МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**

**Факультет безопасности информационных технологий**

**Направление подготовки: 10.03.01 Информационная безопасность**

**Образовательная программа: Информационная безопасность**

**Дисциплина:**

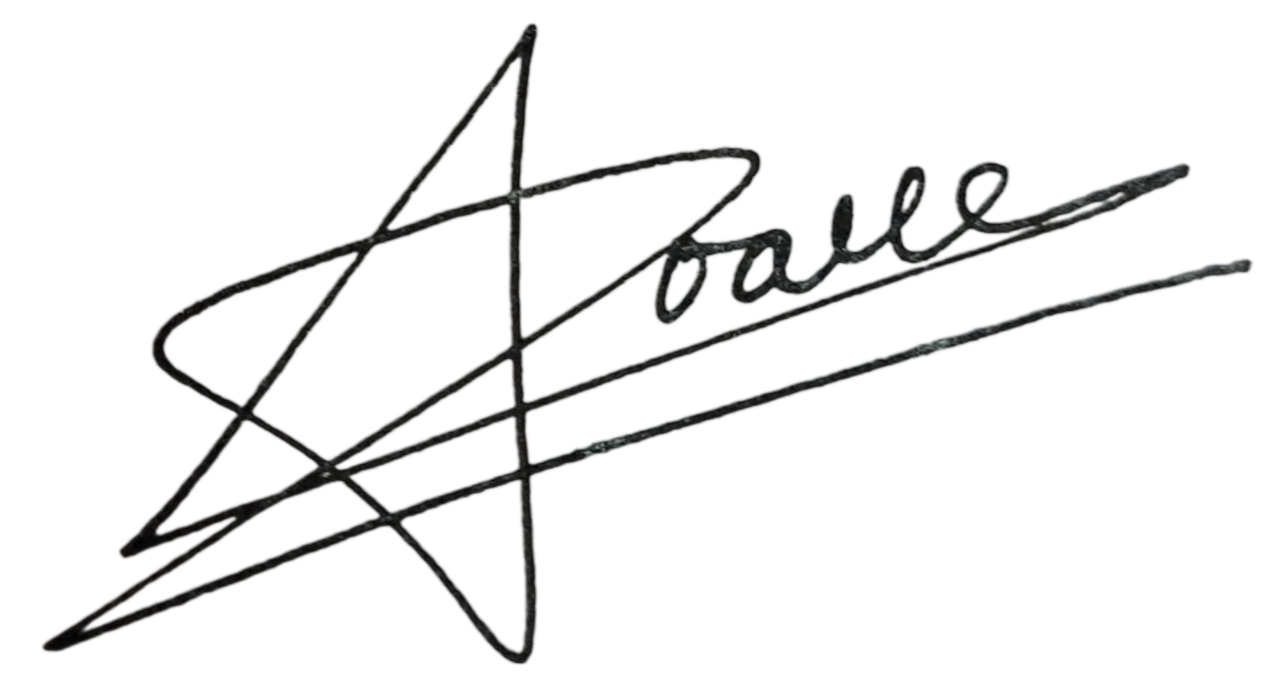
**«Информационная безопасность баз данных»**

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2**

***«Реализация БД в рамках СУБД»***

**Выполнил студент(ы):**

группа/поток: N3347 / ИББД.Nб3 1.5



\_\_\_\_\_\_\_\_Чу Ван Доан \_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_

*ФИО Подпись*

**Проверил:**

Салихов Максим Русланович \_/\_\_\_\_\_\_\_\_\_\_

*ФИО Подпись*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Отметка о выполнении (один из вариантов:*

*отлично, хорошо, удовлетворительно, зачтено)*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Дата*

Санкт-Петербург

2024 г.

**СОДЕРЖАНИЕ**

[**Цель работы 3**](#_heading=h.dsn0t69papw)

[**Задание 4**](#_heading=h.b39kds37v2hx)

[**Ход Работы 5**](#_heading=h.fj4jd1a9mogq)

[1. Выбрать систему управления базами данных (СУБД), которая будет использована в рамках лабораторной работы. Кратко обосновать свой выбор. 5](#_heading=h.91ry3f5v1g9t)

[2. Создать БД в выбранной в вами СУБД на основе итоговой разработанной схемы отношений из ЛР 1. Заполните созданную вами БД информацией, сгенерируйте как минимум 7-8 кортежей с данными для каждой из ваших основных таблиц. В отчете по лабораторной работе укажите следующий SQL-код (написанный вами или сгенерированный средствами администрирования СУБД): 6](#_heading=h.dlq2g5k62omj)

[2.1 Код для создания всех таблиц; 7](#_heading=h.t5p7n77rkbmt)

[2.2 Код для внесения данных в созданные таблицы; 9](#_heading=h.1i8std2hcplu)

[2.3 Код хотя бы одной SQL-команды для модифицирования структуры таблицы; 11](#_heading=h.97htddxjjzax)

[3. Индексировать таблицы. Добавить индексы для атрибутов, по которым происходит объединение таблиц, а также атрибуты по которым выполняется поиск/фильтрация данных. 11](#_heading=h.b3y1hv5jm4tb)

[4. Установить взаимосвязи между таблицами. 12](#_heading=h.57f5d62p84o2)

[5. Дополнительно. Тестовых запросов к вашей БД 13](#_heading=h.b363epfofh27)

[6. Создать представления, составленные в пункте 5 лабораторной 1. 15](#_heading=h.iq938f1d6vrp)

[**Вывод 17**](#_heading=h.m9s4ijudrmu8)

## Цель работы

Получение навыков по работе с современными системами управления базами данных.

## Задание

1. Выбрать систему управления базами данных (СУБД), которая будет использована в рамках лабораторной работы. Кратко обосновать свой выбор.
2. Создать БД в выбранной в вами СУБД на основе итоговой разработанной схемы отношений из ЛР 1. Заполните созданную вами БД информацией, сгенерируйте как минимум 7-8 кортежей с данными для каждой из ваших основных таблиц. В отчете по лабораторной работе укажите следующий SQL-код (написанный вами или сгенерированный средствами администрирования СУБД):

* код для создания всех таблиц;
* код для внесения данных в созданные таблицы;
* код хотя бы одной SQL-команды для модифицирования структуры таблицы;

1. Индексировать таблицы. Добавить индексы для атрибутов, по которым происходит объединение таблиц, а также атрибуты по которым выполняется поиск/фильтрация данных.
2. Установить взаимосвязи между таблицами.
3. **Дополнительно. Тестовых запросов к вашей БД**
4. Создать представления, составленные в пункте 5 лабораторной 1.

## Ход Работы

### Выбрать систему управления базами данных (СУБД), которая будет использована в рамках лабораторной работы. Кратко обосновать свой выбор.

* Использование PostgreSQL
* Причина использования: PostgreSQL — это мощная, открытая система управления реляционными базами данных, соответствующая стандарту SQL. Она способна обрабатывать сложные базы данных, поддерживает многопользовательскую работу и хорошо интегрируется с различными языками программирования. PostgreSQL также широко используется в реальных проектах благодаря своей расширяемости и высокой безопасности.

### Создать БД в выбранной в вами СУБД на основе итоговой разработанной схемы отношений из ЛР 1. Заполните созданную вами БД информацией, сгенерируйте как минимум 7-8 кортежей с данными для каждой из ваших основных таблиц. В отчете по лабораторной работе укажите следующий SQL-код (написанный вами или сгенерированный средствами администрирования СУБД):

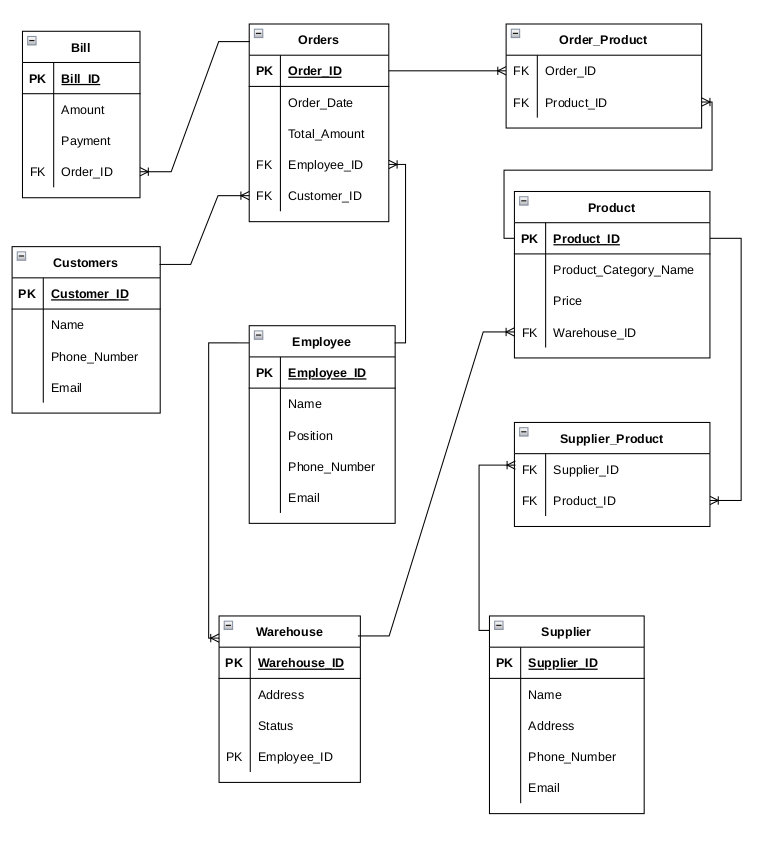
****

Рисунок 1. Итоговая схема предварительных отношений

* Подключение к базе данных:

**sudo systemctl enable --now postgresql.service**

**sudo su postgres -c psql**

* Создать базу данных

**create database coffee\_shop\_db**

**with**

**owner = postgres**

**encoding = 'UTF8'**

**tablespace = pg\_default**

**connection limit = -1**

**is\_template = False;**

* Создайте схему и укажите ее

**coffee\_shop\_db=# create schema coffee\_shop\_schema;**

**CREATE SCHEMA**

**coffee\_shop\_db=# \dn**

**List of schemas**

**Name | Owner**

**--------------------+-------------------**

**coffee\_shop\_schema | postgres**

**public | pg\_database\_owner**

**(2 rows)**

**coffee\_shop\_db=# set search\_path to coffee\_shop\_schema;**

**SET**

**coffee\_shop\_db=# show search\_path ;**

**search\_path**

**--------------------**

**coffee\_shop\_schema**

#### 2.1 Код для создания всех таблиц;

* Таблица Employee

**CREATE TABLE Employee (**

**Employee\_ID SERIAL PRIMARY KEY,**

**Name VARCHAR(100) NOT NULL,**

**Position VARCHAR(50) NOT NULL,**

**Phone\_Number VARCHAR(13),**

**Email VARCHAR(100)**

**);**

* Таблица Supplier

**CREATE TABLE Supplier (**

**Supplier\_ID SERIAL PRIMARY KEY,**

**Name VARCHAR(100) NOT NULL,**

**Address VARCHAR(255),**

**Phone\_Number VARCHAR(13),**

**Email VARCHAR(100)**

**);**

* Таблица Warehouse

**CREATE TABLE Warehouse (**

**Warehouse\_ID SERIAL PRIMARY KEY,**

**Address VARCHAR(255) NOT NULL,**

**Status VARCHAR(50),**

**Employee\_ID INTEGER,**

**FOREIGN KEY (Employee\_ID) REFERENCES Employee(Employee\_ID)**

**);**

* Таблица Product

**CREATE TABLE Product (**

**Product\_ID SERIAL PRIMARY KEY,**

**Product\_Category\_Name VARCHAR(100) NOT NULL,**

**Price NUMERIC(10, 2) NOT NULL,**

**Warehouse\_ID INTEGER,**

**FOREIGN KEY (Warehouse\_ID) REFERENCES Warehouse(Warehouse\_ID)**

**);**

* Таблица Customer

**CREATE TABLE Customer (**

**Customer\_ID SERIAL PRIMARY KEY,**

**Name VARCHAR(100) NOT NULL,**

**Phone\_Number VARCHAR(15),**

**Email VARCHAR(100)**

**);**

* Таблица Orders

**CREATE TABLE Orders(**

**Order\_ID SERIAL PRIMARY KEY,**

**Order\_Date DATE NOT NULL,**

**Total\_Amount NUMERIC(10, 2) NOT NULL,**

**Customer\_ID INTEGER,**

**Employee\_ID INTEGER,**

**FOREIGN KEY (Customer\_ID) REFERENCES Customer(Customer\_ID),**

**FOREIGN KEY (Employee\_ID) REFERENCES Employee(Employee\_ID)**

**);**

* Таблица Bill

**CREATE TABLE Bill (**

**Bill\_ID SERIAL PRIMARY KEY,**

**Amount NUMERIC(10, 2) NOT NULL,**

**Payment\_Method VARCHAR(50) NOT NULL,**

**Order\_ID INTEGER,**

**FOREIGN KEY (Order\_ID) REFERENCES Orders(Order\_ID)**

**);**

* Таблица связи между таблицей Orders и таблицей Product

**CREATE TABLE Order\_Product (**

**Order\_ID INTEGER,**

**Product\_ID INTEGER,**

**PRIMARY KEY (Order\_ID, Product\_ID),**

**FOREIGN KEY (Order\_ID) REFERENCES "Order"(Order\_ID),**

**FOREIGN KEY (Product\_ID) REFERENCES Product(Product\_ID)**

**);**

* Таблица связи между таблицей Supplier и таблицей Product

**CREATE TABLE Supplier\_Product (**

**Supplier\_ID INTEGER,**

**Product\_ID INTEGER,**

**PRIMARY KEY (Supplier\_ID, Product\_ID),**

**FOREIGN KEY (Supplier\_ID) REFERENCES Supplier(Supplier\_ID),**

**FOREIGN KEY (Product\_ID) REFERENCES Product(Product\_ID)**

**);**

#### 2.2 Код для внесения данных в созданные таблицы;

* У нас есть следующие таблицы

**coffee\_shop\_db=# \dt**

**List of relations**

**Schema | Name | Type | Owner**

**--------------------+------------------+-------+----------**

**coffee\_shop\_schema | bill | table | postgres**

**coffee\_shop\_schema | customer | table | postgres**

**coffee\_shop\_schema | employee | table | postgres**

**coffee\_shop\_schema | order\_product | table | postgres**

**coffee\_shop\_schema | orders | table | postgres**

**coffee\_shop\_schema | product | table | postgres**

**coffee\_shop\_schema | supplier | table | postgres**

**coffee\_shop\_schema | supplier\_product | table | postgres**

**coffee\_shop\_schema | warehouse | table | postgres**

**(9 rows)**

* Внесение данных в таблицу Employee

**INSERT INTO employee (name, position, phone\_number, email)**

**VALUES**

**('Nguyen Van A', 'supervisor', '9312828535', 'nguyenvana123@gmail.com'),**

**('Nguyen Thi B', 'salesperson', '92537563834', 'nb4214@gmail.com'),**

**('Artom', 'salesperson', '27582473683', 'artom33@mail.ru'),**

**('Irina', 'salesperson', '8925748253', 'irina8386@mail.ru'),**

**('Tran', 'salesperson', '92846363583', 'trantran4953@gmail.com');**

* Внесение данных в таблицу Supplier

**INSERT INTO supplier (name, address, phone\_number, email)**

**VALUES**

**('Trung Nguyen Coffee', 'Dalat city', '03873532753', 'trungnguyencoffee@gmail.com'),**

**('King Coffee', 'Ho Chi Minh city', '0385636282', 'kingcoffee@gmail.com'),**

**('G7 Coffee', 'Ha Noi', '92834772843', 'g7coffee@gmail.com');**

* Внесение данных в таблицу Warehouse

**INSERT INTO warehouse (address, status, employee\_id)**

**VALUES**

**('Lam Ha', 'In stock', 1),**

**('Tan Ha', 'In stock', 1),**

**('Dan Phuong', 'In stock', 1),**

**('Me Linh', 'In stock', 1);**

* Внесение данных в таблицу Product

**INSERT INTO product (product\_category\_name, price, warehouse\_id)**

**VALUES**

**('Arabica', 100000, 1),**

**('Robusta', 90000, 2),**

**('Bourbon', 96000, 3),**

**('Typica', 92000, 4);**

* Внесение данных в таблицу Customer

**INSERT INTO customer (name, phone\_number, email)**

**VALUES**

**('Alex', '93842727543', 'alex8888@mail.ru'),**

**('Tom', '82736464383', 'tomi7749@mail.ru'),**

**('Anton', '827364646737', 'ton@mail.ru'),**

**('Karababy', '8283747654', 'baby@mail.ru');**

* Внесение данных в таблицу Bill

**INSERT INTO bill (amount, payment, order\_id)**

**VALUES**

**(150000, 'Cash', 1),**

**(599999, 'Bank Transfer', 2),**

**(50000, 'Cash', 3),**

**(20000, 'Online Payment', 3),**

**(6699999, 'Cash', 4);**

* Внесение данных в таблицу Order\_Product

**INSERT INTO order\_product (order\_id, product\_id)**

**VALUES**

**(1, 1),**

**(2, 2),**

**(3, 3),**

**(4, 4);**

* Внесение данных в таблицу Order\_Product

**INSERT INTO supplier\_product (supplier\_id, product\_id)**

**VALUES**

**(1, 1),**

**(1, 2),**

**(2, 3),**

**(3, 4);**

#### 2.3 Код хотя бы одной SQL-команды для модифицирования структуры таблицы;

* Добавление столбца Address в таблицу Customer

**ALTER TABLE Customer ADD COLUMN Address VARCHAR(255);**

* Удаление столбца Address в таблице Customer

**ALTER TABLE Customer DROP COLUMN Address;**

### Индексировать таблицы. Добавить индексы для атрибутов, по которым происходит объединение таблиц, а также атрибуты по которым выполняется поиск/фильтрация данных.

**-- Таблица Employee**

**-- Создание индекса для столбца Employee\_ID для поиска сотрудников**

**CREATE INDEX idx\_employee\_id ON Employee(Employee\_ID);**

**-- Таблица Supplier**

**-- Создание индекса для столбца Supplier\_ID для поиска поставщиков**

**CREATE INDEX idx\_supplier\_id ON Supplier(Supplier\_ID);**

**-- Таблица Product**

**-- Создание индекса для столбца Warehouse\_ID для ускорения связи с таблицей Warehouse**

**CREATE INDEX idx\_product\_warehouse\_id ON Product(Warehouse\_ID);**

**-- Таблица Customer**

**-- Создание индекса для столбца Customer\_ID для поиска клиентов**

**CREATE INDEX idx\_customer\_id ON Customer(Customer\_ID);**

**-- Таблица Order**

**-- Создание индекса для столбцов Customer\_ID и Employee\_ID для ускорения запросов между Order и Customer, Order и Employee**

**CREATE INDEX idx\_order\_customer\_id ON orders(Customer\_ID);**

**CREATE INDEX idx\_order\_employee\_id ON orders(Employee\_ID);**

**-- Таблица Bill**

**-- Создание индекса для столбца Order\_ID для ускорения запросов между Bill и Order**

**CREATE INDEX idx\_bill\_order\_id ON Bill(Order\_ID);**

**-- Таблица Warehouse**

**-- Создание индекса для столбца Employee\_ID для ускорения запросов между Warehouse и Employee**

**CREATE INDEX idx\_warehouse\_employee\_id ON Warehouse(Employee\_ID);**

**-- Связующая таблица Order\_Product**

**-- Создание индекса для столбцов Order\_ID и Product\_ID для ускорения запросов между Order и Product**

**CREATE INDEX idx\_order\_product\_order\_id ON Order\_Product(Order\_ID);**

**CREATE INDEX idx\_order\_product\_product\_id ON Order\_Product(Product\_ID);**

**-- Связующая таблица Supplier\_Product**

**-- Создание индекса для столбцов Supplier\_ID и Product\_ID для ускорения запросов между Supplier и Product**

**CREATE INDEX idx\_supplier\_product\_supplier\_id ON Supplier\_Product(Supplier\_ID);**

**CREATE INDEX idx\_supplier\_product\_product\_id ON Supplier\_Product(Product\_ID);**

### Установить взаимосвязи между таблицами.

* Основные связи между таблицами установлены с использованием внешних ключей, но дополнительные индексы также ускоряют запросы.

**ALTER TABLE Warehouse**

**ADD FOREIGN KEY (Employee\_ID) REFERENCES Employee(Employee\_ID);**

**ALTER TABLE Orders**

**ADD FOREIGN KEY (Customer\_ID) REFERENCES Customer(Customer\_ID),**

**ADD FOREIGN KEY (Employee\_ID) REFERENCES Employee(Employee\_ID);**

**ALTER TABLE Bill**

**ADD FOREIGN KEY (Order\_ID) REFERENCES Orders(Order\_ID);**

**ALTER TABLE Order\_Product**

**ADD FOREIGN KEY (Order\_ID) REFERENCES Orders(Order\_ID),**

**ADD FOREIGN KEY (Product\_ID) REFERENCES Product(Product\_ID);**

**ALTER TABLE Supplier\_Product**

**ADD FOREIGN KEY (Supplier\_ID) REFERENCES Supplier(Supplier\_ID),**

**ADD FOREIGN KEY (Product\_ID) REFERENCES Product(Product\_ID);**

### Дополнительно. Тестовых запросов к вашей БД

* Получить все заказы с информацией о клиентах и сотрудниках, которые их обслуживали:

**coffee\_shop\_db=# SELECT o.Order\_ID, o.Order\_Date, o.Total\_Amount, c.Name AS Customer, e.Name AS Employee**

**FROM Orders o**

**JOIN Customer c ON o.Customer\_ID = c.Customer\_ID**

**JOIN Employee e ON o.Employee\_ID = e.Employee\_ID;**

**order\_id | order\_date | total\_amount | customer | employee**

**----------+------------+--------------+----------+--------------**

**1 | 2024-10-12 | 150000.00 | Alex | Nguyen Thi B**

**2 | 2024-10-13 | 599999.00 | Tom | Nguyen Thi B**

**3 | 2024-10-13 | 70000.00 | Anton | Artom**

**4 | 2024-10-20 | 6699999.00 | Karababy | Irina**

**(4 rows)**

* Показать все продукты, хранящиеся на каждом складе:

**coffee\_shop\_db=# SELECT w.Address AS Warehouse, p.Product\_Category\_Name AS Product, p.Price**

**FROM Warehouse w**

**JOIN Product p ON w.Warehouse\_ID = p.Warehouse\_ID;**

**warehouse | product | price**

**------------+---------+-----------**

**Lam Ha | Arabica | 100000.00**

**Tan Ha | Robusta | 90000.00**

**Dan Phuong | Bourbon | 96000.00**

**Me Linh | Typica | 92000.00**

**(4 rows)**

* Вывести список поставщиков и предоставляемых ими продуктов:

**coffee\_shop\_db=# SELECT s.Name AS Supplier, p.Product\_Category\_Name AS Product**

**FROM Supplier s**

**JOIN Supplier\_Product sp ON s.Supplier\_ID = sp.Supplier\_ID**

**JOIN Product p ON sp.Product\_ID = p.Product\_ID;**

**supplier | product**

**---------------------+---------**

**Trung Nguyen Coffee | Arabica**

**Trung Nguyen Coffee | Robusta**

**King Coffee | Bourbon**

**G7 Coffee | Typica**

**(4 rows)**

* Найти сумму всех заказов, оплаченных наличными:

**coffee\_shop\_db=# SELECT SUM(b.Amount) AS TotalCashPayments**

**FROM Bill b**

**WHERE b.Payment = 'Cash';**

**totalcashpayments**

**-------------------**

**6899999.00**

**(1 row)**

* Получить все заказы, содержащие конкретный продукт, например "Arabica":

**coffee\_shop\_db=# SELECT o.Order\_ID, o.Order\_Date, o.Total\_Amount**

**FROM Orders o**

**JOIN Order\_Product op ON o.Order\_ID = op.Order\_ID**

**JOIN Product p ON op.Product\_ID = p.Product\_ID**

**WHERE p.Product\_Category\_Name = 'Arabica';**

**order\_id | order\_date | total\_amount**

**----------+------------+--------------**

**1 | 2024-10-12 | 150000.00**

**(1 row)**

### Создать представления, составленные в пункте 5 лабораторной 1.

* Представление для сотрудников отдела продаж (Sales Employee View)

**CREATE VIEW sales\_employee\_view AS**

**SELECT**

**e.employee\_ID,**

**e.name AS employee\_name,**

**o.order\_id,**

**o.order\_date,**

**o.total\_amount,**

**c.customer\_id,**

**c.name AS customer\_name,**

**c.phone\_number AS customer\_phone,**

**c.email AS customer\_mail**

**FROM**

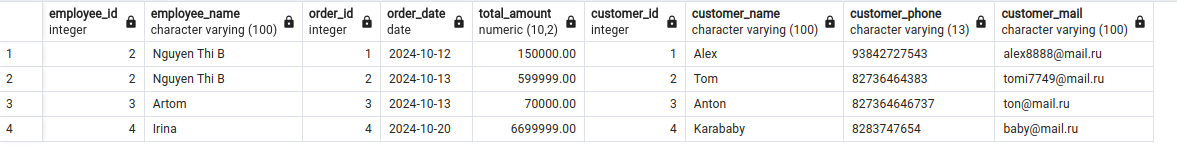
**employee e**

**JOIN**

**orders o ON e.employee\_id = o.employee\_id**

**JOIN**

**customer c ON o.customer\_id = c.customer\_id;**



* Представление для менеджера склада (Warehouse Manager View)

**CREATE VIEW Warehouse\_Manager\_View AS**

**SELECT**

**w.warehouse\_id,**

**w.address AS warehouse\_address,**

**w.status AS warehouse\_status,**

**p.product\_id,**

**p.product\_category\_name,**

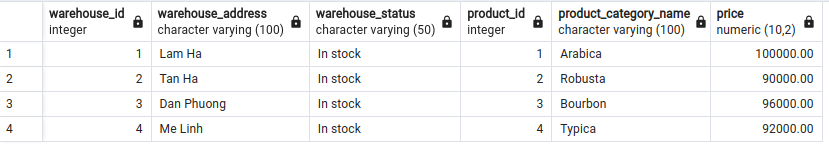
**p.price**

**FROM**

**warehouse w**

**JOIN**

**product p ON w.warehouse\_id = p.warehouse\_id;**



* Представление для клиентов (Customer View)

**CREATE VIEW Customer\_Order\_View AS**

**SELECT**

**c.customer\_id,**

**c.name AS customer\_name,**

**o.order\_id,**

**o.order\_date,**

**o.total\_amount,**

**b.bill\_id,**

**b.amount AS bill\_amount,**

**b.payment**

**FROM**

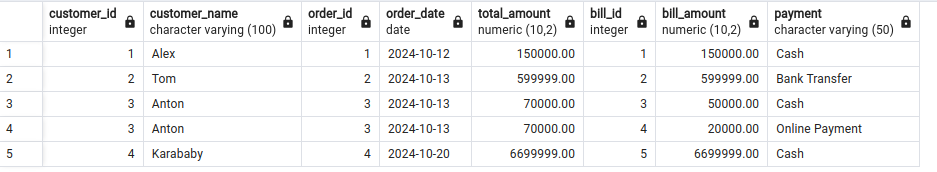
**customer c**

**JOIN**

**orders o ON c.customer\_id = o.customer\_id**

**JOIN**

**bill b ON o.order\_id = b.order\_id;**



## Вывод

В ходе выполнения лабораторной работы были получены навыки по работе с системой управления базами данных PostgreSQL. На основе инфологической модели, разработанной в предыдущей лабораторной работе, была создана база данных для системы управления кафе. Были созданы основные таблицы, включая таблицы сотрудников, клиентов, поставщиков, заказов, продуктов, складов и другие необходимые сущности.

Для каждой таблицы были добавлены необходимые атрибуты и первичные ключи. Также были реализованы внешние ключи, которые обеспечивают связь между таблицами и поддерживают целостность данных. Сгенерированы и добавлены тестовые данные для каждой таблицы, что позволяет проводить анализ данных и проверять функциональность базы данных.

Кроме того, для повышения производительности были созданы индексы на ключевые атрибуты, по которым выполняется поиск и соединение данных. Также были выполнены запросы для изменения структуры таблиц, в частности добавление и удаление столбцов.

В завершение, созданные представления и тестовые запросы показали корректность работы базы данных и возможность использования ее в системе кафе для хранения, обработки и анализа данных.