Objectives

After completing this lab, you will be able to:

- Customize the configuration parameters of your PostgreSQL server instance
- Query the system catalog to retrieve metadata about database objects.

```
theia@theiadocker-rajendraabro:/home/project$ wget
https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/example-guided-project/flig
hts_RUSSIA_small.sql
--2025-12-24 02:30:30--
https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/example-guided-project/flig
hts_RUSSIA_small.sql
Resolving cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud
(cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud)... 169.63.118.104,
169.63.118.104
Connecting to cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud
(cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud)|169.63.118.104|:443...
connected.
HTTP request sent, awaiting response... 200 OK
Length: 103865229 (99M) [application/x-sql]
Saving to: ‘flights_RUSSIA_small.sql’

    fligh   0%       0  --.-KB/s               flight  16%  16.01M  28.3MB/s             flights  32%
32.01M  40.6MB/s            flights_  48%  48.01M  47.6MB/s           flights_R  61%  60.57M
50.1MB/s           flights_RU  68%  68.01M  46.3MB/s         flights_RUS  90%  89.24M  53.5MB/s
lights_RUSS  96%  96.01M  51.4MB/s          flights_RUS 100%  99.05M  52.3MB/s    in 1.9s

2025-12-24 02:30:32 (52.3 MB/s) - ‘flights_RUSSIA_small.sql’ saved [103865229/103865229]
```

Restore to new database

```
postgres=# \i flights_RUSSIA_small.sql
SET
SET
SET
SET
SET
```

```
SET
SET
SET
DROP DATABASE
CREATE DATABASE
psql (15.2 (Ubuntu 15.2-1.pgdg18.04+1), server 13.2)
You are now connected to database "demo" as user "postgres".
SET
SET
SET
SET
SET
SET
SET
SET
CREATE SCHEMA
COMMENT
CREATE EXTENSION
COMMENT
SET
CREATE FUNCTION
CREATE FUNCTION
COMMENT
SET
SET
CREATE TABLE
COMMENT
COMMENT
COMMENT
COMMENT
CREATE VIEW
COMMENT
COMMENT
COMMENT
COMMENT
CREATE TABLE
COMMENT
COMMENT
COMMENT
COMMENT
COMMENT
COMMENT
CREATE VIEW
COMMENT
```

COMMENT
COMMENT
COMMENT
COMMENT
COMMENT
CREATE TABLE
COMMENT
COMMENT
COMMENT
COMMENT
COMMENT
CREATE TABLE
COMMENT
COMMENT
COMMENT
COMMENT
CREATE TABLE
COMMENT
COMMENT
COMMENT
COMMENT
COMMENT
COMMENT
COMMENT
COMMENT
COMMENT
COMMENT
COMMENT
CREATE SEQUENCE
ALTER SEQUENCE
CREATE VIEW
COMMENT
COMMENT
COMMENT
COMMENT
COMMENT
COMMENT
COMMENT
COMMENT
COMMENT
COMMENT
COMMENT
COMMENT
COMMENT

```
COMMENT
COMMENT
COMMENT
COMMENT
COMMENT
COMMENT
COMMENT
COMMENT
CREATE VIEW
COMMENT
COMMENT
COMMENT
COMMENT
COMMENT
COMMENT
COMMENT
COMMENT
COMMENT
COMMENT
COMMENT
CREATE TABLE
COMMENT
COMMENT
COMMENT
COMMENT
CREATE TABLE
COMMENT
COMMENT
COMMENT
COMMENT
COMMENT
CREATE TABLE
COMMENT
COMMENT
COMMENT
COMMENT
COMMENT
COMMENT
ALTER TABLE
COPY 9
COPY 104
COPY 579686
COPY 262788
COPY 33121
```

```
 setval
--------
  33121
(1 row)

COPY 1339
COPY 1045726
COPY 366733
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER DATABASE
ALTER DATABASE
```

Verify restoring database

```
demo=# \dt
               List of relations
  Schema  |      Name       | Type  |  Owner
----------+-----------------+-------+----------
 bookings | aircrafts_data  | table | postgres
 bookings | airports_data   | table | postgres
 bookings | boarding_passes | table | postgres
 bookings | bookings        | table | postgres
 bookings | flights         | table | postgres
 bookings | seats           | table | postgres
 bookings | ticket_flights  | table | postgres
 bookings | tickets         | table | postgres
```

```
(8 rows)
```

A PostgreSQL server instance has a corresponding file named `postgresql.conf` that contains the configuration parameters for the server. By modifying this file, you can enable, disable, or otherwise customize the settings of your PostgreSQL server instance to best suit your needs as a database administrator.

```
demo=# SHOW wal_level;
 wal_level
-----------
 replica
(1 row)
demo=# ALTER SYSTEM SET wal_level = 'logical';
ALTER SYSTEM

demo=# SHOW wal_level;
FATAL:  terminating connection due to administrator command
server closed the connection unexpectedly
        This probably means the server terminated abnormally
        before or while processing the request.
The connection to the server was lost. Attempting reset: Succeeded.
psql (15.2 (Ubuntu 15.2-1.pgdg18.04+1), server 13.2)

demo=# SHOW wal_level;
 wal_level
-----------
 logical
(1 row)
```

Exercise 2: Navigate the System Catalog

`CREATE DATABASE` command, a new database is created on the disk and a new row is automatically inserted into the `pg_database` system catalog table, storing metadata about that database.

Connect to current demo database

```
\connect demo

postgres=# \connect demo
psql (15.2 (Ubuntu 15.2-1.pgdg18.04+1), server 13.2)
You are now connected to database "demo" as user "postgres".
```

```
demo=# SELECT * FROM pg_tables WHERE schemaname = 'bookings';
hastriggers | rowsecurity
------------+-----------------+------------+------------+------------+----------+-------------+-------------
 bookings   | ticket_flights  | postgres   |            | t          | f        | t           | f
 bookings   | boarding_passes | postgres   |            | t          | f        | t           | f
 bookings   | aircrafts_data  | postgres   |            | t          | f        | t           | f
 bookings   | flights         | postgres   |            | t          | f        | t           | f
 bookings   | airports_data   | postgres   |            | t          | f        | t           | f
 bookings   | seats           | postgres   |            | t          | f        | t           | f
 bookings   | tickets         | postgres   |            | t          | f        |
type :/q
```

Suppose as the database administrator, you would like to enable row-level security for the `boarding_passes` table in the `demo` database. When row security is enabled on a table, all normal access to the table for selecting or modifying rows must be specified by a row security policy.

```
demo=# SELECT * FROM pg_tables WHERE schemaname = 'bookings';
demo=# ALTER TABLE boarding_passes ENABLE ROW LEVEL SECURITY;
ALTER TABLE
```

Let's connect your work in the previous section about PostgreSQL instance configuration to the system catalogs. Earlier, you used `SHOW` statements to display configuration parameters. There's also a system catalog called `pg_settings` that stores data about configuration parameters of the PostgreSQL server.

```
demo=# SELECT name, setting, short_desc FROM pg_settings WHERE name = 'wal_level';
demo=#
```