

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 №3-1**

**Тема:**

«Интеграция данных из нескольких источников. Обработка и  
согласование данных из разных источников»

Выполнил(а): Морозова Валерия АДЭУ-211

Преподаватель:

Москва

2025

**Цель работы:** получить практические навыки интеграции, обработки и согласования данных из различных источников с использованием Python и его библиотек.

**Задачи:**

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

**Общее задание.**

Необходимо нарисовать верхнеуровневую архитектуру аналитического решения в <https://draw.io/> .

Использовать следующий шаблон:

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

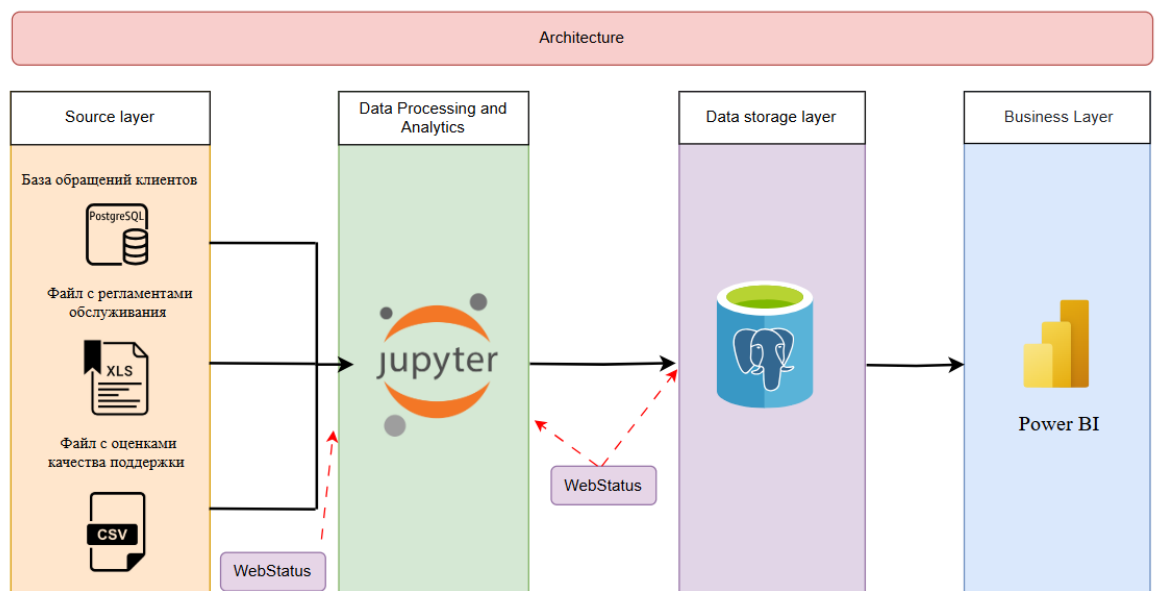


Рисунок 1. Архитектура аналитического решения

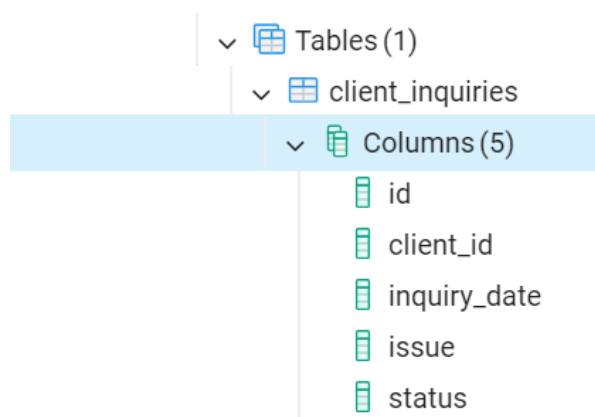
**Вариант 10. Интеграция данных технической поддержки:**

- PostgreSQL база обращений клиентов.
- Excel файл с регламентами обслуживания.
- CSV файл с оценками качества поддержки.

**Задача:** проанализировать качество технической поддержки.

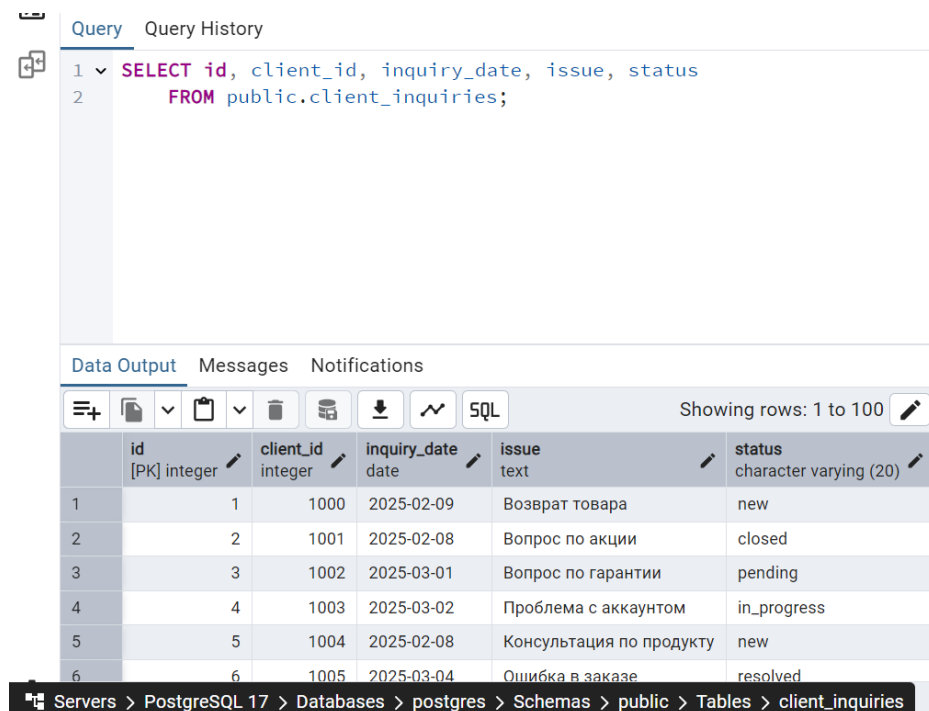
```
Query  Query History
1  CREATE TABLE client_inquiries (
2  id SERIAL PRIMARY KEY,
3  client_id INTEGER NOT NULL,
4  inquiry_date DATE NOT NULL,
5  issue TEXT NOT NULL,
6  status VARCHAR(20) NOT NULL
7  );
```

Рисунок 2. Создание таблицы обращений клиентов в PostgreSQL



Tables (1)
client_inquiries
Columns (5)
id
client_id
inquiry_date
issue
status

Рисунок 3. Таблица успешно создана



```
Query  Query History
1  SELECT id, client_id, inquiry_date, issue, status
2  FROM public.client_inquiries;
```

	id [PK] integer	client_id integer	inquiry_date date	issue text	status character varying (20)
1	1	1000	2025-02-09	Возврат товара	new
2	2	1001	2025-02-08	Вопрос по акции	closed
3	3	1002	2025-03-01	Вопрос по гарантии	pending
4	4	1003	2025-03-02	Проблема с аккаунтом	in_progress
5	5	1004	2025-02-08	Консультация по продукту	new
6	6	1005	2025-03-04	Ошибка в заказе	resolved

Servers > PostgreSQL 17 > Databases > postgres > Schemas > public > Tables > client\_inquiries

Рисунок 4. Таблица заполнена данными

Таблица обращений клиентов содержит данные по айди клиента, дате обращения, проблеме и статусе запроса.

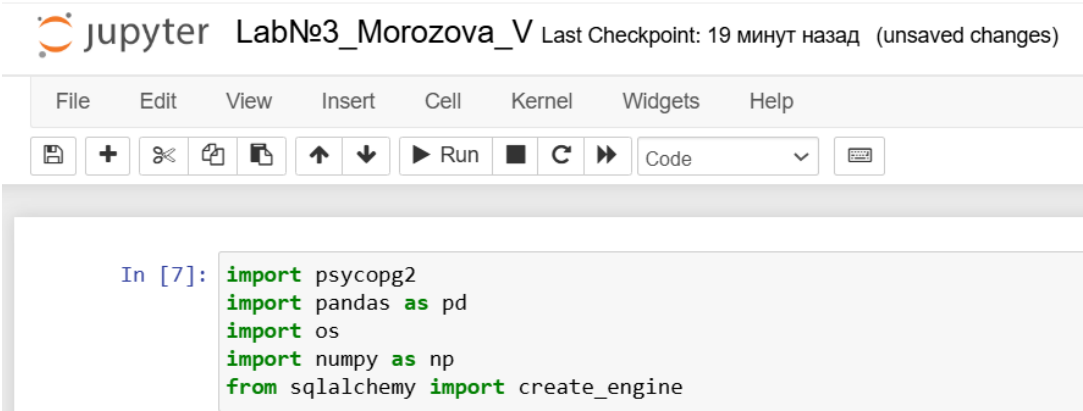


Рисунок 5. Импорт библиотек

1. Чтение данных из разных источников

```
In [8]: # Чтение из CSV
df_csv = pd.read_csv('C:/Users/User/Downloads/support_quality_ratings.csv', sep = ',')
df_csv
```

Out[8]:

	rating_id	inquiry_id	client_id	agent_id	rating_date	overall_rating	resolution_speed_rating	politeness_rating	knowledge_rating	feedback	would_r
0	1	47	1028	9	2025-02-15	2		2	3	2	Разочарован сервисом
1	2	17	1050	2	2025-01-27	4		5	4	3	Все объяснили понятно и доступно
2	3	78	1024	9	2024-10-29	4		3	3	5	Отличный сервис, быстрое решение проблемы
3	4	61	1020	4	2024-12-25	4		5	3	4	Всегда приятно иметь дело с профессионалами

Рисунок 6. Чтение из CSV

```
In [9]: # Чтение из Excel
df_excel = pd.read_excel("C:/Users/User/Downloads/service_regulations.xlsx")
df_excel
```

Out[9]:

	regulation_id		title	department	sla_target	created_date	last_updated	is_active
0	1		Обработка входящих звонков	Отдел продаж	91%	2024-02-28	2025-02-13	False
1	2		Эскалация проблем	Отдел продаж	95%	2023-09-26	2025-02-12	True
2	3		Техническая поддержка первого уровня	Отдел маркетинга	91%	2024-07-28	2025-03-06	True
3	4		Регламент доступа к системе - Специальный	Техническая поддержка	95%	2024-03-09	2024-12-11	False
4	5		Обслуживание VIP-клиентов - Временный	Отдел качества	92%	2024-07-16	2024-12-13	True
...	...		...	...	...	...	...	...
95	96		Регламент резервного копирования - Базовый	Отдел качества	93%	2023-10-03	2024-11-20	False
96	97		Процедура отчетности	Отдел тестирования	99%	2023-11-28	2025-01-12	False
97	98		Процедура обучения новых сотрудников	Отдел маркетинга	91%	2024-04-29	2024-11-07	True
98	99		Регламент документирования	Финансовый отдел	97%	2024-02-01	2025-01-22	True
99	100		Работа с критическими инцидентами	Отдел тестирования	96%	2024-03-31	2024-09-20	True

100 rows × 7 columns

Рисунок 7. Чтение из Excel

```
File Edit View Insert Cell Kernel Widgets Help
[Icons] Run [Buttons] Code [Dropdown] [Menu]

In [3]: # Настройки подключения к базе данных
DB_CONFIG = {
    "dbname": "postgres",
    "user": "postgres",
    "password": "3421",
    "host": "localhost",
    "port": 5433
}

# Подключение к базе данных
try:
    conn = psycopg2.connect(**DB_CONFIG)
    print("Подключение установлено")
except Exception as e:
    print(f"Ошибка подключения: {e}")

# Получение нужной таблицы
try:
    query = "SELECT * FROM public.client_inquiries;"
    table = pd.read_sql_query(query, conn)
    print("client_inquiries:")
    print(table)
except Exception as e:
    print(f"Ошибка выполнения запроса: {e}")

# Закрытие подключения
finally:
    if conn:
        conn.close()
        print("Подключение закрыто")

Подключение установлено
client_inquiries:
   id  client_id  inquiry_date  issue  status
0  301        1000    2025-02-09  Возврат товара  new
```

Рисунок 8. Чтение данных из PostgreSQL

## 2. Очистка данных

```
# 2. Очистка данных

In [10]: df_csv.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 12 columns):
#   Column                Non-Null Count  Dtype
---  ---
0   rating_id             100 non-null    int64
1   inquiry_id            100 non-null    int64
2   client_id             100 non-null    int64
3   agent_id              100 non-null    int64
4   rating_date           100 non-null    object
5   overall_rating        100 non-null    int64
6   resolution_speed_rating 100 non-null    int64
7   politeness_rating     100 non-null    int64
8   knowledge_rating      100 non-null    int64
9   feedback              100 non-null    object
10  would_recommend        100 non-null    bool
11  regulation_followed    100 non-null    bool
dtypes: bool(2), int64(8), object(2)
memory usage: 8.1+ KB

In [16]: # Удаление дубликатов
df_csv = df_csv.drop_duplicates()

In [17]: #проверка на наличие нулевых значений
df_csv.isna().sum().sum()/len(df_csv)

Out[17]: 0.0
```

Рисунок 9. Удаление дубликатов и проверка на наличие нулевых значений в df\_csv

```
jupyter Lab№3_Morozova_V Last Checkpoint: час назад (autosaved)

File Edit View Insert Cell Kernel Widgets Help

[+] [-] [↶] [↷] [↵] [↴] [↵] [↴] [Run] [Stop] [↶] [↷] Code [v] [⌵]

In [18]: # Приведение типов данных
df_csv['rating_date'] = pd.to_datetime(df_csv['rating_date'])

In [19]: # Замена True на 1 и False на 0
df_csv['would_recommend'] = df_csv['would_recommend'].replace({True: 1, False: 0})
df_csv['regulation_followed'] = df_csv['regulation_followed'].replace({True: 1, False: 0})

In [20]: df_csv.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 100 entries, 0 to 99
Data columns (total 12 columns):
#   Column                Non-Null Count  Dtype
---  -
0   rating_id              100 non-null    int64
1   inquiry_id             100 non-null    int64
2   client_id              100 non-null    int64
3   agent_id               100 non-null    int64
4   rating_date            100 non-null    datetime64[ns]
5   overall_rating         100 non-null    int64
6   resolution_speed_rating 100 non-null    int64
7   politeness_rating      100 non-null    int64
8   knowledge_rating       100 non-null    int64
9   feedback               100 non-null    object
10  would_recommend        100 non-null    int64
11  regulation_followed    100 non-null    int64
dtypes: datetime64[ns](1), int64(10), object(1)
memory usage: 10.2+ KB
```

Рисунок 10. Изменение типа данных и замена значений

Как видно согласно повторному анализу, изменения успешно применены.

```
jupyter Lab№3_Morozova_V Last Checkpoint: 2 часа назад (unsaved ch

File Edit View Insert Cell Kernel Widgets Help

[+] [-] [↶] [↷] [↵] [↴] [↵] [↴] [Run] [Stop] [↶] [↷] Code [v] [⌵]

In [12]: df_excel.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 7 columns):
#   Column                Non-Null Count  Dtype
---  -
0   regulation_id          100 non-null    int64
1   title                  100 non-null    object
2   department              100 non-null    object
3   sla_target             100 non-null    object
4   created_date           100 non-null    object
5   last_updated           100 non-null    object
6   is_active              100 non-null    bool
dtypes: bool(1), int64(1), object(5)
memory usage: 4.9+ KB

In [23]: # Удаление дубликатов
df_excel = df_excel.drop_duplicates()

In [24]: # проверка на наличие нулевых значений
df_excel.isna().sum().sum()/len(df_excel)

Out[24]: 0.0
```

Рисунок 11. Удаление дубликатов и проверка на наличие нулевых значений в df\_excel

```
File Edit View Insert Cell Kernel Widgets Help
```

```

In [25]: # Приведение типов данных
df_excel['created_date'] = pd.to_datetime(df_excel['created_date'])
df_excel['last_updated'] = pd.to_datetime(df_excel['last_updated'])

In [26]: # Замена True на 1 и False на 0
df_excel['is_active'] = df_excel['is_active'].replace({True: 1, False: 0})

In [30]: #Удаление посторонних символов (%)
df_excel['sla_target']=df_excel['sla_target'].str.replace(r"%","")

In [31]: #изменение типа данных
df_excel['sla_target']=df_excel['sla_target'].astype('int64')

In [33]: df_excel.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 100 entries, 0 to 99
Data columns (total 7 columns):
#   Column          Non-Null Count  Dtype
---  ---
0   regulation_id    100 non-null    int64
1   title            100 non-null    object
2   department       100 non-null    object
3   sla_target       100 non-null    int64
4   created_date     100 non-null    datetime64[ns]
5   last_updated     100 non-null    datetime64[ns]
6   is_active        100 non-null    int64
dtypes: datetime64[ns](2), int64(3), object(2)
memory usage: 6.2+ KB

```

Рисунок 12. Изменение типа данных, замена значений и удаление посторонних значений

Удаление лишнего символа позволило преобразовать тип данных и в дальнейшем их можно будет использовать для анализа.

```
In [14]: df_postgres.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 5 columns):
#   Column          Non-Null Count  Dtype
---  ---
0   id               100 non-null    int64
1   client_id        100 non-null    int64
2   inquiry_date     100 non-null    object
3   issue            100 non-null    object
4   status           100 non-null    object
dtypes: int64(2), object(3)
memory usage: 4.0+ KB

In [34]: # Удаление дубликатов
df_postgres = df_postgres.drop_duplicates()

In [42]: # Приведение типов данных
df_postgres['inquiry_date'] = pd.to_datetime(df_postgres['inquiry_date'])

In [40]: df_postgres.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 5 columns):
#   Column          Non-Null Count  Dtype
---  ---
0   id               100 non-null    int64
1   client_id        100 non-null    int64
2   inquiry_date     100 non-null    datetime64[ns]
3   issue            100 non-null    object
4   status           100 non-null    object
dtypes: datetime64[ns](1), int64(2), object(2)
memory usage: 4.0+ KB
```

Рисунок 13. Удаление дубликатов и изменение типа данных в df\_postgres

3. Согласование данных

```
# 3. Согласование данных

In [47]: df_csv.rename(columns={'rating_id':'id'}, inplace=True)
df_csv.head(3)

Out[47]:
   id  inquiry_id  client_id  agent_id  rating_date  overall_rating  resolution_speed_rating  politeness_rating  knowledge_rating  feedback  would_recommend
0   1           47       1028         9  2025-02-15              2                2                3                2  Разочарован сервисом  0
1   2           17       1050         2  2025-01-27              4                5                4                3  Все объяснили понятно и доступно  1
2   3           78       1024         9  2024-10-29              4                3                3                5  Отличный сервис, быстрое решение проблемы  1

In [48]: df_excel.rename(columns={'regulation_id':'id'}, inplace=True)
df_excel.head(3)

Out[48]:
   id  title  department  sla_target  created_date  last_updated  is_active
0   1  Обработка входящих звонков  Отдел продаж      91  2024-02-28  2025-02-13      0
1   2  Эскалация проблем  Отдел продаж      95  2023-09-26  2025-02-12      1
2   3  Техническая поддержка первого уровня  Отдел маркетинга      91  2024-07-28  2025-03-06      1
```

Рисунок 14. Согласование данных по id




## # 4. Объединение данных

```
In [62]: # Объединение DataFrame по горизонтали
merge = pd.concat([df_csv, df_excel, df_postgres], axis=1)












print(merge)
```

	id	inquiry_id	client_id	agent_id	rating_date	overall_rating	\
0	1	47	1028	9	2025-02-15	2	
1	2	17	1050	2	2025-01-27	4	
2	3	78	1024	9	2024-10-29	4	
3	4	61	1020	4	2024-12-25	4	
4	5	21	1022	7	2024-09-23	5	
..	...	...	...	...	...	...	
95	96	4	1049	8	2024-10-20	3	
96	97	40	1035	6	2024-09-24	5	
97	98	57	1041	8	2024-11-23	2	
98	99	83	1020	2	2025-02-08	5	
99	100	43	1025	9	2024-09-24	5	

Рисунок 15. Объединение трех dataframe

jupyter Lab№3\_Morozova\_V Last Checkpoint: 3 часа назад (autosaved)  Logo

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

          Code 

```
In [63]: merge_result = pd.DataFrame(merge)
merge_result
```

Out[63]:

	id	inquiry_id	client_id	agent_id	rating_date	overall_rating	resolution_speed_rating	politeness_rating	knowledge_rating	feedback	...	departm
0	1	47	1028	9	2025-02-15	2		2	3	2	Разочарован сервисом	О прс
1	2	17	1050	2	2025-01-27	4		5	4	3	Все объяснили понятно и доступно	О прс
2	3	78	1024	9	2024-10-29	4		3	3	5	Отличный сервис, быстрое решение проблемы	О маркет
3	4	61	1020	4	2024-12-25	4		5	3	4	Всегда приятно иметь дело с профессионалами	Техниче
4	5	21	1022	7	2024-09-23	5		5	5	5	Вежливый и компетентный специалист	О каче
...	...	...	...	...	...	...	...	...	...	...	...	...
95	96	4	1049	8	2024-10-20	3		2	4	2	Решили проблему, но долго	О каче
96	97	40	1035	6	2024-09-24	5		5	5	5	Отличная работа, спасибо!	О тестиров
97	98	57	1041	8	2024-11-23	2		3	1	1	Очень долго решали проблему	О маркет
98	99	83	1020	2	2025-02-08	5		5	4	5	Оперативно решили мою проблему	Финансс
99	100	43	1025	9	2024-09-24	5		5	5	4	Оперативно решили мою проблему	О тестиров

100 rows x 24 columns

Рисунок 16. Преобразование результата

```

In [66]: # Настройки подключения к базе данных
DB_CONFIG = {
    "dbname": "postgres",
    "user": "postgres",
    "password": "3421",
    "host": "localhost",
    "port": 5433
}

# Создание строки подключения
connection_string = f"postgresql+psycopg2://{DB_CONFIG['user']}:{DB_CONFIG['password']}@{DB_CONFIG['host']}:{DB_CONFIG['port']}/"

# Подключение к базе данных
try:
    engine = create_engine(connection_string)
    conn = engine.connect()
    print("Подключение установлено")
except Exception as e:
    print(f"Ошибка подключения: {e}")

# Загрузка нужной таблицы
try:
    merge_result.to_sql('merge_result', conn, if_exists='replace', index=False)
    print("DataFrame успешно выгружен в PostgreSQL!")
except Exception as e:
    print(f"Ошибка выполнения запроса: {e}")

# Закрытие подключения
finally:
    if conn:
        conn.close()
        print("Подключение закрыто")

```

Подключение установлено  
 DataFrame успешно выгружен в PostgreSQL!  
 Подключение закрыто

Рисунок 17. Сохранение результатов в PostgreSQL

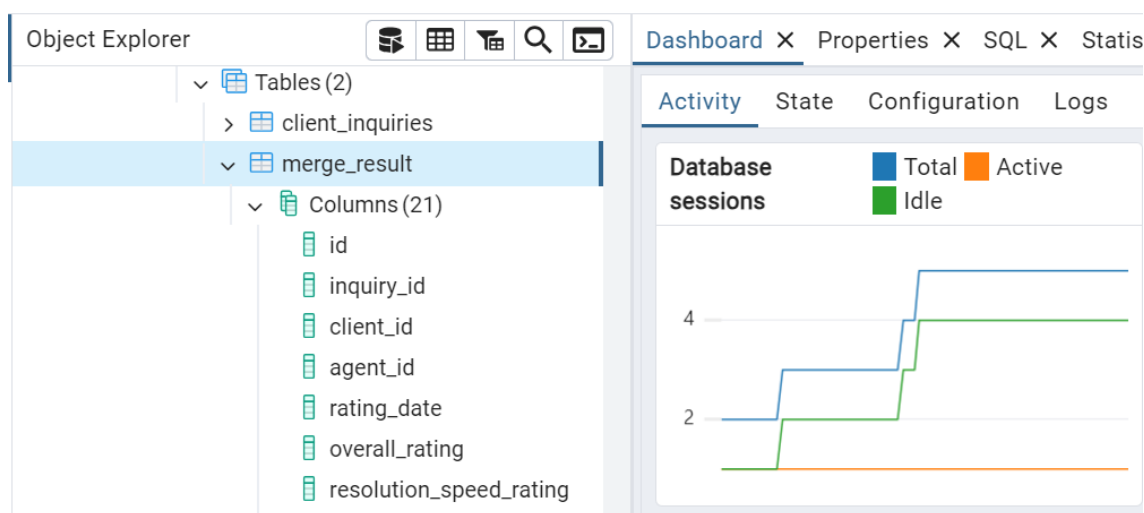


Рисунок 18. Таблица успешно добавлена в мою базу

Данные были загружены в Power BI, где на основании рассчитанных мер и построенных визуализаций, бизнес сможет обращаться к данным и принимать на основе них решения.

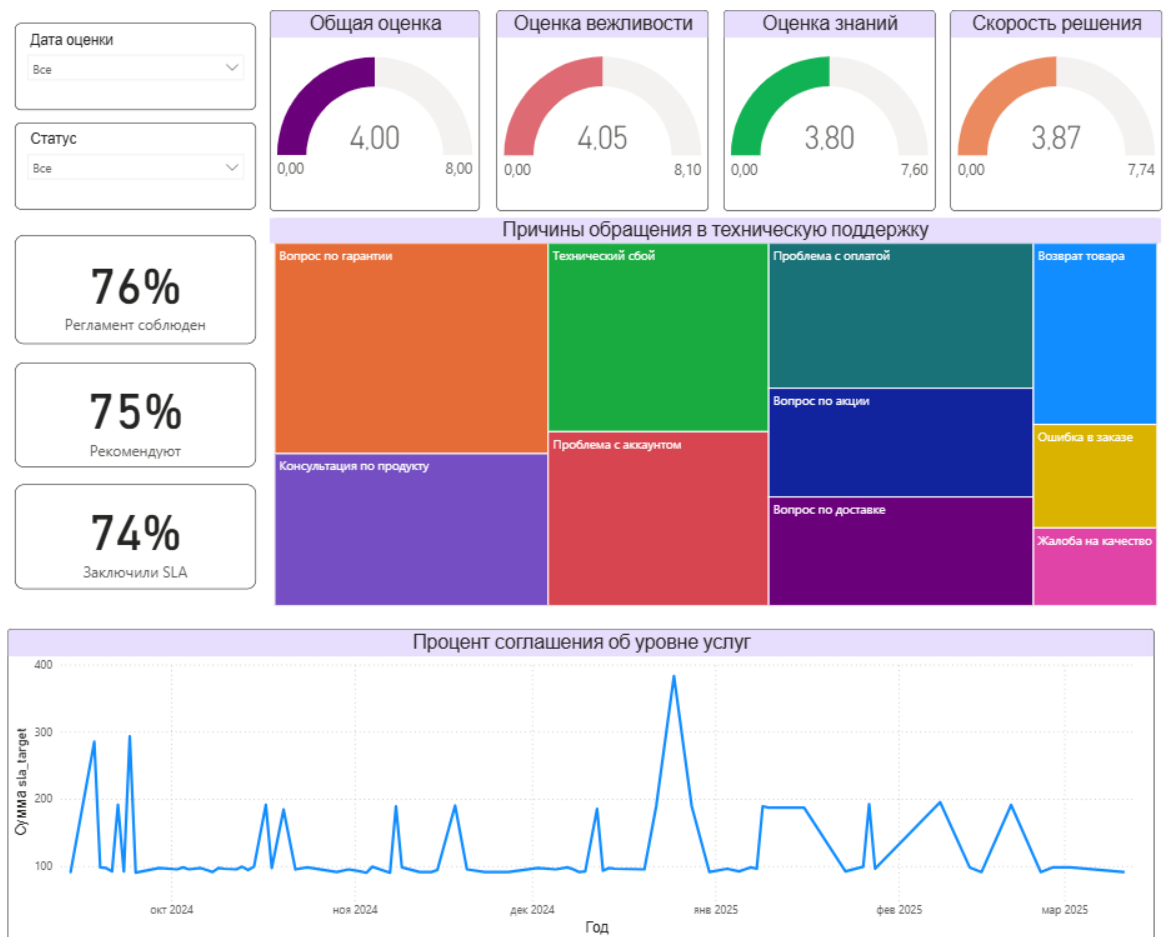


Рисунок 19. Дашборд в Power BI

### Выводы:

1. Изучены методы чтения данных из разных источников.
2. Освоена техника обработки и очистки данных.
3. Согласованы данные из разных источников.
4. Реализовано сохранение обработанных данных в PostgreSQL.
5. Построена архитектура решения.
6. Визуализирован и проведен анализ работы технической службы поддержки.

Согласно дашборду, общая оценка составляет 4 балла, оценка скорости решения вопросов занижает общий показатель.

Регламент соблюден в 76% обращениях.

Больше всего соглашений было заключено к концу 2024 года в декабре.

## 1. Какие методы чтения данных предоставляет pandas?

pandas предлагает множество методов для чтения данных из различных источников:

`pd.read_csv()`: Чтение данных из CSV-файлов.

`pd.read_excel()`: Чтение данных из Excel-файлов.

`pd.read_sql()`: Чтение данных из SQL-баз данных.

`pd.read_json()`: Чтение данных из JSON-файлов.

`pd.read_html()`: Чтение таблиц из HTML-страниц.

`pd.read_parquet()`: Чтение данных из Parquet-файлов.

`pd.read_pickle()`: Чтение данных из файлов, сохраненных с помощью pickle.

## 2. Как обрабатывать пропущенные значения?

`df.isnull()`: Проверка на наличие пропущенных значений.

`df.dropna()`: Удаление строк или столбцов с пропущенными значениями.

`df.fillna()`: Заполнение пропущенных значений с помощью заданного значения или метода (например, среднее, медиана и т.д.).

`df.interpolate()`: Интерполяция пропущенных значений на основе соседних данных.

## 3. Какие типы объединения данных существуют?

*Объединение по индексу*: Использование метода `join()`, чтобы объединить DataFrame по индексам.

*Объединение по столбцам*: Использование метода `merge()`, чтобы объединить DataFrame по значениям в одном или нескольких столбцах.

*Конкатенация*: Использование `pd.concat()` для объединения DataFrame по строкам или столбцам.

## 4. Как проверить качество объединения данных?

*Проверка на дубликаты*: Используйте `df.duplicated()` для поиска дубликатов в объединенных данных.

*Сравнение размеров DataFrame*: Сравните размеры исходных DataFrame и результирующего DataFrame, чтобы убедиться, что данные были объединены корректно.

*Проверка на пропущенные значения:* Используйте `df.isnull().sum()` для проверки наличия пропущенных значений в объединенных данных.

*Визуализация данных:* Визуализация с помощью графиков или таблиц может помочь выявить аномалии в объединенных данных.

## **5. Какие методы дедупликации данных существуют?**

`df.drop_duplicates()`: Удаление дублирующихся строк из DataFrame.

`df.duplicated()`: Возвращает булев массив, указывающий на наличие дубликатов.

`df.groupby()`: Группировка данных с последующим применением агрегирующих функций для удаления дубликатов.