

Выпускной проект

Оглавление

- **Введение**
 - Описание задачи
 - Вводные к проекту
 - Установка инфраструктуры
 - Входные данные
- **Источники данных**
 - Вариант 1. Бакет в S3
 - Вариант 2. Хранилище в PostgreSQL
 - Вариант 3. Стим в Kafka
- **Инструкция по выполнению проекта**
 - Шаг 1. Изучение входных данных
 - Шаг 2. Создание таблиц для хранилища
 - Шаг 3. Пайплайн загрузки данных из систем-источников в staging-слой
 - Шаг 4. Пайплайн обновления витрины данных
 - Шаг 5. Создание дашборда
 - Что должно быть по итогу проекта

Выполнение

- **1. Изучение входных данных**
- **2. Создание таблиц для хранилища**
 - 2.1 Создание STG
 - 2.2 Создание CDM
- **3. Пайплайн Airflow**
 - 3.1 SQL скрипты наполнения витрины
 - 3.2 DAG
 - 3.3 Запуск пайплайна
 - 3.4 Проверка результата работы
- **4. Создание дашборда**
 - 4.1 Требования
 - 4.2 Подключение
 - 4.3 Вспомогательный датасет
 - 4.4 Визуализация

Введение ▲

В этом проекте вы поработаете с данными финтех-стартапа, который предлагает международные банковские услуги через приложение: пользователи могут безопасно переводить деньги в разные страны.

Компания придерживается децентрализованной финансовой системы: в каждой стране, где доступно приложение, есть отдельный сервис, работающий с валютой этой страны. При этом компания ведёт учёт транзакционной активности клиентов внутри и между странами: разработан единый протокол передачи данных, который обеспечивает одинаковую структуру таблиц во всех странах.

Финансовая активность пользователей увеличивается, поэтому пришло время провести единый анализ данных. Вам надо объединить информацию из разных источников и подготовить её для аналитики.

Описание задачи ▲

Команда аналитиков попросила вас собрать данные по транзакционной активности пользователей и настроить обновление таблицы с курсом валют.

Цель — понять, как выглядит динамика оборота всей компании и что приводит к его изменениям.

Данные подготовлены в различных вариантах, так как в компании использовались разные протоколы передачи информации. Поэтому вы можете самостоятельно выбрать источник и решить, каким образом реализовать поставку данных в хранилище.

Варианты размещения данных:

- в S3
- в PostgreSQL
- через Spark Streaming в Kafka



Какую инфраструктуру использовать — выбор за вами. Вы можете работать с той технологией, которая вам больше всего понравилась за время учёбы.

Инструкции к проекту менее детализированные: вам нужно самим решить, как сделать каждый шаг.

Вводные к проекту ▲

Вы должны реализовать пайплайн обработки данных из нескольких источников и полноценное хранилище для финтех-стартапа. Здесь потребуются знания из всех пройденных спринтов.

Вы сами можете выбрать вариант решения. Есть несколько способов реализации задачи:

- Выгрузка данных из Kafka с помощью модулей Apache Hadoop в Data Lake : Apache Spark для сбора данных, HDFS — для сохранения. Последующая поставка данных из Data Lake в DWH.
- Выгрузка данных из PostgreSQL или S3 в DWH с помощью DAG в Airflow : обработка информации в рамках ETL - процесса.

Хранилище в компании должно быть реализовано на Vertica . После построения пайплайна обработки данных необходимо отдельно реализовать пайплайн формирования витрины с помощью Airflow и DAG . Необходимо также реализовать BI - аналитику для компании: подключиться из Metabase к Vertica и реализовать дашборд.

Установка инфраструктуры ▲

Инструменты для проекта вы можете развернуть двумя способами:

1. Локальная инфраструктура: Docker

Выполнив следующую команду, вы скопируете и запустите актуальный Docker -образ:

```
docker run -d -p 8998:8998 -p 8280:8280 -p 15432:5432 --name=de-final-prj-local  
cr.yandex/crp1r8pht0n0g125aug1/de-final-prj:latest  
docker run -d -v sprint-10-final-project:/lessons -p 8998:8998 -p 8280:8280 -p 15432:5432 --name=de-final-prj-local cr.yandex/crp1r8pht0n0g125aug1/de-final-prj:latest
```

Информация по размещению инструментов есть внутри образа, в частности:

- Адрес `Metabase` — `8998`
 - Когда перейдёте по адресу <http://localhost:8998/>, вы попадёте на страницу регистрации. Нужно будет указать имя пользователя и пароль. (`metabase_pass_01`)
- Адрес `Airflow` — `8280`
 - Когда перейдёте по адресу <http://localhost:8280/airflow/>, вы попадёте в меню авторизации:
 - пользователь — `AirflowAdmin`
 - пароль — `airflow_pass`
- Данные по подключению к `PostgreSQL` :
 - хост: `localhost`
 - dbname: `postgres`
 - порт: `15432`
 - пользователь — `jovyan`
 - пароль — `jovyan`

Чтобы запустить ваши `DAG`, подключаться к `Docker` можно через `VSCode`

2. Облачные технологии: `Telegram`-бот

Установите **виртуальную машину** с помощью **Telegram-бота**. ВМ будет генерировать сообщения в `Kafka`, чтобы создать подключение по `SSH`. IP -адрес для ВМ бот выдаст.

При запуске инфраструктуры в веб-интерфейсах, требующих авторизации, вам будет предоставлен логин и пароль. Если вы удаляете инфраструктуру, то всё содержимое тоже удаляется: не забудьте сохранить свои наработки локально.

Входные данные ▲

Данные `transactions`

Содержат в себе информацию о движении денежных средств между клиентами в разных валютах.

Структура данных:

- `operation_id` — `id` транзакции;
- `account_number_from` — внутренний бухгалтерский номер счёта транзакции ОТ КОГО;
- `account_number_to` — внутренний бухгалтерский номер счёта транзакции К КОМУ;
- `currency_code` — трёхзначный код валюты страны, из которой идёт транзакция;
- `country` — страна-источник транзакции;
- `status` — статус проведения транзакции:
 - `queued` («транзакция в очереди на обработку сервисом»),
 - `in_progress` («транзакция в обработке»),
 - `blocked` («транзакция заблокирована сервисом»),
 - `done` («транзакция выполнена успешно»), - `chargeback` («пользователь осуществил возврат по транзакции»).
- `transaction_type` — тип транзакции во внутреннем учёте:
 - `authorisation` («авторизационная транзакция, подтверждающая наличие счёта пользователя»),
 - `sbp_incoming` («входящий перевод по системе быстрых платежей»),
 - `sbp_outgoing` («исходящий перевод по системе быстрых платежей»),
 - `transfer_incoming` («входящий перевод по счёту»),
 - `transfer_outgoing` («исходящий перевод по счёту»),
 - `c2b_partner_incoming` («перевод от юридического лица»),
 - `c2b_partner_outgoing` («перевод юридическому лицу»).

- **amount** — целочисленная сумма транзакции в минимальной единице валюты страны (копейка, цент, куруш);
- **transaction_dt** — дата и время исполнения транзакции до миллисекунд.

Пример данных в таблице:

operation_id	account_number_from	account_number_to	currency_code	country	status	transaction_type	amount	transaction_dt
16f7f0f5-c868-4b8b-b088-d2eb93bf3096	914810	903810	430	russia	done	sbp_incoming	10000	2022-10-01 00:01:21.000
54158aa6-3062-4c0f-b7e5-9d9601f6dd66	903810	9864947	450	china	queued	sbp_incoming	15300	2022-10-01 00:01:21.000
807c51c4-ad6d-4870-aff8-5d191456434b	903810	4398103	420	usa	in_progress	sbp_incoming	170000	2022-10-01 00:01:22.000
46da88fc-fb7d-4009-afa0-5b66aba19616	903810	5282946	430	russia	queued	c2a_incoming	250000	2022-10-01 00:01:23.000
8d9b4f6c-215d-4098-ae98-2358b33707cb	914810	854428	430	russia	blocked	sbp_incoming	78200	2022-10-01 00:01:23.000

Фрагмент данных для быстрого изучения

Данные currencies

Это справочник, который содержит в себе информацию об обновлениях курсов валют и взаимоотношениях валютных пар друг с другом.

Структура данных:

- **date_update** — дата обновления курса валют;
- **currency_code** — трёхзначный код валюты транзакции;
- **currency_code_with** — отношение другой валюты к валюте трёхзначного кода;
- **currency_code_div** — значение отношения единицы одной валюты к единице валюты транзакции.

Пример данных в таблице:

	date_update	currency_code	currency_code_with	currency_with_div
0	2022-10-01 00:00:00.000	430	470	0.92
1	2022-10-01 00:00:00.000	430	460	0.95
2	2022-10-01 00:00:00.000	430	450	1.08
3	2022-10-01 00:00:00.000	430	410	1.02
4	2022-10-01 00:00:00.000	430	420	1.05

Фрагмент данных для быстрого изучения

Источники данных ▲

Как источник вы можете использовать любой из вариантов — **S3**, **PostgreSQL** или **Kafka**. Данные во всех источниках лежат одни и те же, просто в разных форматах.

Вариант 1. Бакет в S3 ▲

Данные подготовлены и доступны для чтения в специальном бакете в **S3** : <https://storage.yandexcloud.net/final-project/> . Данные выгружены батчами: необходимо обрабатывать каждый файл итеративно при запуске **ETL**-процесса.

- История изменения курсов валют лежит в **currencies_history.csv**
- История транзакций — в батчах **transactions_batch_*.csv**

Данные по транзакциям `transactions_batch_*.csv` отсортированы по времени с `01.10.2022` и далее, при этом разбиты по возрастающим батчам на 10 частей. Вы можете обрабатывать файлы, перебирая как по датам, так и по порядковому номеру выгрузок.

Подключение к данным в `S3` осуществляется через библиотеку `boto3`. Вы делали аналогичное подключение в курсе «[ETL: автоматизация подготовки данных](#)». Чтобы подключиться, вам необходимо использовать данные:

```
aws_access_key_id = FakePass  
aws_secret_access_key = FakePass
```

Пример файлов в бакете:

Имя	Размер	Класс хранилища	Последнее изменение	...
currencies_history.csv	159 Б	Стандартное	06.12.2022, в 07:43	...
transactions_batch_1.csv	14.75 МБ	Стандартное	06.12.2022, в 07:41	...
transactions_batch_10.csv	14.75 МБ	Стандартное	06.12.2022, в 07:41	...
transactions_batch_2.csv	14.79 МБ	Стандартное	06.12.2022, в 07:41	...
transactions_batch_3.csv	14.77 МБ	Стандартное	06.12.2022, в 07:41	...
transactions_batch_4.csv	5.28 МБ	Стандартное	06.12.2022, в 07:41	...
transactions_batch_5.csv	1.02 МБ	Стандартное	06.12.2022, в 07:41	...
transactions_batch_6.csv	499.8 КБ	Стандартное	06.12.2022, в 07:41	...
transactions_batch_7.csv	14.73 МБ	Стандартное	06.12.2022, в 07:41	...
transactions_batch_8.csv	14.75 МБ	Стандартное	06.12.2022, в 07:41	...
transactions_batch_9.csv	14.75 МБ	Стандартное	06.12.2022, в 07:41	...

Вариант 2. Хранилище в PostgreSQL ▲

В компании запущена `PostgreSQL` для продакшена. Для построения инфраструктуры аналитики и поставки данных предоставлена отдельная `PostgreSQL`, копия продовой БД. В таблицах `public.transactions` и `public.currencies` есть доступ на загрузку данных.

Для доступа к БД создана учётная запись:

Параметр	Значение
database	db1
host	rc1b-w5d285tmx8jimyn.yandexcloud.net
port	6432
username	student
password	FakePass

Вариант 3. Стим в Kafka ▲

Вы можете запустить поставку данных в режиме реального времени в два топика `Kafka`. Инициируйте запуск топиков через `curl`-вызов:

```
curl -X POST https://order-gen-service.sprint9.tgcloudenv.ru/project/register_kafka \
-H 'Content-Type: application/json; charset=utf-8' \
--data-binary @- << EOF
{
    "student": " ваш_логин",
    "kafka_connect": {
        "host": " ваш_хост_kafka",
        "port": 9091,
        "topic": "transaction-service-input",
        "producer_name": "producer_consumer",
        "producer_password": "пароль_от_продюсера_к_вашей_kafka"
    }
}
EOF
```

Данные будут поступать в виде `JSON`-объектов в топик, который вы указали: `transaction-service-input` (название топика рекомендуемое, можно завести топик с любым названием).

Работать с `Kafka` можно двумя способами:

- Поднять `Kafka`, завести пользователя, выдать пользователю права на отправку и чтение сообщений, создать топик.
- Воспользоваться той `Kafka`, которую вы поднимали в девятом спринте, но создать новый топик.

К вам будут поступать сообщения двух типов:

- транзакции: `object_type = TRANSACTION`
- курсы валют: `object_type = CURRENCY`

Если вы ошиблись или вам надо завести отправку сообщений заново.

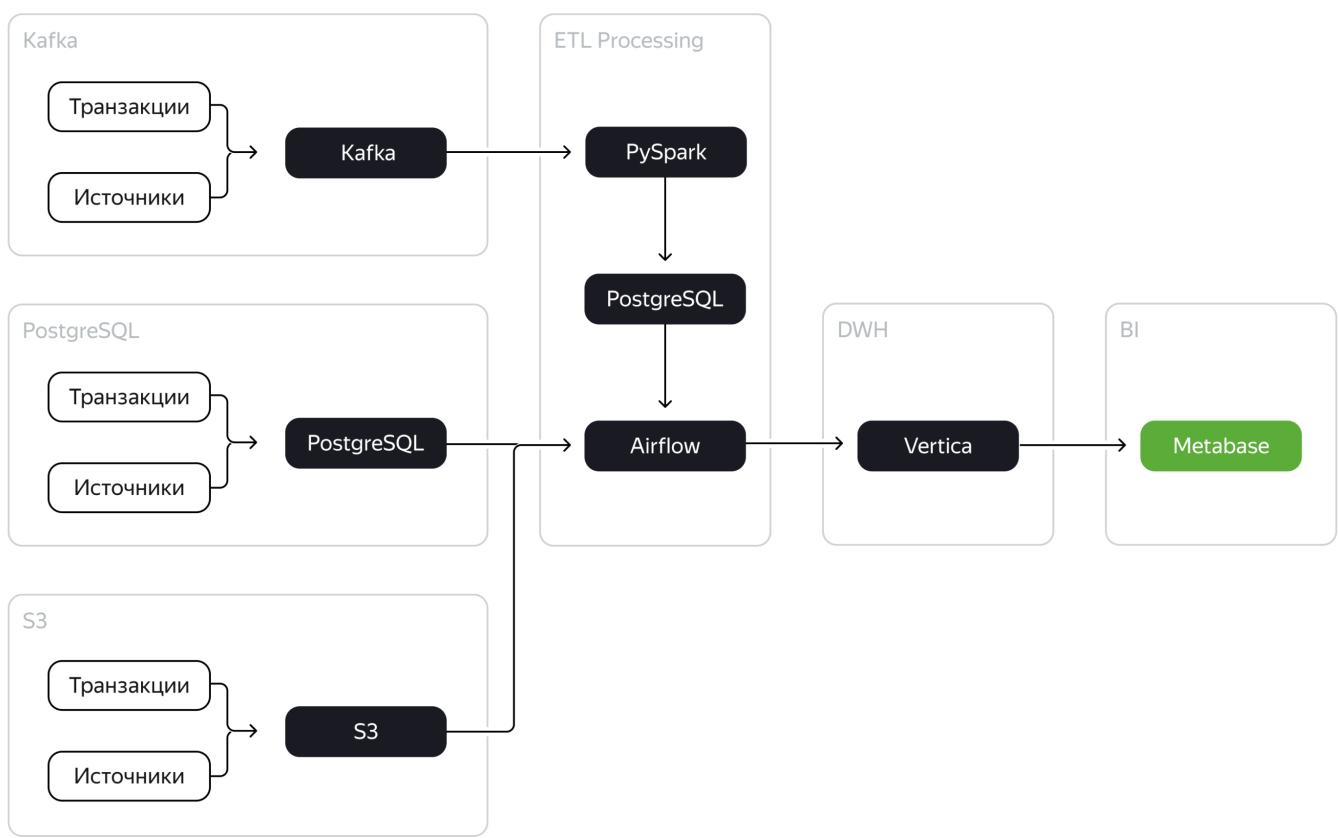
1. Удалите старые настройки:

```
curl -X POST https://order-gen-service.sprint9.tgcloudenv.ru/project/delete_kafka \
-H 'Content-Type: application/json; charset=utf-8' \
--data-binary @- << EOF
{
    "student": " ваш_логин"
}
EOF
```

2. Добавьте новые настройки, как показано выше.

Инструкция по выполнению проекта ▲

Архитектура итогового решения должна выглядеть так:



В схему заложена вариативность, о которой мы писали ранее. Она проявляется на двух этапах:

- источник данных: вы можете выбрать один вариант из трёх в качестве источника;
- пайплайн выгрузки и загрузки данных: вы можете реализовать `ETL`-процесс с помощью `Airflow` или `PySpark`

Выбор источников и способа реализации `ETL` зависит от того, какие навыки вам интереснее закрепить и какую реализацию проекта важнее положить в портфолио.

Шаг 1. Изучение входных данных ▲

- Выберите источник, из которого будете забирать данные.
- Проверьте подключение к выбранному источнику. Убедитесь, что данные доступны в указанных схемах с помощью `IDE`.
- Изучите структуру данных, а также проверьте, что данные полные, и при работе с информацией не выпадает ошибка.

Шаг 2. Создание таблиц для хранилища ▲

Подготовьте инфраструктуру для загрузки сырых данных в схемы `*__STAGING` и `*__DWH` в `Vertica` и в витрину — создайте нужные таблицы.

Принцип наименования схем здесь тот же, что и в шестом спринте: вместо звёздочки укажите вашу почту, которую используете на платформе Практикума. Используйте учётную запись, которую вы создали ранее в спринтах.

1. Подключитесь к `Vertica` со своей учётной записью.

Вы создали её ранее в спринтах.

- Host: `51.250.75.20` или `vertica.tgcloudenv.ru`
- Порт: `5433`
- Login: `stv2023070319`
- Pass: `FakePass`
- Схема: `dwh`

2. Создайте таблицы `transactions` и `currencies` для загрузки сырых данных согласно модели данных:

- а. Реализуйте соответствующую слою сырых данных модель данных полей — присвойте им нужные имена, типы данных.
- б. Создайте проекции по датам.

- с. Добавьте сортировку и сегментирование по хеш-функции от даты и идентификатора транзакции.

3. Создайте витрину `global_metrics` с агрегацией по дням.

Витрина должна помогать отвечать на такие вопросы:

- какая ежедневная динамика сумм переводов в разных валютах;
- какое среднее количество транзакций на пользователя;
- какое количество уникальных пользователей совершают транзакции в валютах;
- какой общий оборот компании в единой валюте.

Для этого включите в витрину такие поля:

- `date_update` — дата расчёта,
- `currency_from` — код валюты транзакции;
- `amount_total` — общая сумма транзакций по валюте в долларах;
- `cnt_transactions` — общий объём транзакций по валюте;
- `avg_transactions_per_account` — средний объём транзакций с аккаунта;
- `cnt_accounts_make_transactions` — количество уникальных аккаунтов с совершёнными транзакциями по валюте.

4. Сделайте так, чтобы витрина обновлялась:

Ежедневно инкрементом на основании данных из таблиц `transactions` и `currencies`. Каждый день должна добавляться новая запись за вчера.

Шаг 3. Пайплайн загрузки данных из систем-источников в `staging`-слой



Вам нужно создать пайплайн для ежедневной выгрузки данных из выбранного источника и загрузки в `staging`-слой хранилища в `Vertica`. Как мы указывали выше, создать пайплайн вы можете двумя способами: через `DAG` в `Airflow` (выгрузка из `PostgreSQL` или `S3`) или через `Spark`-задачу (выгрузка из `Kafka`)

Уточнение про загрузку данных в `Vertica` через `Spark`-задачу

В седьмом спринте вы загружали данные только в `PostgreSQL`, а с `Vertica` у вас такого опыта пока не было.

Здесь же именно в `Vertica` нужно загрузить данные. Поэтому у вас есть три способа решить эту задачу:

- 1. Самостоятельно разобраться с тем, как загружать данные в `Vertica` с помощью `Spark Job`. Это более сложный и рискованный путь, который мы не рекомендуем.
- 1. С помощью `Spark Job` загрузить данные в ваш `Postgres`, а затем оттуда через `DAG` в `Vertica`.
- 1. Выгружать и загружать всё через `DAG`.

1. Определитесь, с помощью каких инструментов вы создадите пайплайн.
2. Спроектируйте пайплайн на основе вводных и нужного состояния данных на выходе.
3. Авторизуйтесь в `Airflow` или `Spark`
4. Реализуйте автоматизированный пайплайн выгрузки данных из источника в `staging`-слой по дням.
5. Запустите пайплайн за выбранный период (октябрь 2022 года), задавая параметр даты: каждый день пайплайн должен отработать самостоятельно.
6. Проверьте, что ежедневные данные инкрементами загружены в схему `*__STAGING` хранилища.
7. Пайплайн должен сохранять результаты в `staging`-слой хранилища, результат один для любой реализации.

Шаг 4. Пайплайн обновления витрины данных



Теперь вам нужно настроить пайплайн переноса данных из слоя `staging` в созданную ранее витрину.

1. Создайте `DAG` в `Airflow`, который будет наполнять итоговую витрину в схеме `*__DWH` хранилища. В рамках пайплайна реализуйте:
 - а. очистку от тестовых аккаунтов (аккаунты < 0).
 - б. ежедневное инкрементальное добавление новой партицию данных по дате за вчерашний день.
 - в. расчёт агрегатов должен быть в соответствии с установленными требованиями по названию и содержанию полей итоговой витрины `global_metrics`

2. Запустите пайплайн за выбранный период (октябрь 2022 года), задавая параметр даты: каждый день пайплайн должен отработать самостоятельно.

Шаг 5. Создание дашборда ▲

Подключите Metabase к витрине и реализуйте интерактивный отчёт для команды аналитиков.

1. Подключите Metabase к схеме *__DWH в Vertica
2. Настройте интеграцию с данными в global_metrics
3. Соберите автобновляемый дашборд:

- а. Визуализируйте метрику суммы переводов с возможностью смотреть общую сумму и выбирать отдельные валюты через фильтр.
- б. Визуализируйте метрику среднего объема транзакций на пользователя.
- в. Визуализируйте метрику количества уникальных пользователей, которые совершают транзакции в валютах.
- г. Визуализируйте метрику общего оборота компании в единой валюте.
- д. Настройте фильтры по дате и выбранным валютам перевода.

С помощью каких графиков визуализировать метрики, как их разместить и как в целом оформить дашборд — вам нужно решить самостоятельно.

⚠ Если Metabase не обнаружит подключение к Vertica или не отображаются доступные БД в Vertica, перезапустите Docker -образ. Вы можете также воспользоваться [официальной инструкцией](#) подключения Metabase к Vertica.

Что должно быть по итогу проекта ▲

В вашем репозитории в GitHub должны быть:

1. DAG, который поставляет данные из источника (на выбор: PostgreSQL, S3, Kafka) в Vertica
2. DAG формирования витрины в Vertica для ответов на вопросы бизнеса.
3. Ссылка на схему и таблицу в Vertica, где расположена итоговая витрина.
4. Скриншоты из вашего дашборда в Metabase, где видны нужные для бизнеса метрики.

Выполнение

1. Изучение входных данных ▲

Содержимое таблиц currencies и transactions в Postgres

```
In [1]: import psycopg2
import pandas as pd

conn_postgre = {'host': 'rc1b-w5d285tmxa8jimyn.mdb.yandexcloud.net',
                'port': '6432',
                'user': 'student',
                'password': 'FakePass',
                'database': 'db1',
                'sslmode': 'require',
                'sslrootcert': 'c:/GitHub/data_engineer/10_final_project/cert/CA.pem'
}

# Подключение к базе данных
conn = psycopg2.connect(**conn_postgre)

cursor = conn.cursor()

# Выполнение запроса
query = """
    SELECT *
    FROM currencies
    LIMIT 10;
"""

cursor.execute(query)
result = cursor.fetchall()
print(result)
```

```

"""
cursor.execute(query)

# Запись результата в переменную df_currencies
df_currencies = pd.DataFrame(cursor.fetchall(), columns=[col[0] for col in cursor.description])

# Закрытие курсора
cursor.close()

cursor = conn.cursor()
query = """
    SELECT *
    FROM transactions
    LIMIT 10;
"""
cursor.execute(query)

df_transactions = pd.DataFrame(cursor.fetchall(), columns=[col[0] for col in cursor.description])

# Закрытие курсора и соединения с базой данных
cursor.close()
conn.close()

# Вывод содержимого
display(df_currencies.head(3))
display(df_transactions.head(3))

```

	date_update	currency_code	currency_code_with	currency_with_div						
0	2022-10-01	430	470	0.920						
1	2022-10-01	430	460	0.950						
2	2022-10-01	430	450	1.080						
	operation_id	account_number_from	account_number_to	currency_code	country	status	transaction_type	amount	transaction_dt	
0	7f9922e8-48b8-4097-9ff6-3eab131e80d9		914810	4454155	430	russia	queued	sbp_incoming	230900	2022-10-01 00:00:02
1	e8793d0b-a562-4bcf-8c04-155792f3ec93		914810	5371058	430	russia	queued	sbp_incoming	62000	2022-10-01 00:00:02
2	2e5f9d2e-2798-4527-a416-69b2f87c7289		914810	6633598	470	turkey	queued	sbp_incoming	30000	2022-10-01 00:00:04

2. Создание таблиц для хранилища ▲

2.1 Создание STG ▲

```

-- STG currencies

CREATE TABLE IF NOT EXISTS STV2023070319__STAGING.currencies (
    date_update      timestamp,
    currency_code    int,
    currency_code_with int,
    currency_with_div numeric(5, 3)
);

CREATE PROJECTION IF NOT EXISTS currencies_date_projection (
    date_update,
    currency_code,
    currency_code_with,
    currency_with_div
)
AS
SELECT
    date_update,
    currency_code,
    currency_code_with,
    currency_with_div
FROM STV2023070319__STAGING.currencies
ORDER BY date_update

```

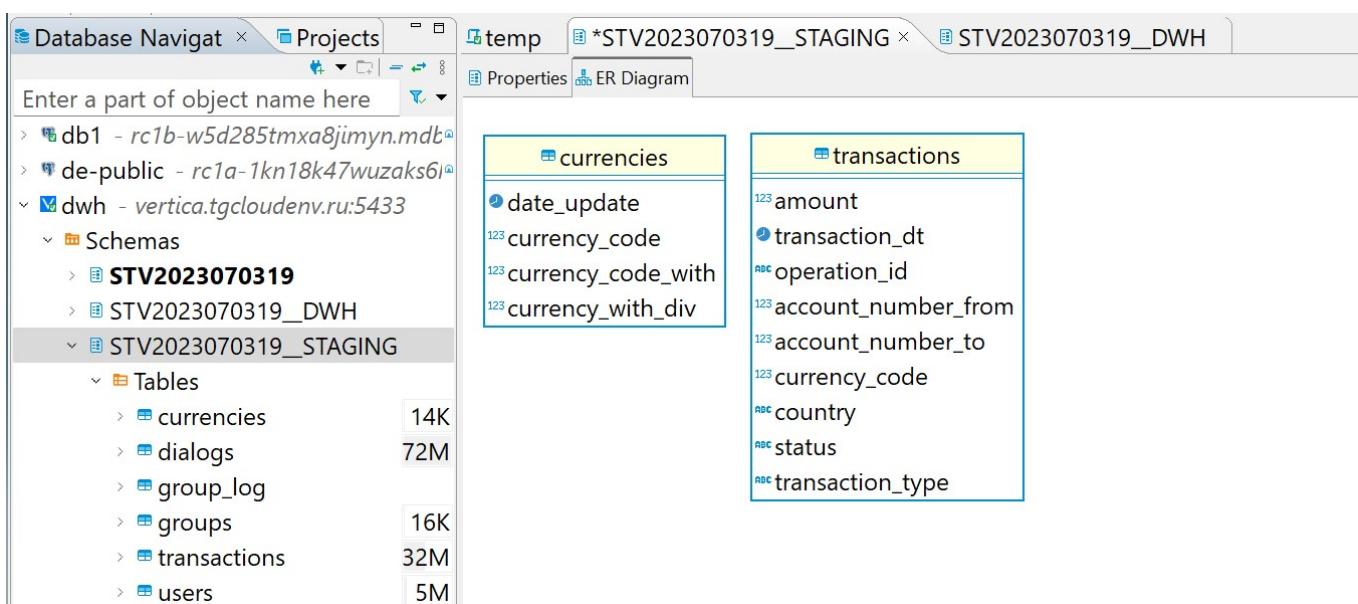
```
SEGMENTED BY hash(date_update) ALL NODES;

-- STG transactions

CREATE TABLE IF NOT EXISTS STV2023070319__STAGING.transactions (
    operation_id          varchar(60),
    account_number_from   int,
    account_number_to     int,
    currency_code         int,
    country                varchar(30),
    status                 varchar(30),
    transaction_type      varchar(30),
    amount                 int,
    transaction_dt        timestamp
);

CREATE PROJECTION IF NOT EXISTS transactions_date_projection (
    operation_id,
    account_number_from,
    account_number_to,
    currency_code,
    country,
    status,
    transaction_type,
    amount,
    transaction_dt
)
AS
SELECT
    operation_id,
    account_number_from,
    account_number_to,
    currency_code,
    country,
    status,
    transaction_type,
    amount,
    transaction_dt
FROM STV2023070319__STAGING.transactions
ORDER BY transaction_dt, operation_id
SEGMENTED BY hash(transaction_dt, operation_id) ALL NODES;
```

Результат



2.2 Создание CDM ▲

```
-- ВИТРИНА global_metrics  
  
CREATE TABLE IF NOT EXISTS STV2023070319__DWH.global_metrics (
```

```

        currency_from          int,
        amount_total           int,
        cnt_transactions       int,
        avg_transactions_per_account numeric(14, 2),
        cnt_accounts_make_transactions int,
        PRIMARY KEY (date_update, currency_from)
    )
;

```

Результат

The screenshot shows the Database Navigator interface. On the left, the database tree is visible, showing the connection to 'dwh - vertica.tgcloudenv.ru:5433' and the schema 'STV2023070319'. Under 'Tables', the 'global_metrics' table is selected, showing its columns: date_update, currency_from, amount_total, cnt_transactions, avg_transactions_per_account, and cnt_accounts_make_transactions. The table has 2.5K rows.

	global_metrics
date_update	
currency_from	
amount_total	
cnt_transactions	
avg_transactions_per_account	
cnt_accounts_make_transactions	

3. Пайплайн Airflow ▲

3.1 SQL скрипты наполнения STG и CDM ▲

```
/lessons/dags/sql/stg_currencies_fill.sql
```

```
COPY STV2023070319__STAGING.currencies (
    date_update,
    currency_code,
    currency_code_with,
    currency_with_div
)
FROM LOCAL '/lessons/data/currencies.csv'
DELIMITER ','
;
```

```
/lessons/dags/sql/stg_transactions_fill.sql
```

```
COPY STV2023070319__STAGING.transactions (
    operation_id      ,
    account_number_from,
    account_number_to,
    currency_code,
    country,
    status,
    transaction_type,
    amount,
    transaction_dt
)
FROM LOCAL '/lessons/data/transactions.csv'
DELIMITER ','
;
```

```
/lessons/dags/sql/global_metrics_fill.sql
```

```
INSERT INTO STV2023070319__DWH.global_metrics (
    date_update,
    currency_from,
    amount_total,
    cnt_transactions,
    avg_transactions_per_account,
```

```

cnt_accounts_make_transactions
)
-- 2 шаг.
-- Создаем второй подзапрос
-- где джойном подзапрос actual_currencies со справочником валют к таблице транзакций
WITH filtered_data AS (
    -- 1 шаг.
    -- сначала создаем подзапрос, который выведет актуальный список валют на нужный день
    -- дополнительно добавим строку с соотношением usd/usd = 1
    WITH actual_currencies AS (
        SELECT
            DATE(date_update) AS date_update,
            currency_code,
            currency_code_with,
            currency_with_div
        FROM STV2023070319__STAGING.currencies AS c
        WHERE DATE(date_update) = DATE('{{ds}}') - INTERVAL '1 day'
            AND currency_code_with = 420
        UNION -- присоединяя недостающую строку с usd
        SELECT
            DATE(date_update) AS date_update,
            420 AS currency_code,
            420 AS currency_code_with,
            1 AS currency_with_div
        FROM STV2023070319__STAGING.currencies AS c
        WHERE DATE(date_update) = DATE('{{ds}}') - INTERVAL '1 day'
    )
    SELECT
        c.date_update,
        t.operation_id,
        t.account_number_from,
        t.currency_code AS currency_from,
        t.amount,
        c.currency_with_div,
        t.amount * c.currency_with_div AS usd_amount
    FROM STV2023070319__STAGING.transactions AS t
    LEFT JOIN actual_currencies AS c ON t.currency_code = c.currency_code
    WHERE
        DATE(transaction_dt) = DATE('{{ds}}') - INTERVAL '1 day' AND
        t.status = 'done' AND
        t.account_number_from > 0
    )
-- 3 шаг.
-- Получаем необходимую витрину
SELECT
    date_update AS date_update,
    currency_from AS currency_from,
    SUM(usd_amount) AS amount_total,
    COUNT(date_update) AS cnt_transactions,
    (
        SUM(COUNT(date_update)) OVER() /
        COUNT(DISTINCT account_number_from)
    ) AS avg_transactions_per_account,
    COUNT(DISTINCT account_number_from) AS cnt_accounts_make_transactions
FROM filtered_data AS f
GROUP BY date_update, currency_from
;

```

3.2 DAG ▲

/lessons/dags/final_project.py

```

from airflow import DAG
from airflow.operators.python import PythonOperator
from airflow.providers.common.sql.operators.sql import SQLExecuteQueryOperator

from airflow.decorators import dag
import psycopg2
import pandas as pd
from datetime import datetime, timedelta

conn_postgre = {'host': 'rc1b-w5d285ttxa8jimyn.mdb.yandexcloud.net',
                'port': '6432',

```

```

        'user': 'student',
        'password': 'FakePass',
        'database': 'db1',
        'sslmode': 'require',
        'sslrootcert': '/lessons/cert/CA.pem'
    }

# PythonOperator (Скачиваем currencies) @ /Lessons/data/currencies.csv

def download_currencies_f(ds, conn_info=conn_postgre):
    with psycopg2.connect(**conn_info) as conn:

        querry = f"""
            SELECT *
            FROM currencies
            WHERE date_update = DATE('{ds}') - INTERVAL '1 day'
        ;
        """

        cur = conn.cursor()
        cur.execute(querry)
        df_currencies = pd.DataFrame(cur.fetchall(),
                                      columns=[col[0] for col in cur.description])

        df_currencies.to_csv('/lessons/data/currencies.csv',
                             sep=',',
                             index=False,
                             header=True)

    return 300

# PythonOperator (Скачиваем transactions) @ /Lessons/data/transactions.csv

def download_transactions_f(ds, conn_info=conn_postgre):
    with psycopg2.connect(**conn_info) as conn:

        querry = f"""
            SELECT *
            FROM transactions
            WHERE transaction_dt >= DATE('{ds}') - INTERVAL '1 day'
                AND transaction_dt <= DATE('{ds}')
        ;
        """

        cur = conn.cursor()
        cur.execute(querry)
        df_transactions = pd.DataFrame(cur.fetchall(),
                                         columns=[col[0] for col in cur.description])

        df_transactions.to_csv('/lessons/data/transactions.csv',
                             sep=',',
                             index=False,
                             header=True)

    return 300

# ----- DAG -----
@dag(
    start_date=datetime(2022, 10, 1),
    end_date=datetime(2022, 10, 31),
    schedule='@daily',
    catchup=True
)
def final_project_20():

    download_currencies = PythonOperator(
        task_id='download_currencies',
        python_callable=download_currencies_f
    )

    download_transactions = PythonOperator(
        task_id='download_transactions',

```

```

        python_callable=download_transactions_f
    )

    stg_currencies_fill = SQLExecuteQueryOperator(
        task_id='stg_currencies_fill',
        sql='sql/stg_currencies_fill.sql',
        conn_id='vertica_conn'
    )

    stg_transactionss_fill = SQLExecuteQueryOperator(
        task_id='stg_transactions_fill',
        sql='sql/stg_transactions_fill.sql',
        conn_id='vertica_conn'
    )

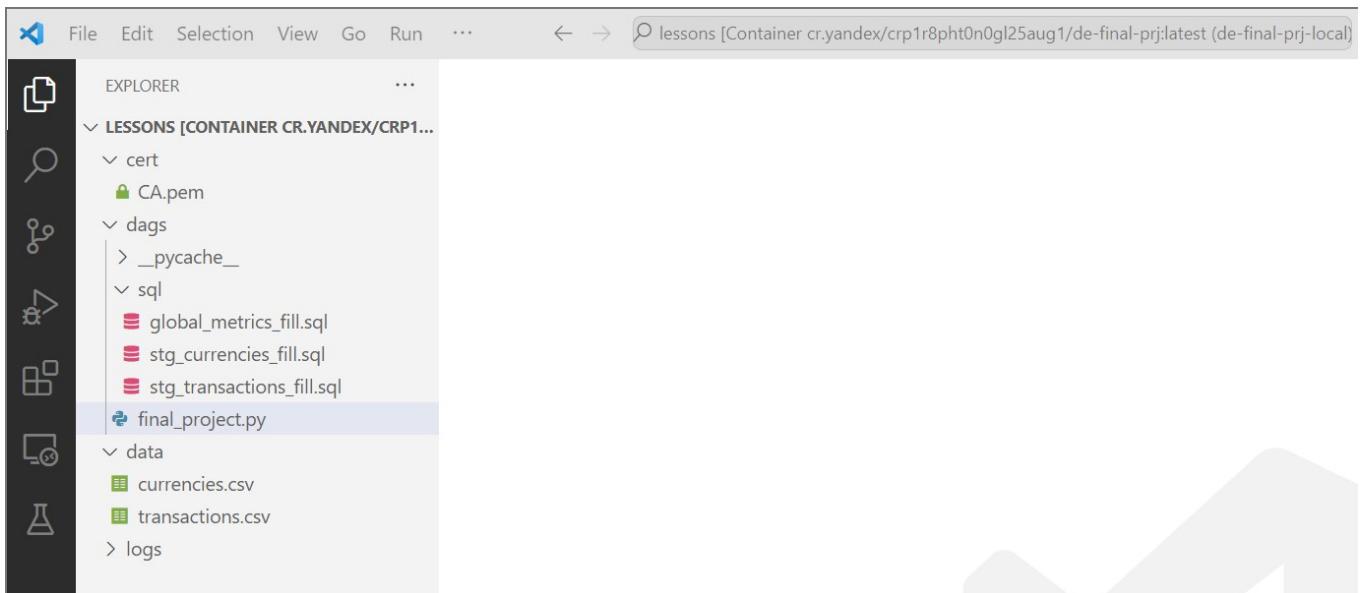
    global_metrics_fill = SQLExecuteQueryOperator(
        task_id='global_metrics_fill',
        sql='sql/global_metrics_fill.sql',
        conn_id='vertica_conn'
    )

    [download_currencies, download_transactions] >> stg_currencies_fill >> stg_transactionss_fill >>
    global_metrics_fill

dag_project = final_project_20()

```

Итого структура



3.3 Запуск пайплайна ▲

Подключение к Airflow

<http://localhost:8280/airflow/>

- пользователь — `AirflowAdmin`
- пароль — `airflow_pass`

Создание переменной

Connection successfully tested



Edit Connection

Connection Id *	vertica_conn
Connection Type *	Vertica
Connection Type missing? Make sure you've installed the corresponding Airflow Provider Package.	
Description	
Host	vertica.tgcloudenv.ru
Schema	dwh
Login	stv2023070319
Password	*****
Port	5433
Extra	

SaveTest

List Connection

Search



Actions



Record Count: 1

	Conn Id	Conn Type	Description	Host	Port	Is Encrypted	Is Extra Encrypted
<input type="checkbox"/>	vertica_conn	vertica		vertica.tgcloudenv.ru	5433	False	False

Граф

The screenshot shows the Airflow web interface with the following details:

- Header:** Airflow logo, DAGs, Datasets, Security, Browse, Admin, Docs, 18:06 UTC, NN.
- DAG Information:** DAG: final_project_10, Schedule: @daily, Next Run: 2022-10-01, 00:00:00.
- Toolbars:** Grid, Graph (selected), Calendar, Task Duration, Task Tries, Landing Times, Gantt, Details, Code, Audit Log.
- Timeline:** Date: 2023-12-18T18:05:54Z, Runs: 25, Run dropdown, Layout, Left > Right, Update button, No DAG runs yet, Find Task... button.
- Operators:** PythonOperator, SQLExecuteQueryOperator.
- Status Legend:** deferred (purple), failed (red), queued (yellow), running (green), scheduled (orange), skipped (pink), success (teal), up_for_reschedule (light green), up_for_retry (light blue), upstream_failed (light orange), no_status (grey).
- Graph View:** Shows tasks: download_currencies, stg_currencies_fill, stg_transactions_fill, global_metrics_fill, download_transactions. download_currencies and download_transactions have arrows pointing to stg_currencies_fill. stg_currencies_fill has arrows pointing to stg_transactions_fill and global_metrics_fill.

Запуск DAG

Triggered final_project_20, it should start any moment now.

DAG: final_project_20

Schedule: @daily | Next Run: 2022-10-01, 00:00:00

Grid Graph Calendar Task Duration Task Tries Landing Times Gantt Details Code Audit Log

18.12.2023 19:13:58 25 All Run Types All Run States Clear Filters

deferred failed queued running scheduled skipped success up_for_reschedule up_for_retry upstream_failed no_status

Auto-refresh

Duration Oct 01, 00:00 Oct 11, 00:00 Oct 21, 00:00

0.00:43
0.00:21
0.00:00

download_currencies
download_transactions
stg_currencies_fill
stg_transactions_fill
global_metrics_fill

DAG final_project_20

DAG Details

DAG Runs Summary

Total Runs Displayed	24
Total success	21
Total running	3

First Run Start 2023-12-18, 19:13:58 UTC

Last Run Start 2023-12-18, 19:13:58 UTC

Max Run Duration 00:00:43

Mean Run Duration 00:00:24

Min Run Duration 00:00:00

DAG Summary

Total Tasks	5
PythonOperators	2

3.4 Проверка результата работы ▾

STV2023070319__STAGING.currencies

temp global_metrics currencies transactions

currencies Enter a SQL expression to filter results (use Ctrl+Space)

Grid	date_update	currency_code	currency_code_with	currency_with_div
1	2022-10-05 00:00:00.000	440	420	0.95
2	2022-10-05 00:00:00.000	440	410	1.04
3	2022-10-05 00:00:00.000	440	450	0.91
4	2022-10-05 00:00:00.000	440	460	1.04
5	2022-10-05 00:00:00.000	440	470	0.98
6	2022-10-05 00:00:00.000	440	430	0.97
7	2022-10-05 00:00:00.000	420	440	1.04
8	2022-10-05 00:00:00.000	420	410	1.08
9	2022-10-05 00:00:00.000	420	450	0.9
10	2022-10-05 00:00:00.000	420	460	1
11	2022-10-05 00:00:00.000	420	470	1
12	2022-10-05 00:00:00.000	420	430	0.96
13	2022-10-05 00:00:00.000	410	440	1.09
14	2022-10-05 00:00:00.000	410	420	1.01
15	2022-10-05 00:00:00.000	410	450	0.94
16	2022-10-05 00:00:00.000	410	460	0.98
17	2022-10-05 00:00:00.000	410	470	1.09
18	2022-10-05 00:00:00.000	410	430	1
19	2022-10-05 00:00:00.000	450	440	1.07
20	2022-10-05 00:00:00.000	450	420	0.96

STV2023070319__STAGING.transactions

temp global_metrics currencies transactions

transactions Enter a SQL expression to filter results (use Ctrl+Space)

Grid	operation_id	account_number_from	account_number_to	currency_code	country	status	transaction_type	amount	transaction_dt
1	f2e13ea0-b518-4b93-a3dc-0de852d774ae	8,974,921	1,339,976	450	china	queued	c2b_partner_incoming	100,000	2022-10-05 00:00:04.000
2	a8ac7f87-55aa-461a-ad5c-b58adb3b056b	5,447,231	732,193	460	england	queued	sbp_incoming	34,600	2022-10-05 00:00:10.000
3	92f57479-2e3a-4764-a919-247c3b370c61	6,126,863	818,011	430	russia	in_progress	c2b_partner_incoming	400,000	2022-10-05 00:00:18.000
4	36b5480a-96e5-4eec-87be-91076f5019cf	7,127,384	6,898,713	430	russia	queued	c2a_incoming	245,000	2022-10-05 00:00:24.000
5	0a180cad-b831-49c6-81a9-db9852e66295	6,503,791	903,810	430	russia	queued	sbp_incoming	28,000	2022-10-05 00:00:27.000
6	5ed0672e-a412-4f79-9e79-549092be0343	587,692	914,810	430	russia	queued	sbp_incoming	267,400	2022-10-05 00:00:33.000
7	8ab4d77d-872c-4d91-add7-38a7cf629911	637,197	903,810	430	russia	queued	sbp_incoming	20,000	2022-10-05 00:00:34.000
8	fe5a3ba2-6018-4d95-9657-11d4b39f66e8	3,238,888	6,710,317	430	russia	queued	sbp_incoming	6,000	2022-10-05 00:00:39.000
9	722c7885-2d33-4c33-b337-0da09e3c0b27	275,282	2,589,745	430	russia	done	sbp_outgoing	-32,900	2022-10-05 00:00:44.000
10	a43daada-67cd-44ac-b59f-fd9756bfc7ee	-1	6,513,762	460	england	queued	c2b_partner_incoming	100,000	2022-10-05 00:00:44.000
11	5ed0672e-a412-4f79-9e79-549092be0343	587,692	914,810	430	russia	in_progress	sbp_incoming	267,400	2022-10-05 00:00:45.000
12	b64438c4-97dd-4994-ad1e-0e2e3233ad28	7,201,816	685,444	430	russia	queued	sbp_incoming	212,000	2022-10-05 00:00:46.000
13	087a0343-cbb6-4ea3-bcd6-4782ab91962e	8,632,112	3,816,782	430	russia	queued	c2a_incoming	600,000	2022-10-05 00:00:49.000
14	0a180cad-b831-49c6-81a9-db9852e66295	6,503,791	903,810	430	russia	done	sbp_incoming	28,000	2022-10-05 00:00:51.000
15	5751edd8-b6f6-4c00-812c-23686cd48b2e	947,426	673,565	430	russia	done	sbp_outgoing	-300,000	2022-10-05 00:00:53.000
16	5ed0672e-a412-4f79-9e79-549092be0343	587,692	914,810	430	russia	done	sbp_incoming	267,400	2022-10-05 00:00:57.000
17	f07fb6c9-e895-4146-968c-0c139d6bf54f	2,874,217	914,810	410	canada	in_progress	sbp_incoming	10,000	2022-10-05 00:01:00.000
18	5a4727c3-9f8d-4b12-af75-0024bedfce91	7,822,222	4,458,968	430	russia	queued	sbp_incoming	80,000	2022-10-05 00:01:03.000
19	fe5a3ba2-6018-4d95-9657-11d4b39f66e8	3,238,888	6,710,317	430	russia	done	sbp_incoming	6,000	2022-10-05 00:01:03.000
20	2081435d-10ad-4bb7-8991-b2b6d82f186	2,538,753	903,810	430	russia	queued	sbp_incoming	51,800	2022-10-05 00:01:05.000

STV2023070319__DWH.global_metrics

Grid	date_update	currency_from	amount_total	cnt_transactions	avg_transactions_per_account	cnt_accounts_make_transactions
1	2022-10-05	420	784,186,003	4,614	8.58	4,572
2	2022-10-05	430	2,983,253,028	18,621	2.19	17,938
3	2022-10-05	440	431,795,870	2,893	13.61	2,881
4	2022-10-05	450	504,893,215	3,126	12.64	3,102
5	2022-10-05	460	545,311,419	3,481	11.36	3,451
6	2022-10-05	470	712,919,861	3,963	9.97	3,932
7	2022-10-08	430	39,001,746,652	9,448	2.47	8,138
8	2022-10-08	460	7,759,359,758	1,852	11.17	1,797
9	2022-10-09	450	5,940,000	1	5	1
10	2022-10-09	460	9,456,433	1	5	1
11	2022-10-18	420	2,193,919,414	466	17.21	214
12	2022-10-18	460	1,800,822,720	354	21.66	170
13	2022-10-23	410	422,433,860	2,598	15.02	2,589
14	2022-10-23	450	522,887,578	3,093	12.64	3,078
15	2022-10-23	460	547,167,591	3,524	11.12	3,499
16	2022-10-05	410	427,184,825	2,508	15.69	2,499
17	2022-10-08	410	6,367,141,019	1,340	15.37	1,306
18	2022-10-08	420	10,059,445,891	2,299	9.08	2,211
19	2022-10-09	410	206,001	1	5	1
20	2022-10-18	430	8,408,782,405	1,704	5.04	731

date_update	currency_from	amount_total	cnt_transactions	avg_transactions_per_account	cnt_accounts_make_transactions
2022-10-05	420	784186003	4614	8.58	4572
2022-10-05	430	2983253028	18621	2.19	17938
2022-10-05	440	431795870	2893	13.61	2881
2022-10-05	450	504893215	3126	12.64	3102
2022-10-05	460	545311419	3481	11.36	3451
2022-10-05	470	712919861	3963	9.97	3932
2022-10-08	430	39001746652	9448	2.47	8138
2022-10-08	460	7759359758	1852	11.17	1797
2022-10-09	450	5940000	1	5.00	1
2022-10-09	460	9456433	1	5.00	1
2022-10-18	420	2193919414	466	17.21	214
2022-10-18	460	1800822720	354	21.66	170
2022-10-23	410	422433860	2598	15.02	2589
2022-10-23	450	522887578	3093	12.64	3078
2022-10-23	460	547167591	3524	11.12	3499
2022-10-05	410	427184825	2508	15.69	2499
2022-10-08	410	6367141019	1340	15.37	1306
2022-10-08	420	10059445891	2299	9.08	2211
2022-10-09	410	206001	1	5.00	1
2022-10-18	430	8408782405	1704	5.04	731
2022-10-23	420	741183528	4488	8.74	4450
2022-10-23	430	2887501270	18440	2.18	17861
2022-10-08	440	6029238253	1472	14.01	1433
2022-10-08	450	6610901349	1625	12.69	1582
2022-10-08	470	9738134001	2036	10.20	1968
2022-10-09	430	64800000	1	5.00	1
2022-10-09	470	62400000	1	5.00	1
2022-10-18	410	777591016	214	34.74	106
2022-10-18	440	1834715521	288	27.27	135
2022-10-18	450	1310712772	272	27.48	134
2022-10-18	470	1592801336	384	19.80	186
2022-10-23	440	450570830	2894	13.52	2877

date_update	currency_from	amount_total	cnt_transactions	avg_transactions_per_account	cnt_accounts_make_transactions
2022-10-23	470	611890648	3857	10.16	3829

4. Создание дашборда ▲

4.1 Требования ▲

- Визуализируйте метрику суммы переводов с возможностью смотреть общую сумму и выбирать отдельные валюты через фильтр.
- Визуализируйте метрику среднего объема транзакций на пользователя.
- Визуализируйте метрику количества уникальных пользователей, которые совершают транзакции в валютах.
- Визуализируйте метрику общего оборота компании в единой валюте.
- Настройте фильтры по дате и выбранным валютам перевода.

4.2 Подключение ▲

Подключение к Metabse

<http://localhost:8998/>

- Пользователь: mustdayker@yandex.ru
- Компания: mustdayker_team
- Пароль: metabase_pass_01

Создаем подключение к Vertica

- Host: 51.250.75.20 или vertical.tgcloudenv.ru
- Порт: 5433
- Login: stv2023070319
- Pass: FakePass
- Схема: dwh

Database type

Vertica

Need help setting up Vertica?
Our docs can help.

Name

Vertica_DWH

Host

vertica.tgcloudenv.ru

Port

5433

Database name

dwh

Username

stv2023070319

Password

The screenshot shows the Metabase interface. At the top, there's a navigation bar with tabs like 'Metabase', 'Browse', and 'Explore'. Below it is a search bar with placeholder 'Search...'. In the main area, the sidebar shows 'OUR DATA' with three datasets: 'STV2023070319', 'STV2023070319_DWH', and 'STV2023070319_STAGING'. The 'STV2023070319_DWH' dataset is currently selected. The top right corner has a 'Learn about our data' button.

4.3 Вспомогательный датасет ▲

Создадим вспомогательный датасет, где привяжем регион валюты в виде текста, для удобной фильтрации и вывода информации.

Для того чтобы можно было подключить фильтр используем заглушки для переменных формата `[[WHERE cc.country = {{country}}]]`

```
WITH currency_codes AS (
  SELECT DISTINCT
    currency_code,
    country
  FROM STV2023070319_STAGING.transactions as t
)
SELECT *
FROM STV2023070319_DWH.global_metrics AS gm
LEFT JOIN currency_codes AS cc ON gm.currency_from = cc.currency_code
[[WHERE cc.country = {{country}}]]
```



Search...

Ask a question



cdm_w_currency ▾ Edited 31 minutes ago by you

Our analytics Vertica_DWH

Vertica_DWH ▾

Aa Country

```
1 WITH currency_codes AS (
2     SELECT DISTINCT
3         currency_code,
4         country
5     FROM STV2023070319__STAGING.transactions as t
6 )
7 SELECT *
8 FROM STV2023070319__DWH.global_metrics AS gm
9 LEFT JOIN currency_codes AS cc ON gm.currency_from = cc.currency_code
10 [[WHERE cc.country = {{country}}]]
```

date_update	currency_from	amount_total (\$)	cnt_transactions	avg_transactions_per_account	cnt_accounts_make_transactions	country
23 October, 2022	460	547 167 591,00	3,524	11.12	3,499	england
18 October, 2022	460	1 800 822 720,00	354	21.66	170	england
9 October, 2022	460	9 456 433,00	1	5	1	england
8 October, 2022	460	7 759 359 758,00	1,852	11.17	1,797	england
5 October, 2022	460	545 311 419,00	3,481	11.36	3,451	england
5 October, 2022	430	2 983 253 028,00	18,621	2.19	17,938	russia
9 October, 2022	430	64 800 000,00	1	5	1	russia

4.4 Визуализация ▲



Search...

Ask a question



Vertica DWH

Our analytics • Edited a few seconds ago by you



Date Range

Country

\$116 201 M
Полный доход компании в млн \$

Суммы переводов по датам

Total amount by date



Total amount by date and country in million \$

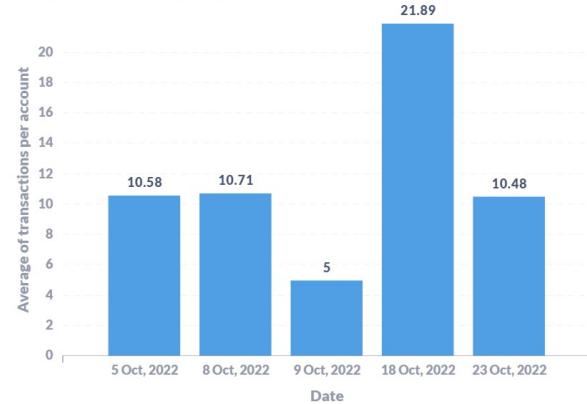
date	canada	china	england	italy	russia	turkey	usa	Row totals
5 Oct, 2022	\$427 M	\$505 M	\$545 M	\$432 M	\$2 983 M	\$713 M	\$784 M	\$6 390 M
8 Oct, 2022	\$6 367 M	\$6 611 M	\$7 759 M	\$6 029 M	\$39 002 M	\$9 738 M	\$10 059 M	\$85 566 M
9 Oct, 2022	\$0 M	\$6 M	\$9 M		\$65 M	\$62 M		\$143 M
18 Oct, 2022	\$778 M	\$1 311 M	\$1 801 M	\$1 835 M	\$8 409 M	\$1 593 M	\$2 194 M	\$17 919 M
23 Oct, 2022	\$422 M	\$523 M	\$547 M	\$451 M	\$2 888 M	\$612 M	\$741 M	\$6 184 M
Grand totals	\$7 995 M	\$8 955 M	\$10 662 M	\$8 746 M	\$53 346 M	\$12 718 M	\$13 779 M	\$116 201 M

Date Range

Country

Среднее количество транзакций на аккаунт

Average transactions per account



Количество уникальных пользователей

count_accounts_make_transactions

