

ДЕПАРТАМЕНТ ОБРАЗОВАНИЯ И НАУКИ ГОРОДА МОСКВЫ

Государственное автономное образовательное учреждение

высшего образования города Москвы

«Московский городской педагогический университет»

(ГАОУ ВО МГПУ)

Институт цифрового образования

Департамент информатики, управления и технологий

Практическая(лабораторная) работа № 2.2

по дисциплине «Платформы Data Engineering»

Выполнил:

студент группы БД-251м

Направление подготовки/Специальность

38.04.05 - Бизнес-информатика

Бобылева Варвара Владимировна

(Ф.И.О.)

Проверил:

Кандидат технических наук

(ученая степень, звание)

Босенко Тимур Муртазович

(Ф.И.О.)

Москва 2025

--

-- Автор: Бобылева Варвара

-- Группа: БД-251м

-- Проект: superstore_dwh

-- Вариант индивидуального задания: 2 Когортный анализ клиентов

--

Слой Intermediate (Промежуточный): Этот слой объединяет и преобразует данные из начального (stg) слоя в более сложные, осмысленные модели. Его цель — создание промежуточных, переиспользуемых представлений данных, которые служат "строительными блоками" для конечных витрин данных. Модели в этом слое не предназначены для прямого использования конечными пользователями, но являются ключевым этапом в подготовке данных для анализа. В данном случае, `int_sales_orders` является примером промежуточной модели.

Слой Marts (Витрины данных): Этот слой, также называемый презентационным или отчетным, содержит готовые к использованию данные, оптимизированные для бизнес-аналитики (BI) и отчетности. Витрины данных ориентированы на конкретные бизнес-направления или отделы (например, продажи или маркетинг). Данные в этом слое легко доступны для конечных пользователей, которые могут использовать их для анализа, создания дашбордов и принятия решений. Примерами витрин данных на графике являются `mart_customer_itv` и `mart_monthly_sales`.

1. Архитектура DWH

На рисунке ниже отображена схема Lineage Graph проекта superstore:

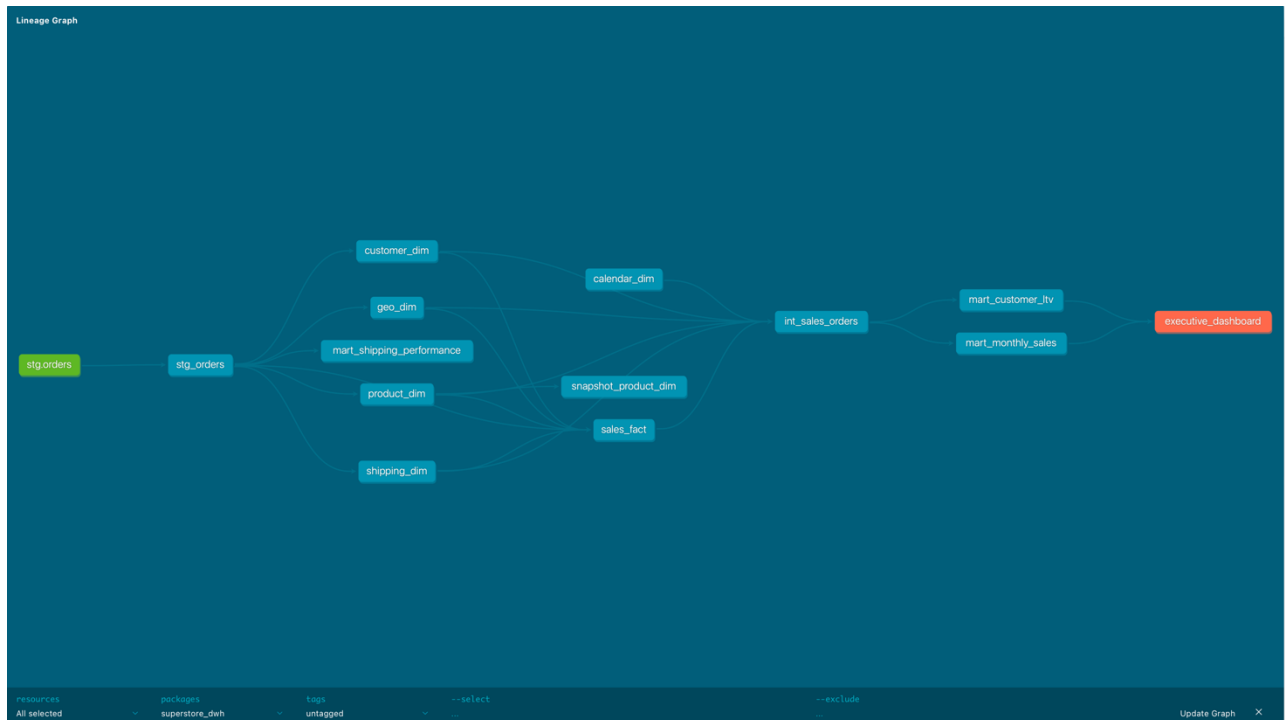


Рисунок 1 - Lineage Graph проекта superstore

На рисунке ниже отображена схема Lineage Graph проекта с учетом индивидуального задания:

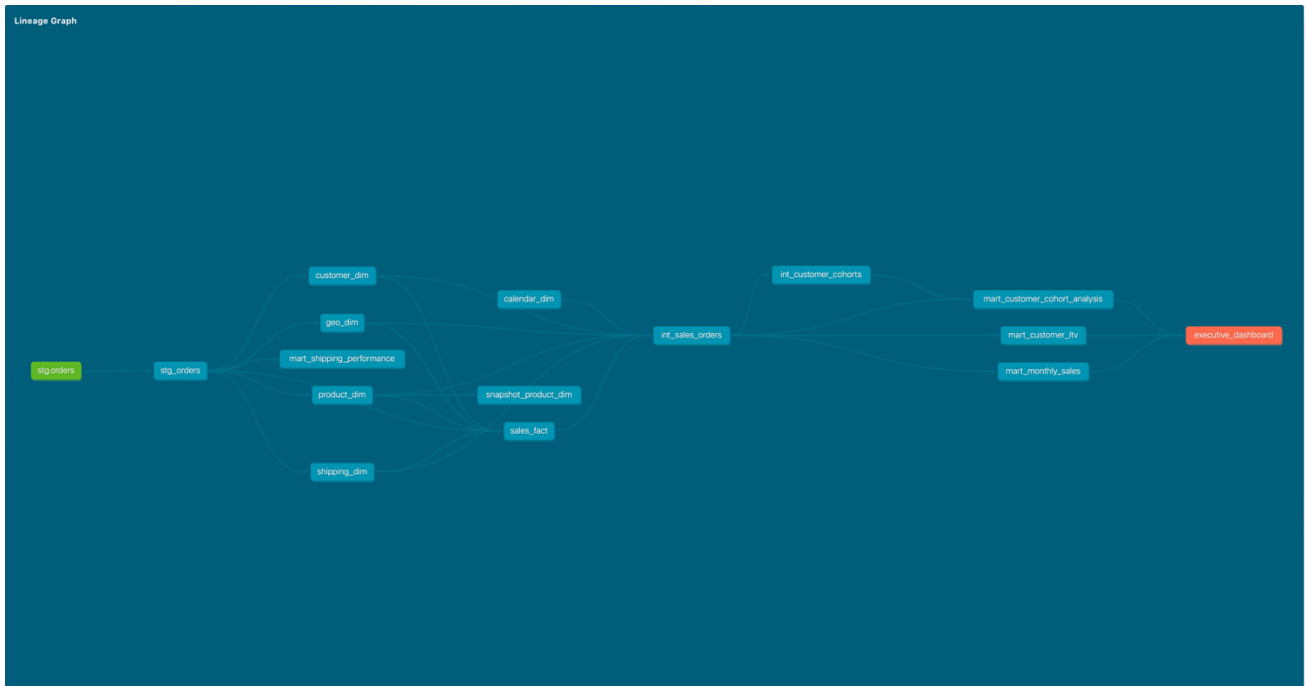


Рисунок 2 - Lineage Graph проекта с учетом индивидуального задания

2. Ключевые фрагменты кода.

Код модели int_sales_orders.sql

```
-- models/intermediate/int_sales_orders.sql
-- Эта модель объединяет факты со всеми измерениями, создавая
-- широкую, денормализованную таблицу для легкого использования в витринах.
```

SELECT

-- Ключи

f.order_id,

-- Измерения из customer_dim

c.customer_id,

c.customer_name,

-- Измерения из product_dim

p.product_id,

p.product_name,

p.category,

p.sub_category,

p.segment,

-- Измерения из geo_dim

g.city,

g.state,

-- Измерения из shipping_dim

s.ship_mode,

-- Даты из calendar_dim (с правильными псевдонимами)

cal_order.date as order_date,

cal_ship.date as ship_date,

-- Метрики из sales_fact

```

    f.sales,
    f.profit,
    f.quantity,
    f.discount

FROM {{ ref('sales_fact') }} AS f
LEFT JOIN {{ ref('customer_dim') }} AS c ON f.cust_id = c.cust_id
LEFT JOIN {{ ref('product_dim') }} AS p ON f.prod_id = p.prod_id
LEFT JOIN {{ ref('shipping_dim') }} AS s ON f.ship_id = s.ship_id
LEFT JOIN {{ ref('geo_dim') }} AS g ON f.geo_id = g.geo_id
-- ИСПРАВЛЕНО: Добавляем псевдонимы, так как календарь используется дважды
LEFT JOIN {{ ref('calendar_dim') }} AS cal_order ON f.order_date_id = cal_order.dateid
LEFT JOIN {{ ref('calendar_dim') }} AS cal_ship ON f.ship_date_id = cal_ship.dateid

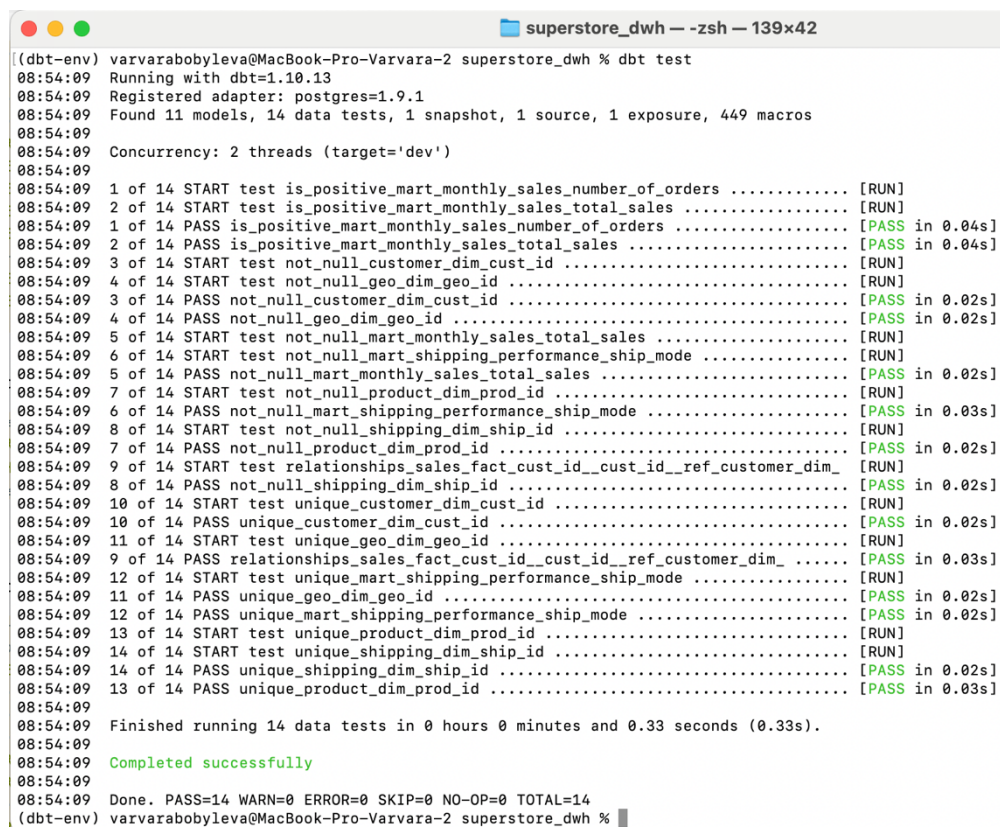
```

Код кастомного теста test_is_positive.sql:

```

{% test is_positive(model, column_name) %} SELECT *
FROM {{ model }}
WHERE {{ column_name }} < 0
{% endtest %}

```



```

(dbt-env) varvarabobyleva@MacBook-Pro-Varvara-2 superstore_dwh % dbt test
08:54:09 Running with dbt=1.10.13
08:54:09 Registered adapter: postgres=1.9.1
08:54:09 Found 11 models, 14 data tests, 1 snapshot, 1 source, 1 exposure, 449 macros
08:54:09 Concurrency: 2 threads (target='dev')
08:54:09
08:54:09 1 of 14 START test is_positive_mart_monthly_sales_number_of_orders ..... [RUN]
08:54:09 2 of 14 START test is_positive_mart_monthly_sales_total_sales ..... [RUN]
08:54:09 1 of 14 PASS is_positive_mart_monthly_sales_number_of_orders ..... [PASS in 0.04s]
08:54:09 2 of 14 PASS is_positive_mart_monthly_sales_total_sales ..... [PASS in 0.04s]
08:54:09 3 of 14 START test not_null_customer_dim_cust_id ..... [RUN]
08:54:09 4 of 14 START test not_null_geo_dim_geo_id ..... [RUN]
08:54:09 3 of 14 PASS not_null_customer_dim_cust_id ..... [PASS in 0.02s]
08:54:09 4 of 14 PASS not_null_geo_dim_geo_id ..... [PASS in 0.02s]
08:54:09 5 of 14 START test not_null_mart_monthly_sales_total_sales ..... [RUN]
08:54:09 6 of 14 START test not_null_mart_shipping_performance_ship_mode ..... [RUN]
08:54:09 5 of 14 PASS not_null_mart_monthly_sales_total_sales ..... [PASS in 0.02s]
08:54:09 7 of 14 START test not_null_product_dim_prod_id ..... [RUN]
08:54:09 6 of 14 PASS not_null_mart_shipping_performance_ship_mode ..... [PASS in 0.03s]
08:54:09 8 of 14 START test not_null_shipping_dim_ship_id ..... [RUN]
08:54:09 7 of 14 PASS not_null_product_dim_prod_id ..... [PASS in 0.02s]
08:54:09 9 of 14 START test relationships_sales_fact_cust_id__cust_id_ref_customer_dim_ ..... [RUN]
08:54:09 8 of 14 PASS not_null_shipping_dim_ship_id ..... [PASS in 0.02s]
08:54:09 10 of 14 START test unique_customer_dim_cust_id ..... [RUN]
08:54:09 11 of 14 START test unique_geo_dim_geo_id ..... [RUN]
08:54:09 9 of 14 PASS relationships_sales_fact_cust_id__cust_id_ref_customer_dim_ ..... [PASS in 0.03s]
08:54:09 12 of 14 START test unique_mart_shipping_performance_ship_mode ..... [RUN]
08:54:09 11 of 14 PASS unique_geo_dim_geo_id ..... [PASS in 0.02s]
08:54:09 12 of 14 PASS unique_mart_shipping_performance_ship_mode ..... [PASS in 0.02s]
08:54:09 13 of 14 START test unique_product_dim_prod_id ..... [RUN]
08:54:09 14 of 14 START test unique_shipping_dim_ship_id ..... [RUN]
08:54:09 13 of 14 PASS unique_product_dim_prod_id ..... [PASS in 0.03s]
08:54:09 14 of 14 PASS unique_shipping_dim_ship_id ..... [PASS in 0.02s]
08:54:09 Finished running 14 data tests in 0 hours 0 minutes and 0.33 seconds (0.33s).
08:54:09 Completed successfully
08:54:09 Done. PASS=14 WARN=0 ERROR=0 SKIP=0 NO-OP=0 TOTAL=14
(dbt-env) varvarabobyleva@MacBook-Pro-Varvara-2 superstore_dwh %

```

Рисунок 3 – Проведение тестирования

Код модели snapshot_product_dim.sql:

```

{% snapshot snapshot_product_dim %} {{
config(
target_schema='dw_snapshots', strategy='check', unique_key='prod_id',
check_cols=['segment', 'category'],
) }}
SELECT prod_id, product_id, segment, category FROM {{ ref('product_dim') }}
{% endsnapshot %}

```

Выполнение индивидуального задания:

Для решения этой задачи нужно создать две модели: одну промежуточную (intermediate), чтобы определить когорту каждого клиента, и одну mart-модель, которая будет использовать эту промежуточную модель для расчета совокупных продаж.

1. Промежуточная модель: int_customer_cohorts.sql

Эта модель будет определять месяц первой покупки для каждого клиента. Она относится к промежуточному слою, так как эту информацию можно будет повторно использовать для других видов когортного анализа.

Файл: models/intermediate/int_customer_cohorts.sql

```
with customer_first_purchase as (  
  select  
    customer_id,  
    min(date_trunc('month', order_date))::date as cohort_month  
  from {{ ref('int_sales_orders') }}  
  group by 1  
)  
  
select * from customer_first_purchase
```

2. Mart-модель: mart_customer_cohort_sales.sql

Эта модель будет использовать int_customer_cohorts для расчета совокупных продаж за первые 3 месяца для каждой когорты.

Файл: models/marts/mart_customer_cohort_sales.sql

```
with orders as (  
  select  
    customer_id,  
    order_date,  
    sales  
  from {{ ref('int_sales_orders') }}  
) ,  
  
customers as (  
  select * from {{ ref('int_customer_cohorts') }}  
) ,  
  
orders_with_cohort as (  
  select  
    o.customer_id,  
    o.order_date,  
    o.sales,  
    c.cohort_month,  
    (extract(year from o.order_date) - extract(year from c.cohort_month)) * 12 +  
    (extract(month from o.order_date) - extract(month from c.cohort_month)) as  
months_since_first_purchase  
  from orders as o  
  join customers as c  
    on o.customer_id = c.customer_id  
) ,  
  
three_month_sales as (  
  select  
    cohort_month,
```

```

        sum(sales) as cumulative_sales_3_months
    from orders_with_cohort
    where months_since_first_purchase between 0 and 2
    group by 1
)

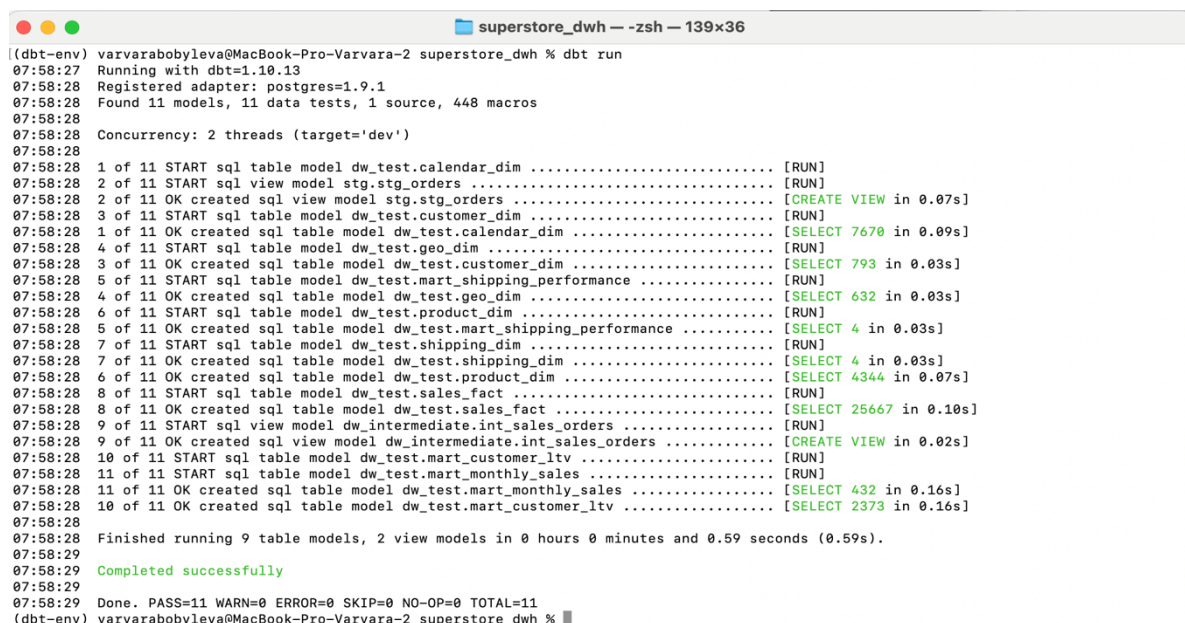
select * from three_month_sales
order by cohort_month

```

Эта модель сначала объединяет данные о заказах с когортой клиента, затем вычисляет, сколько месяцев прошло с первой покупки (*months_since_first_purchase*), фильтрует данные, оставляя только первые 3 месяца (от 0 до 2), и наконец, агрегирует общие продажи по месяцу когорты.

3. Результаты.

Результат выполнения команды `dbt run` для проекта `superstore_dwh`:



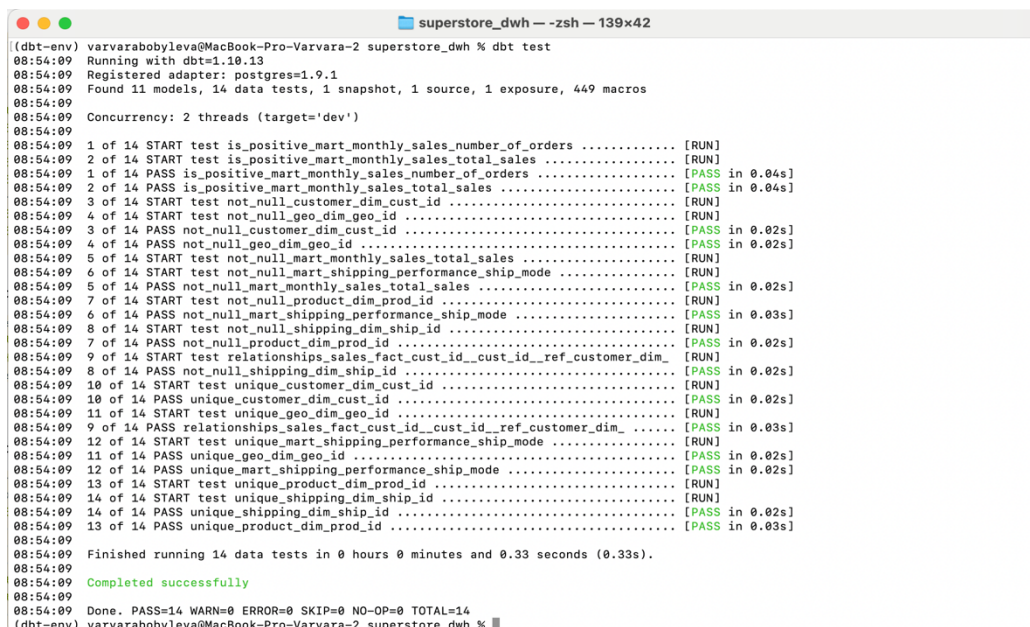
```

superstore_dwh -- -zsh -- 139x36
(dbt-env) varvarabobyleva@MacBook-Pro-Varvara-2 superstore_dwh % dbt run
07:58:27 Running with dbt=1.10.13
07:58:28 Registered adapter: postgres=1.9.1
07:58:28 Found 11 models, 11 data tests, 1 source, 448 macros
07:58:28
07:58:28 Concurrency: 2 threads (target='dev')
07:58:28
07:58:28 1 of 11 START sql table model dw_test.calendar_dim ..... [RUN]
07:58:28 2 of 11 START sql view model stg.stg_orders ..... [RUN]
07:58:28 2 of 11 OK created sql view model stg.stg_orders ..... [CREATE VIEW in 0.07s]
07:58:28 3 of 11 START sql table model dw_test.customer_dim ..... [RUN]
07:58:28 1 of 11 OK created sql table model dw_test.calendar_dim ..... [SELECT 7670 in 0.09s]
07:58:28 4 of 11 START sql table model dw_test.geo_dim ..... [RUN]
07:58:28 3 of 11 OK created sql table model dw_test.customer_dim ..... [SELECT 793 in 0.03s]
07:58:28 5 of 11 START sql table model dw_test.mart_shipping_performance ..... [RUN]
07:58:28 4 of 11 OK created sql table model dw_test.geo_dim ..... [SELECT 632 in 0.03s]
07:58:28 6 of 11 START sql table model dw_test.product_dim ..... [RUN]
07:58:28 5 of 11 OK created sql table model dw_test.mart_shipping_performance ..... [SELECT 4 in 0.03s]
07:58:28 7 of 11 START sql table model dw_test.shipping_dim ..... [RUN]
07:58:28 6 of 11 OK created sql table model dw_test.shipping_dim ..... [SELECT 4 in 0.03s]
07:58:28 8 of 11 START sql table model dw_test.product_dim ..... [SELECT 4344 in 0.07s]
07:58:28 9 of 11 START sql table model dw_test.sales_fact ..... [RUN]
07:58:28 8 of 11 OK created sql table model dw_test.sales_fact ..... [SELECT 25667 in 0.10s]
07:58:28 9 of 11 START sql view model dw_intermediate.int_sales_orders ..... [RUN]
07:58:28 10 of 11 OK created sql view model dw_intermediate.int_sales_orders ..... [CREATE VIEW in 0.02s]
07:58:28 10 of 11 START sql table model dw_test.mart_customer_ltv ..... [RUN]
07:58:28 11 of 11 START sql table model dw_test.mart_monthly_sales ..... [RUN]
07:58:28 11 of 11 OK created sql table model dw_test.mart_monthly_sales ..... [SELECT 432 in 0.16s]
07:58:28 10 of 11 OK created sql table model dw_test.mart_customer_ltv ..... [SELECT 2373 in 0.16s]
07:58:28
07:58:28 Finished running 9 table models, 2 view models in 0 hours 0 minutes and 0.59 seconds (0.59s).
07:58:29
07:58:29 Completed successfully
07:58:29
07:58:29 Done. PASS=11 WARN=0 ERROR=0 SKIP=0 NO-OP=0 TOTAL=11
(dbt-env) varvarabobyleva@MacBook-Pro-Varvara-2 superstore_dwh %

```

Рисунок 4 – Выполнение команды `dbt run`

Результат выполнения команды `dbt test` для проекта `superstore_dwh`:



```

superstore_dwh -- -zsh -- 139x42
(dbt-env) varvarabobyleva@MacBook-Pro-Varvara-2 superstore_dwh % dbt test
08:54:09 Running with dbt=1.10.13
08:54:09 Registered adapter: postgres=1.9.1
08:54:09 Found 11 models, 14 data tests, 1 snapshot, 1 source, 1 exposure, 449 macros
08:54:09
08:54:09 Concurrency: 2 threads (target='dev')
08:54:09
08:54:09 1 of 14 START test is_positive_mart_monthly_sales_number_of_orders ..... [RUN]
08:54:09 2 of 14 START test is_positive_mart_monthly_sales_total_sales ..... [RUN]
08:54:09 1 of 14 PASS is_positive_mart_monthly_sales_number_of_orders ..... [PASS in 0.04s]
08:54:09 2 of 14 PASS is_positive_mart_monthly_sales_total_sales ..... [PASS in 0.04s]
08:54:09 3 of 14 START test not_null_customer_dim_cust_id ..... [RUN]
08:54:09 4 of 14 START test not_null_geo_dim_geo_id ..... [RUN]
08:54:09 3 of 14 PASS not_null_customer_dim_cust_id ..... [PASS in 0.02s]
08:54:09 4 of 14 PASS not_null_geo_dim_geo_id ..... [PASS in 0.02s]
08:54:09 5 of 14 START test not_null_mart_monthly_sales_total_sales ..... [RUN]
08:54:09 6 of 14 START test not_null_mart_shipping_performance_ship_mode ..... [RUN]
08:54:09 5 of 14 PASS not_null_mart_monthly_sales_total_sales ..... [PASS in 0.02s]
08:54:09 7 of 14 START test not_null_product_dim_prod_id ..... [RUN]
08:54:09 6 of 14 PASS not_null_mart_shipping_performance_ship_mode ..... [PASS in 0.03s]
08:54:09 8 of 14 START test not_null_shipping_dim_ship_id ..... [RUN]
08:54:09 7 of 14 PASS not_null_product_dim_prod_id ..... [PASS in 0.02s]
08:54:09 9 of 14 START test relationships_sales_fact_cust_id_cust_id_ref_customer_dim ..... [RUN]
08:54:09 8 of 14 PASS not_null_shipping_dim_ship_id ..... [PASS in 0.02s]
08:54:09 10 of 14 START test unique_customer_dim_cust_id ..... [RUN]
08:54:09 10 of 14 PASS unique_customer_dim_cust_id ..... [PASS in 0.02s]
08:54:09 11 of 14 START test unique_geo_dim_geo_id ..... [RUN]
08:54:09 9 of 14 PASS relationships_sales_fact_cust_id_cust_id_ref_customer_dim ..... [PASS in 0.03s]
08:54:09 12 of 14 START test unique_mart_shipping_performance_ship_mode ..... [RUN]
08:54:09 11 of 14 PASS unique_geo_dim_geo_id ..... [PASS in 0.02s]
08:54:09 12 of 14 PASS unique_mart_shipping_performance_ship_mode ..... [PASS in 0.02s]
08:54:09 13 of 14 START test unique_product_dim_prod_id ..... [RUN]
08:54:09 14 of 14 START test unique_shipping_dim_ship_id ..... [RUN]
08:54:09 14 of 14 PASS unique_product_dim_prod_id ..... [PASS in 0.02s]
08:54:09 13 of 14 PASS unique_shipping_dim_ship_id ..... [PASS in 0.03s]
08:54:09
08:54:09 Finished running 14 data tests in 0 hours 0 minutes and 0.33 seconds (0.33s).
08:54:09
08:54:09 Completed successfully
08:54:09
08:54:09 Done. PASS=14 WARN=0 ERROR=0 SKIP=0 NO-OP=0 TOTAL=14
(dbt-env) varvarabobyleva@MacBook-Pro-Varvara-2 superstore_dwh %

```

Рисунок 5 – выполнения команды `dbt test`

Результат выполнения команды dbt snapshot для проекта superstore_dwh:

```
superstore_dwh — -zsh — 139x16
(dbt-env) varvarabobyleva@MacBook-Pro-Varvara-2 superstore_dwh % dbt snapshot
08:23:01 Running with dbt=1.10.13
08:23:01 Registered adapter: postgres=1.9.1
08:23:02 Found 11 models, 14 data tests, 1 snapshot, 1 source, 449 macros
08:23:02
08:23:02 Concurrency: 2 threads (target='dev')
08:23:02
08:23:02 1 of 1 START snapshot dw_snapshots.snapshot_product_dim ..... [RUN]
08:23:02 1 of 1 OK snapshot dw_snapshots.snapshot_product_dim ..... [SELECT 4344 in 0.09s]
08:23:02
08:23:02 Finished running 1 snapshot in 0 hours 0 minutes and 0.22 seconds (0.22s).
08:23:02
08:23:02 Completed successfully
08:23:02
08:23:02 Done. PASS=1 WARN=0 ERROR=0 SKIP=0 NO-OP=0 TOTAL=1
(dbt-env) varvarabobyleva@MacBook-Pro-Varvara-2 superstore_dwh %
```

Рисунок 6 – Выполнение команды dbt snapshot

Результат выполнения команды dbt run для проекта superstore_dwh с учетом индивидуального задания:

```
superstore_dwh — -zsh — 139x40
(dbt-env) varvarabobyleva@MacBook-Pro-Varvara-2 superstore_dwh % dbt run
09:31:23 Running with dbt=1.10.13
09:31:23 Registered adapter: postgres=1.9.1
09:31:23 Found 13 models, 14 data tests, 1 snapshot, 1 source, 1 exposure, 449 macros
09:31:23
09:31:23 Concurrency: 2 threads (target='dev')
09:31:23
09:31:24 1 of 13 START sql table model dw_test.calendar_dim ..... [RUN]
09:31:24 2 of 13 START sql view model stg.stg_orders ..... [RUN]
09:31:24 2 of 13 OK created sql view model stg.stg_orders ..... [CREATE VIEW in 0.09s]
09:31:24 3 of 13 START sql table model dw_test.customer_dim ..... [RUN]
09:31:24 1 of 13 OK created sql table model dw_test.calendar_dim ..... [SELECT 7670 in 0.11s]
09:31:24 4 of 13 START sql table model dw_test.geo_dim ..... [RUN]
09:31:24 3 of 13 OK created sql table model dw_test.customer_dim ..... [SELECT 793 in 0.06s]
09:31:24 5 of 13 START sql table model dw_test.mart_shipping_performance ..... [RUN]
09:31:24 4 of 13 OK created sql table model dw_test.geo_dim ..... [SELECT 632 in 0.05s]
09:31:24 6 of 13 START sql table model dw_test.product_dim ..... [RUN]
09:31:24 5 of 13 OK created sql table model dw_test.mart_shipping_performance ..... [SELECT 4 in 0.04s]
09:31:24 7 of 13 START sql table model dw_test.shipping_dim ..... [RUN]
09:31:24 7 of 13 OK created sql table model dw_test.shipping_dim ..... [SELECT 4 in 0.03s]
09:31:24 6 of 13 OK created sql table model dw_test.product_dim ..... [SELECT 4344 in 0.07s]
09:31:24 8 of 13 START sql table model dw_test.sales_fact ..... [RUN]
09:31:24 8 of 13 OK created sql table model dw_test.sales_fact ..... [SELECT 25667 in 0.10s]
09:31:24 9 of 13 START sql view model dw_intermediate.int_sales_orders ..... [RUN]
09:31:24 9 of 13 OK created sql view model dw_intermediate.int_sales_orders ..... [CREATE VIEW in 0.02s]
09:31:24 10 of 13 START sql view model dw_intermediate.int_customercohorts ..... [RUN]
09:31:24 10 of 13 OK created sql view model dw_intermediate.int_customercohorts ..... [CREATE VIEW in 0.02s]
09:31:24 11 of 13 START sql table model dw_test.mart_customer_ltv ..... [RUN]
09:31:24 11 of 13 OK created sql table model dw_test.mart_customer_ltv ..... [SELECT 2373 in 0.16s]
09:31:24 12 of 13 START sql table model dw_test.mart_customer_cohort_analysis ..... [RUN]
09:31:24 12 of 13 OK created sql table model dw_test.mart_customer_cohort_analysis ..... [SELECT 432 in 0.16s]
09:31:28 13 of 13 OK created sql table model dw_test.mart_customer_cohort_analysis ..... [SELECT 42 in 3.46s]
09:31:28
09:31:28 Finished running 10 table models, 3 view models in 0 hours 0 minutes and 4.11 seconds (4.11s).
09:31:28
09:31:28 Completed successfully
09:31:28
09:31:28 Done. PASS=13 WARN=0 ERROR=0 SKIP=0 NO-OP=0 TOTAL=13
(dbt-env) varvarabobyleva@MacBook-Pro-Varvara-2 superstore_dwh %
```

Рисунок 7 – Выполнение команды dbt run с индивидуальным заданием

Включение mart-модели в дашборд exposures.yml:

```
# models/marts/exposures.yml
version: 2
exposures:
- name: executive_dashboard
  type: dashboard
  maturity: high
  owner:
    name: "Sales Department"
    email: "sales@superstore.com"
  depends_on:
    - ref('mart_monthly_sales')
    - ref('mart_customer_ltv')
    - ref('mart_customer_cohort_analysis')
  description: "Дашборд для руководства с ключевыми метриками продаж и клиентов"
```

Результат построения:

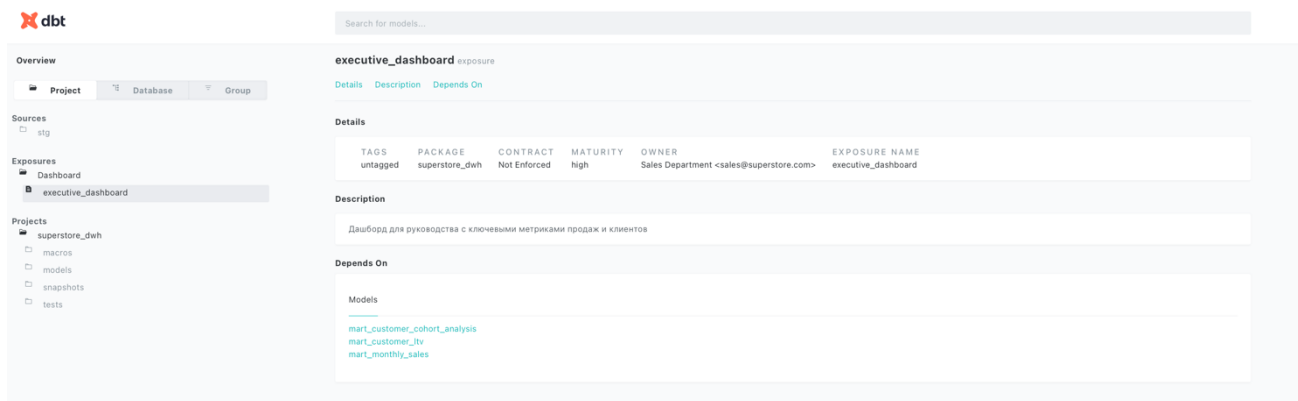


Рисунок 8 – Вывод новой модели в дашборд

Выводы

Использование промежуточных моделей и витрин данных вместо работы с одной большой таблицей фактов имеет несколько ключевых преимуществ:

1. Повышение производительности и эффективности

- **Оптимизация запросов:** Витрины данных (marts) создаются для конкретных бизнес-задач и содержат только нужные данные. Это позволяет BI-инструментам и аналитикам выполнять более быстрые и простые запросы, так как не требуется каждый раз обрабатывать огромную, сложную таблицу фактов.

- **Снижение нагрузки:** Промежуточные модели (intermediate) позволяют разбить сложную логику преобразования данных на более мелкие, управляемые этапы. Вместо того чтобы выполнять один гигантский SQL-запрос, который может быть медленным и ресурсоемким, вы выполняете несколько более простых, что снижает нагрузку на базу данных.

2. Улучшение управляемости и удобства разработки

- **Переиспользуемая логика:** Промежуточные модели содержат общую, переиспользуемую логику (например, очистка, дедупликация или обогащение данных). Эту модель можно использовать как источник для нескольких витрин данных, что исключает дублирование кода и упрощает поддержку.

- **Четкое разделение ответственности:** Каждый слой (staging, intermediate, marts) имеет свою четкую задачу. Это делает проект более понятным и масштабируемым. Разработчикам легче разобраться в структуре данных, найти и исправить ошибки.

3. Гибкость и масштабируемость

- **Адаптация к изменениям:** Если бизнес-логика меняется (например, меняется формула расчета метрики), достаточно внести правки только в соответствующую промежуточную модель. Все зависимые витрины данных автоматически обновятся, не требуя ручных изменений в каждом отдельном запросе.

- **Создание новых витрин:** Разделение на слои позволяет легко добавлять новые витрины данных для новых отделов или отчетов, используя уже существующие промежуточные модели, что значительно ускоряет процесс разработки.

4. Упрощение тестирования

- Разделение сложной логики на более мелкие промежуточные модели позволяет проводить тестирование каждого этапа преобразования по отдельности. Это упрощает поиск и устранение ошибок, так как можно быстро изолировать проблемный участок кода.