

PGHOST='ep-sparkling-thunder-15701210.eu-central-1.aws.neon.tech'
PGDATABASE='exam_db'
PGUSER='andrei.lavrov2014'
PGPASSWORD='Hlt0NhP4WsKg'

Andrei Lavrov, Group 212

Дети: Хранит информацию о каждом ребенке.

Адреса: Содержит информацию об адресах, где проживают дети.

Письма: Хранит информацию о письмах, отправленных детьми.

Категории подарков: Содержит информацию о категориях подарков.

Подарки: Содержит информацию о подарках.

Статусы подарков: Описывает различные состояния подарков.

Пользователи: Содержит информацию об эльфах и Деде Морозе.

Права доступа: Определяет уровень доступа разных пользователей к различным элементам данных.

-- Таблица "Дети"

```
CREATE TABLE Children (  
    child_id SERIAL PRIMARY KEY,  
    name VARCHAR(255) NOT NULL,  
    age INT CHECK (age <= 18),  
    behavior_status BOOLEAN NOT NULL,  
    address_id INT NOT NULL,  
    FOREIGN KEY (address_id) REFERENCES Addresses(address_id)  
);
```

-- Таблица "Адреса"

```
CREATE TABLE Addresses (  
    address_id SERIAL PRIMARY KEY,  
    country VARCHAR(255) NOT NULL,  
    region VARCHAR(255) NOT NULL,  
    house_number INT NOT NULL,  
    apartment_number INT,  
    floor INT,  
    postal_code VARCHAR(20) NOT NULL  
);
```

-- Таблица "Письма"

```
CREATE TABLE Letters (  
    letter_id SERIAL PRIMARY KEY,  
    child_id INT NOT NULL,  
    send_date DATE NOT NULL,  
    receive_date DATE NOT NULL,  
    letter_text TEXT,  
    storage_slot INT,  
    FOREIGN KEY (child_id) REFERENCES Children(child_id)  
);
```

-- Таблица "Категории подарков"

```
CREATE TABLE GiftCategories (  
    category_id SERIAL PRIMARY KEY,  
    name VARCHAR(255) UNIQUE NOT NULL  
);
```

-- Таблица "Подарки"

```
CREATE TABLE Gifts (  
    gift_id SERIAL PRIMARY KEY,  
    name VARCHAR(255) NOT NULL,  
    child_id INT,  
    letter_id INT,  
    category_id INT,  
    status_id INT,  
    FOREIGN KEY (child_id) REFERENCES Children(child_id),  
    FOREIGN KEY (letter_id) REFERENCES Letters(letter_id),  
    FOREIGN KEY (category_id) REFERENCES GiftCategories(category_id),  
    FOREIGN KEY (status_id) REFERENCES GiftStatuses(status_id)  
);
```

-- Таблица "Статусы подарков"

```
CREATE TABLE GiftStatuses (  
    status_id SERIAL PRIMARY KEY,  
    status_name VARCHAR(255) NOT NULL  
);
```

-- Таблица "Пользователи"

```
CREATE TABLE Users (  
    user_id SERIAL PRIMARY KEY,  
    name VARCHAR(255) NOT NULL,  
    role VARCHAR(255) NOT NULL -- Дед Мороз или эльф  
);
```

-- Таблица "Права доступа"

```
CREATE TABLE AccessRights (  
    right_id SERIAL PRIMARY KEY,  
    user_id INT NOT NULL,  
    access_level VARCHAR(255) NOT NULL, -- Чтение, добавление, изменение  
    data_element VARCHAR(255) NOT NULL, -- Письма, дети, подарки, категории, адреса  
    FOREIGN KEY (user_id) REFERENCES Users(user_id)  
);
```

Затем я заполнил таблицы тестовыми данными, чтобы было немного удобнее выполнять запросы.

После того, как все таблицы были заполнены приступаем к выполнению запросов из задания.

Task 2

SELECT

A.country AS Страна,
GS.status_name AS Состояние,
COALESCE(COUNT(G.gift_id), 0) AS Количество

FROM

(Addresses A
CROSS JOIN GiftStatuses GS)

LEFT JOIN Children C ON A.address_id = C.address_id

LEFT JOIN Gifts G ON C.child_id = G.child_id AND G.status_id = GS.status_id

GROUP BY

A.country, GS.status_name

ORDER BY

A.country, GS.status_name;

	abc Страна	abc Состояние	123 Количество
1	Россия	Доставлен	1
2	Россия	Назначен	2
3	Россия	Подготовлен	1
4	Россия	Свободен	2
5	Россия	Упакован	1

Task 3

Для оптимизации производительности запроса, который возвращает количество подарков для детей из каждой страны в каждом из возможных состояний, можно рассмотреть создание следующих индексов:

1. Индекс на столбце address_id в таблице Children: Это ускорит операцию соединения (JOIN) между таблицами Children и Addresses, так как address_id является внешним ключом в таблице Children.

```
CREATE INDEX idx_children_address_id ON Children(address_id);
```

2. Индекс на столбце country в таблице Addresses: Этот индекс поможет ускорить группировку и фильтрацию по стране.

```
CREATE INDEX idx_addresses_country ON Addresses(country);
```

3. Индекс на столбцах child_id и status_id в таблице Gifts: Поскольку запрос соединяет таблицу Gifts с таблицами Children и GiftStatuses, индекс по этим столбцам может ускорить выборку данных.

```
CREATE INDEX idx_gifts_child_id_status_id ON Gifts(child_id, status_id);
```

4. Индекс на столбце status_id в таблице GiftStatuses: Этот индекс может ускорить соединение таблицы GiftStatuses с таблицей Gifts.

```
CREATE INDEX idx_gift_statuses_status_id ON GiftStatuses(status_id);
```

Эти индексы помогут базе данных быстрее находить и фильтровать данные, уменьшая время выполнения запроса, особенно при работе с большими объемами данных.

Task 4

SELECT

C.name AS Имя,

C.age AS Возраст,

COALESCE(L.letter_count, 0) AS Количество_писем,

COALESCE(G.gift_count, 0) AS Количество_доставленных_подарков

FROM

Children C

LEFT JOIN

(SELECT child_id, COUNT(*) AS letter_count

FROM Letters

GROUP BY child_id) L ON C.child_id = L.child_id

LEFT JOIN

(SELECT child_id, COUNT(*) AS gift_count

FROM Gifts

WHERE status_id = 5

GROUP BY child_id) G ON C.child_id = G.child_id

WHERE

COALESCE(L.letter_count, 0) > COALESCE(G.gift_count, 0);

	Имя	Возраст	Количество_писем	Количество_доставленных_подарков
1	Екатерина	8	2	0
2	Иван	12	1	0
3	Анна	6	1	0
4	Николай	15	1	0
5	Алексей	10	1	0

Task 5

Для ускорения запроса, который возвращает список детей, отправивших больше писем, чем получили доставленных подарков, можно рассмотреть создание следующих индексов:

1. Индекс на столбце `child_id` в таблице `Letters`: Поскольку в запросе осуществляется подсчет количества писем для каждого ребенка, индекс на `child_id` в таблице `Letters` ускорит группировку и агрегацию данных.

```
CREATE INDEX idx_letters_child_id ON Letters(child_id);
```

2. Индекс на столбце `child_id` в таблице `Children`: Это ускорит соединение (JOIN) с подзапросами.

```
CREATE INDEX idx_children_child_id ON Children(child_id);
```

Task 6

Предположим, что у нас есть столбец elf_id в таблице Gifts, представляющий эльфа, который сформировал подарок.

Но у нас такого нет, поэтому необходимо изменить таблицу, добавив туда elf_id

```
ALTER TABLE Gifts  
ADD COLUMN elf_id INT;
```

Затем необходимо его заполнить:

```
UPDATE Gifts  
SET elf_id = (SELECT user_id FROM Users WHERE role = 'Эльф' ORDER BY RANDOM() LIMIT  
1);
```

И наконец сам запрос

```
SELECT  
    C.name AS Ребенок,  
    L.letter_id AS Письмо,  
    G.name AS Подарок,  
    U.name AS Эльф  
FROM  
    Gifts G  
JOIN  
    Gifts G2 ON G.letter_id = G2.letter_id AND G.gift_id <> G2.gift_id AND G.name = G2.name  
JOIN  
    Users U ON G.elf_id = U.user_id  
JOIN  
    Users U2 ON G2.elf_id = U2.user_id AND U.user_id <> U2.user_id  
JOIN  
    Letters L ON G.letter_id = L.letter_id  
JOIN  
    Children C ON L.child_id = C.child_id  
GROUP BY  
    C.name, L.letter_id, G.name, U.name  
HAVING  
    COUNT(*) > 1;
```


Task 7

Для ускорения запроса на поиск дубликатов подарков, основываясь на данных в таблице Gifts, следует рассмотреть создание следующих индексов:

1. Комбинированный индекс на столбцах name и letter_id:

Этот индекс будет полезен, так как запрос на дубликаты будет искать совпадения по названию подарка и идентификатору письма, чтобы определить дублирующиеся записи.

```
CREATE INDEX idx_gifts_name_letter_id ON Gifts(name, letter_id);
```

2. Индекс на столбце elf_id:

Поскольку запрос также фильтрует результаты по различным elf_id, индекс по этому столбцу поможет ускорить соответствующую фильтрацию.

```
CREATE INDEX idx_gifts_elf_id ON Gifts(elf_id);
```

3. Индекс на столбце gift_id если он уже не является первичным ключом:

Индекс по первичному ключу обычно создается автоматически, но если gift_id не был определен как первичный ключ, то его стоит добавить для ускорения операций поиска и соединения, особенно если таблица Gifts часто используется в запросах.

```
CREATE INDEX idx_gifts_gift_id ON Gifts(gift_id);
```