Lecture 04: Importing and Exporting Data

DATA 351: Data Management with SQL

Lucas P. Cordova, Ph.D.

2026-01-26

This lecture covers the importing and exporting data.

Table of contents

0.1 Today's Agenda

Part 1: Importing Data with `\COPY`

- Create a clean CSV
- Load data into PostgreSQL
- Verify the import

Part 2: Exporting Data with `\COPY`

- Export tables and queries
- Customize export options
- Validate exported files

1 Getting Data Into PostgreSQL

1.1 Why Not Just INSERT Everything?

Imagine you have 65,000 survey responses...

```sql
INSERT INTO survey VALUES (1, 'USA', 'Developer', 85000);
INSERT INTO survey VALUES (2, 'UK', 'Designer', 72000);
INSERT INTO survey VALUES (3, 'Germany', 'Developer', 91000);
-- ... 64,997 more times
```

This is:

- Slow (each INSERT is a separate transaction)
- Error-prone (one typo and you start over)
- Painful (your fingers will hate you)

Solution: Bulk import with `\COPY`

1.2 The Office: Our Sample Dataset

Today we will work with some Dunder Mifflin employee data again.

1.3 The Dataset

employee_id	full_name	department	salary_usd
1	Michael Scott	Management	75000.00
2	Dwight Schrute	Sales	62000.00
3	Pam Beesly	Reception	42000.00
4	Jim Halpert	Sales	61000.00

1.4 Expanded Dataset for Today

I have added a few more employees and columns so we can practice more SQL:

employee_id	full_name	department	salary_usd	performance_rating	years_experience
1	Michael Scott	Management	75000.00	3.2	15
2	Dwight Schrute	Sales	62000.00	4.8	12
3	Pam Beesly	Reception	42000.00	3.9	8
4	Jim Halpert	Sales	61000.00	4.1	10

em-ployee_id	full_name	department	salary_usd	performance_rat-ing	years_experience
5	Angela Martin	Accounting	52000.00	4.5	11
6	Kevin Malone	Accounting	48000.00	2.1	9
7	Oscar Martinez	Accounting	54000.00	4.7	13
8	Stanley Hudson	Sales	58000.00	3.0	20

1.5 Hands-On: Create the CSV File

Step 1: Create a file named `employees_import.csv`, I recommend in your Downloads directory for now.

Copy this exact content (hover over the data and click the copy button that appearsto the right):

```
employee_id,full_name,department,email,hire_date,salary_usd,is_manager,performance_rating,yea
1,Michael Scott,Management,michael.scott@dundermifflin.com,2005-03-24,75000.00,true,3.2,15,,
2,Dwight Schrute,Sales,dwight.schrute@dundermifflin.com,2006-04-12,62000.00,false,4.8,12,0.08
3,Pam Beesly,Reception,pam.beesly@dundermifflin.com,2007-07-02,42000.00,false,3.9,8,,
4,Jim Halpert,Sales,jim.halpert@dundermifflin.com,2005-10-05,61000.00,false,4.1,10,0.07,2026-
5,Angela Martin,Accounting,angela.martin@dundermifflin.com,2006-08-15,52000.00,false,4.5,11,
6,Kevin Malone,Accounting,kevin.malone@dundermifflin.com,2007-02-28,48000.00,false,2.1,9,,202
7,Oscar Martinez,Accounting,oscar.martinez@dundermifflin.com,2005-06-01,54000.00,false,4.7,13
8,Stanley Hudson,Sales,stanley.hudson@dundermifflin.com,2004-11-20,58000.00,false,3.0,20,0.05
```

1.6 Hands-On: Move the CSV to a Safe Location

Why a "safe" location?

PostgreSQL needs permission to read your file. Some folders are restricted.

macOS / Linux:

```
cp ~/Downloads/employees_import.csv /tmp/employees_import.csv
ls -l /tmp/employees_import.csv
```

Windows (PowerShell):

```
Copy-Item $HOME\Downloads\employees_import.csv C:\Users\Public\employees_import.csv
Get-Item C:\Users\Public\employees_import.csv
```

1.7 Hands-On: Connect to PostgreSQL

Open your terminal and connect:

```
psql -U postgres -h localhost
```

You should see a prompt like:

```
postgres=#
```

> **Warning**
>
> - If your prompt spits out `command not found: psql` or something similar, psql is not in your PATH. Check out the resource on Canvas → Week 2 Lesson Plan → Adding PSQL to your PATH.
> - If you see `psql: could not connect to server: No such file or directory` or something similar, you are not connected to the database or there are credential issues. Try `psql -U postgres -h localhost` instead.

1.8 Hands-On: Create the Database

At the `postgres=#` prompt, run:

```
DROP DATABASE IF EXISTS office_db;
CREATE DATABASE office_db;
\c office_db
```

You should see:

```
You are now connected to database "office_db" as user "postgres".
```

> **Note**
>
> The `\c` command connects you to the new database.

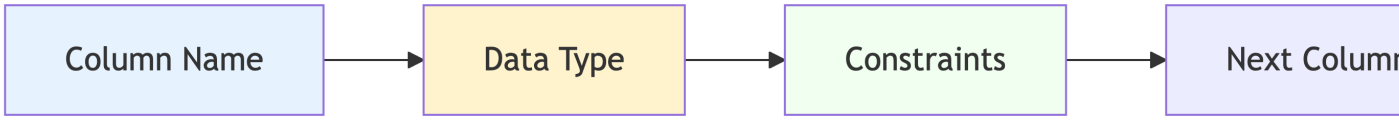
1.9 Hands-On: Create the Table

Now we need a table that matches our CSV columns exactly. Copy this exact content (hover over the data and click the copy button that appears to the right):

```
CREATE TABLE employees (
    employee_id         INTEGER PRIMARY KEY,
    full_name           TEXT NOT NULL,
    department          TEXT NOT NULL,
    email               TEXT,
    hire_date           DATE NOT NULL,
    salary_usd          NUMERIC(10,2) NOT NULL,
    is_manager          BOOLEAN NOT NULL,
    performance_rating  NUMERIC(2,1),
    years_experience    INTEGER,
    commission_rate     NUMERIC(3,2),
    last_login          TIMESTAMP
);
```

Verify with `\d employees` to see the structure.

1.10 Understanding the CREATE TABLE Statement



Each column definition has:

- **Name:** What you call the column (e.g., `salary_usd`)
- **Data Type:** What kind of data it holds (e.g., `NUMERIC(10,2)`)
- **Constraints:** Rules the data must follow (e.g., `NOT NULL`)

1.11 Hands-On: Import the CSV

The moment of truth!

macOS / Linux:

```
\COPY employees FROM '/tmp/employees_import.csv' WITH (FORMAT csv, HEADER true)
```

Windows:

```
\COPY employees FROM 'C:\Users\Public\employees_import.csv' WITH (FORMAT csv, HEADER true)
```

You should see: `COPY 8`

That means 8 rows were imported successfully!

1.12 Hands-On: Verify Your Data

Count the rows:

```
SELECT COUNT(*) FROM employees;
```

count
8

View all the data:

```
SELECT * FROM employees ORDER BY employee_id;
```

Take a moment to verify the data looks correct.

1.13 The Anatomy of `\COPY`

```
\COPY employees FROM '/tmp/employees_import.csv' WITH (FORMAT csv, HEADER true)
```

Let's break this down:

Part	Meaning
`\COPY`	Client-side copy command
`employees`	Target table name
`FROM '/tmp/...'`	Source file path
`FORMAT csv`	File is comma-separated
`HEADER true`	First row is column names

Feature	\COPY	COPY

1.14 \COPY vs COPY: What's the Difference?

Feature	\COPY	COPY
Runs on	Your computer (client)	Database server
File location	Your filesystem	Server filesystem
Permissions	Your user permissions	postgres user permissions
Best for	Development, small files	Production, large files

Rule of thumb: Use \COPY in this class. It is safer and easier.

1.15 Exercise: Check Your Import

Write queries to answer (work with a neighbor if your system is acting up):

1. How many employees are in the Sales department?
2. What is Michael Scott's email?
3. Which employee has the highest performance rating?

Take 3 minutes, then we will review.

1.16 Exercise Solutions

1. Sales department count:

```
1 SELECT COUNT(*) FROM employees WHERE department = 'Sales';
```

count
3

2. Michael's email:

```
1 SELECT email FROM employees WHERE full_name = 'Michael Scott';
```

email
michael.scott@dundermifflin.com

3. Highest performance rating:

```
SELECT full_name, performance_rating
FROM employees
ORDER BY performance_rating DESC
LIMIT 1;
```

full_name	performance_rating
Dwight Schrute	4.8

2 Getting Data Out of PostgreSQL

2.1 Why Export Data?

Real-world scenarios where you need to get data OUT of PostgreSQL:

- Share query results with colleagues who do not use SQL
- Create reports for stakeholders in Excel or Google Sheets
- Feed data into visualization tools like Tableau or Power BI
- Backup specific tables or query results
- Transfer data to another database system

2.2 The `\COPY` Command for Export

Just like importing, we use `\COPY` for exporting:

```
\COPY (SELECT * FROM employees) TO '/tmp/employees_export.csv' WITH (FORMAT csv, HEADER true)
```

You should see: `COPY 8`

That means 8 rows were exported successfully!

2.3 Anatomy of `\COPY` for Export

```
\COPY (SELECT * FROM employees) TO '/tmp/employees_export.csv' WITH (FORMAT csv, HEADER true)
```

Part	Meaning
`\COPY`	Client-side copy command
`(SELECT * FROM employees)`	Query to export (must be in parentheses)
`TO '/tmp/...'`	Destination file path
`FORMAT csv`	Output as comma-separated values
`HEADER true`	Include column names as first row

2.4 Exporting a Table vs Exporting a Query

Export an entire table:

```
\COPY employees TO '/tmp/all_employees.csv' WITH (FORMAT csv, HEADER true)
```

Export a query result:

```
\COPY (SELECT full_name, salary_usd FROM employees WHERE department = 'Sales')
TO '/tmp/sales_team.csv' WITH (FORMAT csv, HEADER true)
```

> **Note**
>
> When exporting a query, you MUST wrap it in parentheses.

2.5 Hands-On: Export Sales Department

Let's export just the Sales team with their commission information:

macOS / Linux:

```
\COPY (SELECT full_name, salary_usd, commission_rate
       FROM employees
       WHERE department = 'Sales'
       ORDER BY salary_usd DESC)
TO '/tmp/sales_export.csv' WITH (FORMAT csv, HEADER true)
```

Windows:

```
\COPY (SELECT full_name, salary_usd, commission_rate
       FROM employees
       WHERE department = 'Sales'
       ORDER BY salary_usd DESC)
TO 'C:\Users\Public\sales_export.csv' WITH (FORMAT csv, HEADER true)
```

2.6 Verify Your Export

Check the file contents (from your terminal, not psql):

macOS / Linux:

```
cat /tmp/sales_export.csv
```

Windows (PowerShell):

```
Get-Content C:\Users\Public\sales_export.csv
```

```
full_name,salary_usd,commission_rate
Dwight Schrute,62000.00,0.08
Jim Halpert,61000.00,0.07
Stanley Hudson,58000.00,0.05
```

2.7 Export Options

Common options for `\COPY ... TO`:

Option	Description	Example
`FORMAT csv`	Comma-separated values	Most common
`FORMAT text`	Tab-separated (default)	For TSV files
`HEADER true`	Include column headers	Usually want this
`DELIMITER ';'`	Custom delimiter	European CSV format
`NULL 'NA'`	Represent NULL as 'NA'	For R compatibility
`QUOTE '"'`	Character for quoting strings	Default is double quote

2.8 Hands-On: Export with Custom Delimiter

Some European systems use semicolons because commas are decimal separators:

```
\COPY (SELECT full_name, salary_usd FROM employees)
TO '/tmp/employees_euro.csv'
WITH (FORMAT csv, HEADER true, DELIMITER ';')
```

Result:

```
full_name;salary_usd
Michael Scott;75000.00
Dwight Schrute;62000.00
```

2.9 Exporting Aggregated Data

Export summary statistics, not raw data:

```
\COPY (
    SELECT
        department,
        COUNT(*) AS employee_count,
        ROUND(AVG(salary_usd), 2) AS avg_salary,
        SUM(salary_usd) AS total_salary
    FROM employees
    GROUP BY department
    ORDER BY avg_salary DESC
) TO '/tmp/dept_summary.csv' WITH (FORMAT csv, HEADER true)
```

This exports a 4-row summary instead of all 8 employee records.

2.10 Handling NULL Values in Export

By default, NULL exports as empty string. You can customize this:

```
\COPY (SELECT full_name, commission_rate FROM employees)
TO '/tmp/with_nulls.csv'
WITH (FORMAT csv, HEADER true, NULL 'N/A')
```

```
full_name,commission_rate
Michael Scott,N/A
Dwight Schrute,0.08
Pam Beesly,N/A
```

Useful when the receiving system needs explicit NULL markers.

2.11 Exercise: Export a Targeted Report

Create a CSV export that shows:

- Employee name
- Department
- Salary

Only include employees who are **not** managers and make more than $50,000.

Export to `/tmp/employee_report.csv` (or `C:\Users\Public\` on Windows).

Take 4 minutes.

2.12 Exercise Solution

```
\COPY (
    SELECT
        full_name,
        department,
        salary_usd
    FROM employees
    WHERE is_manager = FALSE
      AND salary_usd > 50000
    ORDER BY salary_usd DESC
) TO '/tmp/employee_report.csv' WITH (FORMAT csv, HEADER true)
```

```
full_name,department,salary_usd
Dwight Schrute,Sales,62000.00
Jim Halpert,Sales,61000.00
Stanley Hudson,Sales,58000.00
Oscar Martinez,Accounting,54000.00
Angela Martin,Accounting,52000.00
```

2.13 Common Export Mistakes

1. Forgetting parentheses around SELECT:

```
-- WRONG: No parentheses
\COPY SELECT * FROM employees TO '/tmp/file.csv' ...

-- RIGHT: With parentheses
\COPY (SELECT * FROM employees) TO '/tmp/file.csv' ...
```

2. File permission issues:

```
-- May fail if you don't have write permission
\COPY employees TO '/etc/employees.csv' ...

-- Use a directory you can write to
\COPY employees TO '/tmp/employees.csv' ...
```

2.14 `\COPY` vs `COPY` for Export

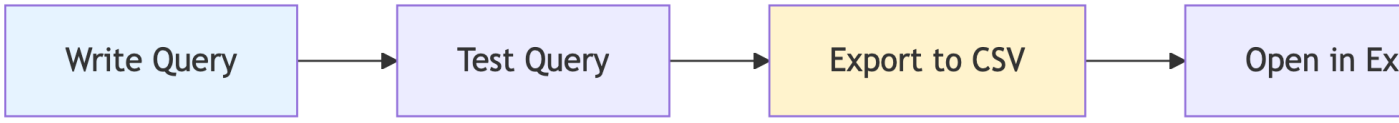
Just like with import, there are two versions:

Feature	`\COPY ... TO`	`COPY ... TO`
Runs on	Your computer (client)	Database server
File location	Your filesystem	Server filesystem
Permissions	Your user permissions	postgres user permissions
Best for	Development, sharing files	Server backups, ETL

> **Tip**
>
> **Rule of thumb:** Use `\COPY` in this class. The file ends up on YOUR machine.

2.15 Real-World Export Workflow

A typical data export workflow:



1. Write and test your query in psql
2. Export results to CSV
3. Open in spreadsheet software
4. Share with non-technical colleagues

2.16 Quick Reference: Import vs Export

Task	Command
Import entire CSV	`\COPY table FROM 'file.csv' WITH (FORMAT csv, HEADER true)`
Export entire table	`\COPY table TO 'file.csv' WITH (FORMAT csv, HEADER true)`
Export query result	`\COPY (SELECT ...) TO 'file.csv' WITH (FORMAT csv, HEADER true)`

> **Note**
>
> Remember:
>
> - `FROM` = bringing data IN
> - `TO` = sending data OUT

2.17 Questions?

We covered:

- Importing CSV files with `\COPY`
- Validating data after import
- Exporting tables and query results
- Controlling export format options

What questions do you have?