Как сдать проект

Docker-тренажёр

① Перейдите в Docker-тренажёр, чтобы получить доступ ко всем необходимым для проекта данным и инструментам. Чтобы открыть страницу проекта, используйте следующую команду:

Скопировать код

BASH

```
docker run -d --rm -p 3000:3000 -p 15432:5432 \
--name=de-project-sprint-3-server \
cr.yandex/crp1r8pht0n0gl25aug1/project-sprint-3:latest
```

docker run -d --rm -p 3000:3000 -p 15432:5432 --name=de-project-sprint-3-server cr.yandex/crp1r8pht0n0gl25aug1/project-sprint-3:latest

① Когда решение будет готово, сохраните и выгрузите его в нужный репозиторий в GitHub.
② Вернитесь на платформу. Чтобы отправить проект на проверку ревьюеру, нажмите в конце урока «Проект» на кнопку `Отправить на проверку`. В течение суток вы получите комментарии. Их нужно учесть: доработать проект и вернуть ревьюеру обновлённый вариант.
③ Возможно, вы будете дорабатывать кейс по комментариям несколько раз. Не переживайте, это нормально.
④ Проект завершён, когда одобрены все доработки.

Работа в GitHub

Вы можете пройти небольшой дополнительный курс по Git и GitHub, если не работали с этими инструментами раньше или просто хотите освежить знания. Вы найдете его в личном кабинете внизу списка всех курсов профессии «Инженер данных».

Шаг 1. Привязка аккаунта

① Для начала привяжите аккаунт в GitHub. Для этого нажмите одноимённую кнопку на странице проекта в Практикуме.
② Откроется страница, где вам нужно будет авторизоваться в своем GitHub-аккаунте, а затем согласиться на авторизацию yandex-praktikum.
③ Готово: вы видите надпись «Github настроен». Репозиторий появится в списке репозиториев на GitHub — в него вы и будете заливать ваше решение. Название репозитория будет таким: «de-project-sprint-N», где N — номер спринта.
④ Обязательно ознакомьтесь с файлом `Readme.md`, который вы найдете в репозитории, чтобы получить детальные инструкции по работе с проектом.

Шаг 2. Клонирование репозитория на свой компьютер

① Откройте папку, в которую вы хотите поместить ваше решение. Вызовите терминал и используйте команду `git clone ссылка-на-репозиторий.git`
② Вы видите запрос доступа Git Credential Manager — соглашайтесь.
③ Репозиторий клонирован.

Шаг 3. Первый коммит и отправка задания

① Выполните задание в этой папке.
② Запустите терминал. Чтобы добавить изменения, используйте команду `git commit -m "комментарий" -a`. Для загрузки коммитов из локального репозитория в удалённый (Github) используйте команду `git push origin main` (где main — название ветки).
③ Изменения сохранены. Чтобы убедиться в этом, зайдите на страницу репозитория в GitHub.
④ Вернитесь на платформу, чтобы отправить проект на проверку ревьюеру.

Проектная работа по ETL и автоматизации подготовки данных

В этом проекте вы потренируетесь изменять процессы в пайплайне так, чтобы они соответствовали новым задачам бизнеса.

Мы разделили задание на три этапа развития пайплайна — вы будете постепенно дорабатывать его, получая новые вводные от заказчика. В итоге должен получится код, который включает в себя решения для каждой ситуации.

Мы положили эталонный пайплайн в Docker-контейнер — примерно такой, какой должен был получиться у вас в конце этого спринта. Вы можете доработать его или написать все самостоятельно.

Этап 1

Пайплайн, который вы сделали в этом спринте, работает, и ваши коллеги уже успешно им пользуются. На данных из витрины `mart.f_sales` BI-аналитики построили графики `total revenue` для различных срезов.

Система магазина продолжает развиваться: команда разработки добавила функционал отмены заказов и возврата средств (`refunded`). Значит, процессы в пайплайне нужно обновить.

Исходные данные

Новые инкременты с информацией о продажах приходят по API, спецификация к которому лежит ниже, и содержат статус заказа (`shipped`/`refunded`).

Спецификация API

POST /generate_report

Метод `/generate_report` инициализирует формирование отчёта. В заголовке запроса нужно указать:

- `X-Nickname` — ваш никнейм (например, sashanikonov).
- `X-Project` со значением `True` — так мы узнаем, что вы выполняете итоговый проект.
- `X-Cohort` со значением номера вашей когорты (например, 1).
- `X-API-KEY` со значением `5f55e6c0-e9e5-4a9c-b313-63c01fc31460`. Если не указать верный ключ к API, то вместо ожидаемого результата выйдет ошибка доступа.

Установите предварительно утилиту curl, чтобы с помощью неё обратиться к API. Выполните следующие команды в командной строке:

Скопировать код

JSON

```
Remove-Item Alias:curl

curl --location --request POST
'https://d5dg1j9kt695d30blp03.apigw.yandexcloud.net/generate_report' `
--header 'X-Nickname: Pavel_Kovalchuk' `
--header 'X-Cohort: 31' `
--header 'X-Project: True' `
--header 'X-API-KEY: 5f55e6c0-e9e5-4a9c-b313-63c01fc31460' `
--data-raw ''
```

рабочий вариант

```
& 'C:\Program Files\Git\mingw64\bin\curl.exe' --location --request POST
'https://d5dg1j9kt695d30blp03.apigw.yandexcloud.net/generate_report' `
--header 'X-Nickname: Pavel_Kovalchuk' `
--header 'X-Cohort: 31' `
--header 'X-Project: True' `
--header 'X-API-KEY: 5f55e6c0-e9e5-4a9c-b313-63c01fc31460'
```

В качестве переменных, указанных в `{{ }}`, передайте ваши актуальные данные.

Метод возвращает `task_id` — ID задачи, в результате выполнения которой должен сформироваться отчёт.

GET /get_report

Метод `get_report` используется для получения отчёта после того, как он будет сформирован на сервере.

С помощью утилиты curl обратитесь к API:

Скопировать код

JSON

```
curl --location --request GET
'https://d5dg1j9kt695d30blp03.apigw.yandexcloud.net/get_report?task_id={{ task_id }}'
\
--header 'X-Nickname: {{ your_nickname }}' \
--header 'X-Cohort: {{ your_cohort_number }}' \
--header 'X-Project: True' \
--header 'X-API-KEY: {{ api_key }}'
```

Пока отчёт будет формироваться, будет возвращаться статус `RUNNING`.

Если отчёт сформирован, то метод вернёт статус `SUCCESS` и `report_id`.

Сформированный отчёт содержит четыре файла:

- `customer_research.csv`,
- `user_order_log.csv`,
- `user_activity_log.csv`,
- `price_log.csv`.

Файлы отчетов можно получить по URL из параметра `s3_path` или сформировать URL самостоятельно по следующему шаблону:

Скопировать код

JSON

```
https://storage.yandexcloud.net/s3-sprint3/cohort_{{ your_cohort_number }}/{{
your_nickname }}/project/{{ report_id }}/{{ file_name }}
```

GET /get_increment

Метод `get_increment` используется для получения данных за те даты, которые не вошли в основной отчёт. Дата обязательно в формате `2020-01-22T00:00:00`.

С помощью утилиты curl обратитесь к API:

Скопировать код

JSON

```
curl --location --request GET
'https://d5dg1j9kt695d30blp03.apigw.yandexcloud.net/get_increment?report_id={{
report_id }}&date={{ date }}' \
--header 'X-Nickname: {{ your_nickname }}' \
--header 'X-Cohort: {{ your_cohort_number }}' \
--header 'X-Project: True' \
--header 'X-API-KEY: {{ api_key }}'
```

Если инкремент сформирован, то метод вернёт статус `SUCCESS` и `increment_id`. Если инкремент не сформируется, то вернётся `NOT FOUND` с описанием причины.

Сформированный инкремент содержит три файла:

- `customer_research_inc.csv`,
- `user_order_log_inc.csv`,
- `user_activity_log_inc.csv`.

Файлы отчетов можно получить по URL из параметра `s3_path` или сформировать URL самостоятельно по следующему шаблону:

Скопировать код

JSON

```
https://storage.yandexcloud.net/s3-sprint3/cohort_{{ your_cohort_number }}/{{ your_nickname }}/project/{{ increment_id }}/{{ file_name }}
```

Ваша задача

Адаптируйте ваш пайплайн для текущей задачи:

- Учтите в витрине `mart.f_sales` статусы `shipped` и `refunded`. Все данные в витрине следует считать `shipped`.
- Обновите пайплайн с учётом статусов и backward compatibility.

Подсказки:

① Необходимо провести миграцию схемы и данных в таблице `mart.f_sales`.
② Таблица фактов `mart.f_sales` должна приобрести вид транзакционной таблицы фактов. Будьте внимательны со статусом `refunded`.
③ Чтобы `total revenue` правильно рассчитывался, строки с `refunded` добавляйте со знаком `-`.
④ Первый инкремент приходит со старым форматом данных — без статуса заказа. Проверьте, что ваш код правильно проставляет статус в этом случае.

Этап 2

Ваши коллеги решили лучше изучить поведение клиентов. Для этого они хотят исследовать возвращаемость клиентов, или customer retention.

Выяснилось, что на текущий момент отчёт по customer retention строится очень долго. Коллеги попросили вас вычислить нужные метрики в дополнительной витрине.

Эта витрина должна отражать следующую информацию:

- Рассматриваемый период — weekly.
- Возвращаемость клиентов:
  - `new` — кол-во клиентов, которые оформили один заказ за рассматриваемый период;

    - `returning` — кол-во клиентов, которые оформили более одного заказа за рассматриваемый период;
    - `refunded` — кол-во клиентов, которые вернули заказ за рассматриваемый период.
  - Доход (`revenue`) и `refunded` для каждой категории покупателей.

Благодаря витрине можно будет выяснить, какие категории товаров лучше всего удерживают клиентов.

Коллеги подготовили для вас схему витрины `mart.f_customer_retention`:

Скопировать код

```
mart.f_customer_retention
1. new_customers_count — кол-во новых клиентов (тех, которые сделали только один заказ за рассматриваемый промежуток времени).
2. returning_customers_count — кол-во вернувшихся клиентов (тех, которые сделали только несколько заказов за рассматриваемый промежуток времени).
3. refunded_customer_count — кол-во клиентов, оформивших возврат за рассматриваемый промежуток времени.
4. period_name — weekly.
5. period_id — идентификатор периода (номер недели или номер месяца).
6. item_id — идентификатор категории товара.
7. new_customers_revenue — доход с новых клиентов.
8. returning_customers_revenue — доход с вернувшихся клиентов.
9. customers_refunded — количество возвратов клиентов.
```

Ваша задача

На основе пайплайна из сквозной задачи спринта наполните витрину `mart.f_customer_retention` данными по «возвращаемости клиентов» в разрезе недель.

Этап 3 — опционально

📌 Мы примем проект без этого этапа, если предыдущие два сделаны верно. Этот этап для тех, кто любит челлендж и готов разобраться с идемпотентностью.

В понедельник вы пришли на работу, проверили систему мониторинга и не нашли никаких неполадок. Однако коллеги-разработчики сообщили, что из-за бага в бэкенде данные из источника за воскресенье пришли неполные. Баг починили, данные инкремента восстановили, и теперь вам нужно сделать так, чтобы данные в аналитической системе отражали реальную картину.

Ваша задача

- Перезапустить пайплайн и убедиться, что после перезапуска не появилось дубликатов в витринах `mart.f_sales` и `mart.f_customer_retention`.

Источники данных

Получить новый инкремент можно по API, используя метод `GET /get_increment`.

Подсказка:

Проверьте, что каждый этап пайплайна учитывает удаление предыдущих данных за расчётный период.