```
-- Удаление таблицы, если она существует
DROP TABLE IF EXISTS index_test_table;

-- Создание новой таблицы
CREATE TABLE index_test_table(
    random_num INTEGER,
    random_text TEXT,
    bool_field BOOLEAN
);

-- Вставка данных в таблицу
INSERT INTO index_test_table(random_num, random_text, bool_field)
SELECT
    s.id,
    chr((32 + random() * 94)::INTEGER),
    random() < 0.01
FROM generate_series(1, 100000) AS s(id)
ORDER BY random();

-- Создание индекса по числовому полю
CREATE INDEX ON index_test_table(random_num);

-- Анализ таблицы для обновления статистики
ANALYZE index_test_table;

-- План запроса для выборки по точному значению
EXPLAIN SELECT * FROM index_test_table WHERE random_num = 1;

-- План запроса для выборки по диапазону значений
EXPLAIN SELECT * FROM index_test_table WHERE random_num < 100;

-- Создание индекса по текстовому полю
CREATE INDEX ON index_test_table(random_text);

-- План запроса с условиями по двум полям
EXPLAIN SELECT * FROM index_test_table WHERE random_num <= 100 AND
random_text = 'a';
```

```
-- Просмотр корреляции столбцов в таблице
SELECT attname, correlation FROM pg_stats WHERE tablename =
'index_test_table';

-- План запроса с широким диапазоном значений
EXPLAIN SELECT * FROM index_test_table WHERE random_num <= 40000;

-- Создание составного индекса
CREATE INDEX ON index_test_table(random_num, random_text);

-- План запроса с использованием составного индекса
EXPLAIN SELECT * FROM index_test_table WHERE random_num <= 100 AND
random_text = 'a';

-- План запроса для булевого поля
EXPLAIN SELECT * FROM index_test_table WHERE bool_field = TRUE;

-- Создание условного индекса для булевого поля
CREATE INDEX ON index_test_table(bool_field) WHERE bool_field;

-- Удаление индекса
DROP INDEX IF EXISTS index_test_table_bool_field_idx;

-- Проверка размера страниц индекса
SELECT relpages FROM pg_class WHERE
relname='index_test_table_bool_field_idx';

-- Создание тестовой таблицы с уникальными значениями
CREATE TABLE test_unique_index AS
SELECT
    generate_series AS id,
    generate_series::TEXT || (random() * 10)::TEXT AS col2,
    (ARRAY['Yes', 'No', 'Maybe'])[floor(random() * 3 + 1)] AS is_okay
FROM generate_series(1, 50000);

-- Создание уникального индекса с условием
CREATE UNIQUE INDEX idx_test_unique_index_id ON test_unique_index(id) WHERE
id IS NOT NULL;
```

```
-- Экспланация запроса на выборку
EXPLAIN SELECT * FROM index_test_table WHERE bool_field = TRUE;

-- Создание индекса по полю bool_field с условием
CREATE INDEX ON index_test_table(bool_field) WHERE bool_field;

-- Удаление индекса
DROP INDEX index_test_table_bool_field_idx;

-- Проверка размера страницы индекса
SELECT relpages FROM pg_class WHERE
relname='index_test_table_bool_field_idx';

-- Создание индекса по нижнему регистру текстового поля
CREATE INDEX ON index_test_table(LOWER(random_text));

-- Анализ таблицы для обновления статистики
ANALYZE index_test_table;

-- Экспланация анализируемого запроса с условием на текстовое поле
EXPLAIN ANALYZE SELECT * FROM index_test_table WHERE LOWER(random_text) =
'a';

-- Создание таблицы событий с JSONB полем
CREATE TABLE events (
    id SERIAL PRIMARY KEY,
    data JSONB
);

-- Вставка данных в таблицу событий
INSERT INTO events (data) VALUES
('{"type": "click", "user": {"id": 1, "name": "Alice"}}'),
('{"type": "click", "user": {"id": 2, "name": "Bob"}}'),
('{"type": "view", "user": {"id": 3, "name": "Charlie"}, "page": {"id": 1,
"title": "Homepage"}}');

-- Создание GIN индекса для JSONB поля
CREATE INDEX idx_events_data ON events USING gin (data);

-- Отключение последовательного сканирования для демонстрации использования
```

```
индекса
SET enable_seqscan = OFF;

-- Анализ таблицы событий
ANALYZE events;

-- Экспланация запроса с использованием JSONB оператора
EXPLAIN SELECT * FROM events WHERE data @> '{"type": "click"}';

-- Включение последовательного сканирования обратно
SET enable_seqscan = ON;

-- Создание таблицы статей
CREATE TABLE articles (
    id SERIAL PRIMARY KEY,
    title TEXT,
    content TEXT
);

-- Вставка данных в таблицу статей
INSERT INTO articles (title, content) VALUES
('PostgreSQL Tutorial', 'This tutorial covers the basics of PostgreSQL.'),
('Full-Text Search in PostgreSQL', 'Learn how to use full-text search in
PostgreSQL.'),
('Advanced PostgreSQL Features', 'Explore advanced features of PostgreSQL,
including GIN indexes and full-text search.');

-- Добавление столбца tsvector с автоматическим заполнением для
полнотекстового поиска
ALTER TABLE articles ADD COLUMN content_tsvector TSVECTOR GENERATED ALWAYS
AS (to_tsvector('english', content)) STORED;

-- Просмотр содержимого таблицы статей
SELECT * FROM articles;

-- Создание GIN индекса для столбца tsvector
CREATE INDEX idx_articles_content_tsvector ON articles USING gin
(content_tsvector);

-- Отключение последовательного сканирования для демонстрации использования
```

```
индекса
SET enable_seqscan = OFF;

-- Экспланация запроса полнотекстового поиска по статьям
EXPLAIN SELECT title, content FROM articles WHERE content_tsvector @@
to_tsquery('english', 'PostgreSQL & full-text');

-- Включение последовательного сканирования обратно
SET enable_seqscan = ON;

SELECT
TABLE_NAME,
        pg_size_pretty(table_size) AS table_size,
        pg_size_pretty(indexes_size) AS indexes_size,
        pg_size_pretty(total_size) AS total_size
FROM (
        SELECT
        TABLE_NAME,
                pg_table_size(TABLE_NAME) AS table_size,
                pg_indexes_size(TABLE_NAME) AS indexes_size,
                pg_total_relation_size(TABLE_NAME) AS total_size
        FROM (
                SELECT ('"' || table_schema || '"."' || TABLE_NAME || '"')
AS TABLE_NAME
                FROM information_schema.tables
        ) AS all_tables
        ORDER BY total_size DESC
        ) AS pretty_sizes;


--Неиспользуемые индексы
SELECT s.schemaname,
       s.relname AS tablename,
       s.indexrelname AS indexname,
       pg_size_pretty(pg_relation_size(s.indexrelid)) AS index_size,
       s.idx_scan
FROM pg_catalog.pg_stat_user_indexes s
   JOIN pg_catalog.pg_index i ON s.indexrelid = i.indexrelid
WHERE s.idx_scan < 10      -- has never been scanned
  AND 0 <>ALL (i.indkey)  -- no index column is an expression
```

```
  AND NOT i.indisunique   -- is not a UNIQUE index
  AND NOT EXISTS          -- does not enforce a constraint
         (SELECT 1 FROM pg_catalog.pg_constraint c
          WHERE c.conindid = s.indexrelid)
ORDER BY pg_relation_size(s.indexrelid) DESC;
```