Индивидуальный проект

Space Colony Management System

Описание: SCMS™ преобразует управление вашей космической колонией в высокоэффективный, синергетический процесс. Используя передовые технологии, включая искусственный интеллект и блокчейн для управления ресурсами, системой жизнеобеспечения и межзвездными коммуникациями, SCMS™ позволяет вашей колонии достичь новых высот процветания. Если ваша колония не будет процветать в течение трех месяцев после установки, мы вернем вам деньги!

Компания не несет ответственности за астероидные удары, инопланетные нападения, солнечные бури или другие обстоятельства непреодолимой силы.

Предполагаемые пользователи: Управляющий колонии, его заместители, инженеры, исследователи, обычные колонисты.

Пользователи и их потребности:

1. Управляющий Колонии
   a. Установление приоритетов для развития колонии
   b. Получение отчетов об эффективности
   c. Распределение ресурсов
   d. Узнавание текущего состояния безопасности

2. Инженеры
   a. Получение текущих задач и их дедлайнов
   b. Отслеживание статуса систем жизнеобеспечения
   c. Совершение запросов на материалы и ресурсы

3. Финансовый Аналитик
   a. Мониторинг экономического состояния колонии
   b. Управление тратами
   c. Получение финансовых отчетов
   d. Внесение финансовых данных в систему

4. Шеф Безопасности
   a. Назначение задач охране
   b. Создание расписания патрулирования
   c. Мониторинг состояния оборудования безопасности

5. Охрана
   a. Получение своего расписания
   b. Отмечание выполненных задач

6. Заведующий Лабораторией
   a. Установление научных приоритетов
   b. Отслеживание прогресса исследований
   c. Управление ресурсами лаборатории

7. Герольд (Коммуникатор)
   a. Получение и передача коммуникаций между колонией и Землей

8. Повара
   a. Создание меню на основе доступных продуктов
   b. Заказ необходимых продуктов
   c. Проверка состояния запасов продуктов

Таким образом, SCMS™ может стать незаменимым инструментом для эффективного управления космической колонией, обеспечивая удовлетворение потребностей всех категорий пользователей.

Функциональные требования для SCMS™:

1. Управление колонией:

   a. Управляющий Колонии должен иметь возможность устанавливать приоритеты для развития колонии.

   b. Получение отчетов об эффективности работы колонии, включая финансовые и ресурсные данные.

   c. Распределение ресурсов между различными секторами колонии.

   d. Мониторинг текущего состояния безопасности в колонии.

2. Инженеры:

   a. Получение списка текущих задач и их дедлайнов.

   b. Отслеживание статуса систем жизнеобеспечения и возможность совершать запросы на материалы и ресурсы.

3. Финансовый аналитик:

   a. Мониторинг экономического состояния колонии, включая доходы и расходы.

   b. Управление бюджетом колонии и получение финансовых отчетов.

   c. Внесение финансовых данных в систему SCMS™.

4. Шеф Безопасности:

   a. Назначение задач охране и создание расписания патрулирования.

   b. Мониторинг состояния оборудования безопасности в колонии.

5. Охрана:

   a. Получение расписания задач.

   b. Отмечание выполненных задач в системе.

6. Заведующий Лабораторией:

   a. Установление научных приоритетов для исследований.

   b. Отслеживание прогресса исследовательских проектов и управление ресурсами лаборатории.

7. Герольд (Коммуникатор):

a. Получение и передача коммуникаций между колонией и Землей.

8. Повара:

a. Создание меню на основе доступных продуктов в колонии.

b. Заказ необходимых продуктов для приготовления блюд.

c. Проверка состояния запасов продуктов в кухне.

9. Система заданий:

a. Управляющий Колонии и другие уполномоченные пользователи должны иметь возможность создавать, назначать, отслеживать и завершать задания в системе SCMS™.

b. Задания должны включать в себя описание, дедлайн и ответственного исполнителя.

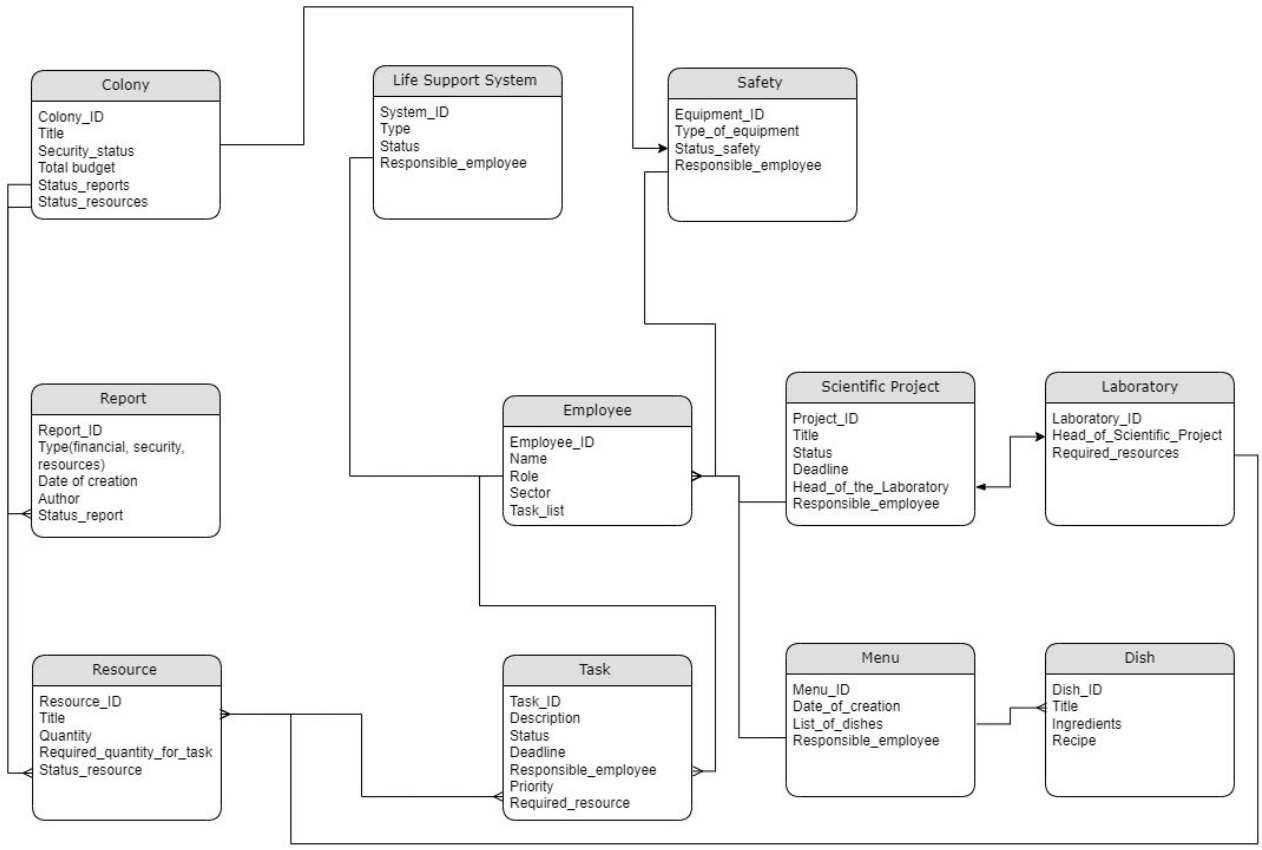
10. Прислуга:

a. Получение списка своих задач в системе.

b. Изменение статуса своих задач (назначено/в исполнении/отменено/выполнено).

c. Получение информации по каждой задаче, включая описание и дедлайн.

Такие функциональные требования позволят SCMS™ эффективно управлять космической колонией и удовлетворять потребности различных категорий пользователей.



Ограничения данных:

1. Каждый ресурс колонии может быть назначен только на один сектор одновременно для предотвращения конфликтов ресурсов.

2. Финансовое состояние колонии должно обновляться в режиме реального времени, чтобы отражать самые последние транзакции и избегать несоответствий.

3. Обновления статуса систем жизнеобеспечения должны быть снабжены временными метками для обеспечения мониторинга в реальном времени и соответствия нормам безопасности.

4. Сотрудники могут иметь только одну роль в один момент времени, чтобы обеспечить чёткую ответственность и подотчетность.

5. Назначение задач должно учитывать установленные сроки; дата начала задачи не может быть позже её дедлайна.

6. Оборудование безопасности должно подвергаться регулярным проверкам, и система должна флагать любые просроченные инспекции.

7. Научные проекты должны иметь определённый статус в каждый момент времени; возможные статусы могут включать {Начат, В Процессе, Завершён, Опубликован}.

8. Каждое сообщение, переданное Герольдом от колонии к Земле и обратно, должно быть зарегистрировано для ведения записей и обеспечения подотчетности.

9. Меню, создаваемое поваром, должно использовать только ингредиенты, доступные в текущем инвентаре колонии, чтобы предотвратить планирование блюд вокруг несуществующих ресурсов.

10. Задачи в системе должны иметь чёткий статус в любой момент времени; они не могут быть неопределенными.

11. По задаче, которая выполнена, должна быть запись со временем завершения для отслеживания показателей эффективности.

12. Любой запрос на ресурсы от сотрудников должен проходить состояние одобрения перед выполнением; возможные состояния включают {Подан на Рассмотрение, Отклонён, Одобрен}.

Эти ограничения обеспечат эффективную работу системы с четкой ответственностью и возможностью отслеживания, что критически важно в контексте управления космической колонией.

Функциональные зависимости по таблицам:

Colony
Colony_ID ⇒ Title, Security_status, Total_budget, Status_reports, Status_resources

Life Support System
System_ID ⇒ Type, Status, Responsible_employee

Safety
Equipment_ID ⇒ Type_of_equipment, Status_safety, Responsible_employee

Employee
Employee_ID ⇒ Name, Role, Sector, Task_list

Report
Report_ID ⇒ Type, Date_of_creation, Author, Status_report

Resource
Resource_ID ⇒ Title, Quantity, Required_quantity_for_task, Status_resource

Task
Task_ID ⇒ Description, Status, Deadline, Responsible_employee, Priority, Required_resource

Scientific Project
Project_ID ⇒ Title, Status, Deadline, Head_of_the_Laboratory, Responsible_employee

Laboratory
Laboratory_ID ⇒ Head_of_Scientific_Project, Required_resources

Menu
Menu_ID ⇒ Date_of_creation, List_of_dishes, Responsible_employee

Dish
Dish_ID ⇒ Title, Ingredients, Recipe

Эти функциональные зависимости подразумевают, что каждый ID в системе уникален и определяет все остальные атрибуты в записи. К примеру, если известен ID сотрудника (Employee_ID), то по этому ID можно узнать имя, роль, сектор и список задач этого сотрудника. Эти зависимости необходимы для обеспечения нормализации базы данных и предотвращения избыточности и несоответствия данных.

Трудности при отсутствии нормализации:

Отсутствие нормализации в системе управления космической колонией может привести к значительной избыточности данных и связанным с этим рискам для безопасности. Каждое изменение, будь то статус ресурса или сведения о сотруднике, потребовало бы множественных обновлений в разных таблицах. Это не только увеличивает объем работы, но и ведет к возможности несогласованности информации, что критично для систем, где точность данных напрямую связана с жизнью и здоровьем людей.

Кроме того, аномалии вставки и удаления данных могут осложнить добавление новых задач или проектов и управление ими. В сложной среде космической колонии, где каждый аспект жизнеобеспечения должен функционировать безупречно, ненормализованные данные могут привести к сбоям в работе жизненно важных систем, увеличивая вероятность катастрофических событий.

Создание таблиц:

```
-- Создание таблицы Colony
CREATE TABLE Colony (
    Colony_ID SERIAL PRIMARY KEY,
    Title VARCHAR(255),
    Security_status VARCHAR(100),
    Total_budget NUMERIC(15, 2),
    Status_reports TEXT,
    Status_resources TEXT
);

-- Создание таблицы Life Support System
CREATE TABLE Life_Support_System (
    System_ID SERIAL PRIMARY KEY,
    Type VARCHAR(255),
    Status VARCHAR(100),
    Responsible_employee INT,
    FOREIGN KEY (Responsible_employee) REFERENCES Employee(Employee_ID)
);

-- Создание таблицы Safety
CREATE TABLE Safety (
    Equipment_ID SERIAL PRIMARY KEY,
    Type_of_equipment VARCHAR(255),
    Status_safety VARCHAR(100),
    Responsible_employee INT,
    FOREIGN KEY (Responsible_employee) REFERENCES Employee(Employee_ID)
);

-- Создание таблицы Employee
CREATE TABLE Employee (
    Employee_ID SERIAL PRIMARY KEY,
    Name VARCHAR(255),
    Role VARCHAR(255),
    Sector VARCHAR(255),
    Task_list TEXT
);

-- Создание таблицы Report
CREATE TABLE Report (
    Report_ID SERIAL PRIMARY KEY,
    Type VARCHAR(100),
    Date_of_creation DATE,
    Author INT,
    Status_report VARCHAR(100),
    FOREIGN KEY (Author) REFERENCES Employee(Employee_ID)
);
```

```
-- Создание таблицы Resource
CREATE TABLE Resource (
   Resource_ID SERIAL PRIMARY KEY,
   Title VARCHAR(255),
   Quantity INT,
   Required_quantity_for_task INT,
   Status_resource VARCHAR(100)
);

-- Создание таблицы Task
CREATE TABLE Task (
   Task_ID SERIAL PRIMARY KEY,
   Description TEXT,
   Status VARCHAR(100),
   Deadline DATE,
   Responsible_employee INT,
   Priority INT,
   Required_resource INT,
   FOREIGN KEY (Responsible_employee) REFERENCES Employee(Employee_ID),
   FOREIGN KEY (Required_resource) REFERENCES Resource(Resource_ID)
);

-- Создание таблицы Scientific Project
CREATE TABLE Scientific_Project (
   Project_ID SERIAL PRIMARY KEY,
   Title VARCHAR(255),
   Status VARCHAR(100),
   Deadline DATE,
   Head_of_the_Laboratory INT,
   Responsible_employee INT,
   FOREIGN KEY (Head_of_the_Laboratory) REFERENCES Employee(Employee_ID),
   FOREIGN KEY (Responsible_employee) REFERENCES Employee(Employee_ID)
);

-- Создание таблицы Laboratory
CREATE TABLE Laboratory (
   Laboratory_ID SERIAL PRIMARY KEY,
   Head_of_Scientific_Project INT,
   Required_resources TEXT,
   FOREIGN KEY (Head_of_Scientific_Project) REFERENCES Scientific_Project(Project_ID)
);

-- Создание таблицы Menu
CREATE TABLE Menu (
   Menu_ID SERIAL PRIMARY KEY,
   Date_of_creation DATE,
   List_of_dishes TEXT,
   Responsible_employee INT,
   FOREIGN KEY (Responsible_employee) REFERENCES Employee(Employee_ID)
);
```

-- Создание таблицы Dish

```
CREATE TABLE Dish (
   Dish_ID SERIAL PRIMARY KEY,
   Title VARCHAR(255),
   Ingredients TEXT,
   Recipe TEXT
);
```

Заполнение таблиц данными:

-- Вставка данных в таблицу Colony

```
INSERT INTO Colony (Title, Security_status, Total_budget, Status_reports, Status_resources)
VALUES
('Alpha Base', 'Secure', 2000000.00, 'All systems operational', 'Adequate'),
('Beta Site', 'Alert', 1500000.00, 'Minor issues reported', 'Sufficient'),
('Gamma Outpost', 'Secure', 1800000.00, 'No issues', 'Adequate'),
('Delta Station', 'High Alert', 1200000.00, 'Security breach reported', 'Critical'),
('Epsilon Complex', 'Secure', 2500000.00, 'Routine maintenance ongoing', 'Adequate'),
('Zeta Hub', 'Moderate', 1750000.00, 'Minor technical glitches', 'Manageable'),
('Eta Facility', 'Secure', 1600000.00, 'Fully operational', 'Adequate'),
('Theta Command', 'Alert', 1400000.00, 'Environmental systems check', 'Sufficient'),
('Iota Platform', 'Secure', 2100000.00, 'New upgrades installed', 'Adequate'),
('Kappa Quarters', 'High Alert', 1100000.00, 'External threat detected', 'Critical');
```

-- Вставка данных в таблицу Employee

```
INSERT INTO Employee (Name, Role, Sector, Task_list)
VALUES
('Alice Johnson', 'Chief Engineer', 'Engineering', 'Task1, Task2'),
('Bob Brown', 'Biologist', 'Science', 'Task3, Task4'),
('Charlie Yen', 'Safety Officer', 'Security', 'Task5, Task6'),
('Diana Winters', 'Systems Analyst', 'IT', 'Task7, Task8'),
('Edward Mills', 'Medical Officer', 'Healthcare', 'Task9, Task10'),
('Fiona Gupta', 'Research Scientist', 'Science', 'Task11, Task12'),
('George Kwame', 'Maintenance Worker', 'Engineering', 'Task13, Task14'),
('Hannah Lee', 'Agricultural Specialist', 'Agriculture', 'Task15, Task16'),
('Ian Cheng', 'Logistics Coordinator', 'Operations', 'Task17, Task18'),
('Julia Espinoza', 'Communications Officer', 'Communications', 'Task19, Task20');
```

-- Вставка данных в таблицу Resource

```
INSERT INTO Resource (Title, Quantity, Required_quantity_for_task, Status_resource)
VALUES
('Oxygen', 500, 50, 'Available'),
('Water', 300, 30, 'Available'),
('Food Supplies', 250, 25, 'Available'),
('Fuel', 150, 15, 'Low'),
('Medical Kits', 100, 10, 'Available'),
('Spare Parts', 200, 20, 'Available'),
('Power Cells', 400, 40, 'Available'),
('Research Equipment', 50, 5, 'Maintenance Required'),
('Protective Suits', 75, 7, 'Available'),
('Communication Devices', 60, 6, 'Available');
```

-- Вставка данных в таблицу Life Support System

```
INSERT INTO Life_Support_System (Type, Status, Responsible_employee)
VALUES
('Air Filtration', 'Operational', 1),
('Water Recycling', 'Maintenance', 2),
('Temperature Control', 'Operational', 3),
('Gravity Simulation', 'Operational', 4),
('Waste Management', 'Operational', 5),
('Oxygen Generation', 'Critical', 6),
('Food Production', 'Operational', 7),
('Power Generation', 'Operational', 8),
('Emergency Systems', 'Operational', 9),
('Communication Systems', 'Operational', 10);
```

-- Вставка данных в таблицу Safety

```
INSERT INTO Safety (Type_of_equipment, Status_safety, Responsible_employee)
VALUES
('Fire Suppression', 'Good', 3),
('Pressure Suits', 'Good', 4),
('Emergency Alarms', 'Maintenance', 5),
('Hazard Detectors', 'Good', 6),
('Security Systems', 'Good', 7),
('Emergency Lights', 'Good', 8),
('Escape Pods', 'Maintenance', 9),
('Medical Equipment', 'Good', 10),
('Radiation Shields', 'Critical', 1),
('Robot Assistants', 'Good', 2);
```

-- Вставка данных в таблицу Report

```
INSERT INTO Report (Type, Date_of_creation, Author, Status_report)
VALUES
('Resource Usage', '2023-01-01', 1, 'Filed'),
('Safety Audit', '2023-01-15', 2, 'Pending'),
('Maintenance Log', '2023-02-01', 3, 'Filed'),
('Research Findings', '2023-02-15', 4, 'Under Review'),
('Health Report', '2023-03-01', 5, 'Filed'),
('Engineering Update', '2023-03-15', 6, 'Under Review'),
('Agricultural Report', '2023-04-01', 7, 'Filed'),
('Operational Review', '2023-04-15', 8, 'Pending'),
('Communication Logs', '2023-05-01', 9, 'Filed'),
('Education and Training', '2023-05-15', 10, 'Under Review');
```

-- Вставка данных в таблицу Task

```
INSERT INTO Task (Description, Status, Deadline, Responsible_employee, Priority,
Required_resource)
VALUES
('Check air filters', 'In Progress', '2023-06-01', 1, 1, 1),
('Water recycling analysis', 'Planned', '2023-06-15', 2, 2, 2),
('Temperature system maintenance', 'In Progress', '2023-07-01', 3, 3, 3),
('Inspect gravity simulators', 'Completed', '2023-07-15', 4, 4, 4),
('Waste system overhaul', 'Planned', '2023-08-01', 5, 5, 5),
('Oxygen generator repair', 'Urgent', '2023-08-15', 6, 1, 6),
('Harvest crops in hydroponics', 'In Progress', '2023-09-01', 7, 2, 7),
('Solar panel alignment', 'Completed', '2023-09-15', 8, 3, 8),
('Emergency drill coordination', 'Planned', '2023-10-01', 9, 4, 9),
('Communication system upgrade', 'In Progress', '2023-10-15', 10, 5, 10);
```

-- Вставка данных в таблицу Scientific Project

```
INSERT INTO Scientific_Project (Title, Status, Deadline, Head_of_the_Laboratory,
Responsible_employee)
VALUES
('Mars Soil Analysis', 'Active', '2023-11-01', 1, 2),
('Space Radiation Effects', 'Active', '2023-11-15', 3, 4),
('Hydroponic Farming Techniques', 'Concluded', '2023-12-01', 5, 6),
('Zero Gravity Health Studies', 'Active', '2023-12-15', 7, 8),
('Asteroid Mining Prospects', 'Planning', '2024-01-01', 9, 10),
('Energy Efficiency in Space', 'Concluded', '2024-01-15', 2, 1),
('Robotic Automation', 'Active', '2024-02-01', 4, 3),
('Deep Space Communication', 'Planning', '2024-02-15', 6, 5),
('Alien Flora Studies', 'Active', '2024-03-01', 8, 7),
('Spacecraft Design Innovations', 'Concluded', '2024-03-15', 10, 9);
```

-- Вставка данных в таблицу Laboratory

```
INSERT INTO Laboratory (Head_of_Scientific_Project, Required_resources)
VALUES
(1, 'Lab Equipment, Test Samples'),
(2, 'Radiation Detectors, Safety Gear'),
(3, 'Seeds, Growth Mediums'),
(4, 'Medical Instruments, Test Kits'),
(5, 'Mining Tools, Geology Kits'),
(6, 'Power Analyzers, Efficiency Meters'),
(7, 'Robotics Kits, AI Modules'),
(8, 'Satellite Dishes, Communication Arrays'),
(9, 'Biological Samples, Microscopes'),
(10, 'Design Software, Engineering Tools');
```

-- Вставка данных в таблицу Menu

```
INSERT INTO Menu (Date_of_creation, List_of_dishes, Responsible_employee)
VALUES
('2023-06-01', 'Space Pasta, Martian Salad', 1),
('2023-06-15', 'Lunar Curry, Asteroid Stew', 2),
('2023-07-01', 'Galactic Pizza, Comet Soup', 3),
('2023-07-15', 'Solar Sandwiches, Meteorite Meatballs', 4),
('2023-08-01', 'Orbit Omelettes, Nebula Noodles', 5),
('2023-08-15', 'Starlight Sushi, Eclipse Eclairs', 6),
('2023-09-01', 'Photon Pie, Quantum Quiche', 7),
('2023-09-15', 'Black Hole Burgers, Supernova Spaghetti', 8),
('2023-10-01', 'Cosmic Cakes, Void Veggies', 9),
('2023-10-15', 'Planetary Pancakes, Rocket Risotto', 10);
```

```
INSERT INTO Dish (Title, Ingredients, Recipe)
VALUES
('Space Pasta', 'Pasta, Tomato Sauce', 'Boil pasta; add sauce'),
('Martian Salad', 'Lettuce, Tomatoes, Cucumbers', 'Chop vegetables; mix with dressing'),
('Lunar Curry', 'Chicken, Curry Powder, Rice', 'Cook chicken; add curry; serve with rice'),
('Asteroid Stew', 'Beef, Potatoes, Carrots', 'Brown beef; simmer with vegetables'),
('Galactic Pizza', 'Dough, Cheese, Pepperoni', 'Prepare dough; add toppings; bake'),
('Comet Soup', 'Chicken, Noodles, Vegetables', 'Boil chicken; add noodles and vegetables'),
('Solar Sandwiches', 'Bread, Ham, Cheese', 'Assemble ham and cheese on bread'),
('Meteorite Meatballs', 'Ground Beef, Spices, Tomato Sauce', 'Form meatballs; cook in sauce'),
('Orbit Omelettes', 'Eggs, Cheese, Ham', 'Whisk eggs; cook with ham and cheese'),
('Nebula Noodles', 'Noodles, Vegetables, Soy Sauce', 'Cook noodles; stir-fry with vegetables'),
('Starlight Sushi', 'Rice, Fish, Seaweed', 'Roll sushi with fish and rice'),
('Eclipse Eclairs', 'Pastry Dough, Cream, Chocolate', 'Bake pastry; fill with cream; top with
chocolate'),
('Photon Pie', 'Crust, Berries, Sugar', 'Prepare crust; fill with berries; bake'),
('Quantum Quiche', 'Eggs, Cheese, Spinach', 'Mix eggs and cheese; add spinach; bake'),
('Black Hole Burgers', 'Beef, Buns, Lettuce, Tomato', 'Grill beef; assemble burgers with toppings'),
('Supernova Spaghetti', 'Spaghetti, Garlic, Olive Oil', 'Cook spaghetti; sauté with garlic and oil'),
('Cosmic Cakes', 'Flour, Sugar, Eggs', 'Mix ingredients; bake'),
('Void Veggies', 'Assorted Vegetables, Olive Oil, Herbs', 'Roast vegetables with herbs'),
('Planetary Pancakes', 'Flour, Eggs, Milk', 'Mix batter; cook on griddle'),
('Rocket Risotto', 'Rice, Chicken Broth, Parmesan', 'Cook rice in broth; stir in cheese');
```

Изменения в таблицах:

Чтобы связать между собой таблицы Menu и Dish, нам нужно ввести внешний ключ в одну из этих таблиц. Обычно это делается путем добавления поля внешнего ключа в таблицу Dish, которое будет указывать на Menu_ID из таблицы Menu. Это позволит нам связать каждое блюдо с определенным меню.

Сначала обновим структуру таблицы Dish, добавив внешний ключ, а затем вставим данные, уже учитывая эту связь.

Изменение структуры таблицы Dish:

```
ALTER TABLE Dish
ADD COLUMN Menu_ID INT,
ADD FOREIGN KEY (Menu_ID) REFERENCES Menu(Menu_ID);
```

Удаление столбца List_of_dishes из таблицы Menu:

```
ALTER TABLE Menu
DROP COLUMN List_of_dishes;
```

Очистка всей таблицы Menu (удаление всех записей):

```
DELETE FROM Menu;
```

Новое заполнение таблицы Menu:

```
INSERT INTO Menu (Date_of_creation, Responsible_employee)
VALUES
('2023-06-01', 1),
('2023-06-15', 2),
('2023-07-01', 3),
('2023-07-15', 4),
('2023-08-01', 5),
('2023-08-15', 6),
('2023-09-01', 7),
('2023-09-15', 8),
('2023-10-01', 9),
('2023-10-15', 10);
```

Заполнение таблицы Dish:

```
INSERT INTO Dish (Title, Ingredients, Recipe, menu_id)
VALUES
('Space Pasta', 'Pasta, Tomato Sauce', 'Boil pasta; add sauce', 11),
('Martian Salad', 'Lettuce, Tomatoes, Cucumbers', 'Chop vegetables; mix with dressing', 11),
('Lunar Curry', 'Chicken, Curry Powder, Rice', 'Cook chicken; add curry; serve with rice', 12),
('Asteroid Stew', 'Beef, Potatoes, Carrots', 'Brown beef; simmer with vegetables', 12),
('Galactic Pizza', 'Dough, Cheese, Pepperoni', 'Prepare dough; add toppings; bake', 13),
('Comet Soup', 'Chicken, Noodles, Vegetables', 'Boil chicken; add noodles and vegetables', 13),
('Solar Sandwiches', 'Bread, Ham, Cheese', 'Assemble ham and cheese on bread', 14),
('Meteorite Meatballs', 'Ground Beef, Spices, Tomato Sauce', 'Form meatballs; cook in sauce',
14),
('Orbit Omelettes', 'Eggs, Cheese, Ham', 'Whisk eggs; cook with ham and cheese', 15),
('Nebula Noodles', 'Noodles, Vegetables, Soy Sauce', 'Cook noodles; stir-fry with vegetables', 15),
('Starlight Sushi', 'Rice, Fish, Seaweed', 'Roll sushi with fish and rice', 16),
('Eclipse Eclairs', 'Pastry Dough, Cream, Chocolate', 'Bake pastry; fill with cream; top with
chocolate', 16),
('Photon Pie', 'Crust, Berries, Sugar', 'Prepare crust; fill with berries; bake', 17),
('Quantum Quiche', 'Eggs, Cheese, Spinach', 'Mix eggs and cheese; add spinach; bake', 17),
('Black Hole Burgers', 'Beef, Buns, Lettuce, Tomato', 'Grill beef; assemble burgers with toppings',
18),
('Supernova Spaghetti', 'Spaghetti, Garlic, Olive Oil', 'Cook spaghetti; sauté with garlic and oil',18),
('Cosmic Cakes', 'Flour, Sugar, Eggs', 'Mix ingredients; bake', 19),
('Void Veggies', 'Assorted Vegetables, Olive Oil, Herbs', 'Roast vegetables with herbs', 19),
('Planetary Pancakes', 'Flour, Eggs, Milk', 'Mix batter; cook on griddle', 20),
('Rocket Risotto', 'Rice, Chicken Broth, Parmesan', 'Cook rice in broth; stir in cheese', 20);
```

Удаление записей с нулевым id, которые были добавлены изначально, до внесения всех изменений:

```
DELETE FROM Dish
WHERE Menu_ID IS NULL;
```

Проверка работы новой схемы:

Вывести все блюда из Меню № 15:

```
SELECT m.Date_of_creation, d.Title
FROM Menu m
JOIN Dish d ON m.Menu_ID = d.Menu_ID
WHERE m.Menu_ID = 15;
```

Запросы:

Запрос на выборку всех задач, просроченных по дедлайну:

```
SELECT * FROM Task
WHERE Deadline < CURRENT_DATE AND Status <> 'Completed';
```

Подсчет количества сотрудников в каждой роли:

```
SELECT Role, COUNT(*) FROM Employee
GROUP BY Role;
```

Список задач с приоритетом, сгруппированный по сотрудникам:

```
SELECT e.Name, t.Description, t.Priority FROM Task t
JOIN Employee e ON t.Responsible_employee = e.Employee_ID
ORDER BY e.Name, t.Priority DESC;
```

Запрос на выборку всех ресурсов, количество которых ниже порогового значения:

```
SELECT * FROM Resource
WHERE Quantity < (SELECT AVG(Quantity) FROM Resource);
```

Вывод всех задач, для которых необходимы ресурсы, находящиеся в критическом состоянии:

```
SELECT t.* FROM Task t
JOIN Resource r ON t.Required_resource = r.Resource_ID
WHERE r.Status_resource = 'Critical';
```

Список всех отчетов, составленных конкретным сотрудником:

```
SELECT * FROM Report
WHERE Author = (SELECT Employee_ID FROM Employee WHERE Name = 'Diana Winters');
```

Обновление статуса безопасности для всех колоний на основе определенных критериев:

```
UPDATE Colony SET Security_status = 'High Alert'
WHERE Total_budget < 1500000.00;
```

Список всех лабораторных проектов с указанием руководителя и ответственного сотрудника:

```
SELECT p.*, e1.Name AS Head, e2.Name AS Responsible FROM Scientific_Project p
JOIN Employee e1 ON p.Head_of_the_Laboratory = e1.Employee_ID
JOIN Employee e2 ON p.Responsible_employee = e2.Employee_ID;
```

Подсчет общего количества каждого типа ресурсов, используемых в задачах:
```
SELECT r.Title, SUM(t.Required_resource) FROM Task t
JOIN Resource r ON t.Required_resource = r.Resource_ID
GROUP BY r.Title;
```

Вывод всех блюд, которые можно приготовить, исходя из текущих запасов продуктов:
```
SELECT DISTINCT d.Title FROM Dish d
JOIN Resource r ON d.Ingredients LIKE CONCAT('%', r.Title, '%')
WHERE r.Quantity > 0;
```

Выборка сотрудников, у которых нет текущих задач:
```
SELECT * FROM Employee e
WHERE NOT EXISTS (SELECT 1 FROM Task t WHERE t.Responsible_employee = e.Employee_ID);
```

Вывод списка задач, которые должны быть завершены в следующем квартале:
```
SELECT * FROM Task
WHERE Deadline BETWEEN CURRENT_DATE AND (CURRENT_DATE + INTERVAL '3 months');
```

Определение колоний с самым высоким и самым низким бюджетом:
```
SELECT
MAX(Total_budget), MIN(Total_budget)
FROM Colony;
```

Список отчетов по системам жизнеобеспечения за последние 6 месяцев:
```
SELECT * FROM Report
WHERE Type = 'Life Support' AND Date_of_creation > CURRENT_DATE - INTERVAL '6 months';
```

Обновление статуса всех задач, связанных с определенным проектом:
```
UPDATE Task SET Status = 'In Progress'
WHERE Task_ID IN (SELECT t.Task_ID FROM Task t
JOIN Scientific_Project p ON t.Responsible_employee = p.Responsible_employee
WHERE p.Title = 'Mars Soil Analysis');
```

Сравнение количества выполненных и невыполненных задач по каждому сектору:
```
SELECT e.Sector, SUM(CASE WHEN t.Status = 'Completed' THEN 1 ELSE 0 END) AS
Completed_Tasks, SUM(CASE WHEN t.Status <> 'Completed' THEN 1 ELSE 0 END) AS
Pending_Tasks FROM Employee e JOIN Task t ON e.Employee_ID = t.Responsible_employee
GROUP BY e.Sector;
```

Список всех задач и ресурсов, необходимых для их выполнения, с количеством каждого ресурса:
```
SELECT t.Description, r.Title, r.Quantity FROM Task t
JOIN Resource r ON t.Required_resource = r.Resource_ID;
```

Список ресурсов, которые необходимо заказать (количество ниже заданного порога):
```
SELECT Title, Quantity FROM Resource
WHERE Quantity < (SELECT AVG(Quantity) * 0.5 FROM Resource);
```

Получение списка всех блюд, в которые входит определенный ингредиент, и колонии, где они подаются:

```
SELECT d.Title, c.Title AS Colony_Name FROM Dish d
JOIN Menu m ON d.Menu_ID = m.Menu_ID
JOIN Colony c ON m.Responsible_employee = c.Colony_ID
WHERE d.Ingredients LIKE '%Tomato%';
```

Вывод колоний, где нет ни одного сотрудника из определенного сектора:

```
SELECT c.Title FROM Colony c
WHERE NOT EXISTS (SELECT 1 FROM Employee e
WHERE e.Sector = 'IT' AND e.Employee_ID = c.Colony_ID);
```

Все можно проверить в базе данных:

User: funny56fox
Password: YZEGou3TPmM7
База Данных: Space_Colony_Management_System
Хост: ep-polished-poetry-56003604.eu-central-1.aws.neon.tech
Порт: 5432

Проект выполнен студентом 3 курса ПАД:
Лавров Андрей Сергеевич
2023