Практика по базам данных
ОТЧЕТ

Голофастов Лев
Группа 372

Предметная область: «Кофейная компания»

Реализация: PostgreSQL

Содержание

ОПИСАНИЕ СИСТЕМЫ

Требования

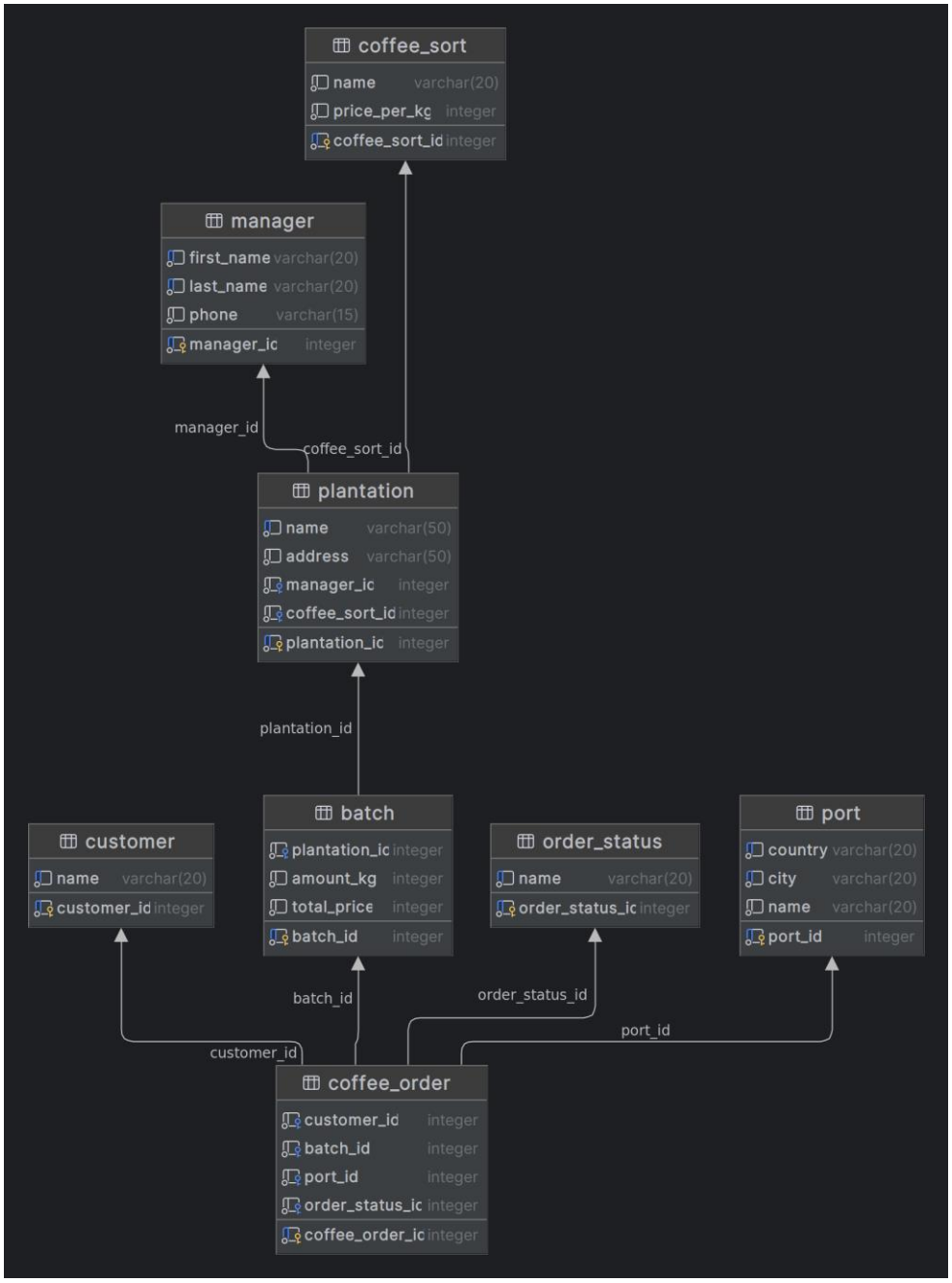
Кофейная компания выращивает, перевозит и продает перекупщикам кофе.
Компания владеет несколькими плантациями в Южной Америке, о них известны названия, адреса, данные об управляющих, сорта выращиваемого кофе и цены продукции.

Кофе упаковывается на плантации в стандартные мешки и отправляется партиями в ближайший порт.
Заказчиком партии является, как правило, какая-либо крупная компания, фасующая и продающая кофе-продукты.
Компания может использовать более чем один порт.

Нужно создать информационную систему, с помощью которой можно будет отлеживать перемещение партий кофе с плантаций в порты, поскольку компании-заказчики в любой момент могут запросить подробную информацию о состоянии своего очередного заказа.

Реализация контроля морских перевозок планируется в следующей версии системы.

Модель данных



Функциональность

Серверная часть

Хранимые процедуры\функции	*Реализация*	*Комментарии*
Получение информации об управляющем по id плантации	GetManagerInfo(plant_id INTEGER)	(id, имя, фамилия, телефон)
Подсчет количества партий, принадлежащих конкретной плантации	GetPlantationBatchCount(plant_id INT)	
Проверка статуса заказа по id	GetOrderStatus(order_id INT)	
Подсчет количества прошедших дней после оформления заказа		
Добавление заказа		
Удаление заказа		
Определение средней стоимости партии в заданном диапазоне дат		Для статистики
Подсчет количества заказов по id статуса		Для статистики

Триггеры	*Реализация*	*Комментарии*
Обновление конечной стоимости партии после обновления цены за килограмм кофе определенного сорта	Tr_After_Update_Coffee_Price	
Удаление соответствующего заказа после удаления заказчика	Tr_After_Delete_Customer	
Присвоение статуса заказа по умолчанию при отсутствии конкретно указанного	Tr_Assign_Initial_Order_Status	Для единообразия данных
Запрет на изменение или отмену уже завершенных заказов (за исключением статуса “On Hold”)		
Ограничение удаления информации о сорте кофе, если есть связанные с ним плантации		
Отслеживание заказов, которые находятся в состоянии "On Hold" на протяжении более определенного времени и автоматического обновления статуса “On Hold” на “Cancelled” для таких заказов		

Представления	*Реализация*	*Комментарии*
Клиенты и количестве их заказов	V_CustomerOrderCount	(id заказчика, название компании, количество заказов)
Партии кофе и связанные с ними портах	V_BatchPortInfo	(id партии, кол-во кг, название порта, город, страна)
Плантации и сорта кофе, выращиваемые на них	V_PlantationCoffeeSort	(название, адрес, сорт кофе, цена за кг)
Статистика по заказам кофе за определенный период времени		(год, месяц, количество заказов, общая стоимость)
Количество заказов по каждому сорту кофе		(название сорта, количество заказов)
Клиенты и информация об их заказах		(название компании, количество заказов, общая стоимость)
Сорта кофе в наличии		(название сорта кофе, название плантации, адрес плантации)

Клиентская часть

Экранные формы основные	*дополнительные*	*Реализация* *(запрос)*	*Что здесь можно использовать из серверной части*
Реестр плантаций		(03) Вывести информацию о плантации и ее управляющем для всех партий кофе Представление, показывающее информацию о плантациях и сортах кофе, выращиваемых на них	V_PlantationCoffeeSort
	Фильтры	(07) Вывести информацию о плантациях, у которых управляющего зовут David (09) Вывести информацию о плантации, на которой выращивают сорт Arabica (12) Вывести все плантации, на которых выращивается кофе с ценой выше средней цены всех	

		сортов, и отсортировать их по убыванию цены кофе Функция для подсчета количества партий, принадлежащих конкретной плантации	GetPlantationBatchCount(plant_id INT)
Список заказчиков		Представление, показывающее информацию о клиентах и количестве их заказов (05) Вывести информацию о заказчике, партии кофе и статусе заказа	V_CustomerOrderCount
	Удаление заказчика		Tr_After_Delete_Customer
	Фильтр (по хозяевам, по видам и т.д.)	(11) Вывести суммарную стоимость всех заказов для каждого заказчика, отсортированную по убыванию суммарной стоимости (10) Вывести информацию о заказчиках, у которых суммарное количество кофе в заказах больше 100 кг (04) Вывеси список всех заказчиков и плантаций, отсортированных по алфавиту	
Сорта кофе		(13) Вывести информацию о сортах кофе	
	Изменить		Tr_After_Update_Coffee_Price
Реестр менеджеров		Функция для получения информации об управляющем по id плантации	GetManagerInfo(plant_id INTEGER)
Реестр портов		(08) Вывести количество партий кофе для каждого порта Представление, показывающее информацию о партиях кофе и связанных с ними портах	V_BatchPortInfo
	Фильтры	(01) Вывести список портов, в которые были доставлены партии кофе с ценой выше средней цены всех партий	
Реестр заказов			
	Фильтр	(06) Вывести все партии кофе, у которых количество кофе больше 100 кг и цена за килограмм меньше 12	

	Новый заказ	Триггер присвоения статуса заказа по умолчанию при создании заказа	Tr_Assign_Initial_Order_Status
	Узнать статус	Функция для проверки статуса заказа по id	GetOrderStatus(order_id INT)
Служебные запросы		(02) Посчитать общее количество килограмм кофе в каждой партии	

СКРИПТЫ

Серверная часть

Хранимые процедуры и функции

```
-----------------------------------------------------------------------------------------------------------------------------------------------------------
--  Получение информации об управляющем по id плантации
-----------------------------------------------------------------------------------------------------------------------------------------------------------
CREATE OR REPLACE FUNCTION get_manager_info(plant_id INTEGER)
RETURNS TABLE(manager_id INTEGER, first_name VARCHAR, last_name VARCHAR, phone VARCHAR) AS
$$
BEGIN
    RETURN QUERY
    SELECT m.Manager_ID, m.First_Name, m.Last_Name, m.Phone
    FROM Manager AS m
    JOIN Plantation AS p ON p.Manager_ID = m.Manager_ID
    WHERE p.Plantation_ID = plant_id;
END;
$$
LANGUAGE plpgsql;

-- Пример вызова функции:
-- SELECT * FROM get_manager_info(1);

-- DROP FUNCTION get_manager_info(plant_id INTEGER);
-----------------------------------------------------------------------------------------------------------------------------------------------------------
--  Подсчет количества партий, принадлежащих конкретной плантации
-----------------------------------------------------------------------------------------------------------------------------------------------------------
CREATE OR REPLACE FUNCTION get_plantation_batch_count(plant_id INT)
RETURNS INT AS
$$
DECLARE
    batch_count INT;
BEGIN
    SELECT COUNT(*) INTO batch_count FROM Batch WHERE Plantation_ID = plant_id;
    RETURN batch_count;
END;
$$
LANGUAGE plpgsql;

-- Пример вызова функции:
-- SELECT get_plantation_batch_count(1) AS batch_count;

-- DROP FUNCTION get_plantation_batch_count(plant_id INT);
-----------------------------------------------------------------------------------------------------------------------------------------------------------
--  Проверка статуса заказа по id
-----------------------------------------------------------------------------------------------------------------------------------------------------------
CREATE OR REPLACE FUNCTION get_order_status(order_id INT)
RETURNS VARCHAR(20) AS $$
DECLARE
    status_name VARCHAR(20);
BEGIN
    SELECT Name INTO status_name
    FROM Order_Status
    JOIN Coffee_Order ON Order_Status.Order_Status_ID = Coffee_Order.Order_Status_ID
    WHERE Coffee_Order.Coffee_Order_ID = order_id;
    RETURN status_name;
END;
$$ LANGUAGE plpgsql;

-- Пример вызова функции:
-- SELECT get_order_status(1) AS status;

-- DROP FUNCTION get_order_status(order_id INT);
```

Триггеры

```
-----------------------------------------------------------------------------------------------------------------------------------------------------------
-- Триггер на обновление цены кофе:
-----------------------------------------------------------------------------------------------------------------------------------------------------------
CREATE OR REPLACE FUNCTION update_batch_total_price()
RETURNS TRIGGER AS
$$
BEGIN
   UPDATE Batch
   SET Total_Price = NEW.Price_Per_Kg * Amount_Kg
   WHERE NEW.Coffee_Sort_ID = OLD.Coffee_Sort_ID;
   RETURN NEW;
END;
$$
LANGUAGE plpgsql;

CREATE TRIGGER after_update_coffee_price
AFTER UPDATE OF Price_Per_Kg ON Coffee_Sort
FOR EACH ROW
EXECUTE FUNCTION update_batch_total_price();

-- Пример работы:
SELECT * FROM batch
   JOIN plantation ON batch.plantation_id = plantation.plantation_id
   JOIN coffee_sort ON plantation.coffee_sort_id = coffee_sort.coffee_sort_id
      WHERE coffee_sort.coffee_sort_id = 1;

UPDATE Coffee_Sort
SET Price_Per_Kg = 15
WHERE Coffee_Sort_ID = 1;

SELECT * FROM batch
   JOIN plantation ON batch.plantation_id = plantation.plantation_id
   JOIN coffee_sort ON plantation.coffee_sort_id = coffee_sort.coffee_sort_id
      WHERE coffee_sort.coffee_sort_id = 1;

-----------------------------------------------------------------------------------------------------------------------------------------------------------
-- Триггер на удаление клиента, удаляющий также соответствующие заказы
-----------------------------------------------------------------------------------------------------------------------------------------------------------
CREATE OR REPLACE FUNCTION delete_related_orders()
RETURNS TRIGGER AS
$$
BEGIN
   DELETE FROM Coffee_Order
   WHERE Customer_ID = OLD.Customer_ID;
   RETURN OLD;
END;
$$
LANGUAGE plpgsql;

CREATE TRIGGER after_delete_customer
AFTER DELETE ON Customer
FOR EACH ROW
EXECUTE FUNCTION delete_related_orders();

-- Пример работы:
SELECT count(*) from coffee_order;

DELETE FROM Customer
WHERE Customer_ID = 1;

SELECT count(*) from coffee_order;

-----------------------------------------------------------------------------------------------------------------------------------------------------------
-- Триггер присвоения статуса заказа по умолчанию при создании заказа
```

```
---------------------------------------------------------------------------------------------------------------------------------------------------
CREATE OR REPLACE FUNCTION assign_initial_order_status()
RETURNS TRIGGER AS
$$
BEGIN
   NEW.Order_Status_ID := 1;
   RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_assign_initial_order_status
BEFORE INSERT ON Coffee_Order
FOR EACH ROW
EXECUTE FUNCTION assign_initial_order_status();

-- Пример работы
SELECT coffee_order.order_status_id from coffee_order;

INSERT INTO Coffee_Order (Coffee_Order_ID, Customer_ID, Batch_ID, Port_ID, Order_Status_ID)
VALUES (5, 2, 3, 4, NULL);

SELECT coffee_order_id, order_status_id from coffee_order;
```

Представления

```
---------------------------------------------------------------------------------------------------------------------------------------------------
-- Представление, показывающее информацию о клиентах и количестве их заказов
---------------------------------------------------------------------------------------------------------------------------------------------------
CREATE VIEW V_CustomerOrderCount AS
   SELECT c.Customer_ID, c.Name, COUNT(co.Coffee_Order_ID) AS Order_Count
      FROM Customer c
         LEFT JOIN Coffee_Order co ON c.Customer_ID = co.Customer_ID
            GROUP BY c.Customer_ID, c.Name;

-- DROP VIEW V_CustomerOrderCount;

---------------------------------------------------------------------------------------------------------------------------------------------------
-- Представление, показывающее информацию о партиях кофе и связанных с ними портах
---------------------------------------------------------------------------------------------------------------------------------------------------
CREATE VIEW V_BatchPortInfo AS
   SELECT b.Batch_ID, b.Amount_Kg, p.Name AS Port_Name, p.City, p.Country
      FROM Batch b
         INNER JOIN Coffee_Order co ON b.Batch_ID = co.Batch_ID
         INNER JOIN Port p ON co.Port_ID = p.Port_ID;

-- DROP VIEW V_BatchPortInfo;

---------------------------------------------------------------------------------------------------------------------------------------------------
-- Представление, показывающее информацию о плантациях и сортах кофе, выращиваемых на них
---------------------------------------------------------------------------------------------------------------------------------------------------
CREATE VIEW V_PlantationCoffeeSort AS
   SELECT p.Name AS Plantation_Name, p.Address, cs.Name AS Coffee_Sort_Name, cs.Price_Per_Kg
      FROM Plantation p
         INNER JOIN Coffee_Sort cs ON p.Coffee_Sort_ID = cs.Coffee_Sort_ID;

-- DROP VIEW V_PlantationCoffeeSort;
```

Клиентская часть

(запросы для экранных форм и отчетов)

```
-- 1. Вывести список портов, в которые были доставлены партии кофе с ценой выше средней цены всех партий:
SELECT Name FROM Port
```

```
   WHERE Port_ID IN (
      SELECT Port_ID
         FROM Coffee_Order
            JOIN Batch ON Coffee_Order.Batch_ID = Batch.Batch_ID
               WHERE Total_Price > (
                  SELECT AVG(Total_Price)
                     FROM Batch
   )
);


-- 2. Посчитать общее количество килограмм кофе в каждой партии:
SELECT Batch_ID, SUM(Amount_Kg) AS Total_Amount
   FROM Batch
      GROUP BY Batch_ID;

-- 3. Вывести информацию о плантации и ее управляющем для всех партий кофе:
SELECT b.Batch_ID, p.Name AS Plantation_Name, m.First_Name, m.Last_Name
   FROM Batch b
      JOIN Plantation p ON b.Plantation_ID = p.Plantation_ID
      JOIN Manager m ON p.Manager_ID = m.Manager_ID;


-- 4. Вывести список всех заказчиков и плантаций, отсортированных по алфавиту:
(
   SELECT Name AS Entity_Name, 'Customer' AS Entity_Type
      FROM Customer
)
UNION
(
   SELECT Manager.First_Name AS Entity_Name, 'Manager' AS Entity_Type
      FROM Manager
 );


-- 5. Вывести информацию о заказчике, партии кофе и статусе заказа:
SELECT c.Name AS Customer_Name, b.Batch_ID, o.Name AS Order_Status
   FROM Coffee_Order co
      JOIN Customer c ON co.Customer_ID = c.Customer_ID
      JOIN Batch b ON co.Batch_ID = b.Batch_ID
      JOIN Order_Status o ON co.Order_Status_ID = o.Order_Status_ID;

-- 6. Вывести все партии кофе, у которых количество кофе больше 100 кг и цена за килограмм меньше 12:
SELECT * FROM Batch
   WHERE Amount_Kg > 100
   AND Total_Price / Amount_Kg < 12;

-- 7. Вывести информацию о плантациях, у которых управляющего зовут David:
SELECT p.Name AS Plantation_Name, p.Address
   FROM Plantation p
      LEFT JOIN Manager m ON p.Manager_ID = m.Manager_ID
         WHERE m.First_Name = 'David';


-- 8. Вывести количество партий кофе для каждого порта:
SELECT p.Name AS Port_Name, COUNT(co.Batch_ID) AS Total_Batches
   FROM Port p
      LEFT JOIN Coffee_Order co ON p.Port_ID = co.Port_ID
         GROUP BY p.Name;


-- 9. Вывести информацию о плантации, на которой выращивают сорт Arabica:
SELECT P.Name, P.Address FROM Plantation P
   JOIN Coffee_Sort C ON P.Coffee_Sort_ID = C.Coffee_Sort_ID
      WHERE C.Name = 'Arabica';
```

```
-- 10. Вывести информацию о заказчиках, у которых суммарное количество кофе в заказах больше 100 кг:
SELECT c.Name AS Customer_Name, SUM(b.Amount_Kg) AS Total_Coffee
   FROM Customer c
      JOIN Coffee_Order co ON c.Customer_ID = co.Customer_ID
      JOIN Batch b ON co.Batch_ID = b.Batch_ID
         GROUP BY c.Name
            HAVING SUM(b.Amount_Kg) > 100;


-- 11. Вывести суммарную стоимость всех заказов для каждого заказчика, отсортированную по убыванию
суммарной стоимости:
SELECT Customer.Name, SUM(Batch.Total_Price) AS Total_Cost
   FROM Customer
      JOIN Coffee_Order ON Customer.Customer_ID = Coffee_Order.Customer_ID
      JOIN Batch ON Coffee_Order.Batch_ID = Batch.Batch_ID
         GROUP BY Customer.Name
            ORDER BY Total_Cost DESC;

-- 12. Вывести все плантации, на которых выращивается кофе с ценой выше средней цены всех сортов,
-- и отсортировать их по убыванию цены кофе:
SELECT Plantation.Name, Coffee_Sort.Name, Coffee_Sort.Price_Per_Kg
   FROM Plantation
      JOIN Coffee_Sort ON Plantation.Coffee_Sort_ID = Coffee_Sort.Coffee_Sort_ID
         WHERE Coffee_Sort.Price_Per_Kg > (SELECT AVG(Price_Per_Kg) FROM Coffee_Sort)
            ORDER BY Coffee_Sort.Price_Per_Kg DESC;

-- 13. Вывести информацию о сортах кофе:
SELECT * FROM Coffee_Sort;
```

ПРИЛОЖЕНИЕ: Создание и заполнение базы данных

```
-- CREATE DATABASE coffee_company;
-- GO
-- USE coffee_company;


CREATE TABLE Manager(
                Manager_ID        INTEGER      NOT NULL,
                First_Name        VARCHAR(20)  NOT NULL,
                Last_Name         VARCHAR(20)  NOT NULL,
                Phone             VARCHAR(15)  NOT NULL,
                CONSTRAINT Manager_PK PRIMARY KEY (Manager_ID)
)
;

CREATE TABLE Coffee_Sort(
                 Coffee_Sort_ID     INTEGER      NOT NULL,
                 Name               VARCHAR(20)  NOT NULL,
                 Price_Per_Kg       INTEGER      NOT NULL,
                 CONSTRAINT Coffee_Sort_PK PRIMARY KEY (Coffee_Sort_ID)
)
;

CREATE TABLE Plantation(
                Plantation_ID      INTEGER      NOT NULL,
                Name               VARCHAR(50)  NOT NULL,
                Address            VARCHAR(50)  NOT NULL,
                Manager_ID         INTEGER      NOT NULL,
                Coffee_Sort_ID     INTEGER      NOT NULL,
                CONSTRAINT Plantation_PK PRIMARY KEY (Plantation_ID),
                CONSTRAINT Plantation_FK_Manager FOREIGN KEY (Manager_ID) REFERENCES Manager
(Manager_ID) ON DELETE CASCADE,
                CONSTRAINT Plantation_FK_Coffee_Sort FOREIGN KEY (Coffee_Sort_ID) REFERENCES
Coffee_Sort (Coffee_Sort_ID) ON DELETE CASCADE,
                CONSTRAINT Plantation_UQ_Name UNIQUE (Name)
)
;

CREATE TABLE Batch(
               Batch_ID          INTEGER        NOT NULL,
               Plantation_ID     INTEGER        NOT NULL,
               Amount_Kg         INTEGER        NOT NULL,
               Total_Price       INTEGER        NOT NULL,
               CONSTRAINT Batch_PK PRIMARY KEY (Batch_ID),
               CONSTRAINT Batch_FK_Plantation FOREIGN KEY (Plantation_ID) REFERENCES Plantation
(Plantation_ID) ON DELETE CASCADE,
               CONSTRAINT Batch_CHK_Amount_Kg CHECK (Amount_Kg > 0)
)
;

CREATE TABLE Port(
               Port_ID           INTEGER      NOT NULL,
               Country           VARCHAR(20)  NOT NULL,
               City              VARCHAR(20)  NOT NULL,
               Name              VARCHAR(20)  NOT NULL,
               CONSTRAINT Port_PK PRIMARY KEY (Port_ID),
               CONSTRAINT Port_UQ_Country_City UNIQUE (Country, City)
)
;

CREATE TABLE Customer(
                Customer_ID       INTEGER      NOT NULL,
                Name              VARCHAR(20)  NOT NULL,
                CONSTRAINT Customer_PK PRIMARY KEY (Customer_ID),
```

```
                CONSTRAINT Customer_UQ_Name UNIQUE (Name)
)
;

CREATE TABLE Order_Status(
                Order_Status_ID     INTEGER      NOT NULL,
                Name                VARCHAR(20)  NOT NULL,
                CONSTRAINT Order_Status_PK PRIMARY KEY (Order_Status_ID),
                CONSTRAINT Order_Status_UQ_Name UNIQUE (Name)
)
;

CREATE TABLE Coffee_Order(
                Coffee_Order_ID     INTEGER      NOT NULL,
                Customer_ID         INTEGER      NOT NULL,
                Batch_ID            INTEGER      NOT NULL,
                Port_ID             INTEGER      NOT NULL,
                Order_Status_ID     INTEGER,
                CONSTRAINT Coffee_Order_PK PRIMARY KEY (Coffee_Order_ID),
                CONSTRAINT Coffee_Order_FK_Customer FOREIGN KEY (Customer_ID) REFERENCES Customer
(Customer_ID) ON DELETE CASCADE,
                CONSTRAINT Coffee_Order_FK_Batch FOREIGN KEY (Batch_ID) REFERENCES Batch (Batch_ID)
ON DELETE CASCADE,
                CONSTRAINT Coffee_Order_FK_Port FOREIGN KEY (Port_ID) REFERENCES Port (Port_ID) ON
DELETE CASCADE,
                CONSTRAINT Coffee_Order_FK_Order_Status FOREIGN KEY (Order_Status_ID) REFERENCES
Order_Status (Order_Status_ID) ON DELETE CASCADE
)
;

CREATE INDEX idx_Manager_First_Name_Last_Name ON Manager (First_Name, Last_Name);

-- Заполнение таблицы Manager
INSERT INTO Manager (Manager_ID, First_Name, Last_Name, Phone)
VALUES
   (1, 'John', 'Doe', '123456789'),
   (2, 'Jane', 'Smith', '987654321'),
   (3, 'Mike', 'Johnson', '555555555'),
   (4, 'Sarah', 'Williams', '111111111'),
   (5, 'David', 'Brown', '222222222'),
   (6, 'Emily', 'Taylor', '333333333'),
   (7, 'Michael', 'Anderson', '444444444');


-- Заполнение таблицы Coffee_Sort
INSERT INTO Coffee_Sort (Coffee_Sort_ID, Name, Price_Per_Kg)
VALUES
   (1, 'Arabica', 10),
   (2, 'Robusta', 8),
   (3, 'Liberica', 12),
   (4, 'Excelsa', 9),
   (5, 'Catimor', 11),
   (6, 'Typica', 13),
   (7, 'Bourbon', 14);


-- Заполнение таблицы Plantation
INSERT INTO Plantation (Plantation_ID, Name, Address, Manager_ID, Coffee_Sort_ID)
VALUES
   (1, 'Finca El Injerto', '123 Main Street', 1, 1),
   (2, 'Hacienda La Esmeralda', '456 Elm Street', 2, 2),
   (3, 'Fazenda Santa Alina', '789 Oak Street', 3, 1),
   (5, 'Fazelina', '786 Oak Street', 5, 1),
   (4, 'Finca El Puente', '321 Pine Street', 4, 3);

-- Заполнение таблицы Batch
```

```
INSERT INTO Batch (Batch_ID, Plantation_ID, Amount_Kg, Total_Price)
VALUES
    (1, 1, 100, 1000),
    (2, 2, 200, 1500),
    (3, 3, 150, 1200),
    (4, 4, 120, 1100);

-- Заполнение таблицы Port
INSERT INTO Port (Port_ID, Country, City, Name)
VALUES
    (1, 'Brazil', 'Santos', 'Port Santos'),
    (2, 'Colombia', 'Buenaventura', 'Port Buenaventura'),
    (3, 'Vietnam', 'Ho Chi Minh City', 'Port Ho Chi Minh'),
    (4, 'Indonesia', 'Jakarta', 'Port Jakarta');

-- Заполнение таблицы Customer
INSERT INTO Customer (Customer_ID, Name)
VALUES
    (1, 'Starbucks'),
    (2, 'Dunkin Donuts'),
    (3, 'Costa Coffee'),
    (4, 'Tim Hortons'),
    (5, 'Nespresso');

-- Заполнение таблицы Order_Status
INSERT INTO Order_Status (Order_Status_ID, Name)
VALUES
    (1, 'New'),
    (2, 'Processing'),
    (3, 'Completed'),
    (4, 'Cancelled'),
    (5, 'On Hold');

-- Заполнение таблицы Coffee_Order
INSERT INTO Coffee_Order (Coffee_Order_ID, Customer_ID, Batch_ID, Port_ID, Order_Status_ID)
VALUES
    (1, 1, 3, 4, 2),
    (2, 2, 2, 3, 1),
    (3, 3, 1, 2, 3),
    (4, 4, 4, 1, 4);

-----------------------------------------------------------------------------------------------------------------------------------------------------------
-- Удаление таблиц
-----------------------------------------------------------------------------------------------------------------------------------------------------------
/*
DROP TABLE Coffee_Order;
DROP TABLE Order_Status;
DROP TABLE Customer;
DROP TABLE Port;
DROP TABLE Batch;
DROP TABLE Plantation;
DROP TABLE Coffee_Sort;
DROP TABLE Manager;
*/
```