**Setting up Streaming Application**

**Objective:**

Create a stored procedure that inserts rental data on the primary server, and verify that changes replicate to the standby server. Add a logging mechanism to track each operation.

**Tasks to Complete:**

**Set up streaming replication (if not already done):**

* **Primary on port 5432**
* **Standby on port 5433**

**Create a table on the primary:**

**CREATE TABLE rental\_log (**

**log\_id SERIAL PRIMARY KEY,**

**rental\_time TIMESTAMP,**

**customer\_id INT,**

**film\_id INT,**

**amount NUMERIC,**

**logged\_on TIMESTAMP DEFAULT CURRENT\_TIMESTAMP**

**);**

**Ensure this table is replicated.**

1. **Write a stored procedure to:**
2. **Insert a new rental log entry**
3. **Accept customer\_id, film\_id, amount as inputs**
4. **Wrap logic in a transaction with error handling (BEGIN...EXCEPTION...END)**

**CREATE OR REPLACE PROCEDURE sp\_add\_rental\_log(**

**p\_customer\_id INT,**

**p\_film\_id INT,**

**p\_amount NUMERIC**

**)**

**LANGUAGE plpgsql**

**AS $$**

**BEGIN**

**INSERT INTO rental\_log (rental\_time, customer\_id, film\_id, amount)**

**VALUES (CURRENT\_TIMESTAMP, p\_customer\_id, p\_film\_id, p\_amount);**

**EXCEPTION WHEN OTHERS THEN**

**RAISE NOTICE 'Error occurred: %', SQLERRM;**

**END;**

**$$;**

**Call the procedure on the primary:**

**CALL sp\_add\_rental\_log(1, 100, 4.99);**

**On the standby (port 5433):**

**Confirm that the new record appears in rental\_log**

**Run:SELECT \* FROM rental\_log ORDER BY log\_id DESC LIMIT 1;**

**Add a trigger to log any UPDATE to rental\_log**

**Step-1**

* Go to “C:\Program Files\PostgreSQL\17\bin” inside cmd as administrator, run

**initdb -D "C:/primary"**

**initdb -D "C:/secondary"**

* Change the postgresql CONF file PORT = 5433, save and exit
* Run the following commands in cmd

**pg\_ctl -D C:\primary -o "-p 5433" -l c:\primary\logfile start**

**psql -p 5433 -d postgres -c "CREATE ROLE replicator with REPLICATION LOGIN PASSWORD 'repl\_pass';"**

* Open a second cmd as administrator with same “C:\Program Files\PostgreSQL\17\bin” location, run the following commands after deleting all the items inside the C:\secondary folder

**pg\_basebackup -D c:\secondary -Fp -Xs -P -R -h 127.0.0.1 -U replicator -p 5433**

**pg\_ctl -D C:\secondary -o "-p 5435" -l c:\secondary\logfile start**

**psql -p 5433 -d postgres**

* (In another cmd), run

**psql -p 5435 -d postgres**

--------------------------------------

5433 -

**select \* from pg\_stat\_replication;**

5435

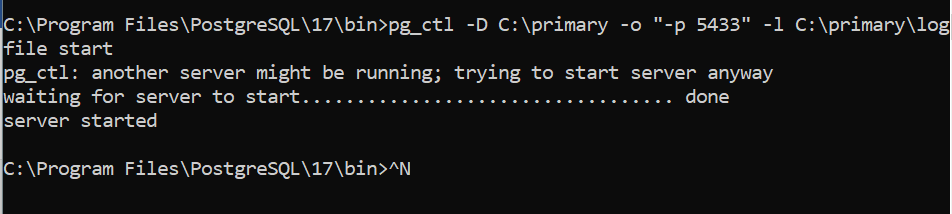
**select pg\_is\_in\_recovery();**

**If you shutdown your streaming replication environment,**

To resume your streaming replication environment, you simply need to:

* Start both servers (primary and standby)
* Verify replication is active
* Open two Command Prompts **as Administrator, go to C:\Program Files\PostgreSQL\17\bin**
* Start the Primary Server (Port 5433)

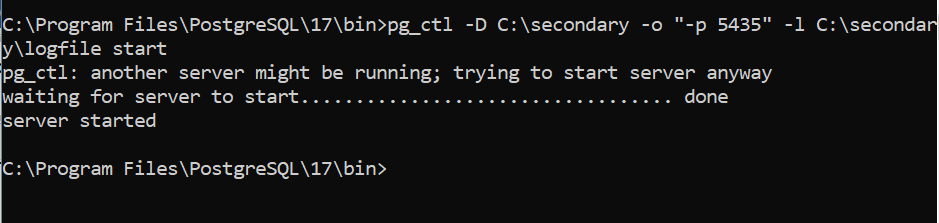
**pg\_ctl -D C:\primary -o "-p 5433" -l C:\primary\logfile start**



This starts the PostgreSQL primary server.

* Start the Standby Server (Port 5435)

**pg\_ctl -D C:\secondary -o "-p 5435" -l C:\secondary\logfile start**



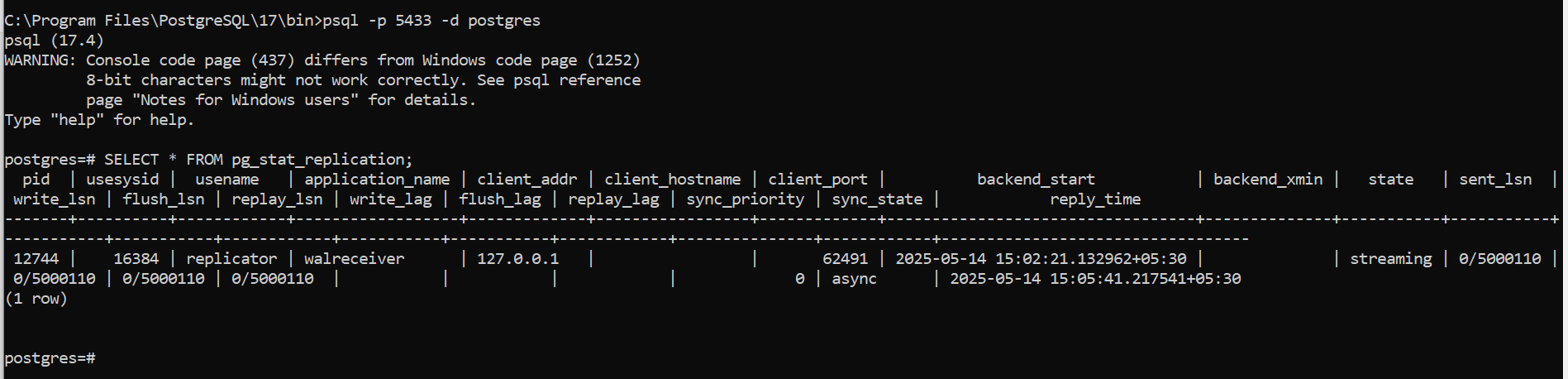
This starts the PostgreSQL standby server, which should automatically start in recovery mode.

* **Verify the setup**
* On Primary (Port 5433), Connect to primary:

**psql -p 5433 -d postgres**

Then check if replication is working:

**SELECT \* FROM pg\_stat\_replication;**



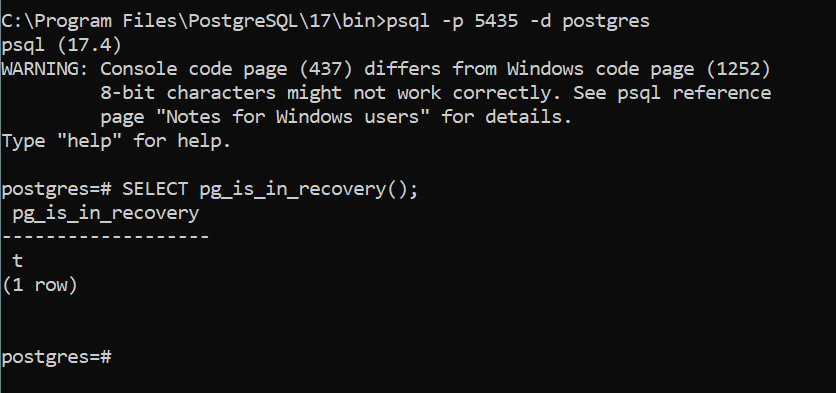
one row showing that the standby is connected and streaming.

* On Standby (Port 5435), connect to standby

**psql -p 5435 -d postgres**

**SELECT pg\_is\_in\_recovery();**

Output should be true, which confirms the standby is running in recovery mode.



**STEP 1: Create the rental\_log table on the primary (port 5433)**

Connect to the primary server:

**psql -p 5433 -d postgres**

Then run,

**CREATE TABLE rental\_log (**

**log\_id SERIAL PRIMARY KEY,**

**rental\_time TIMESTAMP,**

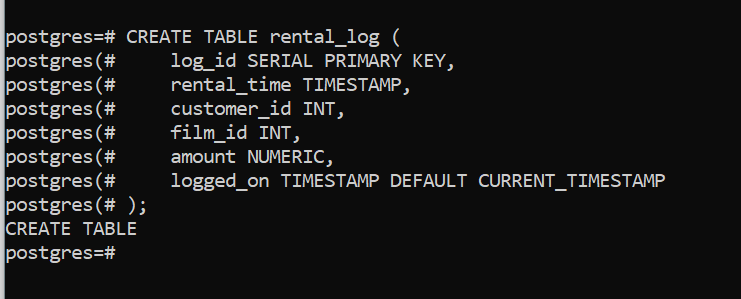
**customer\_id INT,**

**film\_id INT,**

**amount NUMERIC,**

**logged\_on TIMESTAMP DEFAULT CURRENT\_TIMESTAMP**

**);**



**This will automatically replicate to the standby** if replication is set up correctly (streaming replication replicates all WAL changes including schema changes).

**STEP 2: Create the stored procedure on the primary (port 5433)**

**CREATE OR REPLACE PROCEDURE sp\_add\_rental\_log(**

**p\_customer\_id INT,**

**p\_film\_id INT,**

**p\_amount NUMERIC**

**)**

**LANGUAGE plpgsql**

**AS $$**

**BEGIN**

**INSERT INTO rental\_log (rental\_time, customer\_id, film\_id, amount)**

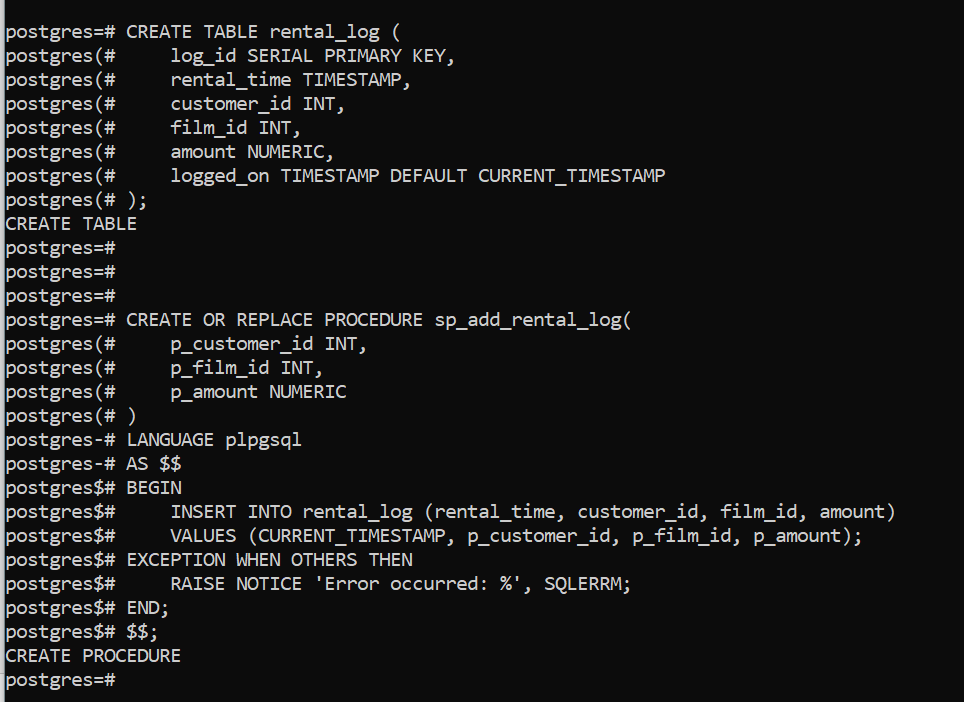
**VALUES (CURRENT\_TIMESTAMP, p\_customer\_id, p\_film\_id, p\_amount);**

**EXCEPTION WHEN OTHERS THEN**

**RAISE NOTICE 'Error occurred: %', SQLERRM;**

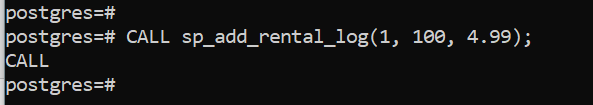
**END;**

**$$;**



**STEP 3: Call the procedure on the primary**

**CALL sp\_add\_rental\_log(1, 100, 4.99);**



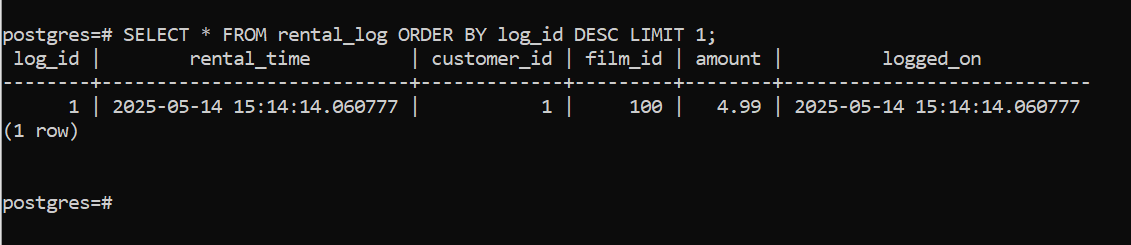
**STEP 4: Verify replication on the standby (port 5435)**

Connect to the standby:

**psql -p 5435 -d postgres**

Then run:

**SELECT \* FROM rental\_log ORDER BY log\_id DESC LIMIT 1;**



If replication is working, you will see the row that was inserted via the procedure on the primary.

**STEP 5: Add a trigger on the primary to log UPDATEs on rental\_log**

First, create a table for logging updates (optional but recommended):

**CREATE TABLE rental\_log\_update\_audit (**

**audit\_id SERIAL PRIMARY KEY,**

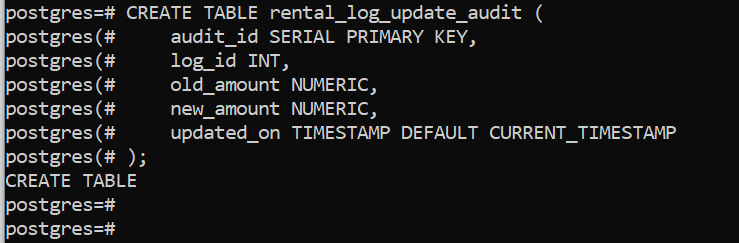
**log\_id INT,**

**old\_amount NUMERIC,**

**new\_amount NUMERIC,**

**updated\_on TIMESTAMP DEFAULT CURRENT\_TIMESTAMP**

**);**



Then create the trigger function:

**CREATE OR REPLACE FUNCTION fn\_log\_rental\_update()**

**RETURNS TRIGGER AS $$**

**BEGIN**

**INSERT INTO rental\_log\_update\_audit (log\_id, old\_amount, new\_amount)**

**VALUES (OLD.log\_id, OLD.amount, NEW.amount);**

**RETURN NEW;**

**END;**

**$$ LANGUAGE plpgsql;**

Now attach the trigger to the table:

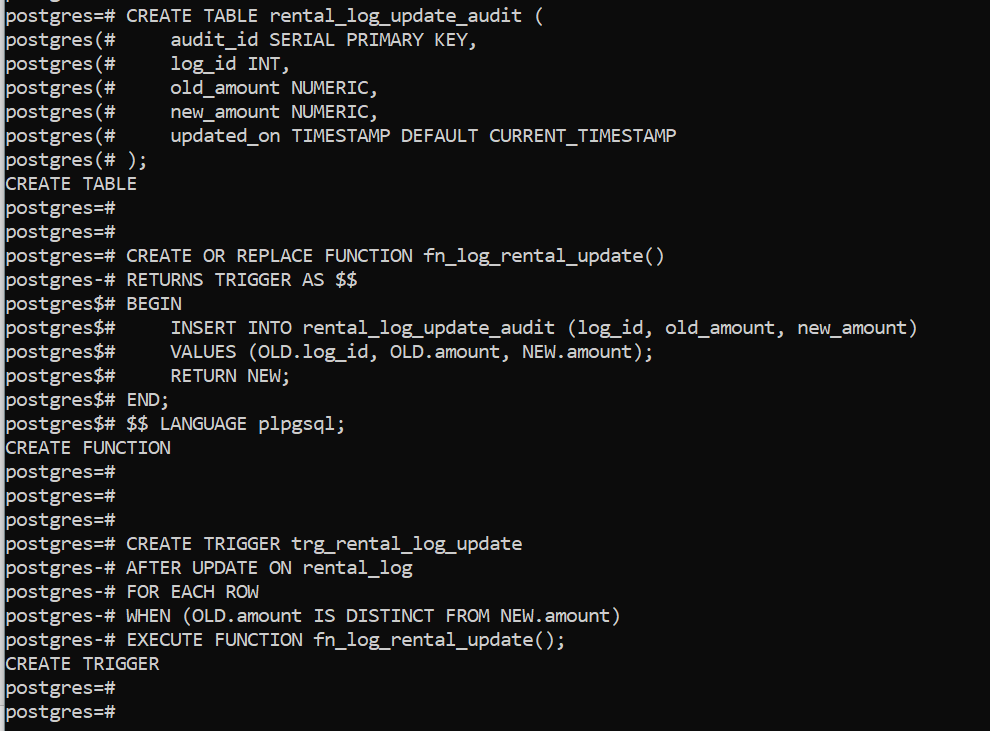
**CREATE TRIGGER trg\_rental\_log\_update**

**AFTER UPDATE ON rental\_log**

**FOR EACH ROW**

**WHEN (OLD.amount IS DISTINCT FROM NEW.amount)**

**EXECUTE FUNCTION fn\_log\_rental\_update();**



**STEP 6: Test the trigger (on primary)**

Update a row in the rental\_log table:

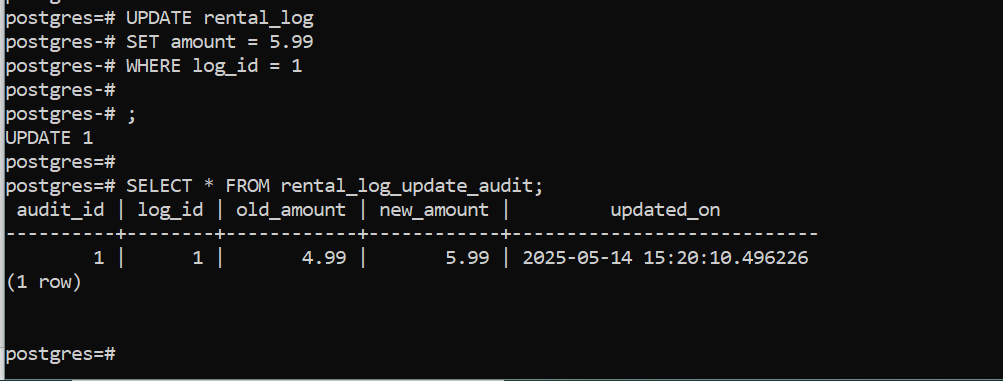
**UPDATE rental\_log**

**SET amount = 5.99**

**WHERE log\_id = 1;**

Then check the audit log:

**SELECT \* FROM rental\_log\_update\_audit;**



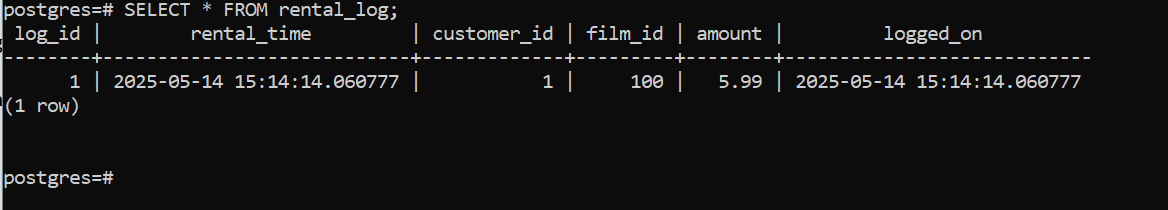
You should see the updated entry logged.

**STEP 7: Verify all data and triggers replicated**

Although **triggers and functions** do **not** replicate to standby, **table data** and schema changes **do**.

So from the **standby (5435)**:

**SELECT \* FROM rental\_log;**



**Summary**

| **Task** | **Where (Port)** | **What to Do** |
| --- | --- | --- |
| Create table | Primary 5433 | CREATE TABLE rental\_log (...) |
| Create procedure | Primary 5433 | CREATE PROCEDURE sp\_add\_rental\_log(...) |
| Call procedure | Primary 5433 | CALL sp\_add\_rental\_log(...) |
| Verify row exists | Standby 5435 | SELECT \* FROM rental\_log ORDER BY log\_id DESC LIMIT 1; |
| Create audit + trigger | Primary 5433 | Function, Trigger, and Audit table |
| Update data to test trigger | Primary 5433 | UPDATE rental\_log SET amount = ... WHERE log\_id = ... |
| Check audit table | Primary 5433 | SELECT \* FROM rental\_log\_update\_audit; |
| Triggers and SPs on standby | Standby 5435 | Not replicated; can only view replicated **data** |