



# Course Work

2024 LR S5 SQL and data processing

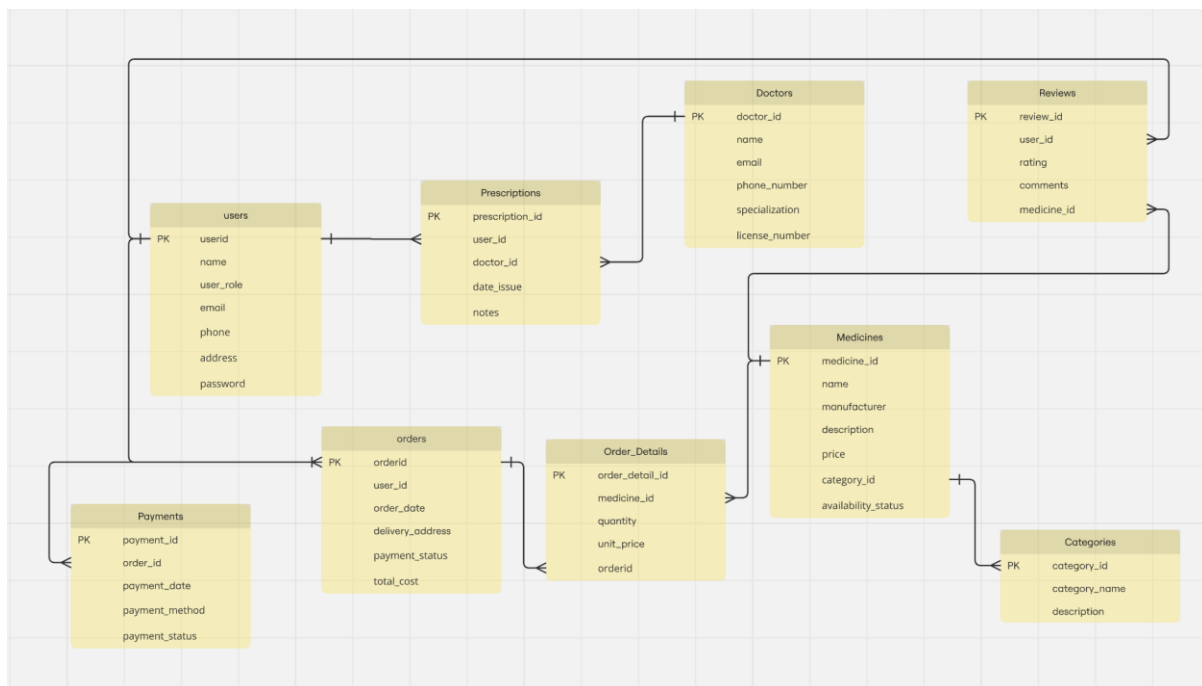
Anastasiya Kupreichyk

## 1. 1 Develop OLTP Solution – Design 3NF Relational Database

### 1. Проектирование схемы:

Таблицы:

1. Users (Пользователи): Хранит информацию о пользователях.
2. Doctors (Врачи): Содержит информацию о врачах.
3. Medicines (Лекарства): Хранит данные о лекарствах.
4. Categories (Категории): Организует лекарства в иерархические категории.
5. Prescriptions (Рецепты): Связывает пользователей с их рецептами, выданными врачами.
6. Reviews (Отзывы): Сохраняет отзывы пользователей о лекарствах.
7. Orders (Заказы): Отслеживает заказы пользователей, включая статус доставки и оплаты.
8. Order Details (Детали заказов): Содержит подробную информацию о лекарствах



## Datasets

## Dataset 1: Users

	A	B	C	D	E	F	G	H
1	user_id;name;email;password;phone;address;role							
2	1;John Doe;john@example.com;hashed_password1;1234567890;123 Main St;user							
3	2;Jane Smith;jane@example.com;hashed_password2;0987654321;456 Elm St;user							
4	3;Alice Johnson;alice@example.com;hashed_password3;1112223333;789 Oak St;user							
5	4;Bob Williams;bob@example.com;hashed_password4;4445556666;101 Pine St;user							
6								
7								

## Dataset 2: Medicines

	A	B	C	D
1	Ibuprofen;1;5.99;HealthCorp;Pain reliever and anti-inflammatory;TRUE			
2	Amoxicillin;2;12.49;PharmaInc;Antibiotic for bacterial infections;TRUE			
3	ColdEase;3;6.99;WellnessCo;Relieves cold symptoms;TRUE			
4	Acetaminophen;1;4.99;MediCare;Pain reliever and fever reducer;FALSE			
5				

### Dataset 3: Doctors

	A	B	C	D	E	F	G	
1	name;specialization;license_number;email;phone							
2	Dr. Alice Johnson;Cardiology;LIC12345;alice.johnson@hospital.com;1112223333							
3	Dr. Bob Williams;Dermatology;LIC67890;bob.williams@clinic.com;4445556666							
4	Dr. Clara Smith;Pediatrics;LIC11223;clara.smith@hospital.com;5556667777							
5	Dr. David Brown;Orthopedics;LIC44556;david.brown@clinic.com;8889990000							
6								

### Dataset 3: Products

	A	B	C	D	E	F	G
1	name;category;price;manufacturer;description;availability_status						
2	Ibuprofen;	Pain Relievers;	5.99;	HealthCorp;	Pain reliever and anti-inflammatory;	True	
3	Amoxicillin;	Antibiotics;	12.49;	PharmaInc;	Antibiotic for bacterial infections;	True	
4	ColdEase;	Cold and Flu;	6.99;	WellnessCo;	Relieves cold symptoms;	True	
5	Acetaminophen;	Pain Relievers;	4.99;	MediCare;	Pain reliever and fever reducer;	False	
6							

### Dataset 3: Orders

	user_id;order_date;delivery_address;payment_status;total_cost	
1;	1/14/2025 10:30;123 Main St, City;completed;18.48	
2;	1/15/2025 14:00;456 Elm St, City;pending;12.49	
3;	1/16/2025 9:45;789 Oak St, City;completed;22.97	
4;	1/17/2025 12:20;101 Pine St, City;pending;30.45	

### Dataset 3: Reviews

1	user_id;medicine_name;rating;comments		
2	1;Ibuprofen;5;Worked great for my headache!		
3	2;ColdEase;4;Helped with my cold, but took a while to work.		
4	3;Amoxicillin;5;Cured my infection quickly.		
5	4;Acetaminophen;3;Effective, but upset my stomach.		

## 1.3 Prepare Script to Load Data from CSV to OLTP Database

```
-- Load data into users table
\COPY users(name, email, password, phone, address, role)
FROM "C:\Users\anast\Downloads\users.csv" DELIMITER ',' CSV HEADER ENCODING 'UTF8' QUOTE '\"' ESCAPE ' ';

-- Load data into doctors table
\COPY doctors(name, specialization, license_number, email, phone)
FROM '/path/to/doctors.csv' DELIMITER ',' CSV HEADER;

-- Load data into categories table
\COPY categories(name, parent_category)
FROM '/path/to/categories.csv' DELIMITER ',' CSV HEADER;

-- Load data into medicines table
\COPY medicines(name, category, price, manufacturer, description, availability_status)
FROM '/path/to/medicines.csv' DELIMITER ',' CSV HEADER;

-- Load data into orders table
\COPY orders(user_id, order_date, delivery_address, payment_status, total_cost)
FROM '/path/to/orders.csv' DELIMITER ',' CSV HEADER;

-- Load data into reviews table
\COPY reviews(user_id, medicine_name, rating, comments)
FROM '/path/to/reviews.csv' DELIMITER ',' CSV HEADER;
```

## 2.1 Develop OLAP solution – design snowflake DWH (2 Facts, 1 SCD Type)

Разработать многомерное хранилище данных (DWH) с использованием snowflake-схемы.

- Реализовать две факт-таблицы и одну измерение с поддержкой SCD Type 2.
- Обеспечить независимость OLAP-структуры от OLTP, добавив агрегации.

### 1. Дизайн схемы:

- Измерения:

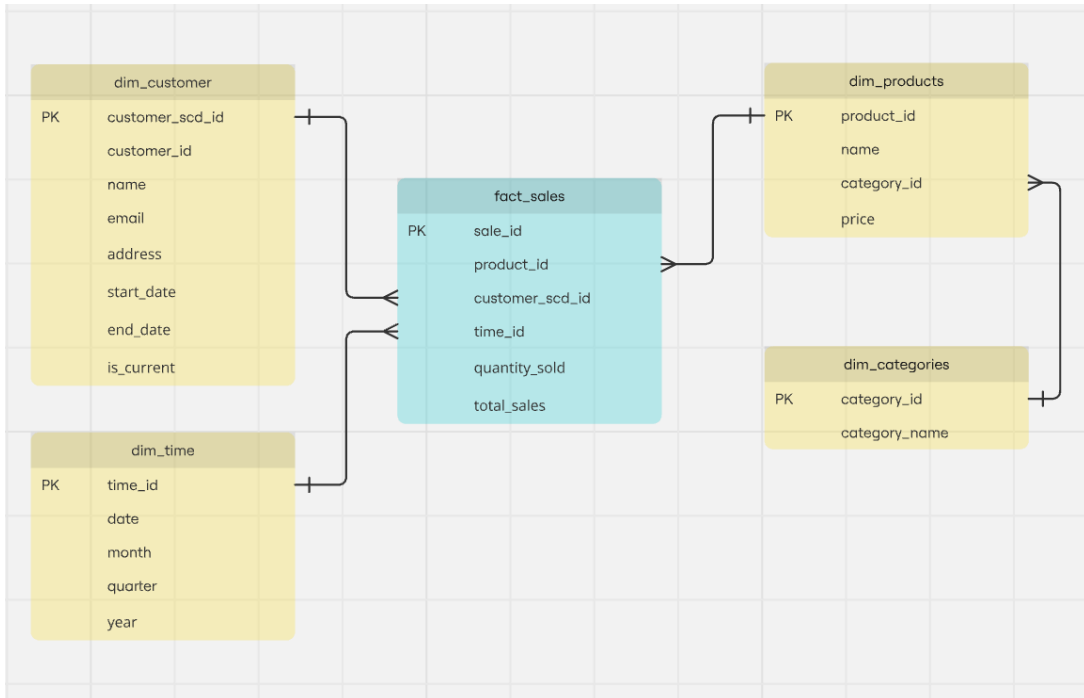
**dim\_customers:** информация о пользователях.

**dim\_products:** информация о продуктах.

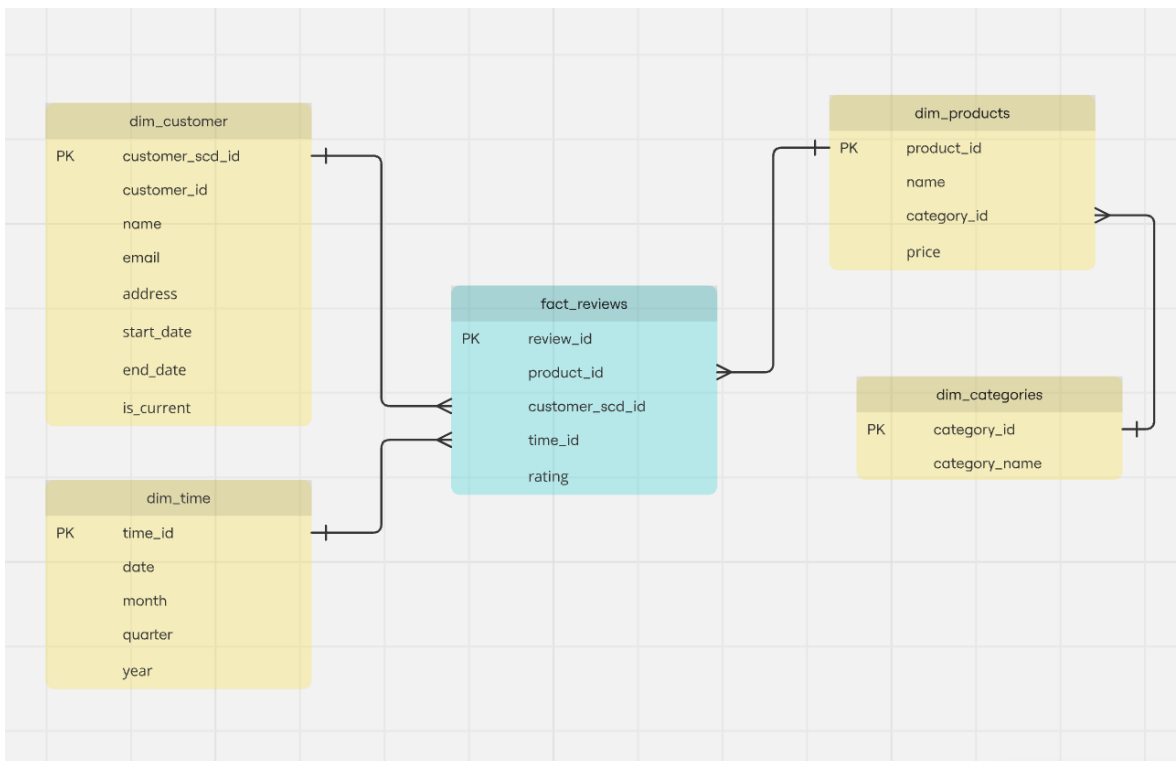
**dim\_time:** временная шкала (SCD Type 2).

- Факт-таблицы:

**fact\_sales:** данные о продажах (количество и общая стоимость).



**fact\_reviews:** данные о пользовательских оценках продуктов.



## 2. Создание таблиц:

```
CREATE TABLE dim_customers (
    customer_id INT,
    name VARCHAR(100),
    email VARCHAR(100),
    address TEXT,
    start_date DATE NOT NULL,
    end_date DATE DEFAULT '9999-12-31',
    is_current BOOLEAN DEFAULT TRUE,
    PRIMARY KEY (customer_id, start_date)
);
```

```
CREATE TABLE dim_products (
    product_id INT PRIMARY KEY,
    name VARCHAR(100),
    category VARCHAR(50),
    price DECIMAL(10, 2)
);
```

```
CREATE TABLE dim_time (
    time_id SERIAL PRIMARY KEY,
    date DATE,
    month INT,
    year INT,
    quarter INT
);
```

```
CREATE TABLE fact_sales (
    sale_id SERIAL PRIMARY KEY,
    product_id INT REFERENCES dim_products(product_id) ON DELETE CASCADE,
    customer_id INT,
    start_date DATE,
    time_id INT REFERENCES dim_time(time_id) ON DELETE CASCADE,
    quantity_sold INT NOT NULL CHECK (quantity_sold > 0),
    total_sales DECIMAL(10, 2) NOT NULL CHECK (total_sales >= 0),
    FOREIGN KEY (customer_id, start_date) REFERENCES dim_customers(customer_id, start_date) ON DELETE CASCADE
);
INSERT INTO fact_sales (sale_id, product_id, customer_id, time_id, quantity_sold, total_sales)
VALUES
(1, 1, 1, 1, 10, 59.90), -- 10 единиц Ibuprofen продано клиенту 1 в январе 2024
(2, 2, 2, 1, 5, 62.45), -- 5 единиц Amoxicillin продано клиенту 2 в январе 2024
(3, 1, 2, 2, 8, 47.92), -- 8 единиц Ibuprofen продано клиенту 2 в феврале 2024
(4, 2, 1, 2, 3, 37.47); -- 3 единицы Amoxicillin продано клиенту 1 в феврале 2024
```

```
CREATE TABLE fact_reviews (
    review_id SERIAL PRIMARY KEY,
    product_id INT REFERENCES dim_products(product_id),
    customer_id INT,
    customer_start_date DATE,
    time_id INT REFERENCES dim_time(time_id),
    rating INT CHECK (rating BETWEEN 1 AND 5),
    FOREIGN KEY (customer_id, customer_start_date) REFERENCES dim_customers(customer_id, start_date)
);
```





```
INSERT INTO dim_time (time_id, date, month, quarter, year)
VALUES
(1, '2024-01-15', 1, 1, 2024),
(2, '2024-02-15', 2, 1, 2024),
(3, '2024-03-15', 3, 1, 2024),
(4, '2024-04-15', 4, 2, 2024);

INSERT INTO fact_reviews (product_id, customer_id, time_id, rating)
VALUES
(1, 1, 1, 5), -- John Doe reviewed Ibuprofen in January
(2, 2, 2, 4), -- Jane Smith reviewed Amoxicillin in February
(3, 3, 3, 3), -- Alice Johnson reviewed ColdEase in March
(4, 1, 4, 5); -- John Doe reviewed Acetaminophen in April
```

**Вопрос: Какую среднюю оценку пользователи поставили лекарствам в разных месяцах?**

SQL-запрос для ответа на вопрос:

```
SELECT
    dt.year,
    dt.month,
    dp.name AS product_name,
    AVG(fr.rating) AS avg_rating
FROM
    fact_reviews fr
JOIN
    dim_time dt ON fr.time_id = dt.time_id
JOIN
    dim_products dp ON fr.product_id = dp.product_id
GROUP BY
    dt.year, dt.month, dp.name
ORDER BY
    dt.year, dt.month, dp.name;
```

	year integer 	month integer 	product_name character varying (100) 	avg_rating numeric 
1	2024	1	Ibuprofen	5.0000000000000000
2	2024	2	Amoxicillin	4.0000000000000000
3	2024	3	ColdEase	3.0000000000000000
4	2024	4	Acetaminophen	5.0000000000000000

**Вопрос: Сколько продуктов было продано и какова их общая стоимость по месяцам?**

SQL-запрос для ответа на вопрос:



```

SELECT
    t.month,
    t.year,
    SUM(s.quantity_sold) AS total_quantity_sold,
    SUM(s.total_sales) AS total_revenue
FROM
    fact_sales s
JOIN
    dim_time t ON s.time_id = t.time_id
GROUP BY
    t.month, t.year
ORDER BY
    t.year, t.month;

```

	month integer	year integer	total_quantity_sold bigint	total_revenue numeric
1	1	2024	15	122.35
2	2	2024	11	85.39

## 3.1 Develop OLTP Solution – Design 3NF Relational Database

### 1. Identify reference OLTP data:

```

SELECT *
FROM orders
WHERE order_date >= CURRENT_DATE - INTERVAL '12 months'; -- Только заказы за последний год

SELECT DISTINCT o.order_id, o.payment_status, r.rating
FROM orders o
LEFT JOIN reviews r ON o.user_id = r.user_id
WHERE o.order_date >= '2024-01-01'
    AND o.payment_status != 'completed'
    AND r.rating >= 4;

```

### 2. Extract data from the source: convert it into a single format for standardized processing.

Извлечение данных о заказах и их деталях за последние 12 месяцев из OLTP базы

```

SELECT
  o.order_id,
  o.user_id,
  o.order_date,
  o.total_cost,
  od.medicine_id,
  od.quantity,
  od.unit_price
FROM
  orders o
JOIN
  order_details od
ON
  o.order_id = od.order_id
WHERE
  o.order_date >= CURRENT_DATE - INTERVAL '12 months';

```

	order_id integer	user_id integer	order_date timestamp without time zone	total_cost numeric (10,2)	medicine_id integer	quantity integer	unit_price numeric (10,2)
1	1	1	2025-01-10 14:30:00	20.97	7	2	5.99
2	1	1	2025-01-10 14:30:00	20.97	8	1	6.99
3	2	2	2025-01-11 15:45:00	12.49	9	1	12.49
4	3	3	2025-01-12 10:15:00	18.98	7	2	4.99
5	4	4	2025-01-13 16:20:00	9.98	7	1	5.99

### 3. Validate Data

Проверим данные на корректность, оставляя только значения в ожидаемых диапазонах.

а) Проверка и исключение некорректных записей:

```

SELECT *
FROM staging_orders
WHERE total_cost < 0
   OR quantity <= 0
   OR medicine_id IS NULL
   OR order_id IS NULL;

```

б) Удаление некорректных записей:

```
DELETE FROM staging_orders
WHERE total_cost < 0
    OR quantity <= 0
    OR medicine_id IS NULL
    OR order_id IS NULL;
```

## 4. Transform Data

### а) Удаление дубликатов:

```
DELETE FROM staging_orders
WHERE order_id IN (
    SELECT order_id
    FROM (
        SELECT order_id, ROW_NUMBER() OVER (PARTITION BY order_id ORDER BY order_date DESC) AS rnk
        FROM staging_orders
    ) t
    WHERE rnk > 1
);
```

### б) Агрегирование данных (например, общий доход по месяцам):

```
SELECT
    DATE_TRUNC('month', order_date) AS month,
    SUM(total_cost) AS monthly_revenue
FROM staging_orders
GROUP BY DATE_TRUNC('month', order_date)
ORDER BY month;
```

## 5. Загрузка данных в staging таблицу:

```
pharmacy=# \copy staging_orders (order_id, user_id, order_date, total_cost, medicine_id, quantity, unit_price)F
ROM 'C:/Users/anast/Downloads/data.csv' WITH CSV HEADER;
COPY 5
pharmacy=#
```

## 6. Publish Data

### а) Вставка новых записей в целевую таблицу:

```
CREATE TABLE target_orders (  
  order_id INT PRIMARY KEY,  
  user_id INT,  
  order_date DATE,  
  total_cost DECIMAL(10, 2),  
  medicine_id INT,  
  quantity INT,  
  unit_price DECIMAL(10, 2)  
);
```

```
INSERT INTO target_orders (order_id, user_id, order_date, total_cost, medicine_id, quantity, unit_price)  
SELECT order_id, user_id, order_date, total_cost, medicine_id, quantity, unit_price  
FROM staging_orders  
ON CONFLICT (order_id) DO NOTHING;
```

## б) Обновление существующих записей:

```
UPDATE target_orders  
SET  
  total_cost = s.total_cost,  
  quantity = s.quantity,  
  unit_price = s.unit_price  
FROM staging_orders s  
WHERE target_orders.order_id = s.order_id  
  AND (target_orders.total_cost != s.total_cost OR target_orders.quantity != s.quantity);
```

## 7. Очистка staging таблицы

```
TRUNCATE TABLE staging_orders;
```

## 8. Мониторинг и журналирование

### а) Создание таблицы журнала загрузки:

```
CREATE TABLE IF NOT EXISTS load_logs (  
  load_id SERIAL PRIMARY KEY,  
  load_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
  inserted_records INT,  
  updated_records INT  
);
```

### б) Запись результатов загрузки:

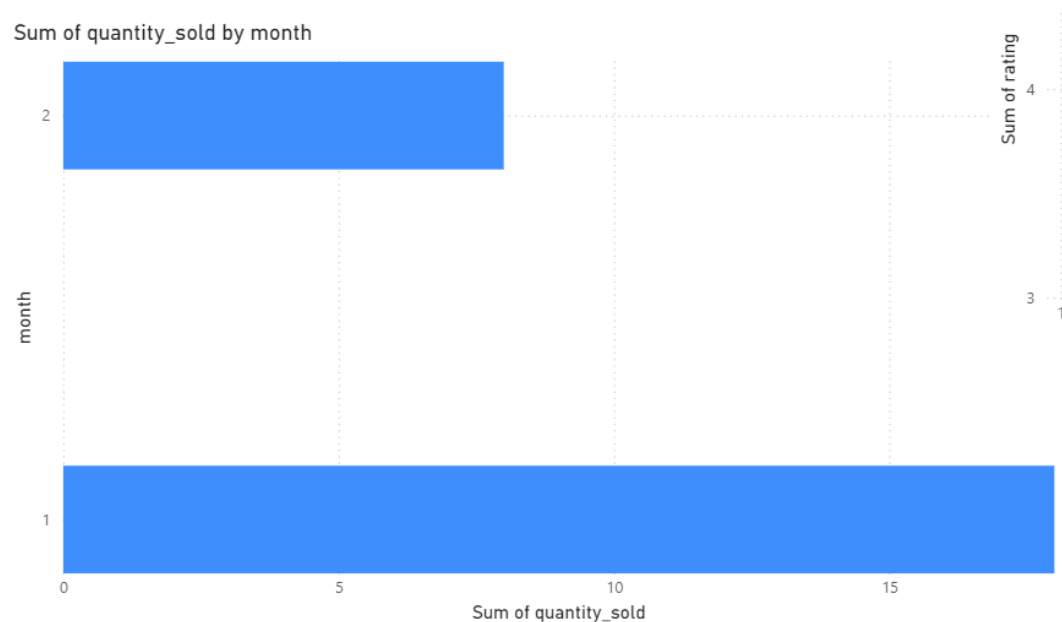
```

INSERT INTO load_logs (inserted_records, updated_records)
VALUES (
    (SELECT COUNT(*) FROM staging_orders s
     LEFT JOIN target_orders t ON s.order_id = t.order_id WHERE t.order_id IS NULL),
    (SELECT COUNT(*) FROM staging_orders s
     JOIN target_orders t ON s.order_id = t.order_id
     WHERE s.total_cost != t.total_cost OR s.quantity != t.quantity)
);

```

## 2.3 Meaningful Power BI report answering analytical questions

### 2.3.1 Количество проданных продуктов и их стоимость по месяцам

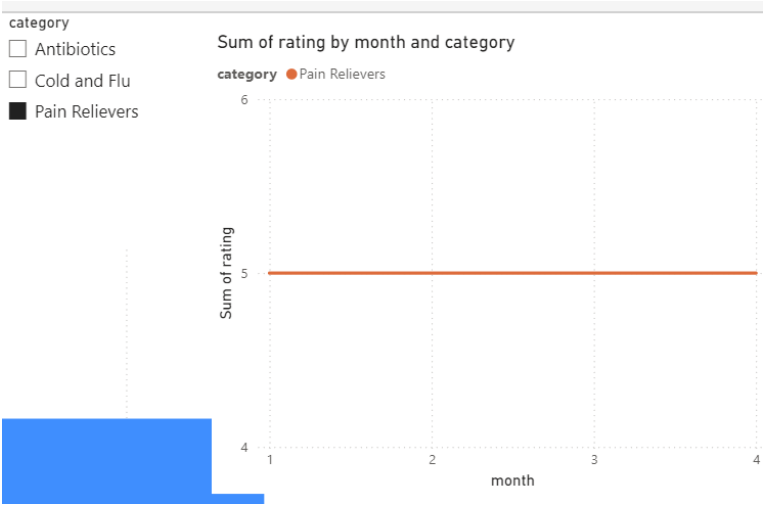
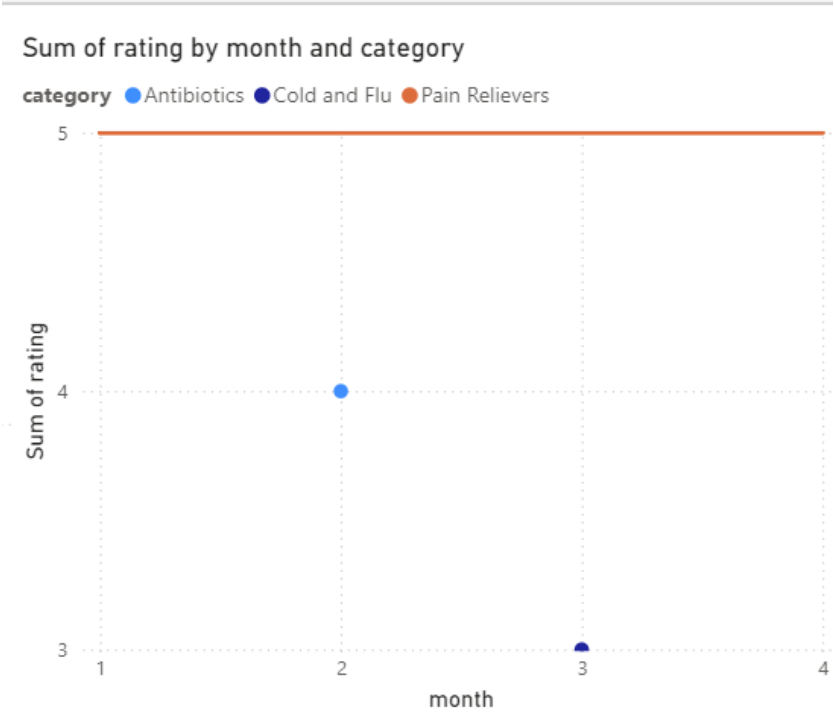


### 2.3.2 Добавление слайсера по категории продукта, году и месяцу

date
1/15/2024
4/15/2024
year
2024
2024

category
☐ Antibiotics
☐ Cold and Flu
☐ Pain Relievers

2.3.3 Средняя оценка продуктов по месяцам



2.3.4 Общее количество и доход от продаж

	Sum of quantity_sold	Sum of total_sales
	18	107.82

## 3. Scripts

### 3.1 Запросы для OLTP базы данных

Цель: Анализ продаж лекарств и отзывов о них по месяцам и годам для выявления сезонных трендов и потребительских предпочтений.

Этот запрос агрегирует данные по продажам лекарств в разрезе месяца и года, считая общее количество проданных единиц каждого продукта.

---Запрос для расчета средней оценки продуктов по месяцам:

```
SELECT
    dt.year,
    dt.month,
    dp.name AS product_name,
    AVG(fr.rating) AS avg_rating
FROM
    fact_reviews fr
JOIN
    dim_time dt ON fr.time_id = dt.time_id
JOIN
    dim_products dp ON fr.product_id = dp.product_id
GROUP BY
    dt.year, dt.month, dp.name
ORDER BY
    dt.year, dt.month, dp.name;
```

-- Запрос для расчета общего количества проданных товаров и общего дохода по месяцам:

```
SELECT
    t.month,
    t.year,
    SUM(s.quantity_sold) AS total_quantity_sold,
    SUM(s.total_sales) AS total_revenue
FROM
    fact_sales s
JOIN
    dim_time t ON s.time_id = t.time_id
GROUP BY
    t.month, t.year
ORDER BY
    t.year, t.month;
```

Этот запрос предоставляет средний рейтинг продуктов по месяцам, чтобы понять, как отзывы изменяются по времени.

### 3.2 Запросы для OLAP базы данных

Здесь используется таблица monthly\_reviews, которая уже агрегирует средние оценки продуктов по месяцам.

```
--Запрос для средней оценки продуктов
SELECT
    dt.year AS review_year,
    dt.month AS review_month,
    dp.name AS product_name,
    AVG(fr.rating) AS avg_rating
FROM
    fact_reviews fr
JOIN
    dim_time dt ON fr.time_id = dt.time_id
JOIN
    dim_products dp ON fr.product_id = dp.product_id
GROUP BY
    dt.year, dt.month, dp.name
ORDER BY
    review_year, review_month, avg_rating DESC;
```

В этом запросе используется таблица monthly\_sales, которая уже агрегирует данные о продажах по месяцам и годам. Это ускоряет процесс получения отчетности.

```
--Запрос для данных о продажах
SELECT
    dt.year AS sales_year,
    dt.month AS sales_month,
    dp.name AS product_name,
    SUM(fs.quantity_sold) AS total_sold
FROM
    fact_sales fs
JOIN
    dim_time dt ON fs.time_id = dt.time_id
JOIN
    dim_products dp ON fs.product_id = dp.product_id
GROUP BY
    dt.year, dt.month, dp.name
ORDER BY
    sales_year, sales_month, total_sold DESC;
```