Advanced SQL Techniques

1. Advanced Joins

LEFT JOIN, RIGHT JOIN, FULL OUTER JOIN

```
SELECT e.name, d.name
FROM employees e
FULL OUTER JOIN departments d
ON e.department_id = d.id;
```

2. Subqueries

Nested Queries:

```
SELECT name FROM employees WHERE age > (SELECT AVG(age) FROM employees);
```

3. Indexing

Importance of Indexes

Creating and Using Indexes:

```
CREATE INDEX idx_name ON employees(name);
```

4. Views

Creating and Querying Views:

```
CREATE VIEW employee_view AS
SELECT name, age FROM employees WHERE age > 30;
```

5. Stored Procedures

Example:

```
CREATE PROCEDURE GetEmployeeCount()
BEGIN
    SELECT COUNT(*) AS TotalEmployees FROM employees;
END;
```

6. Triggers

Example:

```
CREATE TRIGGER before_insert_employee
BEFORE INSERT ON employees
FOR EACH ROW
BEGIN
    IF NEW.age < 18 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Age must be 18 or
above';
```

```
    END IF;
END;
```

7. Transactions

Using COMMIT and ROLLBACK:

```
START TRANSACTION;
UPDATE employees SET age = age + 1 WHERE id = 1;
ROLLBACK;
```

8. Window Functions

ROW_NUMBER(), RANK(), DENSE_RANK():

```
SELECT name, age, RANK() OVER (ORDER BY age DESC) AS Rank FROM
employees;
```

9. Advanced Data Types

JSON, ARRAY, etc.:

```
SELECT JSON_EXTRACT(json_column, '$.key') AS value FROM json_table;
```

10. Performance Optimization

Query Execution Plans (EXPLAIN)
Query Optimization Tips