PostgreSQL Course: Beginner to Advanced

Introduction to PostgreSQL

- **What is PostgreSQL?**
  - Overview of PostgreSQL as an open-source, relational database management system.
  - Features of PostgreSQL: ACID compliance, extensibility, and support for advanced data types.
  - Use cases of PostgreSQL in real-world applications.
- **Installing PostgreSQL**
  - Installation steps for Windows, macOS, and Linux.
  - Setting up PostgreSQL server and pgAdmin.
  - Verifying the installation.
- **PostgreSQL Ecosystem**
  - Overview of tools like psql, pgAdmin, and extensions (e.g., PostGIS).

Section 1: Getting Started with PostgreSQL

1.1 PostgreSQL Basics

- Introduction to databases, schemas, and tables.
- Connecting to PostgreSQL using psql:

```
psql -U postgres
```

- Basic SQL commands to explore databases:

```
\l -- List databases
\dt -- List tables in the current schema
\du -- List users and roles
```

1.2 Database and Table Management

- Creating and selecting a database:

```
CREATE DATABASE school;
\c school;
```

- Creating tables:

```
CREATE TABLE students (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100),
    age INT,
    grade VARCHAR(10)
);
```

- Viewing table structure:

```
\d students;
```

1.3 Basic CRUD Operations

- Inserting data:

```
INSERT INTO students (name, age, grade) VALUES ('John Doe', 15,
'10th');
```

- Reading data:

```
SELECT * FROM students;
```

- Updating data:

```
UPDATE students SET age = 16 WHERE name = 'John Doe';
```

- Deleting data:

```
DELETE FROM students WHERE name = 'John Doe';
```

Section 2: Intermediate PostgreSQL

2.1 Data Types

- Overview of PostgreSQL data types:
  - Numeric: INTEGER, FLOAT, SERIAL.
  - String: CHAR, VARCHAR, TEXT.
  - Date and Time: DATE, TIMESTAMP, TIME.
  - Advanced types: JSON, ARRAY, UUID.
- Choosing appropriate data types for columns.

2.2 Constraints and Keys

- Defining constraints:
  - PRIMARY KEY, FOREIGN KEY, UNIQUE, CHECK, NOT NULL.
- Example:

```
ALTER TABLE students ADD CONSTRAINT chk_age CHECK (age > 5);
```

- Managing keys and constraints.

2.3 Joins

- Understanding JOIN operations:
  - INNER JOIN:

```
SELECT students.name, grades.subject
FROM students
INNER JOIN grades ON students.id = grades.student_id;
```

  - LEFT JOIN, RIGHT JOIN, FULL JOIN.

2.4 Aggregate Functions and Grouping

- Functions: COUNT, SUM, AVG, MAX, MIN.

- Grouping data with GROUP BY:

```
SELECT grade, COUNT(*)
FROM students
GROUP BY grade;
```

- Filtering grouped data with HAVING.

Section 3: Advanced PostgreSQL

3.1 Views

- Creating views for reusable queries:

```
CREATE VIEW student_grades AS
SELECT students.name, grades.subject, grades.score
FROM students
JOIN grades ON students.id = grades.student_id;
```

- Querying views:

```
SELECT * FROM student_grades;
```

- Updating views with `INSTEAD OF` triggers.

3.2 Stored Procedures and Functions

- Writing stored procedures:

```
CREATE OR REPLACE FUNCTION get_student_count() RETURNS INT AS
$$
BEGIN
    RETURN (SELECT COUNT(*) FROM students);
END;
$$ LANGUAGE plpgsql;
```

- Calling functions:

```
SELECT get_student_count();
```

- Difference between procedures and functions.

3.3 Triggers

- Creating triggers:

```
CREATE OR REPLACE FUNCTION log_changes() RETURNS TRIGGER AS $$
BEGIN
    INSERT INTO changes_log(table_name, operation, changed_at)
    VALUES (TG_TABLE_NAME, TG_OP, NOW());
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER student_changes
AFTER INSERT OR UPDATE OR DELETE ON students
FOR EACH ROW
EXECUTE FUNCTION log_changes();
```

- Managing triggers.

3.4 Transactions

- Understanding transactions:

```
BEGIN;
UPDATE students SET grade = '11th' WHERE id = 1;
COMMIT;
```

- Using ROLLBACK for error recovery.
- Savepoints for partial rollbacks:

```
SAVEPOINT sp1;
ROLLBACK TO sp1;
```

3.5 Advanced Query Optimization

- Using EXPLAIN and EXPLAIN ANALYZE:

```
EXPLAIN SELECT * FROM students WHERE age > 15;
```

- Optimizing queries with indexes.
- Managing performance with partitioning.

Section 4: PostgreSQL Administration

4.1 User and Role Management

- Creating roles and users:

```
CREATE ROLE readonly_user WITH LOGIN PASSWORD 'password';
```

- Granting privileges:

```
GRANT SELECT ON ALL TABLES IN SCHEMA public TO readonly_user;
```

- Revoking privileges and managing role inheritance.

4.2 Backup and Restore

- Using `pg_dump` for backups:

```
pg_dump -U postgres school > school_backup.sql
```

- Restoring databases with `psql`:

```
psql -U postgres school < school_backup.sql
```

4.3 Monitoring and Performance Tuning

- Monitoring database activity with `pg_stat_activity`.
- Tuning configuration parameters for performance (e.g., `work_mem`, `shared_buffers`).
- Using extensions like `pg_stat_statements` for query analysis.

Section 5: Real-World Projects

5.1 E-commerce Database

- Designing tables for products, customers, orders, and payments.
- Writing queries for order processing and inventory tracking.

5.2 Employee Management System

- Designing a relational schema for employees, departments, and salaries.
- Using views and stored procedures for reports.

Conclusion

- Recap of PostgreSQL features and capabilities.
- Best practices for working with PostgreSQL.
- Resources for further learning.

Appendix

- Common PostgreSQL commands reference.
- Troubleshooting common errors.