```
EXPLAIN ANALYZE SELECT *
FROM tickets t JOIN ticket_flights tf ON tf.ticket_no = t.ticket_no
WHERE t.ticket_no IN ('0005432000860','0005432000861');

EXPLAIN ANALYZE SELECT *
FROM tickets t JOIN ticket_flights tf ON tf.ticket_no = t.ticket_no;

EXPLAIN ANALYZE SELECT *
FROM tickets t JOIN ticket_flights tf ON tf.ticket_no = t.ticket_no
ORDER BY t.ticket_no;

-- Дроп таблиц
DROP TABLE bus;
DROP TABLE model_bus;
DROP TABLE driver;

-- Создание таблиц
CREATE TABLE bus (id SERIAL, route TEXT, id_model INT, id_driver INT);
CREATE TABLE model_bus (id SERIAL, name TEXT);
CREATE TABLE driver (id SERIAL, first_name TEXT, second_name TEXT);

-- Вставка данных в таблицы
INSERT INTO bus VALUES
    (1, 'Москва-Болшево', 1, 1),
    (2, 'Москва-Пушкино', 1, 2),
    (3, 'Москва-Ярославль', 2, 3),
    (4, 'Москва-Кострома', 2, 4),
    (5, 'Москва-Волгорад', 3, 5),
    (6, 'Москва-Иваново', NULL, NULL);

INSERT INTO model_bus VALUES
    (1, 'ПАЗ'),
    (2, 'ЛИАЗ'),
    (3, 'MAN'),
    (4, 'МАЗ'),
    (5, 'НЕФАЗ');

INSERT INTO driver VALUES
```

```
    (1, 'Иван', 'Иванов'),
    (2, 'Петр', 'Петров'),
    (3, 'Савелий', 'Сидоров'),
    (4, 'Антон', 'Шторкин'),
    (5, 'Олег', 'Зажигаев'),
    (6, 'Аркадий', 'Паровозов');

-- Вывод данных из таблиц
SELECT * FROM bus;
SELECT * FROM model_bus;
SELECT * FROM driver;

-- Прямое соединение с использованием JOIN
EXPLAIN
SELECT *
FROM bus b
JOIN model_bus mb ON b.id_model = mb.id;

-- Прямое соединение без явного указания JOIN (подразумевается INNER JOIN)
EXPLAIN
SELECT *
FROM bus b, model_bus mb
WHERE b.id_model = mb.id;

-- LEFT JOIN для соединения таблиц с возвратом всех записей из левой таблицы
SELECT *
FROM bus b
LEFT JOIN model_bus mb ON b.id_model = mb.id;

-- RIGHT JOIN для соединения таблиц с возвратом всех записей из правой
таблицы
SELECT *
FROM bus b
RIGHT JOIN model_bus mb ON b.id_model = mb.id;

-- LEFT JOIN с условием возврата записей, для которых нет соответствий в
правой таблице
SELECT *
FROM bus b
LEFT JOIN model_bus mb ON b.id_model = mb.id
```

```
WHERE mb.id IS NULL;

-- RIGHT JOIN с условием возврата записей, для которых нет соответствий в левой таблице
SELECT *
FROM bus b
RIGHT JOIN model_bus mb ON b.id_model = mb.id
WHERE b.id IS NULL;

-- FULL JOIN для соединения таблиц с возвратом всех записей из обеих таблиц
SELECT *
FROM bus b
FULL JOIN model_bus mb ON b.id_model = mb.id;

-- FULL JOIN с условием возврата записей, для которых нет соответствий в одной из таблиц
SELECT *
FROM bus b
FULL JOIN model_bus mb ON b.id_model = mb.id
WHERE b.id IS NULL OR mb.id IS NULL;

-- CROSS JOIN для создания декартова произведения таблиц
SELECT *
FROM bus b
CROSS JOIN model_bus mb;

-- Использование CROSS JOIN с условием, эквивалентное INNER JOIN
EXPLAIN
SELECT *
FROM bus b
CROSS JOIN model_bus mb
WHERE b.id_model = mb.id;

-- CROSS JOIN без условий, эквивалентное декартову произведению
SELECT *
FROM bus b, model_bus mb
WHERE 1 = 1;

-- Создание и наполнение тестовых таблиц для демонстрации работы параметров планировщика
```

```
DROP TABLE IF EXISTS test;
CREATE TABLE test AS
    SELECT (RANDOM() * 100)::INT AS id, 'product ' || id AS product
    FROM generate_series(1, 10000) AS id;

SELECT * FROM test;

CREATE TABLE test_2 (id INT);
INSERT INTO test_2 VALUES (1);

-- Установка параметров планировщика
SET enable_hashjoin = ON;
SET enable_mergejoin = ON;
SET enable_nestloop = ON;

-- Изменение параметра join_collapse_limit
SET join_collapse_limit = 8;    -- увеличиваем предел для оптимизации
соединений
SET join_collapse_limit = 1;    -- уменьшаем предел для ограничения
оптимизации

-- Демонстрация влияния параметра join_collapse_limit на план запроса
EXPLAIN
SELECT *
FROM test_2 t2
INNER JOIN test t1 ON t2.id = t1.id
INNER JOIN test_2 t3 ON t3.id = t2.id;

-- Пример использования LATERAL JOIN
CREATE TABLE t_product AS
    SELECT id AS product_id, id * 10 * RANDOM() AS price, 'product ' || id
AS product
    FROM generate_series(1, 1000) AS id;

CREATE TABLE t_wishlist
(
    wishlist_id    INT,
    username       TEXT,
    desired_price  NUMERIC
);
```

```
INSERT INTO t_wishlist VALUES
    (1, 'hans', '450'),
    (2, 'joe', '60'),
    (3, 'jane', '1500');

SELECT * FROM t_product LIMIT 10;
SELECT * FROM t_wishlist;

-- Использование LATERAL JOIN для соединения t_wishlist и t_product с
фильтрацией по цене
SELECT *
FROM t_wishlist AS w
LEFT JOIN LATERAL (
    SELECT *
    FROM t_product AS p
    WHERE p.price < w.desired_price
    ORDER BY p.price DESC
    LIMIT 5
) AS x ON TRUE
ORDER BY wishlist_id, price DESC;

-- Использование LATERAL без LEFT JOIN, что не гарантирует возврат всех
записей из t_wishlist
SELECT *
FROM t_wishlist AS w, LATERAL (
    SELECT *
    FROM t_product AS p
    WHERE p.price < w.desired_price
    ORDER BY p.price DESC
    LIMIT 3
) AS x
ORDER BY wishlist_id, price DESC;

-- Использование операций UNION, INTERSECT и EXCEPT для демонстрации
объединения, пересечения и разности множеств
DROP TABLE IF EXISTS top_rated_films;
CREATE TABLE top_rated_films(title VARCHAR NOT NULL, release_year SMALLINT);

DROP TABLE IF EXISTS most_popular_films;
```

```
CREATE TABLE most_popular_films(title VARCHAR NOT NULL, release_year
SMALLINT);

INSERT INTO top_rated_films(title, release_year) VALUES
   ('The Shawshank Redemption', 1994),
   ('The Godfather', 1972),
   ('12 Angry Men', 1957);

INSERT INTO most_popular_films(title, release_year) VALUES
   ('An American Pickle', 2020),
   ('The Godfather', 1972),
   ('Greyhound', 2020);

-- Объединение множеств
SELECT * FROM top_rated_films
UNION
SELECT * FROM most_popular_films;

-- Объединение множеств с сохранением дубликатов
SELECT * FROM top_rated_films
UNION ALL
SELECT * FROM most_popular_films;

-- Пересечение множеств
SELECT * FROM top_rated_films
INTERSECT
SELECT * FROM most_popular_films;

-- Разность множеств
SELECT * FROM top_rated_films
EXCEPT
SELECT * FROM most_popular_films;
```