Проект

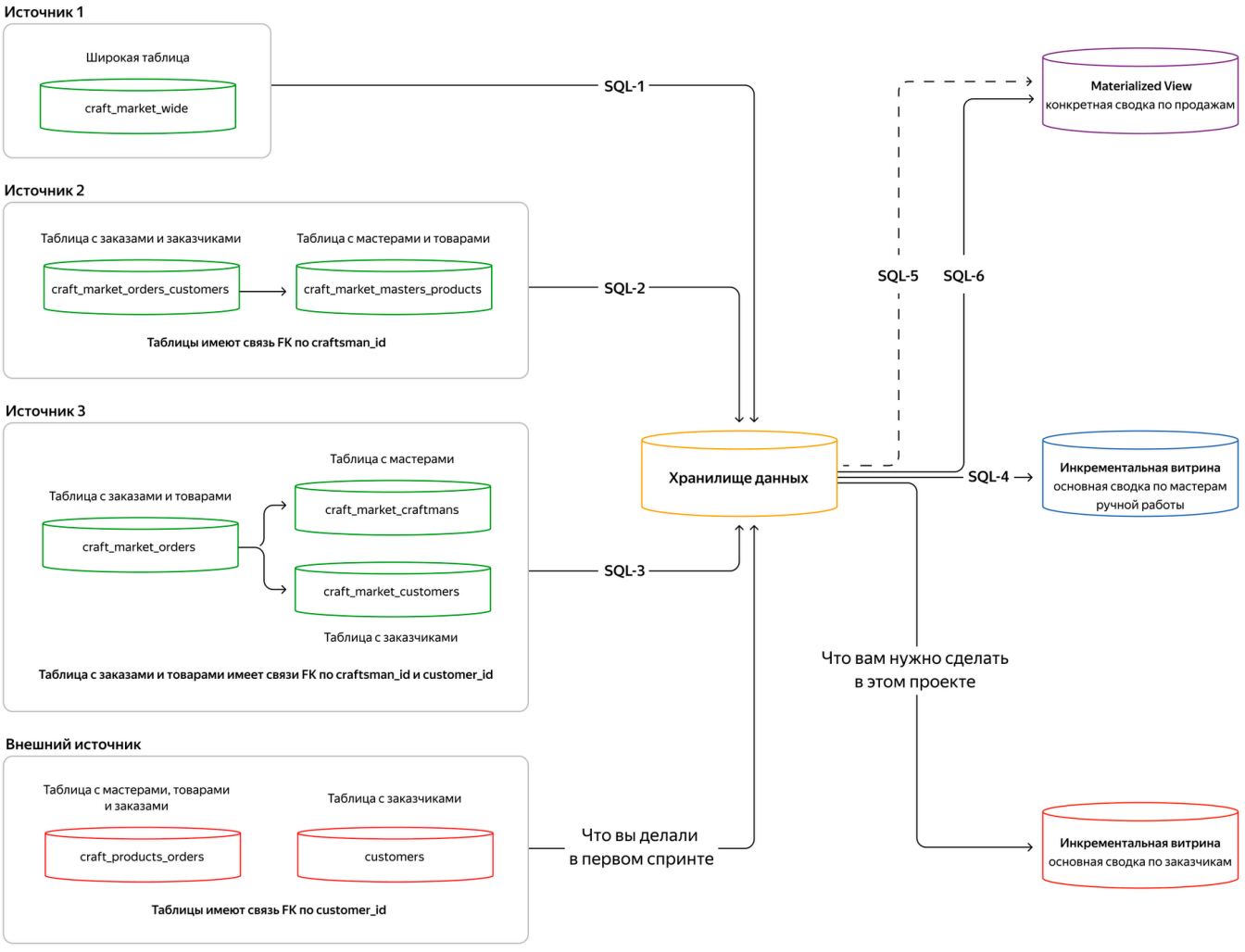
Project_DWH

Важная особенность корпоративных хранилищ — возможность интегрировать новые источники. В этом проекте вы как раз и добавите новый источник в DWH.

Маркетплейс товаров ручной работы набирает популярность и расширяет свою аудиторию. Чтобы привлечь больше покупателей, приобрели ещё один сайт, который предоставляет похожие услуги. Формальная часть сделки завершена, нужно интегрировать данные купленного сайта в хранилище.

Добавление нового источника — тот же процесс, что и загрузка предыдущих источников: нужно интегрировать данные и построить новую витрину. Вы уже строили инкрементальную витрину с отчётом по мастерам, теперь постройте похожую витрину с отчётом по заказчикам. Она пригодится, чтобы внедрить программу лояльности, изучить интересы пользователей, сделать более релевантной систему рекомендаций товаров и скорректировать маркетинговый план по акциям.

Общая схема проекта выглядит следующим образом:



Бóльшую часть работы вы сделали, осталось только загрузить данные нового источника в хранилище и построить витрину.

Инструкция по выполнению

Шаг 1. Подключение к базе

Вам понадобится та же БД, с которой вы работали в течение всего спринта. Настройки соединения с БД те же. Нужные параметры вы получили во время регистрации в сервисе проверок. Напомним их:

- **Хост** — значение поля `host` из полученных параметров подключения.
- **База данных** — значение поля `dbname` из полученных параметров подключения.
- **Порт** — 6432.
- **Аутентификация** — **Database Native**.
- **Пользователь** — значение поля `user` из полученных параметров подключения.
- **Пароль** — значение поля `password` из полученных параметров подключения.

Обязательно выберите пункт **Сохранять пароль локально**. Это позволит не вводить пароль заново при каждом входе.

Теперь напомним схему подключения, таблицы и модель данных:

- В схеме `dwh` — таблицы измерений для мастеров ручной работы (`d_craftsman`), заказчиков (`d_customer`) и товаров (`d_product`). Также там есть таблица фактов о продажах (`f_order`), инкрементальная витрина по мастерам за отчётные периоды (`craftsman_report_datamart`) и `MATERIALIZED VIEW` с отчётом продаж по всему маркетплейсу (`orders_report_materialized_view`).
- В схеме `source1` — широкая таблица первого источника `craft_market_wide`.
- В схеме `source2` — две ненормализованные таблицы: таблица мастеров и товаров (`craft_market_masters_products`) и таблица заказчиков и заказов (`craft_market_orders_customers`).
- В схеме `source3` — три таблицы: мастеров (`craft_market_craftsmans`), заказчиков (`craft_market_customers`), заказов с товарами (`craft_market_orders`).
- Схема `external_source` — новая. Там как раз находятся таблицы нового источника, которые нужно подключить к хранилищу.

Шаг 2. Изучение данных нового источника

Изучите данные в источнике. Там указаны две таблицы: в одной данные по мастерам, товарам и заказам (`craft_products_orders`), а в другой — только по заказчикам (`customers`). Правильно разложите данные по измерениям в хранилище. Сделайте это так же, как вы делали в первом спринте.

Данные новой схемы `external_source` должны оказаться в `dwh` в таблицах измерений и фактов.

Шаг 3. Напишите скрипт переноса данных из источника в хранилище

Загрузка данных из нового источника — операция не на один раз: какое-то время сайт всё ещё будет существовать, пока не будет выполнен ребрендинг. А это значит, что данные из источника нужно постоянно забирать и собирать в хранилище.

Данные должны попасть в схему `dwh` по тому же принципу, что и из схем `source1`, `source2`, `source3`. Дополните скрипт загрузки данных из старых источников новым источником.

Если нужно, изучите скрипт загрузки из старых источников.

Скрипт загрузки из старых источников

Скопировать кодSQL

```
/* создание таблицы tmp_sources с данными из всех источников */
DROP TABLE IF EXISTS tmp_sources;
CREATE TEMP TABLE tmp_sources AS
SELECT  order_id,
        order_created_date,
        order_completion_date,
        order_status,
        craftsman_id,
        craftsman_name,
        craftsman_address,
        craftsman_birthday,
        craftsman_email,
        product_id,
        product_name,
        product_description,
        product_type,
        product_price,
        customer_id,
        customer_name,
        customer_address,
        customer_birthday,
        customer_email
  FROM source1.craft_market_wide
UNION
SELECT  t2.order_id,
        t2.order_created_date,
        t2.order_completion_date,
        t2.order_status,
        t1.craftsman_id,
        t1.craftsman_name,
        t1.craftsman_address,
        t1.craftsman_birthday,
        t1.craftsman_email,
        t1.product_id,
        t1.product_name,
        t1.product_description,
```

```
        t1.product_type,
        t1.product_price,
        t2.customer_id,
        t2.customer_name,
        t2.customer_address,
        t2.customer_birthday,
        t2.customer_email
  FROM source2.craft_market_masters_products t1
    JOIN source2.craft_market_orders_customers t2 ON t2.product_id = t1.product_id
and t1.craftsman_id = t2.craftsman_id
UNION
SELECT  t1.order_id,
        t1.order_created_date,
        t1.order_completion_date,
        t1.order_status,
        t2.craftsman_id,
        t2.craftsman_name,
        t2.craftsman_address,
        t2.craftsman_birthday,
        t2.craftsman_email,
        t1.product_id,
        t1.product_name,
        t1.product_description,
        t1.product_type,
        t1.product_price,
        t3.customer_id,
        t3.customer_name,
        t3.customer_address,
        t3.customer_birthday,
        t3.customer_email
  FROM source3.craft_market_orders t1
    JOIN source3.craft_market_craftsmans t2 ON t1.craftsman_id = t2.craftsman_id
    JOIN source3.craft_market_customers t3 ON t1.customer_id = t3.customer_id;

/* обновление существующих записей и добавление новых в dwh.d_craftsmans */
MERGE INTO dwh.d_craftsman d
USING (SELECT DISTINCT craftsman_name, craftsman_address, craftsman_birthday,
craftsman_email FROM tmp_sources) t
ON d.craftsman_name = t.craftsman_name AND d.craftsman_email = t.craftsman_email
WHEN MATCHED THEN
  UPDATE SET craftsman_address = t.craftsman_address,
craftsman_birthday = t.craftsman_birthday, load_dttm = current_timestamp
WHEN NOT MATCHED THEN
  INSERT (craftsman_name, craftsman_address, craftsman_birthday, craftsman_email,
load_dttm)
  VALUES (t.craftsman_name, t.craftsman_address, t.craftsman_birthday,
t.craftsman_email, current_timestamp);
```

```
/* обновление существующих записей и добавление новых в dwh.d_products */
MERGE INTO dwh.d_product d
USING (SELECT DISTINCT product_name, product_description, product_type, product_price
from tmp_sources) t
ON d.product_name = t.product_name AND d.product_description = t.product_description
AND d.product_price = t.product_price
WHEN MATCHED THEN
  UPDATE SET product_type= t.product_type, load_dttm = current_timestamp
WHEN NOT MATCHED THEN
  INSERT (product_name, product_description, product_type, product_price, load_dttm)
  VALUES (t.product_name, t.product_description, t.product_type, t.product_price,
current_timestamp);

/* обновление существующих записей и добавление новых в dwh.d_customer */
MERGE INTO dwh.d_customer d
USING (SELECT DISTINCT customer_name, customer_address, customer_birthday,
customer_email from tmp_sources) t
ON d.customer_name = t.customer_name AND d.customer_email = t.customer_email
WHEN MATCHED THEN
  UPDATE SET customer_address= t.customer_address,
customer_birthday= t.customer_birthday, load_dttm = current_timestamp
WHEN NOT MATCHED THEN
  INSERT (customer_name, customer_address, customer_birthday, customer_email,
load_dttm)
  VALUES (t.customer_name, t.customer_address, t.customer_birthday, t.customer_email,
current_timestamp);

/* создание таблицы tmp_sources_fact */
DROP TABLE IF EXISTS tmp_sources_fact;
CREATE TEMP TABLE tmp_sources_fact AS
SELECT  dp.product_id,
        dc.craftsman_id,
        dcust.customer_id,
        src.order_created_date,
        src.order_completion_date,
        src.order_status,
        current_timestamp
FROM tmp_sources src
JOIN dwh.d_craftsman dc ON dc.craftsman_name = src.craftsman_name and
dc.craftsman_email = src.craftsman_email
JOIN dwh.d_customer dcust ON dcust.customer_name = src.customer_name and
dcust.customer_email = src.customer_email
JOIN dwh.d_product dp ON dp.product_name = src.product_name and
dp.product_description = src.product_description and dp.product_price =
src.product_price;

/* обновление существующих записей и добавление новых в dwh.f_order */
MERGE INTO dwh.f_order f
```

```
USING tmp_sources_fact t
ON f.product_id = t.product_id AND f.craftsman_id = t.craftsman_id AND f.customer_id
= t.customer_id AND f.order_created_date = t.order_created_date
WHEN MATCHED THEN
  UPDATE SET order_completion_date = t.order_completion_date, order_status =
t.order_status, load_dttm = current_timestamp
WHEN NOT MATCHED THEN
  INSERT (product_id, craftsman_id, customer_id, order_created_date,
order_completion_date, order_status, load_dttm)
  VALUES (t.product_id, t.craftsman_id, t.customer_id, t.order_created_date,
t.order_completion_date, t.order_status, current_timestamp);
```

Шаг 4. Изучите потребности бизнеса в новой витрине

Изучите постановку задачи от бизнес-аналитика и исследуйте хранилище на соответствие данных. Посмотрите, какие данные понадобятся для создания DDL новой витрины.

Бизнесу требуются следующие данные:

- идентификатор записи;
- идентификатор заказчика;
- Ф. И. О. заказчика;
- адрес заказчика;
- дата рождения заказчика;
- электронная почта заказчика;
- сумма, которую потратил заказчик;
- сумма, которую заработала платформа от покупок заказчика за месяц (10% от суммы, которую потратил заказчик);
- количество заказов у заказчика за месяц;
- средняя стоимость одного заказа у заказчика за месяц;
- медианное время в днях от момента создания заказа до его завершения за месяц;
- самая популярная категория товаров у этого заказчика за месяц;
- идентификатор самого популярного мастера ручной работы у заказчика. Если заказчик сделал одинаковое количество заказов у нескольких мастеров, возьмите любого;
- количество созданных заказов за месяц;
- количество заказов в процессе изготовки за месяц;
- количество заказов в доставке за месяц;
- количество завершённых заказов за месяц;
- количество незавершённых заказов за месяц;
- отчётный период, год и месяц.

Шаг 5. Напишите DDL новой витрины

Напишите DDL для витрины, определив типы, названия полей и комментарии. Учитывайте, что витрина должна быть инкрементальной: понадобится дополнительная таблица с датой загрузки

данных.

Витрину назовите `dwh.customer_report_datamart`. Возьмите за основу витрину по мастерам ручной работы. Новая витрина и витрина по мастерам будут во многом схожи.

Шаг 6. Напишите скрипт для инкрементального обновления витрины

Напишите скрипт, который будет выполнять расчёт для новой витрины. Это инкрементальная витрина, значит, и расчёт должен быть инкрементальным. За основу можно взять скрипт обновления инкрементальной витрины по мастерам.

Скрипт обновления витрины по мастерам

Скопировать кодSQL

```
-- DDL витрины данных
DROP TABLE IF EXISTS dwh.craftsman_report_datamart;

CREATE TABLE IF NOT EXISTS dwh.craftsman_report_datamart (
    id BIGINT GENERATED ALWAYS AS IDENTITY NOT NULL, -- идентификатор записи
    craftsman_id BIGINT NOT NULL, -- идентификатор мастера
    craftsman_name VARCHAR NOT NULL, -- Ф.И.О. мастера
    craftsman_address VARCHAR NOT NULL, -- адрес мастера
    craftsman_birthday DATE NOT NULL, -- дата рождения мастера
    craftsman_email VARCHAR NOT NULL, -- электронная почта мастера
    craftsman_money NUMERIC(15,2) NOT NULL, -- сумма, которую заработал мастер (-10%
на платформы) за месяц
    platform_money BIGINT NOT NULL, -- сумма, которую заработала платформа от продаж
мастера за месяц
    count_order BIGINT NOT NULL, -- количество заказов у мастера за месяц
    avg_price_order NUMERIC(10,2) NOT NULL, -- средняя стоимость одного заказа у
мастера за месяц
    avg_age_customer NUMERIC(3,1) NOT NULL, -- средний возраст покупателей
    median_time_order_completed NUMERIC(10,1), -- медианное время в днях от момента
создания заказа до его завершения за месяц
    top_product_category VARCHAR NOT NULL, -- самая популярная категория товаров у
этого мастера за месяц
    count_order_created BIGINT NOT NULL, -- количество созданных заказов за месяц
    count_order_in_progress BIGINT NOT NULL, -- количество заказов в процессе
изготовки за месяц
    count_order_delivery BIGINT NOT NULL, -- количество заказов в доставке за месяц
    count_order_done BIGINT NOT NULL, -- количество завершённых заказов за месяц
    count_order_not_done BIGINT NOT NULL, -- количество незавершённых заказов за
месяц
    report_period VARCHAR NOT NULL, -- отчётный период год и месяц
    CONSTRAINT craftsman_report_datamart_pk PRIMARY KEY (id)
);
```

```
-- DDL таблицы инкрементальных загрузок
DROP TABLE IF EXISTS dwh.load_dates_craftsman_report_datamart;

CREATE TABLE IF NOT EXISTS dwh.load_dates_craftsman_report_datamart (
    id BIGINT GENERATED ALWAYS AS IDENTITY NOT NULL,
    load_dttm DATE NOT NULL,
    CONSTRAINT load_dates_craftsman_report_datamart_pk PRIMARY KEY (id)
);

WITH
dwh_delta AS ( -- определяем, какие данные были изменены в витрине или добавлены в
DWH. Формируем дельту изменений
    SELECT
            dc.craftsman_id AS craftsman_id,
            dc.craftsman_name AS craftsman_name,
            dc.craftsman_address AS craftsman_address,
            dc.craftsman_birthday AS craftsman_birthday,
            dc.craftsman_email AS craftsman_email,
            fo.order_id AS order_id,
            dp.product_id AS product_id,
            dp.product_price AS product_price,
            dp.product_type AS product_type,
            DATE_PART('year', AGE(dcs.customer_birthday)) AS customer_age,
            fo.order_completion_date - fo.order_created_date AS diff_order_date,
            fo.order_status AS order_status,
            TO_CHAR(fo.order_created_date, 'yyyy-mm') AS report_period,
            crd.craftsman_id AS exist_craftsman_id,
            dc.load_dttm AS craftsman_load_dttm,
            dcs.load_dttm AS customers_load_dttm,
            dp.load_dttm AS products_load_dttm
            FROM dwh.f_order fo
                INNER JOIN dwh.d_craftsman dc ON fo.craftsman_id = dc.craftsman_id
                INNER JOIN dwh.d_customer dcs ON fo.customer_id = dcs.customer_id
                INNER JOIN dwh.d_product dp ON fo.product_id = dp.product_id
                LEFT JOIN dwh.craftsman_report_datamart crd ON dc.craftsman_id =
crd.craftsman_id
                    WHERE (fo.load_dttm > (SELECT COALESCE(MAX(load_dttm),'1900-01-
01') FROM dwh.load_dates_craftsman_report_datamart)) OR
                            (dc.load_dttm > (SELECT COALESCE(MAX(load_dttm),'1900-01-
01') FROM dwh.load_dates_craftsman_report_datamart)) OR
                            (dcs.load_dttm > (SELECT COALESCE(MAX(load_dttm),'1900-
01-01') FROM dwh.load_dates_craftsman_report_datamart)) OR
                            (dp.load_dttm > (SELECT COALESCE(MAX(load_dttm),'1900-01-
01') FROM dwh.load_dates_craftsman_report_datamart))
),
dwh_update_delta AS ( -- делаем выборку мастеров ручной работы, по которым были
изменения в DWH. По этим мастерам данные в витрине нужно будет обновить
    SELECT
```

```
            dd.exist_craftsman_id AS craftsman_id
            FROM dwh_delta dd
                WHERE dd.exist_craftsman_id IS NOT NULL
),
dwh_delta_insert_result AS ( -- делаем расчёт витрины по новым данным. Этой
информации по мастерам в рамках расчётного периода раньше не было, это новые данные.
Их можно просто вставить (insert) в витрину без обновления
    SELECT
            T4.craftsman_id AS craftsman_id,
            T4.craftsman_name AS craftsman_name,
            T4.craftsman_address AS craftsman_address,
            T4.craftsman_birthday AS craftsman_birthday,
            T4.craftsman_email AS craftsman_email,
            T4.craftsman_money AS craftsman_money,
            T4.platform_money AS platform_money,
            T4.count_order AS count_order,
            T4.avg_price_order AS avg_price_order,
            T4.avg_age_customer AS avg_age_customer,
            T4.product_type AS top_product_category,
            T4.median_time_order_completed AS median_time_order_completed,
            T4.count_order_created AS count_order_created,
            T4.count_order_in_progress AS count_order_in_progress,
            T4.count_order_delivery AS count_order_delivery,
            T4.count_order_done AS count_order_done,
            T4.count_order_not_done AS count_order_not_done,
            T4.report_period AS report_period
            FROM (
                SELECT     -- в этой выборке объединяем две внутренние выборки по
расчёту столбцов витрины и применяем оконную функцию для определения самой популярной
категории товаров
                        *,
                        RANK() OVER(PARTITION BY T2.craftsman_id ORDER BY
count_product DESC) AS rank_count_product
                        FROM (
                            SELECT -- в этой выборке делаем расчёт по большинству
столбцов, так как все они требуют одной и той же группировки, кроме столбца с самой
популярной категорией товаров у мастера. Для этого столбца сделаем отдельную выборку
с другой группировкой и выполним JOIN
                                T1.craftsman_id AS craftsman_id,
                                T1.craftsman_name AS craftsman_name,
                                T1.craftsman_address AS craftsman_address,
                                T1.craftsman_birthday AS craftsman_birthday,
                                T1.craftsman_email AS craftsman_email,
                                SUM(T1.product_price) - (SUM(T1.product_price) * 0.1)
AS craftsman_money,
                                SUM(T1.product_price) * 0.1 AS platform_money,
                                COUNT(order_id) AS count_order,
                                AVG(T1.product_price) AS avg_price_order,
```

```
                                AVG(T1.customer_age) AS avg_age_customer,
                                PERCENTILE_CONT(0.5) WITHIN GROUP(ORDER BY
diff_order_date) AS median_time_order_completed,
                                SUM(CASE WHEN T1.order_status = 'created' THEN 1 ELSE
0 END) AS count_order_created,
                                SUM(CASE WHEN T1.order_status = 'in progress' THEN 1
ELSE 0 END) AS count_order_in_progress,
                                SUM(CASE WHEN T1.order_status = 'delivery' THEN 1
ELSE 0 END) AS count_order_delivery,
                                SUM(CASE WHEN T1.order_status = 'done' THEN 1 ELSE 0
END) AS count_order_done,
                                SUM(CASE WHEN T1.order_status != 'done' THEN 1 ELSE 0
END) AS count_order_not_done,
                                T1.report_period AS report_period
                                FROM dwh_delta AS T1
                                    WHERE T1.exist_craftsman_id IS NULL
                                        GROUP BY T1.craftsman_id, T1.craftsman_name,
T1.craftsman_address, T1.craftsman_birthday, T1.craftsman_email, T1.report_period
                            ) AS T2
                                INNER JOIN (
                                    SELECT     -- Эта выборка поможет определить
самый популярный товар у мастера ручной работы. Эта выборка не делается в предыдущем
запросе, так как нужна другая группировка. Для данных этой выборки можно применить
оконную функцию, которая и покажет самую популярную категорию товаров у мастера
                                            dd.craftsman_id AS
craftsman_id_for_product_type,
                                            dd.product_type,
                                            COUNT(dd.product_id) AS count_product
                                            FROM dwh_delta AS dd
                                                GROUP BY dd.craftsman_id,
dd.product_type
                                                    ORDER BY count_product DESC) AS
T3 ON T2.craftsman_id = T3.craftsman_id_for_product_type
                ) AS T4 WHERE T4.rank_count_product = 1 ORDER BY report_period --
условие помогает оставить в выборке первую по популярности категорию товаров
),
dwh_delta_update_result AS ( -- делаем перерасчёт для существующих записей витрины,
так как данные обновились за отчётные периоды. Логика похожа на insert, но нужно
достать конкретные данные из DWH
    SELECT
            T4.craftsman_id AS craftsman_id,
            T4.craftsman_name AS craftsman_name,
            T4.craftsman_address AS craftsman_address,
            T4.craftsman_birthday AS craftsman_birthday,
            T4.craftsman_email AS craftsman_email,
            T4.craftsman_money AS craftsman_money,
            T4.platform_money AS platform_money,
            T4.count_order AS count_order,
```

```
            T4.avg_price_order AS avg_price_order,
            T4.avg_age_customer AS avg_age_customer,
            T4.product_type AS top_product_category,
            T4.median_time_order_completed AS median_time_order_completed,
            T4.count_order_created AS count_order_created,
            T4.count_order_in_progress AS count_order_in_progress,
            T4.count_order_delivery AS count_order_delivery,
            T4.count_order_done AS count_order_done,
            T4.count_order_not_done AS count_order_not_done,
            T4.report_period AS report_period
            FROM (
                SELECT     -- в этой выборке объединяем две внутренние выборки по расчёту столбцов витрины и применяем оконную функцию для определения самой популярной категории товаров
                        *,
                        RANK() OVER(PARTITION BY T2.craftsman_id ORDER BY count_product DESC) AS rank_count_product
                        FROM (
                            SELECT -- в этой выборке делаем расчёт по большинству столбцов, так как все они требуют одной и той же группировки, кроме столбца с самой популярной категорией товаров у мастера. Для этого столбца сделаем отдельную выборку с другой группировкой и выполним JOIN
                                T1.craftsman_id AS craftsman_id,
                                T1.craftsman_name AS craftsman_name,
                                T1.craftsman_address AS craftsman_address,
                                T1.craftsman_birthday AS craftsman_birthday,
                                T1.craftsman_email AS craftsman_email,
                                SUM(T1.product_price) - (SUM(T1.product_price) * 0.1) AS craftsman_money,
                                SUM(T1.product_price) * 0.1 AS platform_money,
                                COUNT(order_id) AS count_order,
                                AVG(T1.product_price) AS avg_price_order,
                                AVG(T1.customer_age) AS avg_age_customer,
                                PERCENTILE_CONT(0.5) WITHIN GROUP(ORDER BY diff_order_date) AS median_time_order_completed,
                                SUM(CASE WHEN T1.order_status = 'created' THEN 1 ELSE 0 END) AS count_order_created,
                                SUM(CASE WHEN T1.order_status = 'in progress' THEN 1 ELSE 0 END) AS count_order_in_progress,
                                SUM(CASE WHEN T1.order_status = 'delivery' THEN 1 ELSE 0 END) AS count_order_delivery,
                                SUM(CASE WHEN T1.order_status = 'done' THEN 1 ELSE 0 END) AS count_order_done,
                                SUM(CASE WHEN T1.order_status != 'done' THEN 1 ELSE 0 END) AS count_order_not_done,
                                T1.report_period AS report_period
                                FROM (
                                    SELECT     -- в этой выборке достаём из DWH
```

```
обновлённые или новые данные по мастерам, которые уже есть в витрине
                                        dc.craftsman_id AS craftsman_id,
                                        dc.craftsman_name AS craftsman_name,
                                        dc.craftsman_address AS
craftsman_address,
                                        dc.craftsman_birthday AS
craftsman_birthday,
                                        dc.craftsman_email AS craftsman_email,
                                        fo.order_id AS order_id,
                                        dp.product_id AS product_id,
                                        dp.product_price AS product_price,
                                        dp.product_type AS product_type,
                                        DATE_PART('year',
AGE(dcs.customer_birthday)) AS customer_age,
                                        fo.order_completion_date -
fo.order_created_date AS diff_order_date,
                                        fo.order_status AS order_status,
                                        TO_CHAR(fo.order_created_date, 'yyyy-mm')
AS report_period
                                        FROM dwh.f_order fo
                                            INNER JOIN dwh.d_craftsman dc ON
fo.craftsman_id = dc.craftsman_id
                                            INNER JOIN dwh.d_customer dcs ON
fo.customer_id = dcs.customer_id
                                            INNER JOIN dwh.d_product dp ON
fo.product_id = dp.product_id
                                            INNER JOIN dwh_update_delta ud ON
fo.craftsman_id = ud.craftsman_id
                                ) AS T1
                                    GROUP BY T1.craftsman_id, T1.craftsman_name,
T1.craftsman_address, T1.craftsman_birthday, T1.craftsman_email, T1.report_period
                            ) AS T2
                                INNER JOIN (
                                    SELECT     -- Эта выборка поможет определить
самый популярный товар у мастера. Эта выборка не делается в предыдущем запросе, так
как нужна другая группировка. Для данных этой выборки можно применить оконную
функцию, которая и покажет самую популярную категорию товаров у мастера
                                        dd.craftsman_id AS
craftsman_id_for_product_type,
                                        dd.product_type,
                                        COUNT(dd.product_id) AS count_product
                                        FROM dwh_delta AS dd
                                            GROUP BY dd.craftsman_id,
dd.product_type
                                                ORDER BY count_product DESC) AS
T3 ON T2.craftsman_id = T3.craftsman_id_for_product_type
                ) AS T4 WHERE T4.rank_count_product = 1 ORDER BY report_period
),
```

```
insert_delta AS ( -- выполняем insert новых расчитанных данных для витрины
    INSERT INTO dwh.craftsman_report_datamart (
        craftsman_id,
        craftsman_name,
        craftsman_address,
        craftsman_birthday,
        craftsman_email,
        craftsman_money,
        platform_money,
        count_order,
        avg_price_order,
        avg_age_customer,
        median_time_order_completed,
        top_product_category,
        count_order_created,
        count_order_in_progress,
        count_order_delivery,
        count_order_done,
        count_order_not_done,
        report_period
    ) SELECT
            craftsman_id,
            craftsman_name,
            craftsman_address,
            craftsman_birthday,
            craftsman_email,
            craftsman_money,
            platform_money,
            count_order,
            avg_price_order,
            avg_age_customer,
            median_time_order_completed,
            top_product_category,
            count_order_created,
            count_order_in_progress,
            count_order_delivery,
            count_order_done,
            count_order_not_done,
            report_period
            FROM dwh_delta_insert_result
),
update_delta AS ( -- выполняем обновление показателей в отчёте по уже существующим
мастерам
    UPDATE dwh.craftsman_report_datamart SET
        craftsman_name = updates.craftsman_name,
        craftsman_address = updates.craftsman_address,
        craftsman_birthday = updates.craftsman_birthday,
        craftsman_email = updates.craftsman_email,
```

```
        craftsman_money = updates.craftsman_money,
        platform_money = updates.platform_money,
        count_order = updates.count_order,
        avg_price_order = updates.avg_price_order,
        avg_age_customer = updates.avg_age_customer,
        median_time_order_completed = updates.median_time_order_completed,
        top_product_category = updates.top_product_category,
        count_order_created = updates.count_order_created,
        count_order_in_progress = updates.count_order_in_progress,
        count_order_delivery = updates.count_order_delivery,
        count_order_done = updates.count_order_done,
        count_order_not_done = updates.count_order_not_done,
        report_period = updates.report_period
    FROM (
        SELECT
            craftsman_id,
            craftsman_name,
            craftsman_address,
            craftsman_birthday,
            craftsman_email,
            craftsman_money,
            platform_money,
            count_order,
            avg_price_order,
            avg_age_customer,
            median_time_order_completed,
            top_product_category,
            count_order_created,
            count_order_in_progress,
            count_order_delivery,
            count_order_done,
            count_order_not_done,
            report_period
            FROM dwh_delta_update_result) AS updates
    WHERE dwh.craftsman_report_datamart.craftsman_id = updates.craftsman_id
),
insert_load_date AS ( -- делаем запись в таблицу загрузок о том, когда была совершена
загрузка, чтобы в следующий раз взять данные, которые будут добавлены или изменены
после этой даты
    INSERT INTO dwh.load_dates_craftsman_report_datamart (
        load_dttm
    )
    SELECT GREATEST(COALESCE(MAX(craftsman_load_dttm), NOW()),
                    COALESCE(MAX(customers_load_dttm), NOW()),
                    COALESCE(MAX(products_load_dttm), NOW()))
        FROM dwh_delta
```

```
)
SELECT 'increment datamart'; -- инициализируем запрос CTE
```

Чтобы получить идентификатор самого популярного мастера у заказчика, можно попробовать написать ещё один `INNER JOIN` по аналогии с определением самой популярной категории товаров. Самый популярный мастер определяется по количеству заказов от заказчика. Если заказчик сделал одинаковое количество заказов у нескольких мастеров, возьмите идентификатор любого. Для этого воспользуйтесь командами `DISTINCT ON (COLUMN)` или `FIRST_VALUE`. Или же можно вместо оконной функции `RANK()` воспользоваться оконной функцией `ROW_NUMBER()`.

Как будут проверять мой проект

На что обращают внимание при проверке проектов:

- логично ли написаны SQL-запросы;
- выбраны ли эффективные типы данных в DDL;
- нет ли аномалий, дублей или пропусков в данных.