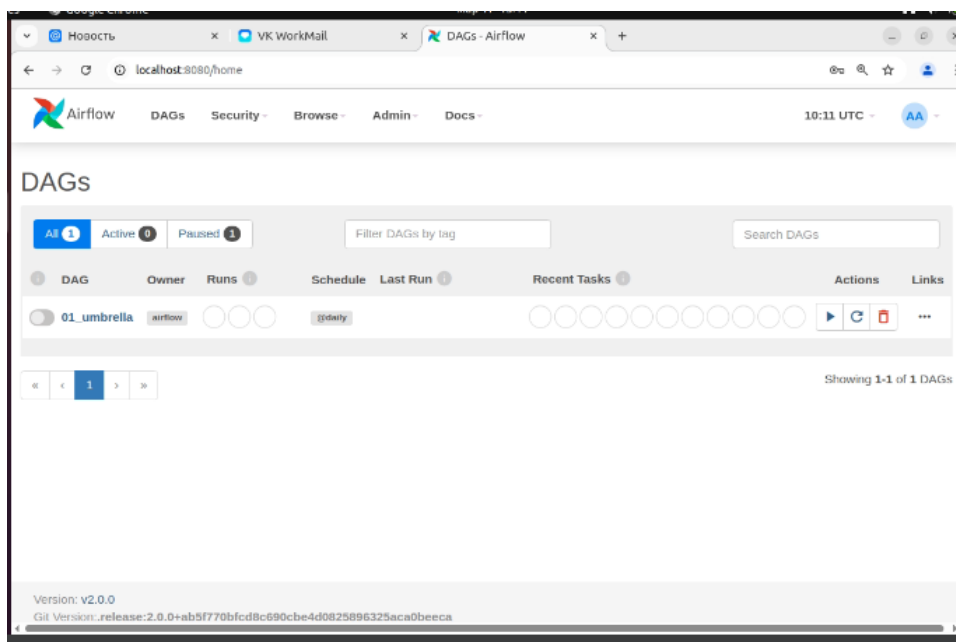


Рисунок 3. Airflow

График дага выглядит следующим образом (рисунок 4).



Рисунок 4. График дага

На рисунке 5 показан apache airflow

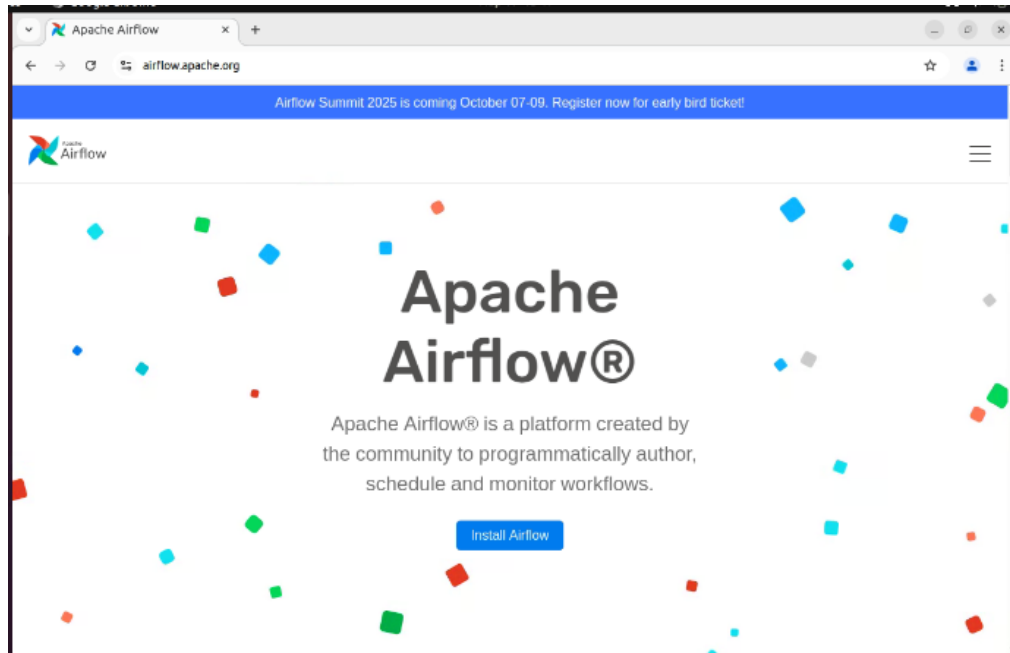


Рисунок 5. Apache airflow

Основные элементы интерфейса Apache Airflow:

1. Веб-сервер: это основной компонент, который предоставляет веб-интерфейс для использования Airflow. Он позволяет пользователям управлять, мониторить и планировать рабочие процессы (DAGs). Веб-сервер можно запустить с помощью команды `airflow webserver`.
2. Панель управления (UI): панель управления предоставляет различные функции, включая:
 - Списки DAG: отображает все доступные DAGs с информацией о их состоянии.
 - График выполнения (Graph View): визуализирует зависимость между задачами (операми) в виде графа.
 - Дерево задач (Tree View): Показано состояние выполнения каждой задачи в рамках DAG.
 - Измерение времени выполнения задач (Task Duration): график времени выполнения задач и DAG.
 - Журнал выполнения задач (Logs): позволяет просматривать логи выполнения для каждой задачи.

3. Проблемы и отделы (XComs): Эти механизмы позволяют передавать данные между задачами в DAG. Это полезно, когда необходимо передать результаты одного шага к другому.

4. Пользовательские настройки: Веб-интерфейс включает разделы для управления переменными, подключениями к внешним сервисам, а также плагинами, что дает возможность интегрировать Airflow с другими сервисами и APIs.

5. Мониторинг и уведомления: Airflow поддерживает мониторинг состояния выполнения задач и отправку уведомлений при возникновении ошибок или при завершении задач.

Архитектура решения представлена на рисунке 6.

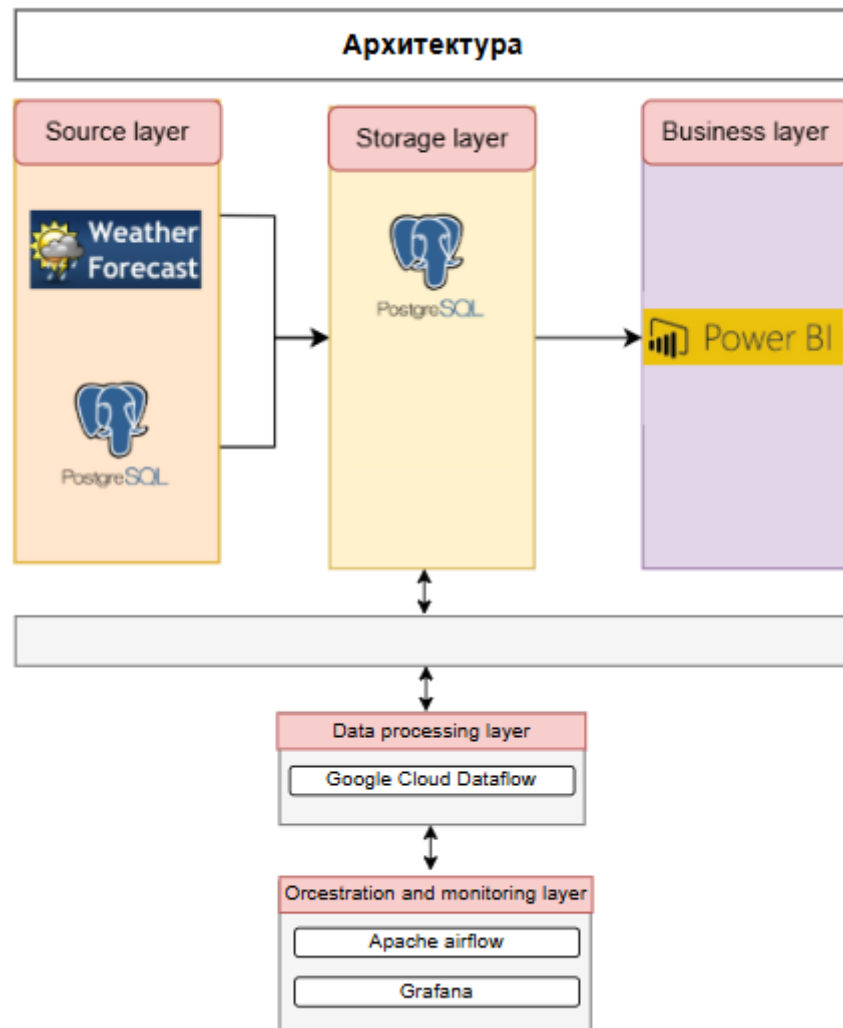


Рисунок 6. Архитектура решения

В качестве источника данных будет использоваться PostgreSQL и API прогноза погоды. На уровне хранения данных используется PostgreSQL, а на бизнес-уровне — Power bi. Обработка данных будет осуществляться в Google Cloud Dataflow, а мониторинг и администрирование — в airflow и Grafana

Как выглядит поток данных:

1. **Конвейер потока данных:**
 - **Источник 1:** API погоды (каждый час) -> извлекает температуру, влажность, давление для определенных городов.
 - **Источник 2:** PostgreSQL (таблица городов) -> содержит информацию о местоположении городов (широта, долгота).
 - **Трансформация:** объединение данных из обоих источников по идентификатору города, преобразование форматов данных, вычисление скользящих средних температур.
 - **Приемник:** BigQuery (таблица с данными о погоде).
2. **Power BI:** подключается к BigQuery и создает отчеты, показывающие динамику температуры по городам, аномальные погодные явления, прогнозы.
3. **Airflow:** Планирует запуск Dataflow Pipeline каждый час.
4. **Grafana:** отслеживает время выполнения Dataflow Pipeline, количество обработанных записей, ошибки API.

Задание 4.2.

Для того, чтобы построить конвейер данных, нужно сначала импортировать csv и загрузить файл (рисунок 7).

What is ETL?

ETL is actually short form of Extract, Transform and Load, a process in which data is acquired, changed/processes and then finally get loaded into data warehouse/database(s).

You can extract data from data sources like Files, Website or some Database, transform the acquired data and then load the final version into database for business usage.

You may ask, Why ETL? well, what ETL does, many of you might already been doing one way or other by writing different functions/scripts to perform tasks but one of the main advantage of ETLs is visualizing your entire data flow pipeline thus help you make decisions according to that.

Let's start with building our own ETL pipeline.

- Extract data from CSV file
- Transform/Manipulate Data
- Load Data into MongoDB

```
[1] # to read data from csv, python provides csv module
import csv
```

To deal with files in Python, we use the open() function, it's a built-in Python function. This function accepts two different arguments (inputs) in the parentheses, always in the following order:

- the name of the file (as a string)
- the mode of working with the file (as a string)

The syntax to open a file in python is:

```
file_obj = open("filename", "mode")
```

```
f = open(r'bonus_program.csv')
# 'f' is a file handler here

csv_reader = csv.reader(f)
print(csv_reader)

<csv.reader object at 0x7b8033f08580>
```

Рисунок 7. Импорт csv и загрузка файла

Далее на рисунке 8 виден сам файл, а также фильтрация и деление столбца о лояльных очках на 100.

1	customer_id	name	email	loyalty_points	purchase_id	purchase_date	amount	promotion_id	start_date	end_date
2	101	Ivan Ivanov	ivan.ivanov@example.com	150	201	15.03.2024 10:00	55.00	5	01.03.2024 00:00	31.03.2024 23:59
3	102	Peter Petrov	peter.petrov@example.com	220	202	20.02.2024 14:30	80.50	5	15.02.2024 00:00	29.02.2024 23:59
4	103	Elena Sidorova	elena.sidorova@example.com	180	203	10.03.2024 16:45	40.00	3	01.03.2024 00:00	15.03.2024 23:59
5	104	Mariya Kuznetsova	mariya.kuznetsova@example.com	300	204	05.04.2024 9:15	120.00	5	01.04.2024 00:00	30.04.2024 23:59
6	105	Alexey Smirnov	alexey.smirnov@example.com	250	205	11.05.2024 12:00	65.75	2	01.05.2024 00:00	15.05.2024 23:59
7	106	Olga Popova	olga.popova@example.com	190	206	18.06.2024 18:30	90.20	4	01.06.2024 00:00	30.06.2024 23:59
8	107	Dmitry Vasil'ev	dmitry.vasiliev@example.com	280	207	01.07.2024 11:45	75.00	1	01.07.2024 00:00	31.07.2024 23:59
9	108	Natalya Fedorova	natalya.fedorova@example.com	210	208	25.08.2024 15:00	110.00	1	01.08.2024 00:00	31.08.2024 23:59
10	109	Sergey Andreev	sergey.andreev@example.com	320	209	30.09.2024 20:35	45.50	1	01.09.2024 00:00	30.09.2024 23:59
11	110	Tat'yana Nikolaeva	tat'yana.nikolaeva@example.com	290	210	08.10.2024 13:20	130.00	3	01.10.2024 00:00	31.10.2024 23:59
12	111	Kirill Morozov	kirill.morozov@example.com	170	211	22.01.2024 11:00	60.00	4	01.01.2024 00:00	31.01.2024 23:59
13	112	Anastasiya Volkova	anastasiya.volkova@example.com	210	212	10.02.2024 15:45	70.00	1	01.02.2024 00:00	28.02.2024 23:59
14	113	Artem Belov	artem.belov@example.com	195	213	18.03.2024 17:00	50.00	5	01.03.2024 00:00	31.03.2024 23:59
15	114	Viktoriya Sokolova	viktoriya.sokolova@example.com	310	214	12.04.2024 10:30	140.00	5	01.04.2024 00:00	30.04.2024 23:59
16	115	Roman Lebedev	roman.lebedev@example.com	240	215	05.05.2024 14:00	75.50	2	01.05.2024 00:00	31.05.2024 23:59
17	116	Valerya Novikova	valerya.novikova@example.com	200	216	25.06.2024 19:15	95.00	3	01.06.2024 00:00	30.06.2024 23:59
18	117	Denis Orlov	denis.orlov@example.com	290	217	10.07.2024 12:30	80.00	4	01.07.2024 00:00	31.07.2024 23:59
19	118	Katerina Pavlova	katerina.pavlova@example.com	225	218	18.08.2024 16:00	115.00	4	01.08.2024 00:00	31.08.2024 23:59
20	119	Igor' Semenov	igor.semenov@example.com	180	219	22.09.2024 21:45	50.50	5	01.09.2024 00:00	30.09.2024 23:59
21	120	Kseniya Filipova	kseniya.filipova@example.com	270	220	01.10.2024 14:00	125.00	2	01.10.2024 00:00	31.10.2024 23:59
22	121	Maksim Petrov	maksim.petrov@example.com	160	221	18.01.2023 9:00	65.00	4	20.12.2023 00:00	10.01.2024 23:59
23	122	Alina Ivanova	alina.ivanova@example.com	240	222	28.02.2024 13:15	75.00	5	01.02.2024 00:00	29.02.2024 23:59
24	123	Stanislav Kuznetsov	stanislav.kuznetsov@example.com	215	223	05.03.2024 16:40	70.00	1	01.03.2024 00:00	15.03.2024 23:59
25	124	Yuliyana Smirnova	julia.smirnova@example.com	325	224	20.04.2024 11:00	145.00	1	01.04.2024 00:00	30.04.2024 23:59
26	125	Mikhail Sokolov	mikhail.sokolov@example.com	250	225	10.05.2024 15:30	85.50	3	01.05.2024 00:00	31.05.2024 23:59
27	126	Nikolay Popov	nikolay.popov@example.com	210	226	05.06.2024 20:00	100.00	4	01.06.2024 00:00	30.06.2024 23:59
28	127	Andrey Vasil'ev	andrey.vasiliev@example.com	300	227	22.07.2024 9:15	85.00	5	01.07.2024 00:00	31.07.2024 23:59

Transforming/Changing the data.

```
[3] assetsCode = ['1','2','3','4']

# initialize empty list
data = []

next(csv_reader, None) # skips the headers

# read csv data row wise
for row in csv_reader:
    if (row[7] in assetsCode):
        row[3] = float(row[3]) / 100
        data.append(row)

# print(csv_reader.line_num)
print(len(data))
print(data[0:2])
```

```
[[('103', 'Elena Sidorova', 'elena.sidorova@example.com', 1.8, '203', '3/10/2024 16:45', '40.00', '3', '3/12/2024 8:00', '3/15/2024 23:59'), ('105', 'Alexey Smirnov', 'alexey.smirnov@example.com', 2.5, '205', '5/12/2024 12:00', '65.75', '2', '01/05/2024 00:00', '15/05/2024 23:59')]]
```

Рисунок 8. Трансформация данных

На рисунке 9 показано подключение к базе данных SQL, удаление таблицы и создание таблицы bonus_program; вывод первой строки.

Loading the data into SQL DB

```
[4] import sqlite3
conn = sqlite3.connect('session.db')
```

```
[5] try:
    conn.execute("DROP TABLE IF EXISTS 'bonus_program' ")
except Exception as e:
    print(str(e))
```

```
[6] # Create a new Table named as Crypto
try:
    conn.execute("""
        CREATE TABLE bonus_program
        (customer_id INTEGER,
         name TEXT NOT NULL,
         start_date datetime,
         email TEXT NOT NULL,
         loyalty_points Float DEFAULT 0,
         promotion_id INTEGER,
         end_date datetime,
         purchase_date datetime,
         amount Float DEFAULT 0);""")
    print ("Table created successfully");
except Exception as e:
    print(str(e))
    print('Table Creation Failed!!!!')
finally:
    conn.close() # this closes the database connection
```

Table created successfully

```
[7] print(data[0])
```

```
[('103', 'Elena Sidorova', 'elena.sidorova@example.com', 1.8, '203', '3/10/2024 16:45', '40.00', '3', '3/1/2024 0:00', '3/15/2024 23:59']
```

Рисунок 9. Создание таблицы

На рисунке 10 видна выборка столбцов.

```
[8] # Some more transformations
sql_data = [(row[1], row[2], row[3], row[5], row[6], row[7], row[8]) for row in data]
sql_data[:2]
```

```
[('Elena Sidorova',
 'elena.sidorova@example.com',
 1.8,
 '3/10/2024 16:45',
 '40.00',
 '3',
 '3/1/2024 0:00'),
 ('Aleksy Smirnov',
 'aleksey.smirnov@example.com',
 2.5,
 '5/12/2024 12:00',
 '65.75',
 '2',
 '5/1/2024 0:00')]
```

```
[9] # lets make new connection to Insert crypto data in SQL DB
conn = sqlite3.connect('session.db')
cur = conn.cursor()
try:
    cur.executemany("INSERT INTO bonus_program(name, email, loyalty_points, purchase_date, amount, promotion_id, start_date) VALUES (?, ?, ?, ?, ?, ?, ?)", sql_data)
    conn.commit()
    print('Data Inserted Successfully')
except Exception as e:
    print(str(e))
    print('Data Insertion Failed')
finally:
    conn.close()
```

Data Inserted Successfully

Рисунок 10. Выборка столбцов

На рисунке 11 показано сохранение в csv файл


```
[10] # Let's Read data from DB to verify it

conn = sqlite3.connect('session.db')
rows = conn.cursor().execute('Select * from bonus_program')

for row in rows:
    print(row)
```

```
(None, 'Elena Sidorova', '3/1/2024 0:00', 'elena.sidorova@example.com', 1.8, 3, None, '3/10/2024 16:45', 40.0)
(None, 'Aleksij Smirnov', '5/1/2024 0:00', 'aleksij.smirnov@example.com', 2.5, 2, None, '5/12/2024 12:00', 65.75)
(None, 'Ol'ga Popova', '6/1/2024 0:00', 'olga.popova@example.com', 1.9, 4, None, '6/18/2024 18:30', 90.2)
(None, 'Dmitrij Vasil'ev', '7/1/2024 0:00', 'dmitry.vasiliev@example.com', 2.8, 1, None, '7/1/2024 11:45', 75.0)
(None, 'Natal'ya Fedorova', '8/1/2024 0:00', 'natalia.fedorova@example.com', 2.1, 1, None, '8/25/2024 15:00', 110.0)
(None, 'Sergej Andreev', '9/1/2024 0:00', 'sergey.andreev@example.com', 3.2, 1, None, '9/30/2024 20:15', 45.5)
(None, 'Tat'jana Nikolaeva', '10/1/2024 0:00', 'tatiana.nikolaeva@example.com', 2.6, 3, None, '10/8/2024 13:30', 130.0)
(None, 'Kirill Morozov', '11/1/2024 0:00', 'kirill.morozov@example.com', 1.7, 4, None, '1/22/2024 11:00', 60.0)
(None, 'Anastasiya Volkova', '2/1/2024 0:00', 'anastasia.volkova@example.com', 2.3, 1, None, '2/10/2024 15:45', 70.0)
(None, 'Roman Lebedev', '5/1/2024 0:00', 'roman.lebedev@example.com', 2.4, 2, None, '5/5/2024 14:00', 75.5)
(None, 'Valeriya Novikova', '6/1/2024 0:00', 'valeria.novikova@example.com', 2.0, 3, None, '6/25/2024 19:15', 95.0)
(None, 'Denis Orlov', '7/1/2024 0:00', 'denis.orlov@example.com', 2.9, 4, None, '7/10/2024 12:30', 80.0)
(None, 'Ekaterina Pavlova', '8/1/2024 0:00', 'ekaterina.pavlova@example.com', 2.25, 4, None, '8/18/2024 16:00', 115.0)
(None, 'Kseniya Filippova', '10/1/2024 0:00', 'ksenia.filippova@example.com', 2.7, 3, None, '10/1/2024 14:00', 135.0)
(None, 'Maksim Petrov', '12/20/2023 0:00', 'maksim.petrov@example.com', 1.6, 4, None, '1/18/2024 9:00', 65.0)
(None, 'Stanislav Kuznecov', '3/1/2024 0:00', 'stanislav.kuznetsov@example.com', 2.15, 3, None, '3/5/2024 18:45', 55.0)
(None, 'Yuliya Smirnova', '4/1/2024 0:00', 'yulia.smirnova@example.com', 3.25, 1, None, '4/20/2024 11:00', 145.0)
(None, 'Mikhail Sokolov', '5/1/2024 0:00', 'mikhail.sokolov@example.com', 2.5, 3, None, '5/10/2024 15:30', 80.5)
(None, 'Evgeniya Popova', '6/1/2024 0:00', 'evgenia.popova@example.com', 2.1, 4, None, '6/5/2024 20:00', 100.0)
(None, 'Nadezhda Fedorova', '8/1/2024 0:00', 'nadezhda.fedorova@example.com', 2.35, 3, None, '8/12/2024 14:45', 120.0)
(None, 'Oksana Nikolaeva', '10/1/2024 0:00', 'oksana.nikolaeva@example.com', 2.8, 1, None, '10/25/2024 10:30', 140.0)
```

Write data in a csv file

```
[12] csvfile = open('bonus_program_sql.csv', 'w')
      csv_writer = csv.writer(csvfile, lineterminator='\n')
      csv_writer.writerow(['name', 'email', 'loyalty_points', 'purchase_date', 'amount', 'promotion_id', 'start_date'])
      csv_writer.writerows(sql_data)
      csvfile.close()
```

Рисунок 11. Сохранение в csv файл

Выводы:

1. Клонирован на ПК задание Бизнес-кейс Umbrella в домашний каталог VM;
2. Запущен контейнер с кейсом, изучить и описать основные элементы интерфейса Apache Airflow;
3. Спроектирована верхнеуровневая архитектура аналитического решения задания Бизнес кейс Umbrella в draw.io.
4. Построен конвейер данных на основании Basic pipeline ETL.rar.