МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
«Омский государственный технический университет»

Факультет информационных технологий и компьютерных систем

Кафедра «Прикладная математика и фундаментальная информатика»

Домашнее задание

по дисциплине SQL и получение данных

на тему Разработка хранилища данных

Студента Скиба Данилы Сергеевича
фамилия, имя, отчество полностью

Курс 3 Группа МО-231

Направление 02.03.03 Математическое обеспечение и администрирование информационных систем
код, наименование

Выполнил 17.01.2026

Проверил Гулянов Р. Г.

Омск 2026

СОДЕРЖАНИЕ

Введение

Отчёт документирует выполнение двух связанных лабораторных работ по проектированию и реализации процессов обработки данных. Основной целью являлось приобретение практических навыков в области построения цикла работы с данными и построения витрин.

Задачами лабораторных работы являются:

1. разработка и реализация полного *ETL*-конвейера для обработки синтетических данных;
2. реализация процедур очистки и трансформации данных средствами *SQL;*
3. организация хранения данных в структурированном и неструктурированном виде;
4. автоматизация процесса загрузки и преобразования данных
5. построение аналитической витрины данных;
6. организация переноса данных между различными СУБД.

1. Описание исходных данных

Используемые данные взяты с открытого ресурса *Kaggle.* Датасет разработан на основе публичных данных о поездках такси Нью-Йорка за 2025 год. Для данного проекта была создана репрезентативная выборка, содержащая ключевые атрибуты поездки. Неструктурированные данные содержат 15 атрибутов в текстовом формате:

- *vendor_id* – поставщик услуг (компания);
- *tpep_pickup_datetime* – дата и время начала поездки;
- *tpep_dropoff_datetime* – дата и время окончания поездки;
- *passenger_count* – количество пассажиров;
- *trip_distance* – дистанция поездки (в милях);
- *retacode_id* – тарифный код;
- *pu_location_id* – идентификатор зоны посадки;

- *do_location_id* – идентифактор зоны высадки;
- *payment_type* – тип оплаты;
- *tip_amount* – сумма чаевых;
- *improvement_surch_arge* – дополнительный сбор;
- *total_amount* – итоговая сумма к оплате;
- *congestion_surcharge* – сбор за пробки;
- *airport_fee* – аэропортный сбор;
- *cbd_congestion_fee* – сбор за выезд в центр города.

2. Extract-Transform-Load (ETL)

Процедура *ETL* представляет с собой очистку, преобразование и загрузку данных в целевую систему хранения. В контексте данной лабораторной работы был реализован классический *ETL*-пайплайн для обработки данных о поездках такси.

Ключевые этапы процесса:

1. ***Extract***: функция *get_dataset()* загружает необработанные датасет в формате *.csv* из локального хранилища. Функция *load_data_to_db()* загружает данные в формате таблицы *t_sql_source_unstructered* в СУБД *PostgreSQL*. Код функции представлен в приложении А.
2. ***Transform***: очистка данных и приведение их к структурированному виду. Основная логика преобразования, включающая валидацию дат, исправление типов данных, фильтрация аномалий и удаление строк с пропусками реализована на *SQL* в функции *fn_etl_data_load(start_date, end_date)*. Код функции представлен в приложении А.
3. ***Load:*** загрузка очищенных данных в целевую таблицу *t_sql_source_structred* выполняется в той же *SQL* функции *fn_etl_data_load(start_date, end_date).*

SQL функция *fn_etl_data_load(start_date, end_date)* может быть вызвана извне функцией *fll_structerd_table(start, end_date),* написанной на *Python.* Код

функции *fll_structerd_table(start, end_date*) продемонстрирован в приложении А.

Весь *ETL*-пайплайн выполнен в единой функции *etl()* на *Python*, работа которой представлена на рисунках 1 и 2.

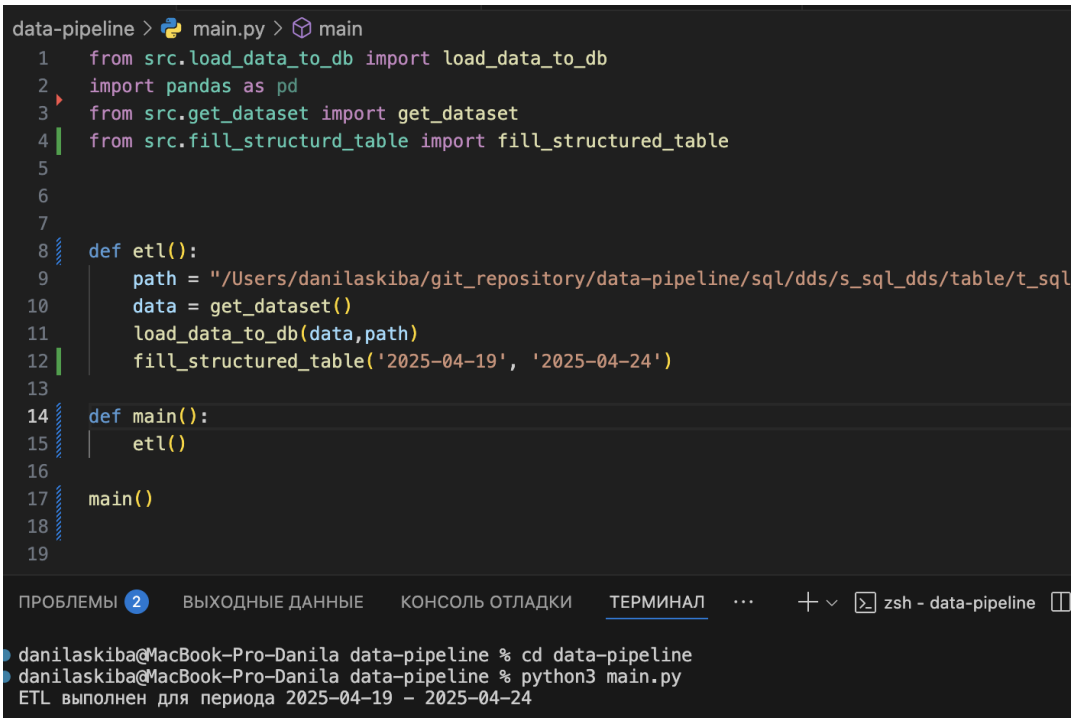
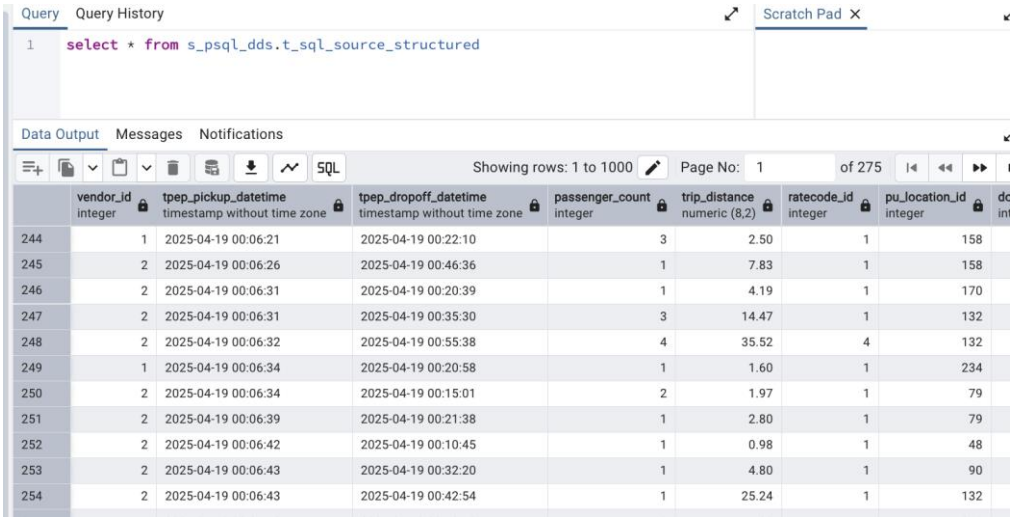


Рисунок 1 – Функция *etl()* и её вызов



	vendor_id integer	tpep_pickup_datetime timestamp without time zone	tpep_dropoff_datetime timestamp without time zone	passenger_count integer	trip_distance numeric (8,2)	ratecode_id integer	pu_location_id integer
244	1	2025-04-19 00:06:21	2025-04-19 00:22:10	3	2.50	1	158
245	2	2025-04-19 00:06:26	2025-04-19 00:46:36	1	7.83	1	158
246	2	2025-04-19 00:06:31	2025-04-19 00:20:39	1	4.19	1	170
247	2	2025-04-19 00:06:31	2025-04-19 00:35:30	3	14.47	1	132
248	2	2025-04-19 00:06:32	2025-04-19 00:55:38	4	35.52	4	132
249	1	2025-04-19 00:06:34	2025-04-19 00:20:58	1	1.60	1	234
250	2	2025-04-19 00:06:34	2025-04-19 00:15:01	2	1.97	1	79
251	2	2025-04-19 00:06:39	2025-04-19 00:21:38	1	2.80	1	79
252	2	2025-04-19 00:06:42	2025-04-19 00:10:45	1	0.98	1	48
253	2	2025-04-19 00:06:43	2025-04-19 00:32:20	1	4.80	1	90
254	2	2025-04-19 00:06:43	2025-04-19 00:42:54	1	25.24	1	132

Рисунок 2 – Результат работы функции *etl()*

Данные поездок такси имеют высокую аналитическую ценность. Представленная *ETL* функция помогает подготовить данные для их дальнейшего анализа.

3. Разработка DWH и витрины данных

Хранилище данных – это совокупность данных, предназначенная для поддержки принятия управленческих решений.

На основе предыдущего этапа *ETL* создан процесс построения хранилища данных (*DWH*) и аналитической витрины данных. *DWH* реализовано следующим образом:

1. Спроектирована схема типа «Звезда». В центре находится фактовая таблица, а вокруг неё таблицы-справочники. Такая структура упрощает запросы и повышает производительность анализа.

2. Созданы справочники (*dimensions*) на основе категориальных признаков из таблицы *t_sql_source_structured* (*pu_location_id, do_location_id, payment_type, retacode_id, vendor_id*). Каждый справочник имеет структуру (*id SERIAL PRIMARY KEY, name VARCHAR*).

3. Создана фактовая таблица *t_dm_task*. В неё загружаются все количественные признаки поездок, а также внешние ключи на соответствующие справочники.

На рисунке 2 представлена схема *DWH*.

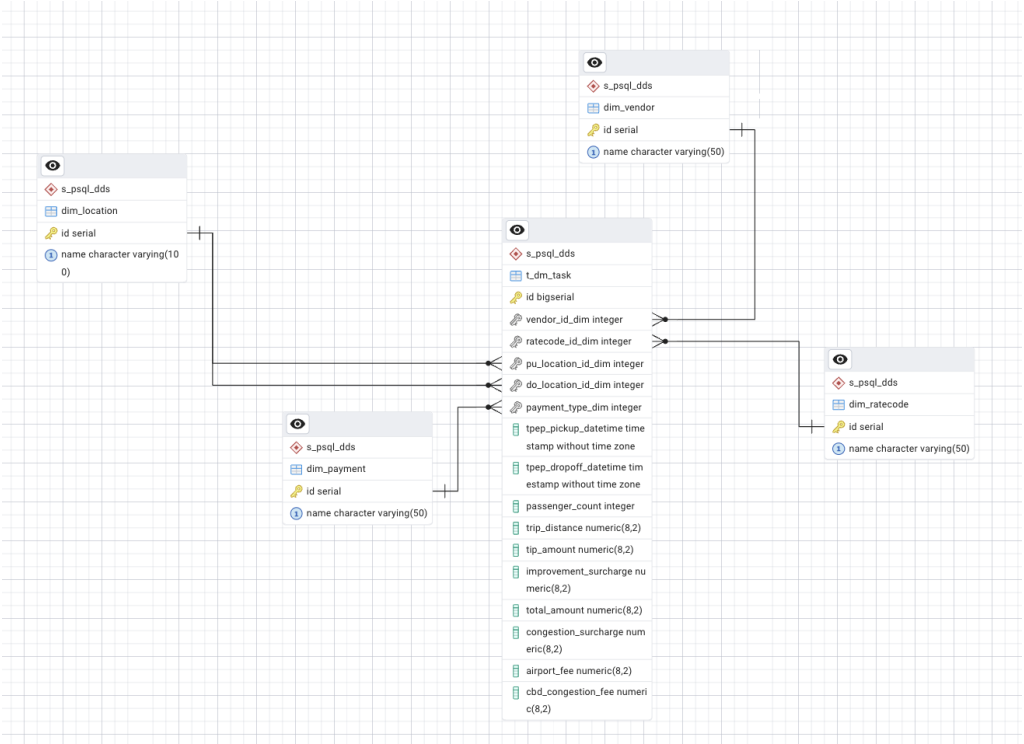


Рисунок 2 – *DWH*

Загрузка данных в таблицу осуществляется *SQL* функцией *fn_dm_data_load(start date, end date).* В качестве аргументов передаётся временной промежуток, за который загружаются данные из *t_sql_source_structured.* Функция представлена в приложении Б.

Вызов функции осуществляется извне с помощью *Python* функции *fill_dm_table(start_date, end_date).* Код и работа данного метода представлены на рисунках 3 и 4.

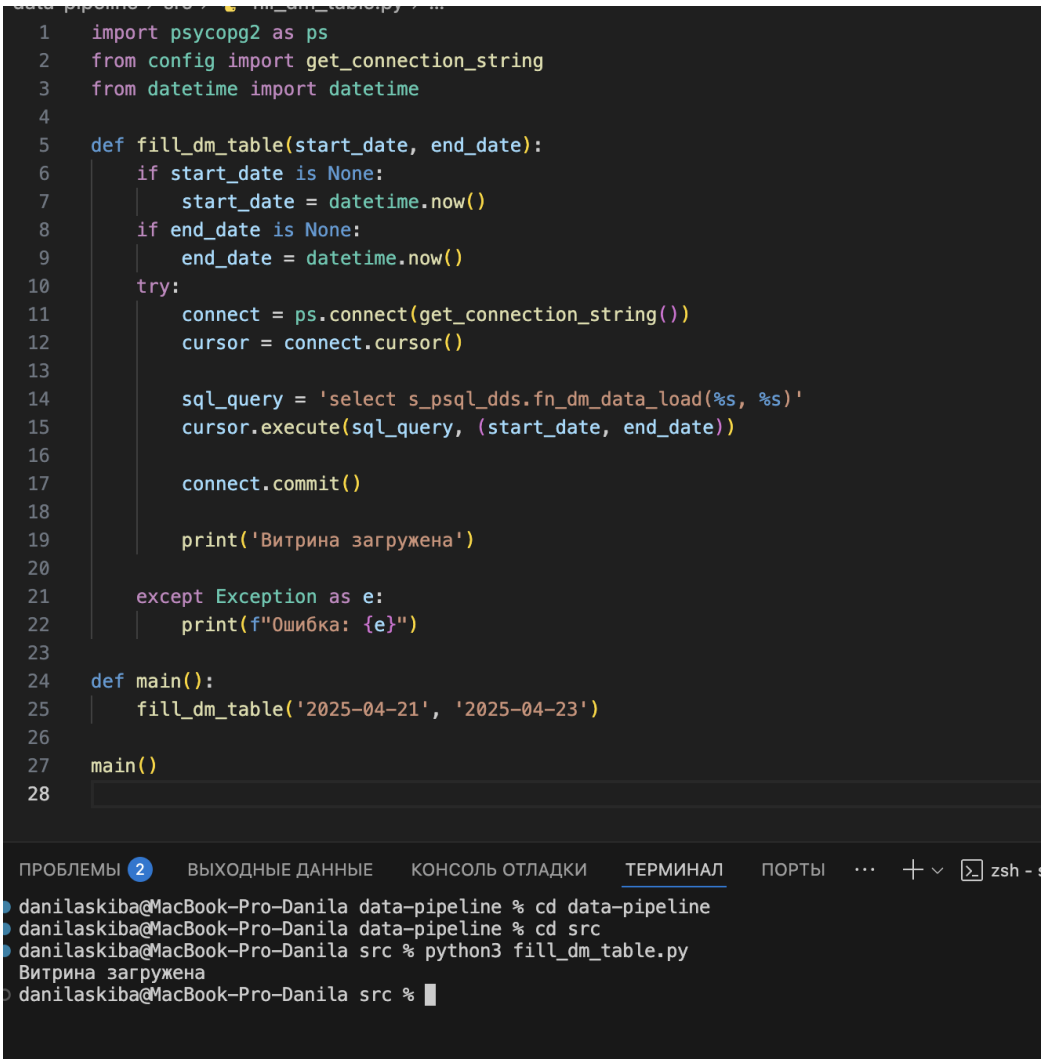


Рисунок 3 – Загрузка данных в таблицу *t_dm_task*

```
select * from s_psql_dds.t_dm_task
```

Data Output | Messages | Notifications

Showing rows: 1 to 1000 | Page No: 1 of 131

	id [PK] bigint	vendor_id_dim integer	ratecode_id_dim integer	pu_location_id_dim integer	do_location_id_dim integer	payment_type_dim integer	tpep_pickup_datetime timestamp without time zone
1	5636993	3	1	140	234	3	2025-04-21 02:37:19
2	5636994	3	1	67	201	3	2025-04-21 12:08:03
3	5636995	3	1	67	26	3	2025-04-21 13:57:58
4	5636996	3	1	71	149	3	2025-04-21 17:38:13
5	5636997	3	2	34	204	3	2025-04-21 19:39:51
6	5636998	3	1	142	149	1	2025-04-21 20:06:39
7	5636999	3	1	34	34	3	2025-04-21 22:03:00
8	5637000	3	1	150	43	1	2025-04-22 13:11:44
9	5637001	3	1	7	251	3	2025-04-22 15:56:22
10	5637002	3	1	93	93	3	2025-04-22 17:26:36
11	5637003	3	1	67	61	3	2025-04-22 17:51:02
12	5637004	3	1	198	156	3	2025-04-22 17:59:24
13	5637005	3	1	240	73	3	2025-04-22 19:32:37
14	5637006	3	1	7	135	3	2025-04-22 22:20:28
15	5637007	3	1	34	65	3	2025-04-22 22:30:32
16	5637008	3	1	76	207	1	2025-04-23 07:45:17
17	5637009	3	1	142	251	3	2025-04-23 09:39:36
18	5637010	3	2	34	16	3	2025-04-23 09:48:07
19	5637011	3	5	40	67	3	2025-04-23 11:36:52
20	5637012	3	1	3	142	3	2025-04-23 12:10:32
21	5637013	3	1	76	7	3	2025-04-23 12:18:55
22	5637014	3	1	93	3	3	2025-04-23 14:24:30
23	5637015	3	1	226	93	3	2025-04-23 14:42:32
24	5637016	3	1	69	170	3	2025-04-23 15:55:07

Рисунок 4 – Полученная таблица *t_dm_task*

Витрина данных – это подмножество хранилища данных, ориентированное на конкретную предметную область или бизнес-задачу. Она представляет пользователям удобный, агрегированный и безопасный доступ к данным.

В проекте витрина реализована с помощью представления (*VIEW*) с названием *v_dm_task*. Представление не хранит данные физически, а является сохранённым *SQL*-запросом (*SELECT * FROM t_dm_task*).

Использование витрин данных распространенно при перекладке данных между СУБД. Финальным шагом в лабораторной работе была организация перекладки данных из витрины *PostgreSQL* в *MySQL:*

1. Создана идентичная таблица в *MySQL* с такой же структурой, как у *t_dm_task.*

2. Реализована функция *migrate_to_my_sql* на *Python*, которая собирает *SQL*-запросом данные представления *v_dm_task*, очищает таблицу *t_dm_task_mysql* и заполняет новыми данными.

Работа функции *migrate_to_my_sql()* представлена на рисунке 5 и 6.

```python
import psycopg2
import mysql.connector
from config import get_connection_string, MYSQL_CONFIG

def migrate_to_mysql():
    pg_conn = psycopg2.connect(get_connection_string())
    pg_cur = pg_conn.cursor()

    pg_cur.execute("""
        SELECT * FROM s_psql_dds.v_dm_task
        order by tpep_pickup_datetime
    """)
    data = pg_cur.fetchall()
    pg_conn.close()


    if not data:
        print('Нет данных')
        return

    mysql_conn = mysql.connector.connect(**MYSQL_CONFIG)
    mysql_cur = mysql_conn.cursor()

    mysql_cur.execute("""
            truncate t_dm_task_mysql
    """)

    for row in data:
        mysql_cur.execute("""
        insert into t_dm_task_mysql
        (vendor_id_dim, ratecode_id_dim, pu_location_id_dim,
        do_location_id_dim, payment_type_dim, tpep_pickup_datetime,
        tpep_dropoff_datetime, passenger_count, trip_distance,
        tip_amount, improvement_surcharge, total_amount,
        congestion_surcharge, airport_fee, cbd_congestion_fee)
        VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)
        """, row[1:])


    mysql_conn.commit()
    mysql_conn.close()

    print('Успешно скопировано')


def main():
    migrate_to_mysql()

main()
```

```
ПРОБЛЕМЫ 2  ВЫХОДНЫЕ ДАННЫЕ  КОНСОЛЬ ОТЛАДКИ  ТЕРМИНАЛ  ПОРТЫ  JUPYTER  PLAYWRIGH

danilaskiba@MacBook-Pro-Danila data-pipeline % cd data-pipeline/src
danilaskiba@MacBook-Pro-Danila src % python3 migrate_to_mysql.py
Успешно скопировано
danilaskiba@MacBook-Pro-Danila src %
```

Рисунок 5 – Реализация перекладки данных

```
mysql> select * from t_dm_task_mysql;
+--------+--------------+------------------+----------------------+---------------------+-------------------+--------------------+---------------------+-----------------+---------------+------------+-----------
---------+--------------+----------------------+-------------+---------------------+
| id     | vendor_id_dim | ratecode_id_dim | pu_location_id_dim | do_location_id_dim | payment_type_dim | tpep_pickup_datetime | tpep_dropoff_datetime | passenger_count | trip_distance | tip_amount | improvemen
t_surcharge | total_amount | congestion_surcharge | airport_fee | cbd_congestion_fee |
+--------+--------------+------------------+----------------------+---------------------+-------------------+--------------------+---------------------+-----------------+---------------+------------+-----------
---------+--------------+----------------------+-------------+---------------------+
|      1 |            3 |                1 |                   93 |                 146 |                 2 | 2025-04-21 00:00:02 | 2025-04-21 00:05:10 |               1 |          1.17 |       2.73 |
       1.00 |        16.38 |                 2.50 |        0.00 |                0.75 |
|      2 |            3 |                1 |                   93 |                  39 |                 2 | 2025-04-21 00:00:10 | 2025-04-21 00:12:13 |               2 |          1.38 |       3.57 |
       1.00 |        21.42 |                 2.50 |        0.00 |                0.75 |
|      3 |            3 |                1 |                   93 |                  47 |                 2 | 2025-04-21 00:00:12 | 2025-04-21 00:13:15 |               2 |          2.85 |       4.27 |
       1.00 |        25.62 |                 2.50 |        0.00 |                0.75 |
|      4 |            3 |                1 |                   30 |                  61 |                 3 | 2025-04-21 00:00:12 | 2025-04-21 00:17:03 |               9 |          4.72 |       0.00 |
       1.00 |        21.74 |                 NULL |        NULL |                0.00 |
|      5 |            3 |                1 |                   93 |                 162 |                 2 | 2025-04-21 00:00:15 | 2025-04-21 00:07:51 |               1 |          1.58 |       2.36 |
       1.00 |        18.11 |                 2.50 |        0.00 |                0.75 |
|      6 |            3 |                1 |                   93 |                  47 |                 2 | 2025-04-21 00:00:15 | 2025-04-21 00:16:49 |               9 |          2.78 |       2.28 |
       1.00 |        25.03 |                 2.50 |        0.00 |                0.75 |
|      7 |            3 |                1 |                   34 |                  40 |                 2 | 2025-04-21 00:00:16 | 2025-04-21 00:16:48 |               1 |         11.49 |      10.00 |
       1.00 |        62.15 |                 0.00 |        1.75 |                0.00 |
|      8 |            3 |                1 |                   40 |                 109 |                 2 | 2025-04-21 00:00:16 | 2025-04-21 00:17:35 |               1 |         13.31 |      13.01 |
       1.00 |        79.80 |                 0.00 |        1.75 |                0.00 |
|      9 |            3 |                1 |                  159 |                 142 |                 1 | 2025-04-21 00:00:18 | 2025-04-21 00:05:19 |               2 |          1.30 |       0.00 |
       1.00 |        13.65 |                 2.50 |        0.00 |                0.75 |
|     10 |            2 |                1 |                  226 |                 162 |                 1 | 2025-04-21 00:00:20 | 2025-04-21 00:09:01 |               1 |          1.60 |       0.00 |
       1.00 |        16.45 |                 2.50 |        0.00 |                0.75 |
|     11 |            3 |                1 |                   93 |                 247 |                 2 | 2025-04-21 00:00:20 | 2025-04-21 00:14:51 |               1 |          3.47 |       4.55 |
       1.00 |        27.30 |                 2.50 |        0.00 |                0.75 |
|     12 |            3 |                1 |                  148 |                 218 |                 1 | 2025-04-21 00:00:23 | 2025-04-21 00:25:59 |               1 |          6.59 |       0.00 |
       1.00 |        36.70 |                 2.50 |        0.00 |                0.00 |
|     13 |            3 |                1 |                  222 |                  68 |                 3 | 2025-04-21 00:00:25 | 2025-04-21 00:27:15 |               9 |         16.07 |       0.00 |
       1.00 |         9.31 |                 NULL |        NULL |                0.75 |
|     14 |            3 |                1 |                   34 |                 110 |                 1 | 2025-04-21 00:00:29 | 2025-04-21 00:23:45 |               2 |         10.96 |       0.00 |
       1.00 |        52.35 |                 0.00 |        1.75 |                0.00 |
|     15 |            3 |                1 |                   40 |                   7 |                 2 | 2025-04-21 00:00:39 | 2025-04-21 00:15:30 |               1 |          9.00 |      10.79 |
       1.00 |        64.73 |                 2.50 |        1.75 |                0.75 |
|     16 |            3 |                1 |                   93 |                 140 |                 2 | 2025-04-21 00:00:43 | 2025-04-21 00:16:10 |               1 |          2.15 |       4.27 |
       1.00 |        25.62 |                 2.50 |        0.00 |                0.75 |
|     17 |            3 |                1 |                  142 |                  66 |                 2 | 2025-04-21 00:00:44 | 2025-04-21 00:05:00 |               1 |          0.58 |       2.31 |
       1.00 |        13.86 |                 2.50 |        0.00 |                0.75 |
|     18 |            3 |                1 |                   45 |                  68 |                 2 | 2025-04-21 00:00:48 | 2025-04-21 00:03:37 |               1 |          0.69 |       2.17 |
       1.00 |        13.02 |                 2.50 |        0.00 |                0.75 |
|     19 |            3 |                1 |                   25 |                 245 |                 3 | 2025-04-21 00:00:59 | 2025-04-21 00:25:57 |               9 |         11.27 |       0.00 |
```

Рисунок 6 – Переложенные данные в таблицу *t_dm_task_mysql*

4. Инструкция по запуску проекта

Для запуска проекта необходимо следующее предустановленное программное обеспечение:

- *Python 3.12.0* (или выше),
- *PostgreSQL 15+*,
- *MySQL 8.0+*,
- *Git* для клонирования репозитория,
- *pip (*менеджер пакетов *Python).*

Порядок действий запуска проекта:

1. скачать исходный датасет по ссылке: https://drive.google.com/file/d/1IMGfftVscy-GR9YLSRmG989C-_krBxtq/view?usp=drive_link
2. с помощью команды *git clone* https://github.com/Danila-Skiba/data-pipeline клонировать репозиторий проекта;
3. поместить файл *df.csv* в папку *src/;*
4. установить зависимости проекта командой *pip install -r requirements.txt;*

5. создать файл *src/passwords.env* с параметрами *DB_PgSQL*=<пароль суперпользователя *PostgreSQL*>, *DB_MySQL*=<пароль суперпользователя *MySQL*>;

6. Создать базы данных *etl_database*, *taxi_dwh* в средах *PostgreSQL*, *MySQL;*

7. выполнить *sql* скрипты создания схемы данных, таблиц, функций и представлений в среде *PostgreSQL* или через функцию *src*/*execute_sql.py* с параметром пути до файла **.sql*;

8. выполнить файл *src*/*create_object_lab2.py* для создания таблиц *DWH;*

9. при необходимости изменить параметры файла *src/config.py;*

10. запустить *ETL* процесс файлом *etl.py;*

11. запустить процесс перекладки данных файлом *src/migrate_to_mysql.py;*

12. проверить таблицу *t_dm_task_mysql* в базе данных *taxi_dwh* СУБД *MySQL.*

Заключение

В результате выполнения лабораторных работ был разработан процесс обработки данных через *ETL*-пайплайн, а также построено хранилище данных *DWH* и продемонстрирована перекладка витрины данных между разными СУБД.

ПРИЛОЖЕНИЕ А

Файл load_data_to_db.py

```
import pandas as pd
import psycopg2 as ps
from src.config import get_connection_string


def load_data_to_db(data:pd.DataFrame, path: str, table_name: str  =
't_sql_source_unstructured'):
    if('Unnamed: 0' in data.columns):
        data = data.drop(columns=['Unnamed: 0'])


    with open(path, 'r') as f:
        sql_script = f.read()


    connect = ps.connect(get_connection_string())
    cursor = connect.cursor()


    cursor.execute(sql_script)


    cursor.execute(f"TRUNCATE TABLE s_psql_dds.{table_name}")


    for _, row in data.iterrows():
        values = [str(val) if pd.notna(val) else None for val in row]
        placeholders = ', '.join(['%s'] * len(values))


        query = "INSERT INTO s_psql_dds.t_sql_source_unstructured VALUES (" +
placeholders + ")"


        cursor.execute(query, values)
```

```
    connect.commit()
    connect.close()
```

Файл fn_etl_data_load.sql

```
create or replace function s_psql_dds.fn_etl_data_load(start_date date, end_date date)
returns void as $$
begin
    truncate s_psql_dds.t_sql_source_structured;

    insert into s_psql_dds.t_sql_source_structured (
        vendor_id,
        tpep_pickup_datetime,
        tpep_dropoff_datetime,
        passenger_count,
        trip_distance,
        ratecode_id,
        pu_location_id,
        do_location_id,
        payment_type,
        tip_amount,
        improvement_surcharge,
        total_amount,
        congestion_surcharge,
        airport_fee,
        cbd_congestion_fee
    )
    select distinct
```

```
--Vendor_ID(Поставщик)--
case
    when vendor_id is null or vendor_id = '' then 0
    when vendor_id in ('1', '2') then vendor_id::INTEGER
    else 0
end as vendor_id,

--tpep_pickup_datetime--
case
    when tpep_pickup_datetime::TIMESTAMP >=
tpep_dropoff_datetime::TIMESTAMP then null
    when tpep_pickup_datetime::TIMESTAMP > CURRENT_TIMESTAMP
then null
    else tpep_pickup_datetime::TIMESTAMP
end as tpep_pickup_datetime,

--tpep_dropoff_datetime--
case
    when tpep_dropoff_datetime::TIMESTAMP > CURRENT_TIMESTAMP
then null
    else tpep_dropoff_datetime::TIMESTAMP
end as tpep_dropoff_datetime,

--passenger_count--
case
    when passenger_count is null or passenger_count = '' then null
    when passenger_count ~ '^-?\d+(\.\d+)?$' then
        case
            -- Сначала в DECIMAL, потом в INTEGER (отбрасывает
дробную часть)
```

```
            when passenger_count::DECIMAL < 0 then null
            when passenger_count::DECIMAL > 9 then 9
            else passenger_count::DECIMAL::INTEGER
        end
    else null
end as passenger_count,

--TRIP_DISTANCE--
case
    when trip_distance is null or trip_distance ='' then 0.0
    when trip_distance ~ '^-?\d+(\.\d+)?$' then
        case
            when trip_distance::DECIMAL < 0 THEN 0.0
            when trip_distance::DECIMAL > 500 then 500.0
            else ROUND(trip_distance::DECIMAL, 2)
        end
    else 0.0
end as trip_distance,

--RATECODE_ID--
case
    when ratecode_id is null or ratecode_id = '' then 1
    when ratecode_id ~ '^-?\d+(\.\d+)?$' then
        case
            when ratecode_id::DECIMAL::INTEGER between 1 and 6
                then ratecode_id::DECIMAL::INTEGER
            else 1
        end
    else 1
```

```
    end as ratecode_id,

    --pu_location--
    case
      when pu_location_id IS NULL OR pu_location_id = '' then NULL
      when pu_location_id ~ '^\d+$' then
        case
          when pu_location_id::INTEGER BETWEEN 1 AND 263 then
pu_location_id::INTEGER
          when pu_location_id::INTEGER IN (264, 265) then
pu_location_id::INTEGER
          else NULL
        end
      else NULL
    end as pu_location_id,

    -- DO_LOCATION_ID--
    case
      when do_location_id is null or do_location_id = '' then NULL
      when do_location_id ~ '^-?\d+(\.\d+)?$' then
        case
          when do_location_id::DECIMAL::INTEGER between 1 and 263
            then do_location_id::DECIMAL::INTEGER
          when do_location_id::DECIMAL::INTEGER in (264, 265)
            then do_location_id::DECIMAL::INTEGER
          else NULL
        end
      else NULL
    end as do_location_id,
```

```
--payment_type--
case
   when upper(payment_type) = 'CASH' then 'Cash'
   when upper(payment_type) = 'CREDIT' then 'Credit'
   when upper(payment_type) = 'CREDIT CARD' then 'Credit'
   when upper(payment_type) = 'NO CHARGE' then 'No charge'
   when upper(payment_type) = 'DISPUTE' then 'Dispute'
   when upper(payment_type) = 'UNKNOWN' then 'Unknown'

   when payment_type ilike '%cash%' then 'Cash'
   when payment_type ilike '%credit%' then 'Credit'
   when payment_type ilike '%card%' then 'Credit'
   when payment_type ilike '%no charge%' then 'No charge'
   when payment_type ilike '%dispute%' then 'Dispute'
   else 'Unknown'
end as payment_type,

--TIP_AMOUNT--
case
   when tip_amount ~ '^-?\d+(\.\d+)?$' then
      case
         when tip_amount::DECIMAL < 0 then NULL
         when tip_amount::DECIMAL > 1000 then 1000.0
         else ROUND(tip_amount::DECIMAL, 2)
      end
   else NULL
end as tip_amount,
```

```
--IMPROVEMENT_SURCHARGE--
case
   when improvement_surcharge ~ '^\d+(\.\d+)?$'
      and improvement_surcharge::DECIMAL BETWEEN 0 AND 1
      then ROUND(improvement_surcharge::DECIMAL, 2)
   else NULL
end as improvement_surcharge,

--total_amount--
case
   when total_amount ~ '^-?\d+(\.\d+)?$' then
      case
         when total_amount::DECIMAL < 0 then NULL
         when total_amount::DECIMAL > 10000 then NULL
         else ROUND(total_amount::DECIMAL, 2)
      end
   else NULL
end as total_amount,

--CONGESTION_SURCHARGE--
case
   when congestion_surcharge ~ '^-?\d+(\.\d+)?$' then
      case
         when congestion_surcharge::DECIMAL < 0 then NULL
         when congestion_surcharge::DECIMAL > 2.5 then NULL
         else ROUND(congestion_surcharge::DECIMAL, 2)
      end
   else NULL
end as congestion_surcharge,
```

```
    --AIRPORT_FEE--
    case
        when airport_fee ~ '^-?\d+(\.\d+)?$' then
            case
                when airport_fee::DECIMAL < 0 then NULL
                when airport_fee::DECIMAL < 10.0 then airport_fee::DECIMAL
                else NULL
            end
        else NULL
    end as airport_fee,

    --CBD_CONGESTION_FEE--
    case
        when cbd_congestion_fee ~ '^-?\d+(\.\d+)?$' then
            case
                when cbd_congestion_fee::DECIMAL < 0 then NULL
                when cbd_congestion_fee::DECIMAL > 0.8 then NULL
                else ROUND(cbd_congestion_fee::DECIMAL, 2)
            end
        else NULL
    end as cbd_congestion_fee

FROM s_psql_dds.t_sql_source_unstructured
where
    tpep_pickup_datetime is not null
    and vendor_id is not null
    and total_amount is not null
```

and tpep_pickup_datetime::DATE BETWEEN start_date AND end_date and tpep_pickup_datetime::DATE is not null

ORDER BY tpep_pickup_datetime;

RAISE NOTICE 'ETL завершен для периода % - %', start_date, end_date;

END;

$$ LANGUAGE plpgsql;

ПРИЛОЖЕНИЕ Б

Файл fn_dm_data_load.sql

```
CREATE OR REPLACE FUNCTION s_psql_dds.fn_dm_data_load(
    start_dt DATE,
    end_dt DATE
)
RETURNS VOID AS $$
BEGIN

    truncate s_psql_dds.t_dm_task;



    INSERT INTO s_psql_dds.dim_vendor (name)
    SELECT DISTINCT CAST(vendor_id AS VARCHAR)
    FROM s_psql_dds.t_sql_source_structured
    WHERE DATE(tpep_pickup_datetime) BETWEEN start_dt AND end_dt
      AND vendor_id IS NOT NULL
    ON CONFLICT (name) DO NOTHING;

    INSERT INTO s_psql_dds.dim_ratecode (name)
    SELECT DISTINCT CAST(ratecode_id AS VARCHAR)
    FROM s_psql_dds.t_sql_source_structured
    WHERE DATE(tpep_pickup_datetime) BETWEEN start_dt AND end_dt
      AND ratecode_id IS NOT NULL
    ON CONFLICT (name) DO NOTHING;



    INSERT INTO s_psql_dds.dim_location (name)
```

```
SELECT DISTINCT location_name
FROM (
    SELECT CAST(pu_location_id AS VARCHAR) AS location_name
    FROM s_psql_dds.t_sql_source_structured
    WHERE DATE(tpep_pickup_datetime) BETWEEN start_dt AND end_dt
      AND pu_location_id IS NOT NULL
    UNION
    SELECT CAST(do_location_id AS VARCHAR) AS location_name
    FROM s_psql_dds.t_sql_source_structured
    WHERE DATE(tpep_pickup_datetime) BETWEEN start_dt AND end_dt
      AND do_location_id IS NOT NULL
) AS locations
ON CONFLICT (name) DO NOTHING;
INSERT INTO s_psql_dds.dim_payment (name)
SELECT DISTINCT payment_type
FROM s_psql_dds.t_sql_source_structured
WHERE DATE(tpep_pickup_datetime) BETWEEN start_dt AND end_dt
  AND payment_type IS NOT NULL
ON CONFLICT (name) DO NOTHING;


INSERT INTO s_psql_dds.t_dm_task (
    vendor_id_dim,
    ratecode_id_dim,
    pu_location_id_dim,
    do_location_id_dim,
    payment_type_dim,
    tpep_pickup_datetime,
    tpep_dropoff_datetime,
```

```
    passenger_count,
    trip_distance,
    tip_amount,
    improvement_surcharge,
    total_amount,
    congestion_surcharge,
    airport_fee,
    cbd_congestion_fee
)
SELECT
    v.id AS vendor_id_dim,
    r.id AS ratecode_id_dim,
    pl.id AS pu_location_id_dim,
    dl.id AS do_location_id_dim,
    p.id AS payment_type_dim,
    s.tpep_pickup_datetime,
    s.tpep_dropoff_datetime,
    s.passenger_count,
    s.trip_distance,
    s.tip_amount,
    s.improvement_surcharge,
    s.total_amount,
    s.congestion_surcharge,
    s.airport_fee,
    s.cbd_congestion_fee
FROM s_psql_dds.t_sql_source_structured s
LEFT JOIN s_psql_dds.dim_vendor v ON CAST(s.vendor_id AS VARCHAR) =
v.name
```

```
    LEFT JOIN s_psql_dds.dim_ratecode r ON CAST(s.ratecode_id AS VARCHAR)
= r.name
    LEFT JOIN s_psql_dds.dim_location pl ON CAST(s.pu_location_id AS
VARCHAR) = pl.name
    LEFT JOIN s_psql_dds.dim_location dl ON CAST(s.do_location_id AS
VARCHAR) = dl.name
    LEFT JOIN s_psql_dds.dim_payment p ON s.payment_type = p.name
    WHERE DATE(s.tpep_pickup_datetime) BETWEEN start_dt AND end_dt;

    RAISE NOTICE 'Загружено данных за период: % - %', start_dt, end_dt;

EXCEPTION
    WHEN OTHERS THEN
        RAISE EXCEPTION 'Ошибка при загрузке данных: %', SQLERRM;
END;
$$ LANGUAGE plpgsql;
```