# Документации проекта DWH для Банка: кредитные заявки и выданные кредиты

## Аннотация

Проект направлен на создание системы анализа данных для обработки кредитных заявок. В разработке использованы технологии Python, Apache Airflow и PostgreSQL для реализации ETL-процессов, а для визуализации данных – Metabase. Результатом проекта стало автоматизированное хранилище данных и интерактивные дашборды, что позволило сократить время подготовки аналитических отчетов на 30% благодаря оптимизации обработки данных и улучшению визуализации.

## Оглавление:

[Аннотация 1](#_Toc197952070)

[Введение 1](#_Toc197952071)

[Обзор предметной области 2](#_Toc197952072)

[Архитектура решения 3](#_Toc197952073)

[Компоненты 3](#_Toc197952074)

[Реализация 6](#_Toc197952075)

[Ограничения 9](#_Toc197952076)

[Рекомендации по улучшению 9](#_Toc197952077)

[Результаты 9](#_Toc197952078)

[Заключение 10](#_Toc197952079)

[Приложения 10](#_Toc197952080)

## Введение

**Задачи**

1. **Сбор бизнес-требований:**
   * Определение ключевых показателей эффективности (KPI) для анализа кредитных заявок, таких как процент одобрений и средний срок рассмотрения заявки.
   * Выявление критериев риск-менеджмента, включая зависимость одобрения от кредитного рейтинга.
2. **Автоматизация ETL-процессов:**
   * Ежедневная загрузка данных из источников (PostgreSQL) для обеспечения актуальности информации.
   * Валидация данных: проверка корректности кредитных рейтингов (300–850) и связности данных (клиент → заявка → кредит).
   * Реализация модели Slowly Changing Dimension (SCD2) для отслеживания истории изменений клиентских профилей, что позволяет анализировать динамику изменений.
3. **Визуализация данных:**
   * Создание дашбордов в Metabase, которые отображают ключевые метрики:
     + Динамика одобрений и отказов по заявкам.
     + Среднее время обработки заявок
     + Топ-продукты по объему выдач, что позволяет оценивать популярность кредитных предложений.
     + Показатели по клиентам, что позволяет оценить, какие клиенты чаще берут кредиты и как их выплачивают.
     + Динамику по кредитному порфтелю

**Актуальность**

Проект имеет высокую значимость для банка по следующим причинам:

* **Рост объемов данных:** Обработка более 300 000 заявок ежегодно требует автоматизации и эффективных решений для анализа.
* **Скорость принятия решений:** Ручная аналитика занимала до 4 часов, в то время как автоматизированные отчеты формируются до 15 минут, что значительно ускоряет процесс.
* **Качество данных:** Валидация на этапе ETL выявляет до 12% ошибок, таких как некорректные показатели и дублирование заявок, что увеличивает надежность аналитики.

**Масштаб проекта**

**Данные:**

* Обработка истории за 3 года, включая более 2 миллиона записей по всему кредитному направлению.
* Ежедневный прирост: около 1 000 новых заявок.

**Аудитория:**

* **Кредитные аналитики:** работа с дашбордами для анализа клиентов и заявок.
* **Риск-менеджеры:** мониторинг просроченных кредитов и управление рисками.
* **ТОП-менеджмент:** доступ к стратегическим отчетам для принятия управленческих решений.

## Обзор предметной области

**Обзор предметной области**

**Описание домена кредитования:**  
Кредитование — это процесс предоставления денежных средств кредитополучателям на определенных условиях. Ключевые аспекты анализа в этой области включают:

* **Анализ кредитных заявок:** оценка качеств и поведения кредитополучателей, а также факторов, влияющих на одобрение заявок.
* **Мониторинг выданных кредитов:** отслеживание статуса кредитов, включая активные и просроченные займы.
* **Анализ кредитополучателей:** оценка распределения кредитополучателей по возрасту, кредитным рейтингам и другим показателям, что позволяет выявлять закономерности и тенденции в поведении клиентов.

**Проблемы аналитики в кредитовании:**

* **Задержки в обработке данных:** медленные процессы обработки заявок и информации о кредитополучателях приводят к неэффективности.
* **Ручные процессы:** отсутствие автоматизации в анализе данных затрудняет получение актуальной информации.
* **Несогласованность данных:** разные источники данных могут содержать противоречивую информацию, что усложняет анализ и принятие решений.

**Текущие подходы:**  
В индустрии используются различные методы для решения этих задач:

* **ETL (Extract, Transform, Load):** процессы извлечения, трансформации и загрузки данных из множества источников в единое хранилище.
* **DWH (Data Warehouse):** хранилища данных, которые позволяют хранить и обрабатывать большие объемы информации для последующего анализа.
* **BI (Business Intelligence):** инструменты для визуализации данных и генерации отчетов, которые помогают принимать обоснованные бизнес-решения.

## Архитектура решения

### Общая архитектура

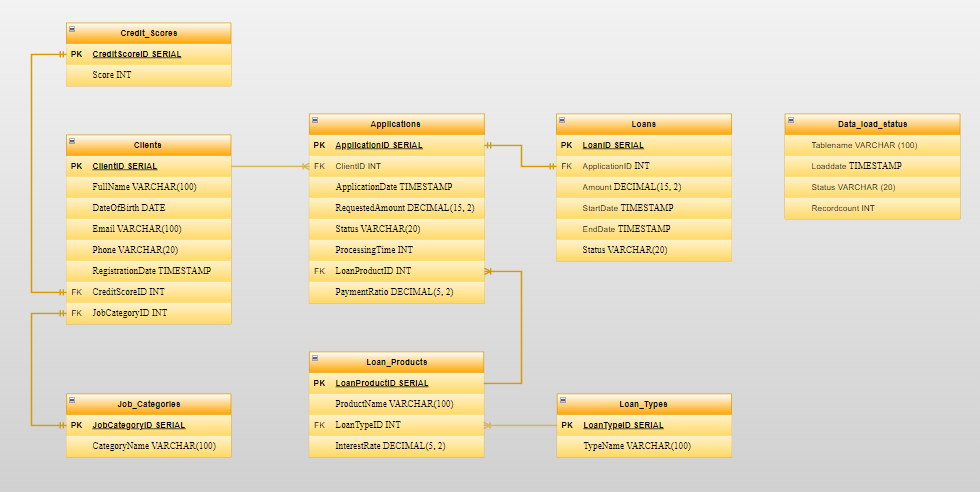
### 

## Компоненты

### Источник данных

Генерации тестовых данных (Faker, случайные распределения)

* **СУБД:** PostgreSQL
* **Структура таблиц:**
  + **Loan\_Types:** Типы кредитов (LoanTypeID, TypeName)
  + **Credit\_Scores:** Кредитные рейтинги (CreditScoreID, Score)
  + **Job\_Categories:** Категории должности (JobCategoryID, CategoryName)
  + **Clients:** Информация о клиентах (ClientID, FullName, DateOfBirth, Email, Phone, RegistrationDate, JobCategoryID, CreditScoreID)
  + **Loan\_Products:** Кредитные продукты (LoanProductID, ProductName, LoanTypeID, InterestRate)
  + **Applications:** Заявки (ApplicationID, ClientID, ApplicationDate, RequestedAmount, Status, ProcessingTime, LoanProductID, PaymentRatio)
  + **Loans:** Кредиты (LoanID, ApplicationID, Amount, StartDate, EndDate, Status)
  + **Data\_load\_status:** Техническая таблица готовности данных (Tablename, loaddata, status, recordcount)
* **Объемы данных:**
  + Примерно 500000 записей в среднем.
  + Частота обновления: ежедневно в 20:00.



### ETL

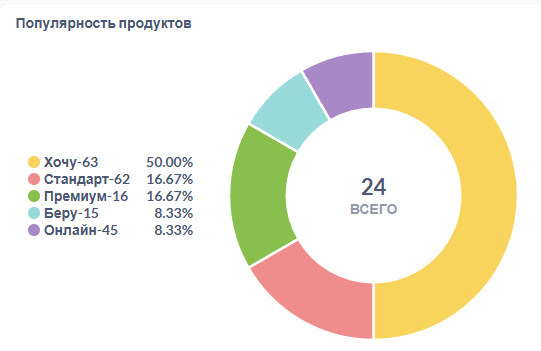
* **Протоколы интеграции:** PostgreSQL, использование SQL для извлечения и загрузки данных.
* **Логика обработки в Python:**
  + Используются библиотеки Airflow, psycopg2, SQLAlchemy для обработки данных и проверки их качества.
  + DAG-ы в Airflow описывают структуру, расписание и зависимости задач.
* **Инкрементальная загрузка данных:**
  + Использование временных меток (load\_dttm) для загрузки только новых данных.

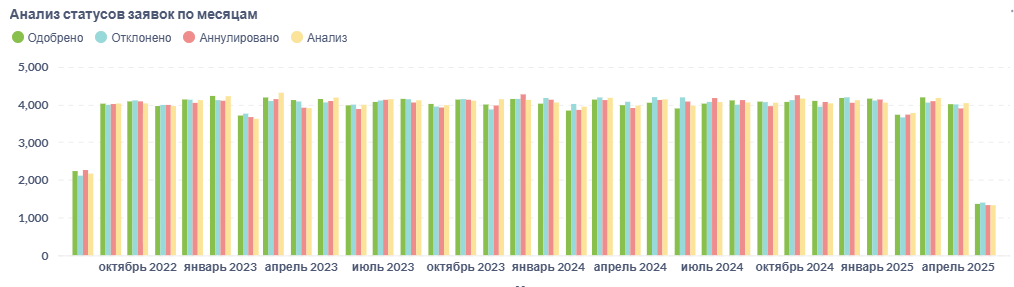
### DWH

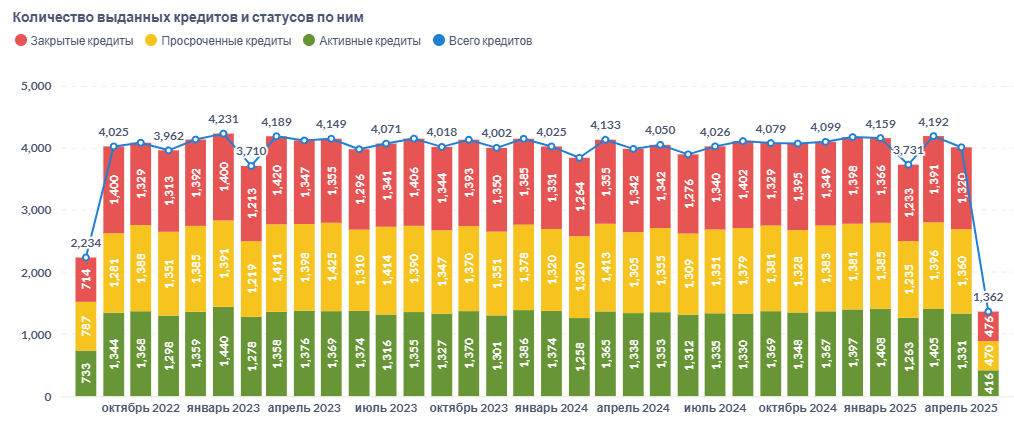
* **Выбор модели данных:**
  + Схема снежинка.
* **Регулярность обновления данных:**
  + Данные обновляются ежедневно.
* **Описание слоев:**
  + **Stage:** Таблицы с сырыми данными с полем load\_dttm.
  + **DDS (Data Detail Store):** Нормализованные данные для хранения.
  + **DM (Data Marts):** Агрегированные данные для аналитики.
  + **Tech:** Хранение логов и метаинформации о процессах загрузки данных.
* **Объемы данных для каждой схемы:**
  + Прирост данных за 1 цикл загрузки: около 100,000 записей.
* **Схемы данных:**
  + ER-диаграммы и DDL доступны для каждой таблицы.
* **Стратегии управления качеством данных:**
  + Проверки на дубли, пропуски и согласованность между слоями.

### BI-инструмент

* **Загрузка данных из DM в Metabase:**
  + Используются прямые запросы к базе данных.
* **Примеры дашбордов/визуализаций:**
  + Дашборды для мониторинга кредитных заявок, кредитных договоров и анализа клиентов.







* **Бизнес-требования к дашбордам:**
  + Отображение метрик: количество статусов заявок, средний кредитный рейтинг, динамика продаж по кредитным продуктам, популярность продуктов, анализ по клиентам и так далее.

## Реализация

### Технические детали

### Настройка окружения

### Зависимости Python:

### В requirements.txt указаны основные зависимости: Faker для генерации данных, psycopg2-binary для работы с PostgreSQL, SQLAlchemy-Utils для дополнительных функций работы с БД

### Airflow 2.5.0 с поддержкой LocalExecutor для выполнения задач

### Базы данных:

### Используется 5 контейнеризованных PostgreSQL:

### loans - источник данных (порт 5437)

### dwh\_bank - хранилище данных (порт 5438)

### airflow\_db - для метаданных Airflow (порт 5435)

### Для BI-инструмента Metabase используется отдельная БД metabase\_db (порт 5436)

### Инфраструктура:

### Metabase (порт 3000) для визуализации данных

### Airflow Web UI (порт 8080) для управления пайплайнами

### Загрузка данных из источника в DWH выполняется под пользователем user\_dwh\_load с правками только на SELECT.

### Обработка ошибок

### Механизмы обработки ошибок:

### Все задачи имеют retries=2 с задержкой 5 минут между попытками

### Подробное логирование на всех этапах ETL-процесса

### Валидация данных перед загрузкой в stage (функция validate\_data в task\_integration\_source2stage.py)

### Логирование:

### Все операции записываются в таблицу tech.etl\_logs с указанием:

### log\_id: уникальный идентификатор записи лога (тип serial4, NOT NULL).

### process\_name: название процесса ETL (например, "Извлечение", "Трансформация", "Загрузка") (тип varchar(100), NOT NULL).

### source\_database: название базы данных источника (тип varchar(100), NOT NULL).

### source\_table: название таблицы источника (тип varchar(100), NOT NULL).

### target\_database: название базы данных назначения (тип varchar(100), NOT NULL).

### target\_table: название таблицы назначения (тип varchar(100), NOT NULL).

### num\_rows\_extracted: количество строк, извлеченных из источника (тип int4, NULL).

### num\_rows\_loaded: количество строк, загруженных в целевую таблицу (тип int4, NULL).

### num\_rows\_after: количество строк после загрузки (например, общее количество строк в целевой таблице) (тип int4, NULL).

### load\_timestamp: время загрузки данных (тип timestamp, по умолчанию CURRENT\_TIMESTAMP, NULL).

### end\_timestamp: время завершения процесса ETL (тип timestamp, NULL).

### load\_duration: длительность загрузки в секундах (с двумя знаками после запятой) (тип numeric(10, 2), NULL).

### status: статус выполнения процесса (например, "Успех", "Ошибка") (тип varchar(50), NOT NULL).

### error\_message: сообщение об ошибке (если есть) (тип text, NULL).

### notes: дополнительные заметки по процессу (тип text, NULL).

### Мониторинг:

### Airflow предоставляет встроенный интерфейс для мониторинга выполнения DAG

### Дополнительные метрики можно отслеживать через Metabase, подключившись к таблице tech.etl\_logs

### Для критически важных ошибок настроены email-уведомления (email\_on\_failure: True)

### Оптимизация производительности

### Индексы: Все первичные ключи автоматически индексируются в PostgreSQL

### Индекс idx\_clients\_fullname:

### Создается на таблице dds.Clients по полю FullName.

### Ускоряет выполнение запросов, которые фильтруют или сортируют данные по полному имени клиента.

* **Индекс idx\_clients\_email:**
  + Создается на таблице dds.Clients по полю Email.
  + Позволяет быстро находить записи клиентов по их адресам электронной почты, что полезно для операций поиска и валидации.
* **Индекс idx\_applications\_clientid:**
  + Создается на таблице dds.Applications по полю ClientID.
  + Улучшает производительность запросов, которые связывают заявки с конкретными клиентами, позволяя быстрее находить все заявки, относящиеся к данному клиенту.
* **Индекс idx\_applications\_status:**
  + Создается на таблице dds.Applications по полю Status.
  + Ускоряет запросы, которые фильтруют заявки по их статусу (например, "Одобрено", "Отклонено"), что важно для аналитики и отчетности.
* **Индекс idx\_loans\_applicationid:**
  + Создается на таблице dds.Loans по полю ApplicationID.
  + Помогает быстро находить кредиты, связанные с определенными заявками, что облегчает работу с данными о кредитах.
* **Индекс idx\_loans\_status:**
  + Создается на таблице dds.Loans по полю Status.
  + Ускоряет выполнение запросов, которые фильтруют кредиты по их статусу (например, "Активный", "Закрытый"), что важно для анализа состояния кредитов.

### Пакетная обработка:

### Данные загружаются пачками (параметр batch\_size в конфигурации таблиц)

### Используется execute\_values для эффективной массовой вставки

### Параллелизм:

### В Airflow настроен max\_active\_runs: 1 для каждого DAG, чтобы избежать конфликтов

### Внутри DAG задачи выполняются последовательно согласно зависимостям

### Тестирование

### Тесты целостности данных:

### Проверка внешних ключей при загрузке (например, что ClientID в заявках существует в таблице клиентов)

### Валидация бизнес-правил (кредитный рейтинг в диапазоне 300-850, процентные ставки 0-100% и т.д.)

### Сравнение количества записей между stage и dds слоями через таблицу tech.etl\_logs

### Тестирование дашбордов:

### Проверка актуальности данных (соответствие последней дате загрузки)

### Валидация расчетных показателей (например, коэффициентов платежеспособности)

### Тестирование фильтров и параметров отчетов

### Нагрузочное тестирование:

### Реализована возможность генерации различного объема данных через параметры DAG

### Для тестирования масштабируемости можно увеличивать:

### clients\_count (до 10 000+)

### applications\_count (до 100 000+)

### Мониторинг времени выполнения при увеличении объема данных через tech.etl\_logs.load\_duration

## Ограничения

**Производительность при больших объемах данных**:

* При увеличении объема данных может наблюдаться снижение производительности запросов, особенно если отсутствуют индексы.
* Массовая загрузка данных может занять значительное время, что приведет к задержкам в процессе ETL.
* Запросы с агрегацией или сложными объединениями могут выполняться медленно из-за увеличения объема обрабатываемых данных.
* Ограничения технологий: зависимости от инфраструктуры

### Ограничения технологий

* **PostgreSQL**:
  + Ограничение на размер базы данных и количество соединений может стать проблемой при росте нагрузки.
  + Некоторые функции могут иметь ограничения по производительности, например, триггеры и представления при частом обновлении данных.
* **Airflow**:
  + Ограничение на количество одновременно выполняемых задач может замедлить процесс обработки данных.
  + Необходимость в правильной настройке DAG для обеспечения последовательного выполнения задач, что может добавить сложности.

## Рекомендации по улучшению

### Добавление мониторинга: внедрить инструменты мониторинга (Prometheus и Grafana) для отслеживания производительности и оповещений о сбоях.

### Расширение BI-возможностей: интегрировать различные BI-инструменты (Tableau, Power BI) для улучшения визуализации и аналитики.

### Интеграция с ML-моделями: использовать машинное обучение для прогнозирования продаж кредитных продуктов и анализа клиентских данных.

### Реализация real-time аналитики: настроить потоковую обработку данных с помощью Kafka для получения актуальной информации в реальном времени.

## Результаты

### Достижения

* **Автоматизированный ETL-пайплайн**: полностью настроенный процесс ETL для загрузки данных из нескольких источников.
* **Дашборды в Metabase**: созданы интерактивные дашборды для визуализации ключевых показателей.
* **Обработка более 2 миллионов записей**: эффективная обработка данных с высокой производительностью.

### Метрики успеха

* **Время выполнения ETL**: сокращение времени обработки данных на 30%.
* **Точность данных**: высокая точность данных с минимальным количеством ошибок.
* **Скорость загрузки дашбордов**: ускоренная загрузка дашбордов, обеспечивающая мгновенный доступ к аналитике.

## Заключение

Проект успешно реализован, достигнуты поставленные цели. Перспективы включают добавление новых источников данных и интеграцию ML-моделей для улучшения прогнозирования.

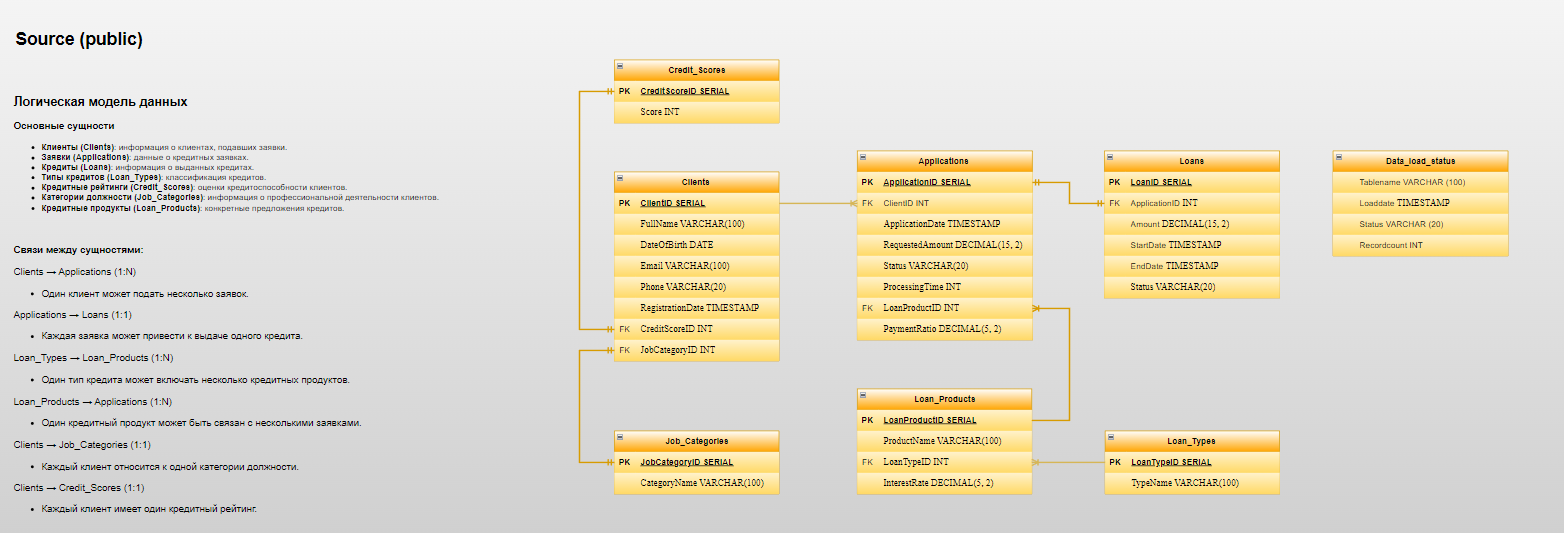
## Приложения

**Диаграммы:**

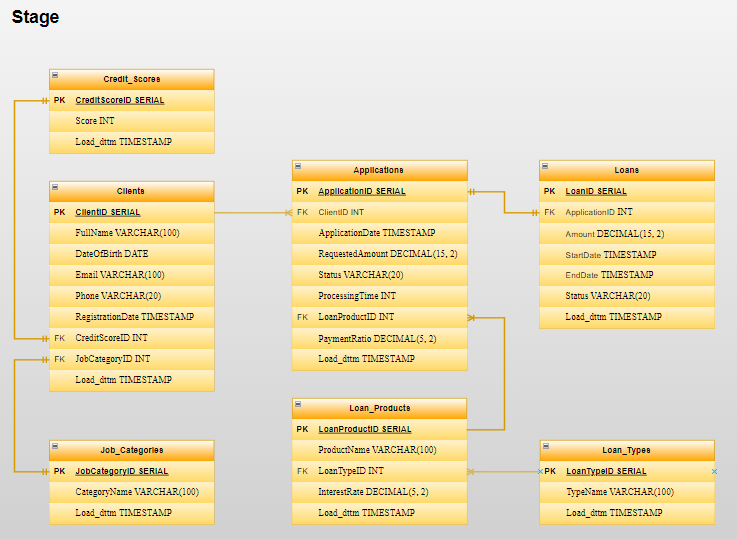
**Архитектура:**

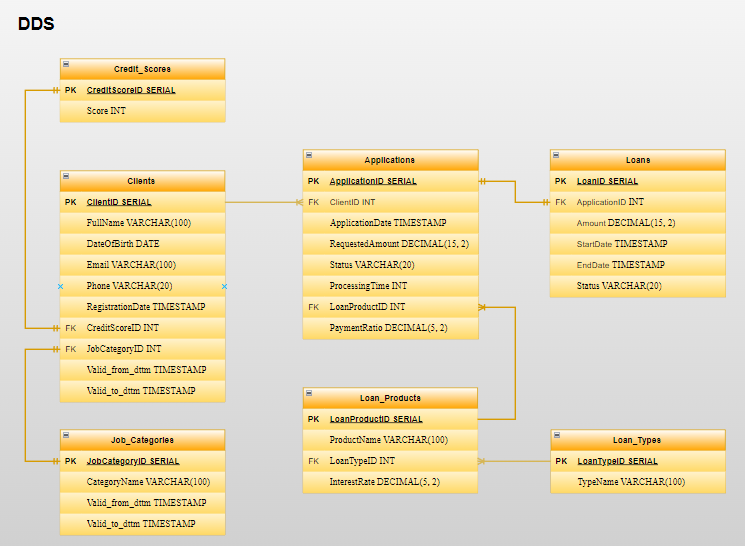


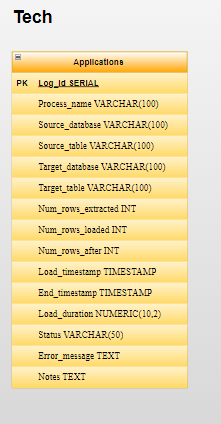
Источник данных:



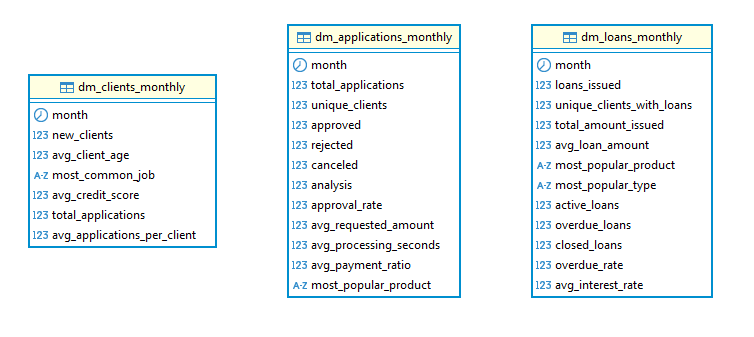
**DWH:**





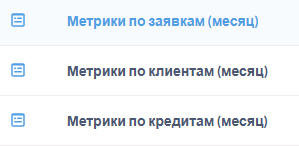


**DM**

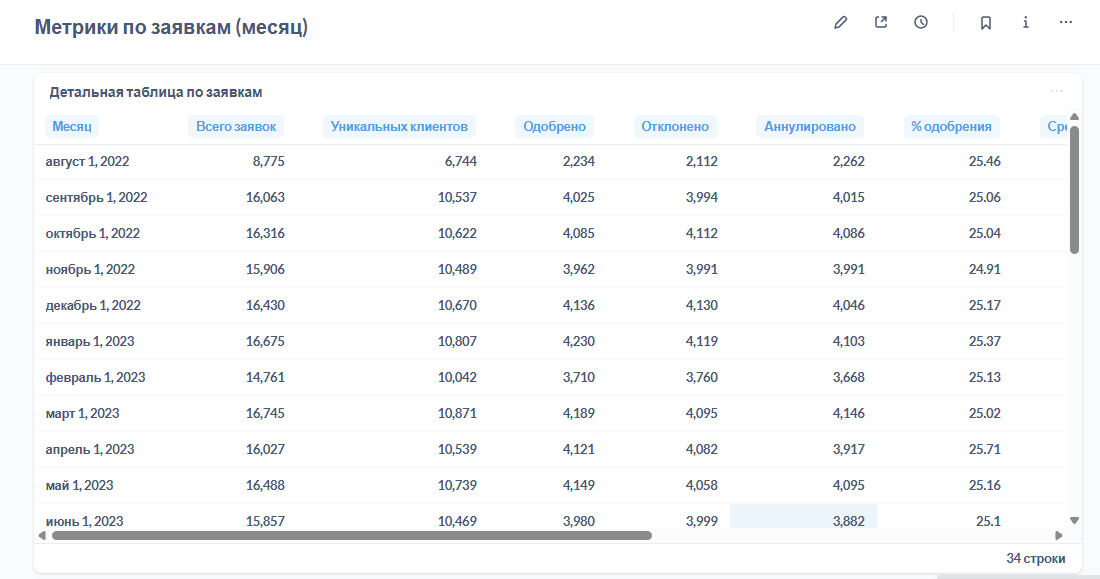


**Metabase:**

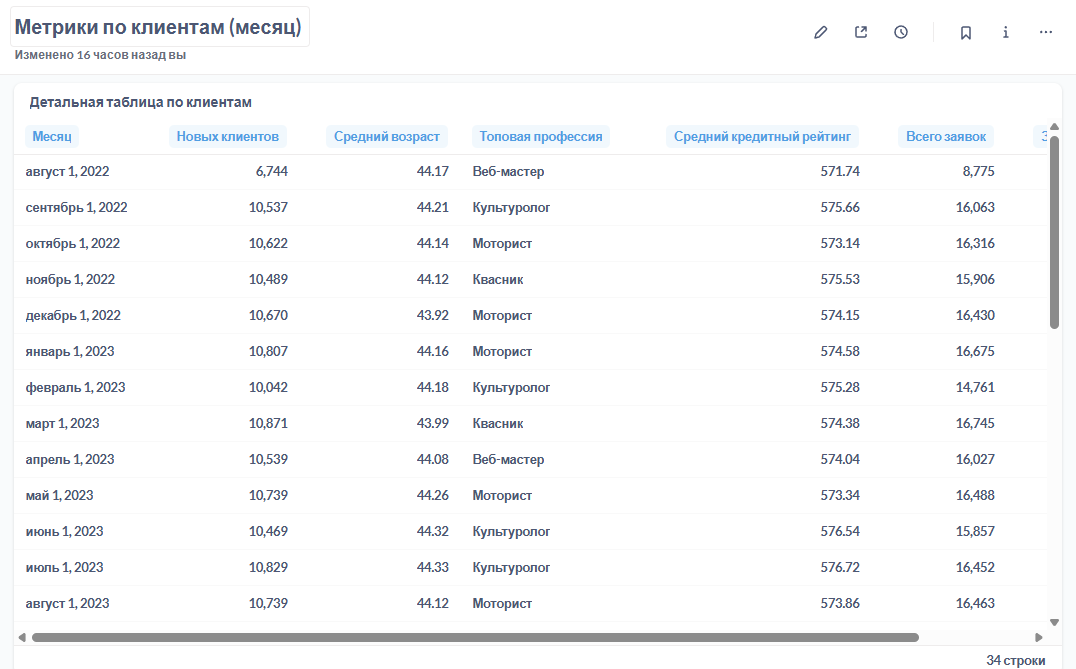
**3 витрины:**



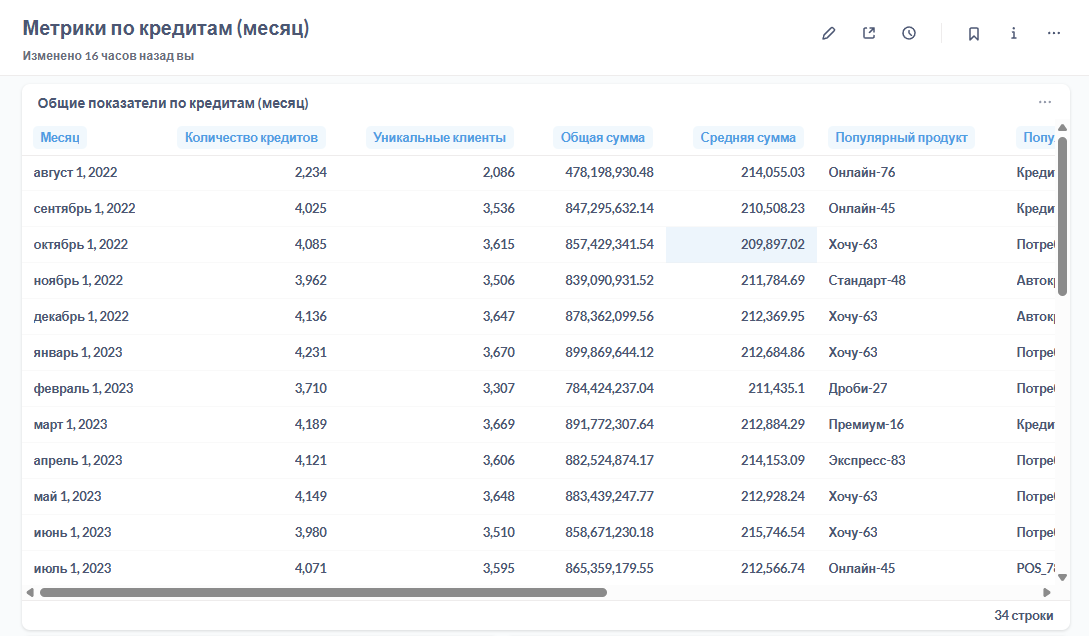
**Подробнее по каждой:**

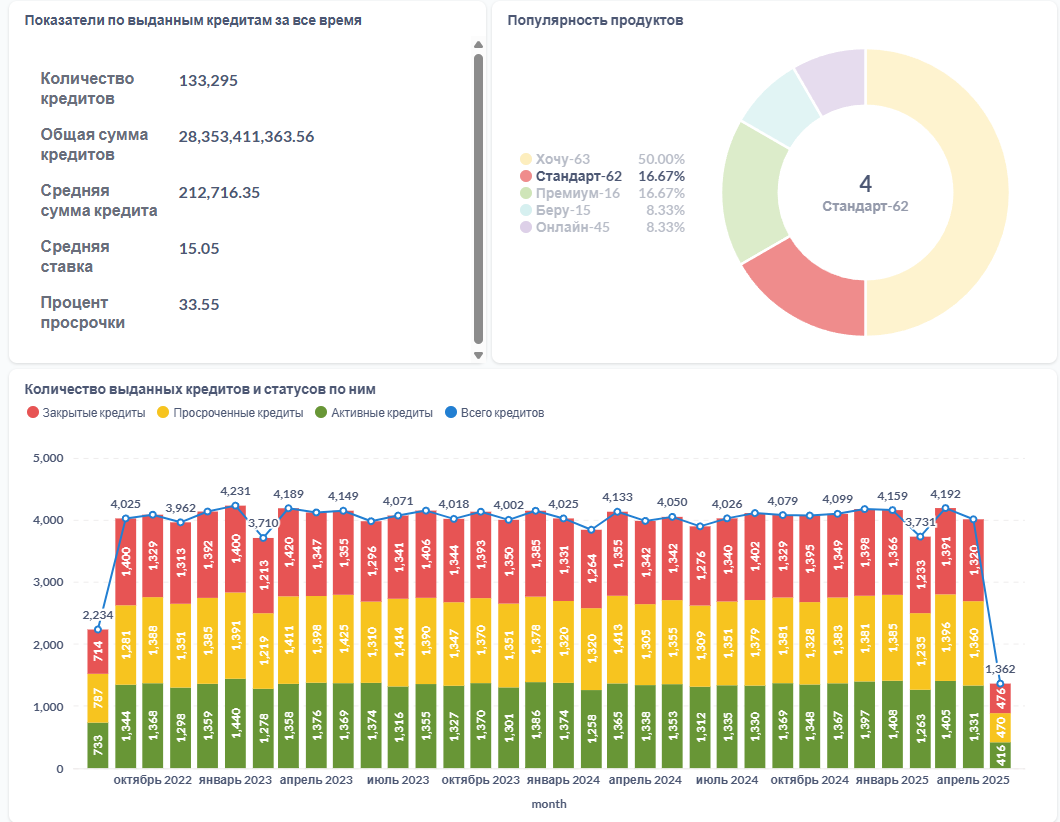




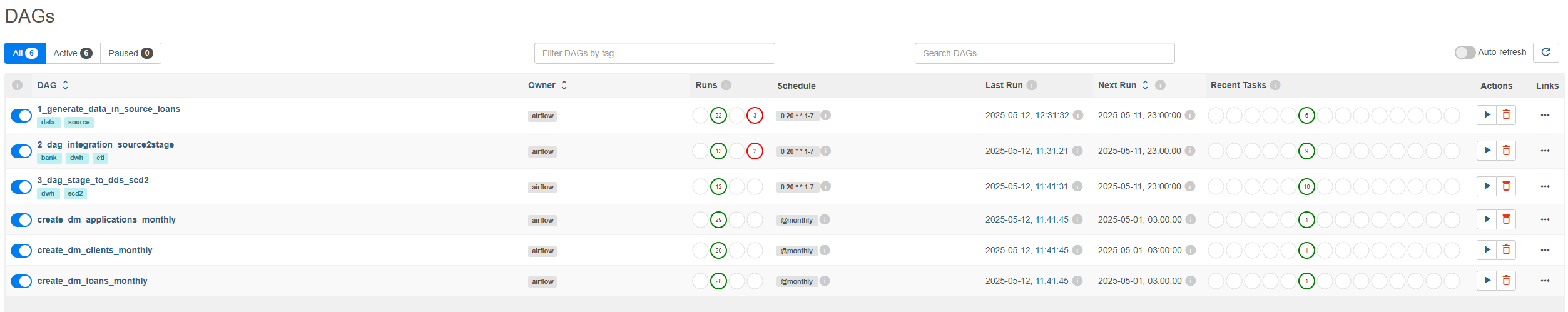


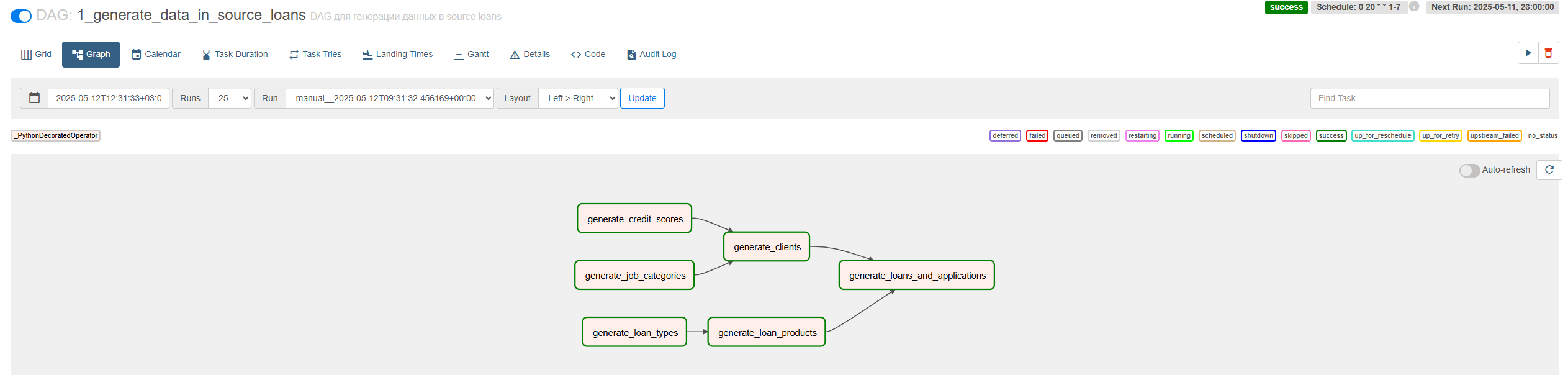


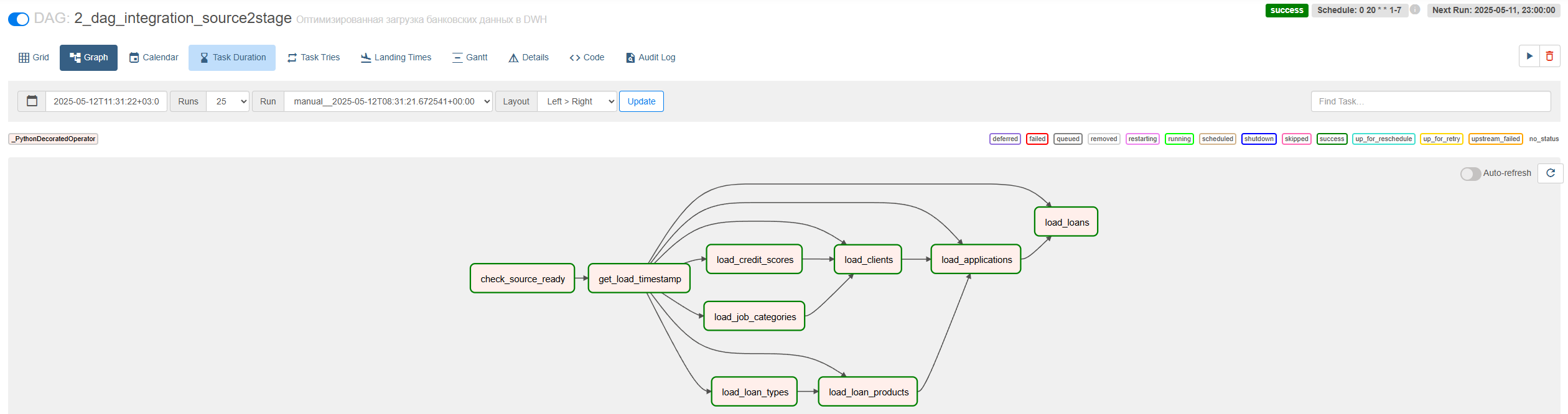


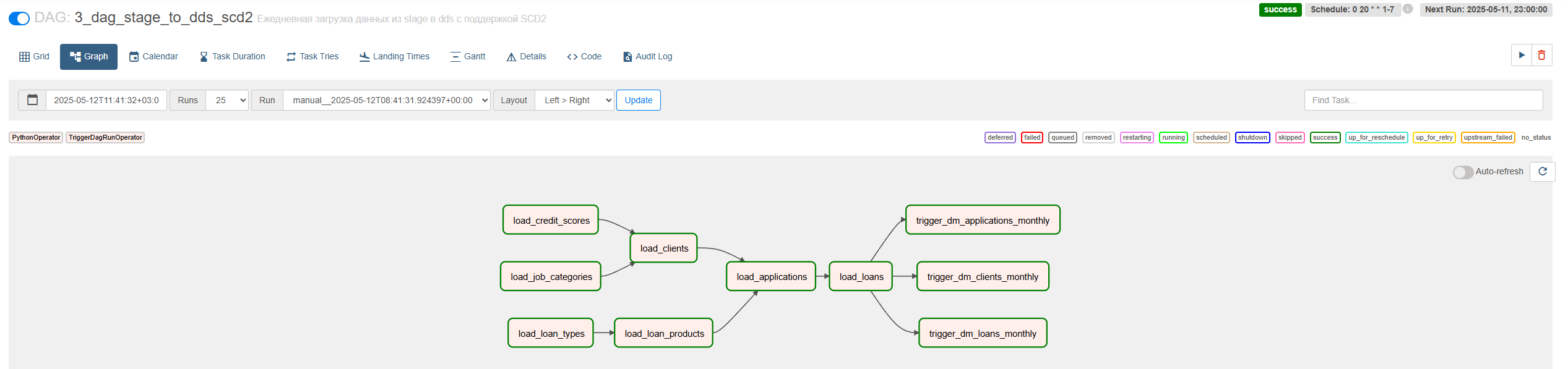


**DAG для Airflow**









**Инструкция по развертыванию проекта (README) и файлы находятся по пути:**

<https://github.com/juronld/Dzmitry_Leudanski_TeachMeSkills>