**Lab Exercise 20- Logging and Troubleshooting Common Issues in PostgreSQL**

**Objective**

* Enable and configure logging in PostgreSQL
* Identify, simulate, and resolve common PostgreSQL issues
* Use logs for performance and security diagnostics

**Prerequisites**

* PostgreSQL installed and running
* Superuser access
* Basic understanding of SQL and system permissions

**1. Enable and Configure PostgreSQL Logging**

**Step 1: Open postgresql.conf**

**File path (Windows):**

C:\Program Files\PostgreSQL\15\data\postgresql.conf

**Step 2: Modify Logging Settings**

Find and update the following parameters:

logging\_collector = on

log\_directory = 'log'

log\_filename = 'postgresql-%Y-%m-%d.log'

log\_statement = 'all'

log\_connections = on

log\_disconnections = on

log\_line\_prefix = '%t [%p]: [%l-1] user=%u,db=%d,app=%a,client=%h '

Save the file.

**Step 3: Restart PostgreSQL**

**2. Simulate Common PostgreSQL Issues**

**Scenario 1: Invalid Login Attempt**

Try connecting with a wrong password:

psql -U postgres -d testdb

Enter an incorrect password.

**Expected Log Message:**

FATAL: password authentication failed for user "postgres"

**Scenario 2: Query Timeout or Long Execution**

Run a slow query:

SELECT pg\_sleep(10);

**Expected Log Output:**

duration: 10000.000 ms statement: SELECT pg\_sleep(10);

**Scenario 3: Table Not Found**

Execute:

SELECT \* FROM nonexistent\_table;

**Expected Log Output:**

ERROR: relation "nonexistent\_table" does not exist

**Scenario 4: Deadlock Detection**

Open two sessions and run:

**Session 1:**

BEGIN;

UPDATE employees SET salary = salary + 1000 WHERE id = 1;

**Session 2:**

BEGIN;

UPDATE employees SET salary = salary + 500 WHERE id = 2;

UPDATE employees SET salary = salary + 500 WHERE id = 1; -- This will wait

Now in **Session 1**, run:

UPDATE employees SET salary = salary + 500 WHERE id = 2;

You will hit a **deadlock**.

**Expected Log Output:**

ERROR: deadlock detected

DETAIL: Process 12345 waits for ShareLock on transaction 5678

**Scenario 5: Disk Full or Out of Space**

Create a loop to fill the disk (use caution on test systems):

CREATE TABLE big\_data AS SELECT generate\_series(1, 100000000) AS id;

If disk fills, the log will show:

ERROR: could not extend file ... No space left on device

**Scenario 6: Permission Denied**

Run a query as a user with no access:

SET ROLE readonly;

DELETE FROM employees;

**Expected Log:**

ERROR: permission denied for table employees

**3. Read PostgreSQL Logs**

**On Windows**

Go to:

C:\Program Files\PostgreSQL\15\data\log

Open the latest log file with Notepad or any viewer.

tail -f postgresql-2025-05-20.log

**4. Troubleshooting Workflow Checklist**

| **Step** | **What to Check** | **Tool / Command** |
| --- | --- | --- |
| 1 | Error message in logs | Check log file or journal logs |
| 2 | Connection issues | pg\_hba.conf and netstat -an |
| 3 | Query timeouts | EXPLAIN ANALYZE |
| 4 | Permissions | \z and \du in psql |
| 5 | Disk usage | df -h or Task Manager (Windows) |
| 6 | Locks or deadlocks | pg\_locks, pg\_stat\_activity |

**5. Advanced Diagnostic Views**

Use the following SQL views:

-- Active queries

SELECT \* FROM pg\_stat\_activity;

-- Locks

SELECT \* FROM pg\_locks;

-- Wait events

SELECT pid, wait\_event\_type, wait\_event FROM pg\_stat\_activity WHERE wait\_event IS NOT NULL;

-- Database-level stats

SELECT \* FROM pg\_stat\_database;

**Summary**

This lab has covered:

* Enabling and reading PostgreSQL logs
* Simulating login, query, and permission errors
* Detecting long queries, deadlocks, and storage issues
* Using built-in monitoring views