МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт информационных технологий (ИИТ)
Кафедра цифровой трансформации (ЦТ)

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ

по дисциплине «Разработка баз данных»

Практическое занятие №3

Студенты группы *ИКБО-22-23 Платонов Т.А.* _____
(подпись)

Старший преподаватель *Черняускас В. В.* _____
(подпись)

Отчет представлен «___»_____2025 г.

Москва 2025 г.

СОДЕРЖАНИЕ

Постановка задачи и выполнение

Задание 1: использование оператора CASE

1. Составить запрос, использующий поисковое выражение CASE для категоризации данных по числовому признаку из БД (цена).

```
SELECT
    ri.id_record_instance,
    al.title,
    ri.purchaseprice,
    CASE
      WHEN ri.purchaseprice >= 30 THEN 'High'
      WHEN ri.purchaseprice BETWEEN 18 AND 29.99 THEN 'Medium'
      WHEN ri.purchaseprice > 0 THEN 'Low'
      ELSE 'Unknown'
    END AS price_band
  FROM recordinstance ri
  JOIN album al ON al.id_album = ri.id_album_fk;
```

cordinstance(+) 1

Введите SQL выражение чтобы отфильтровать результаты

id_record_instance	title	purchaseprice	price_band
1	Abbey Road	25,5	Medium
2	Kind of Blue	30	High
3	Una Mattina	20	Medium

Рисунок 1 -

2. Составить запрос, в котором оператор CASE используется внутри агрегатной функции (например, SUM или COUNT) для выполнения условной агрегации.

```sql
SELECT
    ar.name AS artist,
    SUM(CASE WHEN ri.purchaseprice >= 25 THEN 1 ELSE 0 END) AS expensive_copies,
    SUM(CASE WHEN ri.purchaseprice < 15  THEN 1 ELSE 0 END) AS cheap_copies,
    COUNT(*)                                                AS all_copies
FROM recordinstance ri
JOIN album al ON al.id_album = ri.id_album_fk
JOIN artist ar ON ar.id_artist = al.id_artist_fk
GROUP BY ar.name
ORDER BY all_copies DESC;
```

Введите SQL выражение чтобы отфильтровать результаты

artist	expensive_copies	cheap_copies	all_copies
Ludovico Einaudi	0	0	1
Miles Davis	1	0	1
The Beatles	1	0	1

Рисунок 2 -

Задание 2: использование подзапросов.

Составить и выполнить три запроса, демонстрирующих разные типы подзапросов.

1. Скалярный подзапрос: найти все записи в таблице, у которых значение в некотором числовом столбце превышает среднее (или максимальное/минимальное) значение по этому столбцу.

```sql
SELECT
    ri.id_record_instance,
    al.title,
    ri.purchaseprice
FROM recordinstance ri
JOIN album al ON al.id_album = ri.id_album_fk
WHERE ri.purchaseprice > (SELECT AVG(purchaseprice) FROM recordinstance)
ORDER BY ri.purchaseprice DESC;
```

Введите SQL выражение чтобы отфильтровать результаты

id_record_instance	title	purchaseprice
2	Kind of Blue	30
1	Abbey Road	25,5

2. Многострочный подзапрос с IN: вывести информацию из одной таблицы на основе идентификаторов, полученных из связанной таблицы по определенному критерию (в данном случае, обязательно по дате).

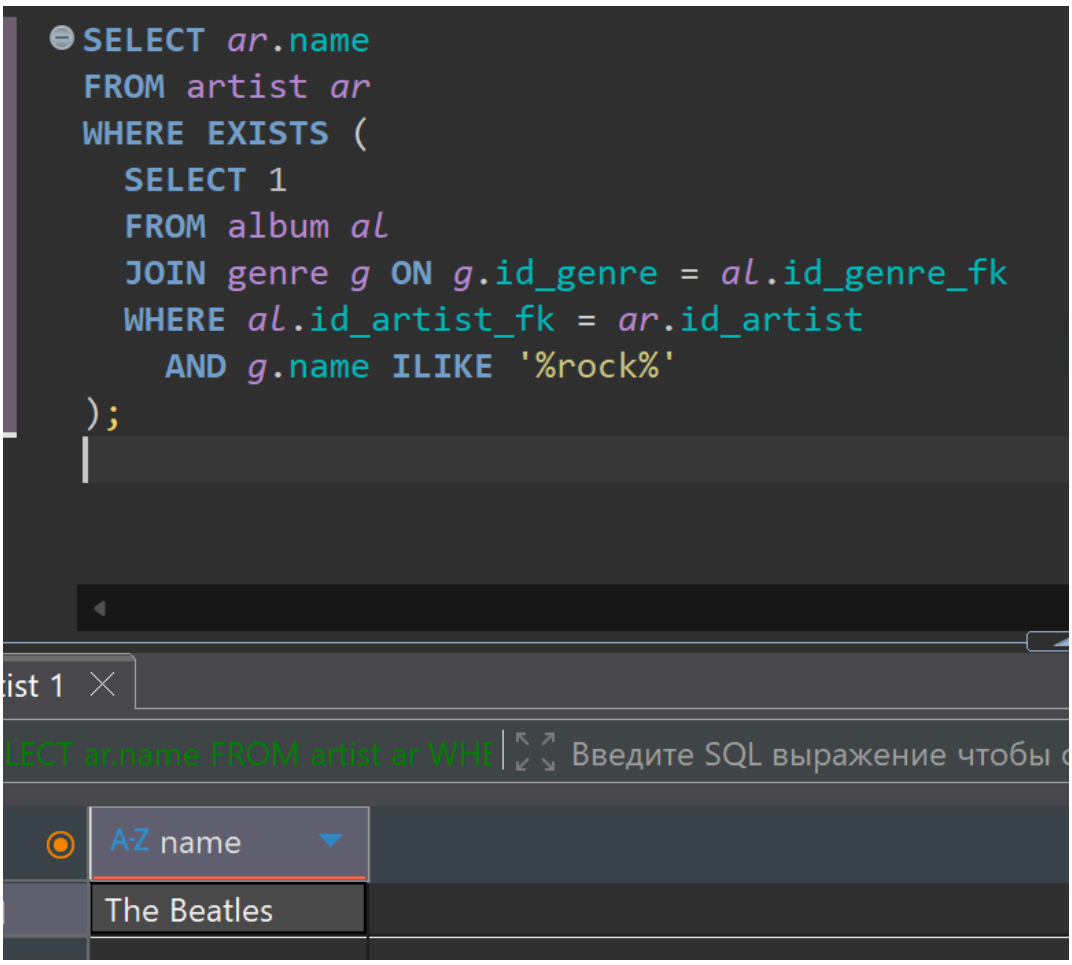
```sql
SELECT al.title, al.releaseyear, g.name AS genre
FROM album al
JOIN genre g ON al.id_genre_fk = g.id_genre
WHERE al.id_genre_fk IN (
    SELECT id_genre
    FROM genre
    WHERE name IN ('Rock', 'Jazz')
)
ORDER BY al.title;
```

m(+) 1

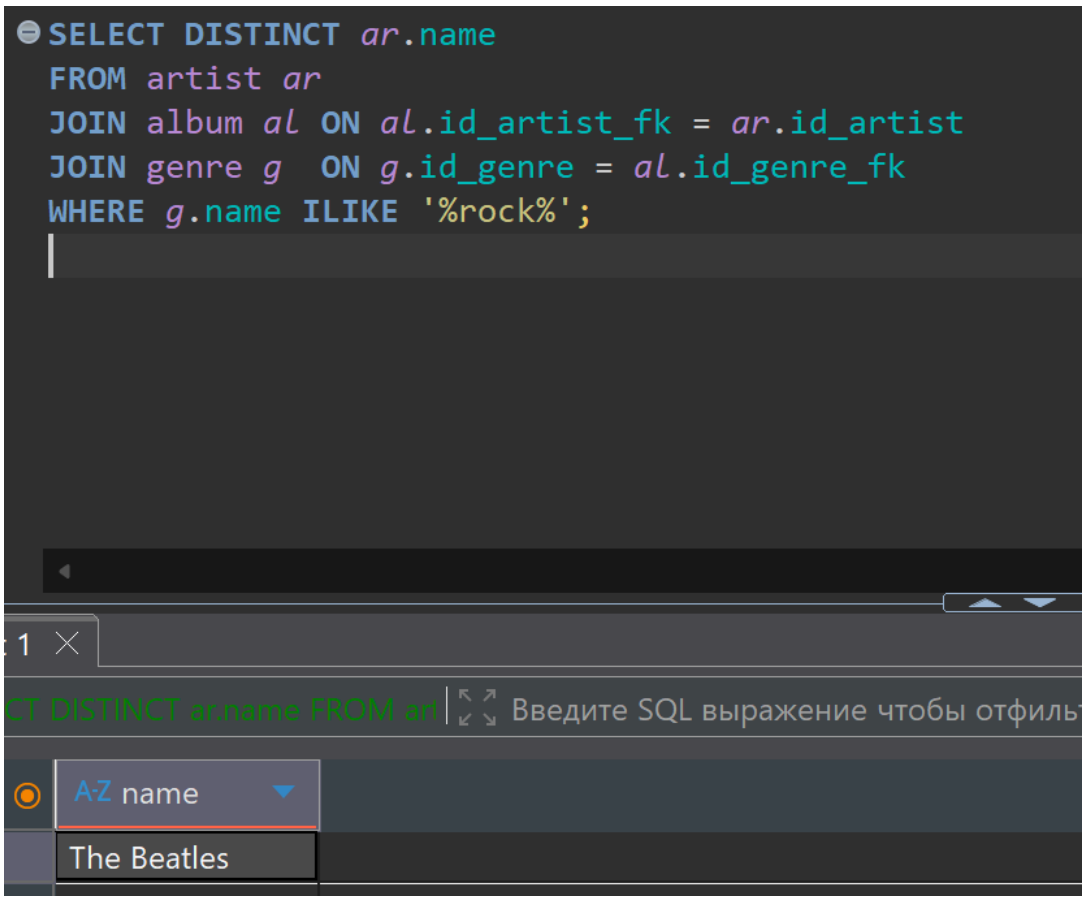
ECT al.title, al.releaseyear, g.nam | Введите SQL выражение чтобы отфильтро

A-Z title	123 releaseyear	A-Z genre
Abbey Road	1 969	Rock
Kind of Blue	1 959	Jazz
To Pimp a Butterfly	2 015	Jazz

3. Коррелированный подзапрос с EXISTS: найти все записи из родительской таблицы, для которых существует хотя бы одна связанная запись в дочерней таблице, удовлетворяющая текстовому условию.



4. Альтернативное решение с JOIN: решите задачу из пункта выше (2.3, Коррелированный подзапрос с EXISTS), но на этот раз с использованием оператора соединения JOIN.

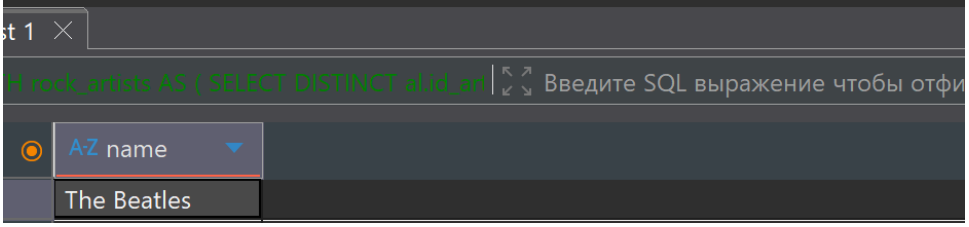


Задание 3: использование обобщенных табличных выражений (CTE).

1. Стандартное CTE: переписать запрос из Задания 2.3 (с коррелированным подзапросом) с использованием обобщенного

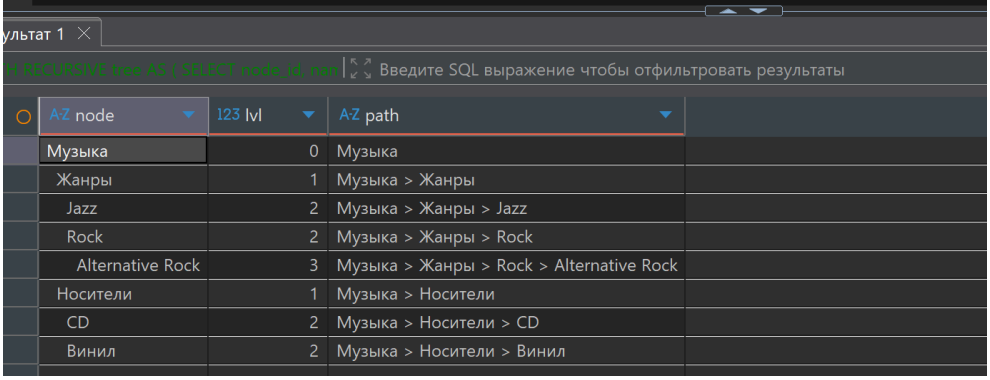
табличного выражения (CTE).

```sql
WITH rock_artists AS (
    SELECT DISTINCT al.id_artist_fk
    FROM album al
    JOIN genre g ON g.id_genre = al.id_genre_fk
    WHERE g.name ILIKE '%Rock%'
)
SELECT ar.name
FROM artist ar
JOIN rock_artists ra ON ra.id_artist_fk = ar.id_artist
ORDER BY ar.name;
```



2. Рекурсивное CTE: используя имеющуюся в вашей схеме данных таблицу с иерархической структурой (например, pharmacists), написать рекурсивный запрос с помощью WITH RECURSIVE для вывода всей иерархии с указанием уровня вложенности

```sql
WITH RECURSIVE tree AS (
    SELECT node_id, name, parent_id, 0 AS lvl, name::text AS path
    FROM demo_music_tree
    WHERE parent_id IS NULL
    UNION ALL
    SELECT c.node_id, c.name, c.parent_id, p.lvl + 1,
           (p.path || ' > ' || c.name)
    FROM demo_music_tree c
    JOIN tree p ON c.parent_id = p.node_id
)
SELECT LPAD('', lvl*2, ' ') || name AS node, lvl, path
FROM tree
ORDER BY path;
```



A-Z node	123 lvl	A-Z path
Музыка	0	Музыка
Жанры	1	Музыка > Жанры
Jazz	2	Музыка > Жанры > Jazz
Rock	2	Музыка > Жанры > Rock
Alternative Rock	3	Музыка > Жанры > Rock > Alternative Rock
Носители	1	Музыка > Носители
CD	2	Музыка > Носители > CD
Винил	2	Музыка > Носители > Винил

Вывод

В ходе выполнения практической работы удалось освоить приёмы реализации условной логики в SQL с помощью оператора CASE, позволяющего динамически категоризировать данные и выполнять условную агрегацию. На примерах базы данных были проработаны различные типы подзапросов: скалярные, многострочные и коррелированные, а также показаны их преимущества и особенности использования в реальных задачах. Отдельное внимание уделялось сравнению подходов через EXISTS и JOIN, что помогло понять различие между проверкой факта существования записи и выборкой связанных данных. Дополнительно изучены обобщённые табличные выражения, которые позволяют структурировать сложные запросы и делают их более читаемыми, а также рекурсивные CTE, применяемые для работы с иерархическими структурами. Работа показала, как встроенные механизмы SQL позволяют решать задачи разного уровня сложности на стороне базы данных, обеспечивая более гибкую и мощную обработку информации.