# Домашнее задание

Форк репозитария расположен по адресу:

<https://github.com/a-v-kalabuhov/dbt-datavault-postgres.git>

Все изменения внесены в ветку [**practice-05.06.25**](https://github.com/a-v-kalabuhov/dbt-datavault-postgres/tree/practice-05.06.25).

Теперь непосредственно ДЗ:

1. **Добавить в любой csv файл новую строку с новым бизнес-ключом. Построить все модели, описать, что произошло с данными на всех этапах.**

---  
Добавил строку в файл source\_customers.csv.  
После загрузки seed’ов новая строка появилась в таблице *source\_customers*, т.к. таблицы seed’ов при загрузке полностью пересоздаются.   
После построения моделей:  
Таблица source\_customers является источником для view *stg\_customers*, который является источником данных для хаба *hub\_customer* и сателлита *sat\_customer*. Соответственно, новая строка из этого view была далее перенесена в raw vault в *hub\_customer*. Остальные строки *hub\_customer* не обрабатывались, т.к. для модели raw\_vault используется инкрементальная материализация. Аналогично новая строка добавилась в сателлит *sat\_customer*.   
В business vault в таблицу *customer\_pit* добавились строки для нового заказчика. Эти строки получаются в результате cross join новой строки из *hub\_customer* и строками таблицы *as\_of\_date*. Т.е. на каждую дату в диапазоне дат из *as\_of\_date* в таблицу *customer\_pit* добавляется строка с первичным ключом строки заказчика из *hub\_customer*.

1. **Добавить в любой csv файл новую строку с существующим бизнес-ключом, но другими свойствами. Построить все модели, описать, что произошло с данными на всех этапах.**

---

Добавил строку в файл source\_customers.csv.  
После загрузки seed’ов новая строка появилась в таблице *source\_customers*.   
После построения моделей:

Новая строка была далее добавлена в raw vault в сателлит *sat\_customers* и теперь в нём хранится две строки для выбранного значения бизнес-ключа, каждое со своей датой загрузки и датой актуальности. Из сателлита можно легко выбрать актуальные данные заказчика на нужную дату. В *hub\_customer* новая строка не добавлялась, т.к. в нём уже есть запись с таким значением бизнес-ключа, а хаб по определению может хранитсья только одна запись на одно значение бизнес-ключа.

В business vault в таблице *customer\_ pit* у всех строк, связанных с изменённым заказчиком обновилась дата загрузки в поле sat\_customer\_ldts. На самом деле, данные не обновились, а были вставлены, т.к. модель customer\_pit всегда компилируется в запрос TRUNCATE TABLE. Поэтому все строки всегда вставляются в таблицу заново.

1. **Удалить строку из любого csv файла. Построить все модели, описать, что произошло с данными на всех этапах.**

---

Удалил строчку из source\_customers.csv.

В результате эта строчка пропала также из таблицы source\_customers. Никаких изменений в других таблицах не произошло. Все ранее загруженные данные в *hub\_customer*, *sat\_customer* и *customer\_pit* остались без изменений.

1. **Создать витрину, которая покажет динамику количества всех заказов в разрезе недели. Описать словами, как вы это сделали и что получилось.**

Для создания витрины:

- в папку проекта models добавил вложенную папку модели cmd, которая уже была объявлена в dbt\_project.yml

- в этой папке создал файл модели mart\_orders\_weekly.sql

- в файл добавил нужные мне sql-запросы

Сама витрина получилась следующего вида:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **ayear** | **aweek** | **week\_start\_date** | **placed** | **completed** | **shipped** | **return\_pending** | **returned** |
| 2018 | 1 | 2018-01-01T00:00:00+00:00 | 0 | 5 | 0 | 0 | 1 |
| 2018 | 2 | 2018-01-08T00:00:00+00:00 | 0 | 4 | 0 | 0 | 1 |
| 2018 | 3 | 2018-01-15T00:00:00+00:00 | 0 | 5 | 0 | 0 | 2 |
| 2018 | 4 | 2018-01-22T00:00:00+00:00 | 0 | 7 | 0 | 1 | 0 |
| 2018 | 5 | 2018-01-29T00:00:00+00:00 | 0 | 7 | 0 | 0 | 0 |
| … | | | | | | | |
| 2018 | 15 | 2018-04-09T00:00:00+00:00 | 1 | 0 | 0 | 0 | 0 |

Чтобы получить такую витрину, я с помощью CTE:

- выбрал минимальную и максимальную даты из sat\_orders

- по этим датам функцией generate\_series создал временную таблицу со списком недель от минимальной даты к максимальной – так я получил первые три столбца витрины

- далее добавил временную таблицу из sat\_orders, в которую добавил колонки номер недели и год

- чтобы посчитать количество заказов в каждой неделе я добавил к таблице недель подзапросы вида select count(\*) from sat\_order where week=:week and year=:year and status=:status

- в таблице source\_orders используется пять статусов заказа, поэтому потребовалось пять подзапросов для вычисления количества заказов по каждому из статусов.

1. **Создать витрину, в которой все клиенты будут отсортированы по количеству заказов со статусом «завершен» от наибольшего к наименьшему.**

Для создания витрины необходимо получить актуальный список клиентов и к нему добавить количество завершённых заказов:

1. Список завершённых заказав по каждому клиенту. Это можно сделать, соединив таблицы *sat\_order* и *link\_customer\_order* примерно так:  
     
   select

        lco.customer\_pk, -- клиент

        so.order\_pk -- заказ

from

        {{ref('sat\_order')}} so

        left join

        {{ref('link\_customer\_order')}} lco

        on (so.order\_pk=lco.order\_pk)

where

        so.status='completed'

1. Для актуального списка клиентов нужно выбрать последние загруженные данные клиента – имя, фамилию и email. Это можно сделать из таблицы *sat\_customer* либо подзапросом, либо оконной функцией, отобрав данные клиента с самой поздней датой загрузки в сателлит. Я сделал CTE, которое нумерует все записи каждого клиента по дате загрузки оконной функцией **ROW\_NUMBER()**:

with actual\_customers as (

  select

        customer\_pk, first\_name, last\_name,

        ROW\_NUMBER() over (PARTITION by customer\_pk order by load\_date desc) as row\_num

    from

        {{ref('sat\_customer')}} sc

)

Чтобы добавить email, нужно соединить полученное CTE с таблицей *hub\_customer* и отобрать только записи с row\_num = 1:

select

        ac.\*,

        hc.customer\_key as email

    from

        actual\_customers ac

        left join

        {{ref('hub\_customer')}} hc

        on (ac.customer\_pk = hc.customer\_pk)

    where

        row\_num = 1

1. Теперь к списку клиентво можно подхапросом добавить колиество завершённых заказов с помощью CTE их пункта 1:  
     
   select

    customer\_pk, first\_name, last\_name, email,

    (select

count(co.\*)   
from   
 completed\_orders co

where   
 co.customer\_pk=acwe.customer\_pk) as completed\_order\_count

from

   actual\_customers\_with\_email acwe

1. Осталось только отсортировать :

select

    \*

from

    customer\_rating

order by

    completed\_order\_count desc,

    last\_name,

    first\_name