

1 Завдання: Створити SELECT запит на базі таблиці planes, відсортувати записи за полем worked_hours в порядку зменшення та вивести тільки ті записи, у яких модель літака Airbus A320 або Boeing 787.

Пояснення:

Для цього запиту використовуємо команду SELECT для вибору потрібних полів з таблиці planes. Використовуємо умову WHERE для відбору тільки тих записів, у яких модель літака Airbus A320 або Boeing 787. Для сортування записів використовуємо команду ORDER BY з параметром worked_hours у порядку зменшення.

SQL-код:

```
SELECT id, board_number, model, worked_hours, seats, capacity
```

```
FROM planes
```

```
WHERE model='Airbus A320' OR model='Boeing 787'
```

```
ORDER BY worked_hours DESC;
```

2 Завдання: Виконати SELECT запит для виведення списку рейсів разом з кількістю вільних місць на цих рейсах.

Пояснення: Для отримання кількості вільних місць на кожному рейсі, потрібно відняти кількість проданих квитків від загальної кількості місць на літаку, що виконує цей рейс.

SQL-код:

```
SELECT f.id AS flight_id, p.board_number, f.departure_point, f.destination_point, f.departure_time,  
f.landing_time,
```

```
    (p.seats - f.sold_tickets_number) AS free_seats
```

```
FROM flights f
```

```
JOIN planes p ON f.plane_id = p.id;
```

3 Завдання: Створити запит, що повертає інформацію про всі рейси, в яких літак з моделлю "Boeing 737" і кількість проданих квитків менше 100 або відправлення відбулося пізніше 2023-03-21 00:00:00.

Пояснення: Для цього запиту потрібно використовувати таблиці "flights" та "planes". Поля, які нам потрібні з таблиці "flights": departure_point, destination_point, departure_time, landing_time, sold_tickets_number. З таблиці "planes" потрібне поле model. Потрібно зв'язати ці дві таблиці за допомогою полів plane_id (з таблиці "flights") та id (з таблиці "planes"). Умови для запиту: model = 'Boeing 737' AND (sold_tickets_number < 100 OR departure_time > '2023-03-21 00:00:00').

SQL-код:

```
SELECT f.departure_point, f.destination_point, f.departure_time, f.landing_time, f.sold_tickets_number
FROM flights f
JOIN planes p ON f.plane_id = p.id
WHERE p.model = 'Boeing 737' AND (f.sold_tickets_number < 100 OR f.departure_time > '2023-03-21
00:00:00')
ORDER BY f.departure_time;
```

4 Завдання: Вибрати всі рейси разом з моделлю літака та прізвищем пілота, якщо відомості про пілота є, використовуючи зовнішній з'єднання Outer Join.

Пояснення: Для цього запиту потрібно об'єднати три таблиці: flights, planes, та pilots. З'єднання між таблицями flights та planes можна зробити з використанням INNER JOIN за допомогою збігу значень стовпця plane_id з flights та id з planes. Для з'єднання таблиці pilots до результату, можна використати LEFT OUTER JOIN за допомогою збігу значень стовпця crew_member_id з pilots та id з crew_members.

SQL-код:

```
SELECT flights.*, planes.model, pilots.last_name
```

```
FROM flights
```

```
INNER JOIN planes ON flights.plane_id = planes.id
```

```
LEFT OUTER JOIN pilots ON planes.id = pilots.crew_member_id;
```

5 Завдання: Напишіть SQL-запити з використанням операторів Like, Between, In, Exists, All, Any для вибірки даних з таблиць.

Пояснення:

1. Оператор Like дозволяє виконати пошук строк, які задовольняють певному шаблону. У прикладі нижче вибираються всі літаки, модель яких починається зі слова "Boeing":

```
SELECT * FROM planes WHERE model LIKE 'Boeing%';
```

2. Оператор Between дозволяє виконати пошук значень, які належать певному діапазону. У прикладі нижче вибираються всі рейси, які відбулися між 2023-03-22 та 2023-03-24:

```
SELECT * FROM flights WHERE departure_time BETWEEN '2023-03-22' AND '2023-03-24';
```

3. Оператор In дозволяє виконати пошук значень, які співпадають зі списком заданих значень. У прикладі нижче вибираються всі рейси, що відбулися на літаках з ідентифікаторами 1, 3 та 5:

```
SELECT * FROM flights WHERE plane_id IN (1, 3, 5);
```

4. Оператор Exists дозволяє перевірити наявність результату запиту до іншої таблиці. У прикладі нижче вибираються всі літаки, для яких існують записи в таблиці flights:

```
SELECT * FROM planes WHERE EXISTS (SELECT * FROM flights WHERE planes.id = flights.plane_id);
```

5. Оператор All дозволяє виконати порівняння з усіма значеннями заданого підзапиту, а оператор Any - з хоча б одним з них. У прикладі нижче вибираються всі пілоти, які дозволені керувати всіма літаками, що знаходяться в таблиці planes:

```
SELECT * FROM pilots WHERE allowed_planes = (SELECT GROUP_CONCAT(model SEPARATOR  
, ') FROM planes) AND last_flight_date IS NOT NULL;
```

6 Завдання: Вивести кількість рейсів для кожної моделі літака.

Пояснення: Для виконання цього завдання потрібно зібрати дані з таблиці `planes` та `flights`, де визначити кількість рейсів для кожної моделі літака, використовуючи підсумовування та групування за моделлю літака.

SQL-код:

```
SELECT planes.model, COUNT(flights.id) AS number_of_flights  
FROM planes  
JOIN flights ON planes.id = flights.plane_id  
GROUP BY planes.model;
```

Виконується операція `SELECT` для вибору полів `model` з таблиці `planes` та `id` з таблиці `flights`.

Виконується операція `JOIN` для об'єднання таблиць `planes` та `flights` по полю `id` з `planes` та `plane_id` з `flights`.

Виконується операція групування `GROUP BY` за полем `model` з таблиці `planes`.

Виконується підсумовування `COUNT(flights.id)` для обчислення кількості рейсів для кожної моделі літака.

Результат виводиться на екрані.

7 Завдання: Напишіть запит, який вибере усі рейси, які здійснюються на літаках, що мають робочі години більше 8000.

Пояснення:

Потрібно вибрати всі рейси з таблиці flights

Рейс повинен мати літак, який знаходиться у таблиці planes

Літак повинен мати робочі години більше 8000

Тому використаємо підзапит у WHERE, щоб вибрати всі літаки з більш ніж 8000 годинами

SQL-код:

SELECT *

FROM flights

WHERE plane_id IN (

SELECT id

FROM planes

WHERE worked_hours > 8000

);

8 Завдання: Вибрати всі рейси, що виконуються на літаках моделі Boeing 737.

Пояснення: Для вирішення завдання необхідно використати під-запит у частині FROM, який вибере всі id літаків моделі Boeing 737 з таблиці planes, а потім в основному запиті використати цей список id для вибору всіх рейсів з таблиці flights, що виконуються на цих літаках.

SQL-код:

```
SELECT f.*  
  
FROM flights f  
  
WHERE f.plane_id IN (  
  
    SELECT p.id  
  
    FROM planes p  
  
    WHERE p.model = 'Boeing 737'  
  
)
```

Пояснення коду: Вибираємо всі рейси з таблиці flights за допомогою основного запиту SELECT f.* FROM flights f, де f.* вибирає всі стовпці з таблиці flights. Умова WHERE вказує на те, що plane_id повинен бути одним з id літаків, вибраних у підзапиті. У підзапиті ми вибираємо всі id літаків моделі Boeing 737 з таблиці planes, за допомогою запиту SELECT p.id FROM planes p WHERE p.model = 'Boeing 737'.

9 Завдання: Повернути список пілотів та їхніх дозволених літаків у вигляді ієрархії за допомогою рекурсивного запиту.

Пояснення: Для цього запиту будемо використовувати рекурсивний запит з операцією об'єднання, щоб злити дані пілотів з їхніми дозволеними літаками в одну таблицю. Потім використаємо функцію CONCAT для відображення ієрархії.

SQL-код:

```
WITH RECURSIVE pilot_hierarchy AS (  
    SELECT id, crew_member_id, allowed_planes, CAST(last_name AS CHAR(200)) AS name, 0 AS level  
    FROM pilots  
    JOIN crew_members ON pilots.crew_member_id = crew_members.id  
    UNION ALL  
    SELECT p.id, p.crew_member_id, p.allowed_planes, CONCAT(ph.name, ' > ', CAST(cm.last_name AS  
    CHAR(200))) AS name, level + 1  
    FROM pilots p  
    JOIN crew_members cm ON p.crew_member_id = cm.id  
    JOIN pilot_hierarchy ph ON p.crew_member_id = ph.crew_member_id  
)  
SELECT id, name, allowed_planes, level  
FROM pilot_hierarchy  
ORDER BY id, level;
```

Цей запит використовує рекурсивний запит для створення CTE (Common Table Expression - це тимчасова таблиця, яка використовується в межах одного запиту SELECT, щоб зберегти проміжні результати запиту та полегшити читання та редагування складних запитів) з назвою "pilot_hierarchy", який містить початкові дані з таблиці "pilots" та "crew_members". Наступний

рядок UNION ALL об'єднує ці дані з допомогою таблиці, що була створена раніше, щоб створити ієрархічний список. Кожен новий запис додається до списку з новим рівнем.

У нашому запиті ми використовуємо CAST для перетворення поля "last_name" у CHAR(200), щоб мати можливість використовувати функцію CONCAT, яка дозволяє додавати рядки разом. Окрім того, ми використовуємо ORDER BY для сортування даних за рівнем.

10 Завдання: Створити SELECT-запит типу CrossTab для виведення кількості проданих квитків на кожен рейс для кожного літака.

Пояснення: Запит типу CrossTab дозволяє здійснювати трансформацію даних, коли ми хочемо отримати результат у вигляді таблиці з рядками і стовпцями, які не відповідають структурі вихідної таблиці. У даному випадку, ми хочемо отримати таблицю, де в рядках будуть ідентифікатори літаків, в стовпцях - ідентифікатори рейсів, а значеннями - кількість проданих квитків на кожен рейс для кожного літака.

SQL-код:

```
SELECT plane_id,  
  
       SUM(CASE WHEN id = 1 THEN sold_tickets_number ELSE 0 END) AS flight_1,  
  
       SUM(CASE WHEN id = 2 THEN sold_tickets_number ELSE 0 END) AS flight_2,  
  
       SUM(CASE WHEN id = 3 THEN sold_tickets_number ELSE 0 END) AS flight_3,  
  
       SUM(CASE WHEN id = 4 THEN sold_tickets_number ELSE 0 END) AS flight_4,  
  
       SUM(CASE WHEN id = 5 THEN sold_tickets_number ELSE 0 END) AS flight_5  
  
FROM flights  
  
GROUP BY plane_id;
```

Цей запит використовує функцію SUM в поєднанні з CASE для обчислення кількості проданих квитків на кожен рейс для кожного літака. Після цього використовується групування за ідентифікатором літака (plane_id), щоб отримати результат у вигляді таблиці CrossTab.

11 Завдання: Оновити інформацію про літак з певним ідентифікатором.

Пояснення: Іноді може знадобитися оновити інформацію про об'єкт в базі даних. В даному випадку ми хочемо оновити інформацію про літак з певним ідентифікатором, наприклад, збільшити кількість годин, які він пролетів.

SQL-код:

```
UPDATE planes
```

```
SET worked_hours = worked_hours + 1000
```

```
WHERE id = 1;
```

Цей запит збільшить кількість годин, пролетітих першим літаком в базі даних, на 1000. Ми використали команду UPDATE, щоб оновити таблицю planes. Потім ми вказали, яку колонку ми хочемо оновити та на скільки годин ми хочемо збільшити значення, використовуючи оператор SET. У нашому випадку ми використали оператор + для додавання 1000 до поточного значення колонки worked_hours. Нарешті, ми вказали, який рядок ми хочемо оновити, використовуючи оператор WHERE. У нашому випадку ми вказали id = 1, щоб оновити рядок з першим літаком в таблиці planes.

12 Завдання: Оновити дані в таблиці planes та pilots для літака з номером борту 'AA101'. Змінити кількість годин роботи літака на 13000 та додати новий дозволений тип літака для пілота з ID=1 - 'Boeing 777'.

Пояснення: Для оновлення даних в таблиці planes та pilots для конкретного літака, ми використовуємо операцію UPDATE з ключовим словом JOIN, щоб зв'язати ці дві таблиці за допомогою зовнішнього ключа plane_id. Після цього, ми використовуємо ключове слово SET, щоб оновити значення кількості годин роботи в таблиці planes та додати новий тип літака для пілота в таблиці pilots, використовуючи операцію конкатенації рядків CONCAT.

SQL-код:

```
UPDATE planes
```

```
JOIN pilots ON planes.id = pilots.crew_member_id
```

```
SET planes.worked_hours = 13000,
```

```
    pilots.allowed_planes = CONCAT(pilots.allowed_planes, ', Boeing 777')
```

```
WHERE planes.board_number = 'AA101';
```

13 Завдання: Додайте новий літак в таблицю planes.

Пояснення: Для додавання нового запису в таблицю planes використовується операція INSERT INTO з вказанням значень для кожного стовпця таблиці.

SQL-код:

```
INSERT INTO planes (board_number, model, worked_hours, seats, capacity)
```

```
VALUES ('AA109', 'Boeing 777', 4000, 200, 7000);
```

14 Завдання: написати запити типу Append (INSERT) для додавання записів з інших таблиць.

Пояснення: запити типу Append дозволяють додавати нові записи до таблиць на основі даних з інших таблиць. Це корисно, коли потрібно копіювати дані з однієї таблиці в іншу, або додавати нові записи на основі існуючих.

SQL-код:

1. Додавання нових літаків з існуючих записів в таблицю planes:

```
INSERT INTO planes (board_number, model, worked_hours, seats, capacity)
```

```
SELECT board_number, model, worked_hours, seats, capacity
```

```
FROM planes
```

```
WHERE worked_hours < 5000;
```

2. Додавання записів про проданих квитків на нові рейси в таблицю flights:

```
INSERT INTO flights (departure_point, destination_point, departure_time, landing_time, plane_id,  
sold_tickets_number)
```

```
SELECT 'Kyiv', 'Paris', '2023-04-01 15:00:00', '2023-04-01 18:00:00', id, 0
```

```
FROM planes
```

```
WHERE model = 'Airbus A320';
```

3. Додавання нових членів екіпажу на основі існуючих записів у таблицю crew_members:


```
INSERT INTO crew_members (last_name, birth_date, address)
```

```
SELECT last_name, birth_date, address
```

```
FROM crew_members
```

```
WHERE id IN (1, 3);
```

4. Додавання нових пілотів на основі існуючих записів у таблицю pilots:

```
INSERT INTO pilots (crew_member_id, allowed_planes, last_flight_date)
```

```
SELECT crew_member_id, allowed_planes, last_flight_date
```

```
FROM pilots
```

```
WHERE crew_member_id = 2;
```

15 Завдання: Видалити всі дані з кожної таблиці.

Пояснення: Щоб видалити всі дані з таблиці, використовують команду DELETE без умови WHERE. Це видаляє всі рядки з таблиці, зберігаючи саму таблицю з її структурою. Також можна використовувати команду TRUNCATE, яка видаляє всі рядки з таблиці та відновлює інкрементний лічильник до початкового значення, але залишає таблицю зі структурою.

SQL-код:

```
DELETE FROM planes;
```

```
DELETE FROM flights;
```

```
DELETE FROM crew_members;
```

```
DELETE FROM pilots;
```

```
DELETE FROM flight_crew;
```

або

```
TRUNCATE TABLE planes;
```

```
TRUNCATE TABLE flights;
```

```
TRUNCATE TABLE crew_members;
```

```
TRUNCATE TABLE pilots;
```

```
TRUNCATE TABLE flight_crew;
```

Обидва коди виконують ту ж операцію видалення всіх даних з кожної таблиці, але команда TRUNCATE є швидшою, особливо для таблиць з великою кількістю рядків. Проте, вона не підходить, якщо існують залежності з іншими таблицями, оскільки це може призвести до помилки з обмеженнями на зовнішні ключі. В такому випадку, використання команди DELETE без умови WHERE є більш безпечним варіантом.

16 Завдання: Видалити всі рейси з проданим кількістю квитків менше 100.

Пояснення: Ми хочемо видалити рейси з таблиці `flights`, де кількість проданих квитків менше 100. Для цього ми скористаємось запитом `DELETE`, використовуючи умову `WHERE` для фільтрації записів.

SQL-код:

```
DELETE FROM flights
```

```
WHERE sold_tickets_number < 100;
```