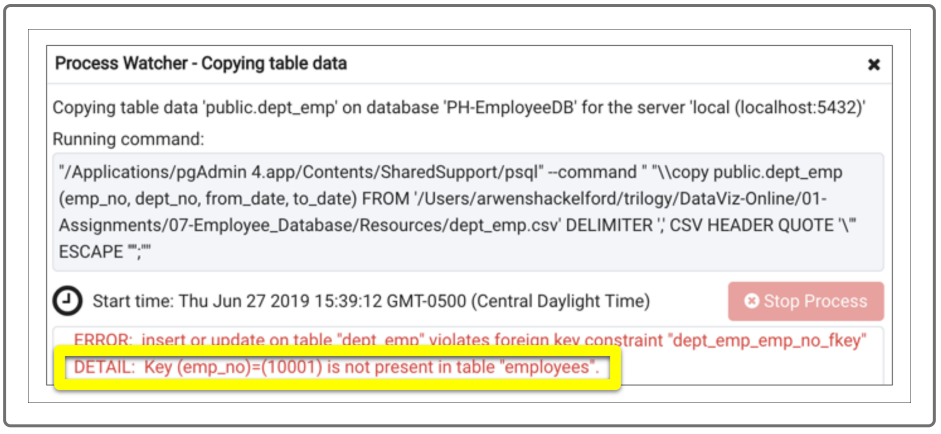
7.2.4 Troubleshoot Imports

Every developer in every coding language will encounter an error. Sometimes they're due to typos, other times the code syntax is incorrect. In each case, there will be an error code provided. These codes help us (the developers) research the error and investigate solutions.

What if we run into an error during our table imports? What could be the possible cause? We'll help Bobby practice troubleshooting because this skill is one that is worth developing early.

Handle Common Errors

What if Bobby runs into an error while he's importing the data? Below is an example of a likely error:



Because the `FOREIGN KEY` constraint references other tables, we need to import the data in a specific order.

NOTE

> The error message above provides sufficient error text for a developer to Google. If searching for the detail in the data or your code doesn't help, Google the error itself for additional troubleshooting assistance.

For example, the dept_emp table references the Employees table through its foreign key. If there is no data in the Employees table, then there are no foreign keys to link to, and an error will occur.

Handle Mismatched Data Types

Another common scenario is when a data type in a table we've created doesn't match the CSV data. What should we do?

Because data within a Postgres database is static, we can't go back and fix a typo in our original table creation code. Remember the "table already exists" error from earlier? Postgres has this check in place so we don't accidentally overwrite good data.

If you encounter an error during data import that corresponds to the incorrect data type in the table, you'll need to start fresh by deleting the

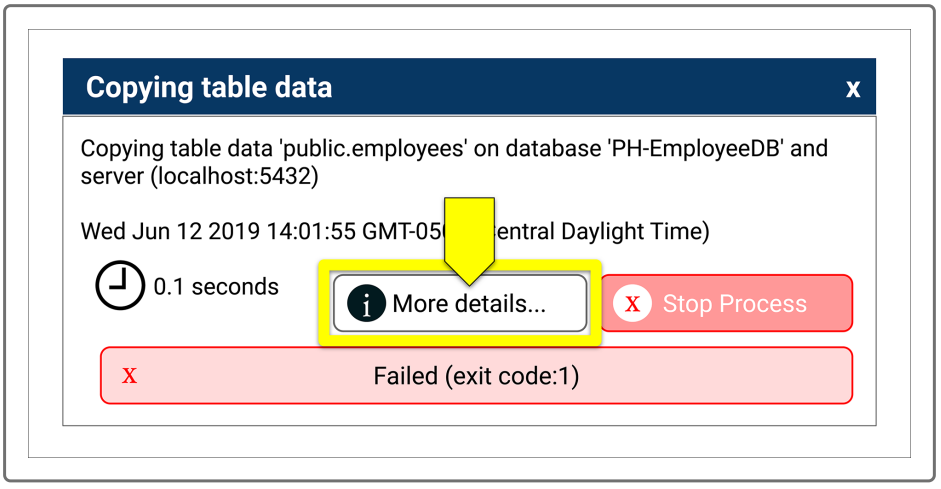
table and creating a new one in its place.

Trying to update the `CREATE TABLE` statement and then re-executing it won't work because SQL doesn't allow duplicates or data overrides. Creating an entirely new table with a new name would work, but that adds two more complications:

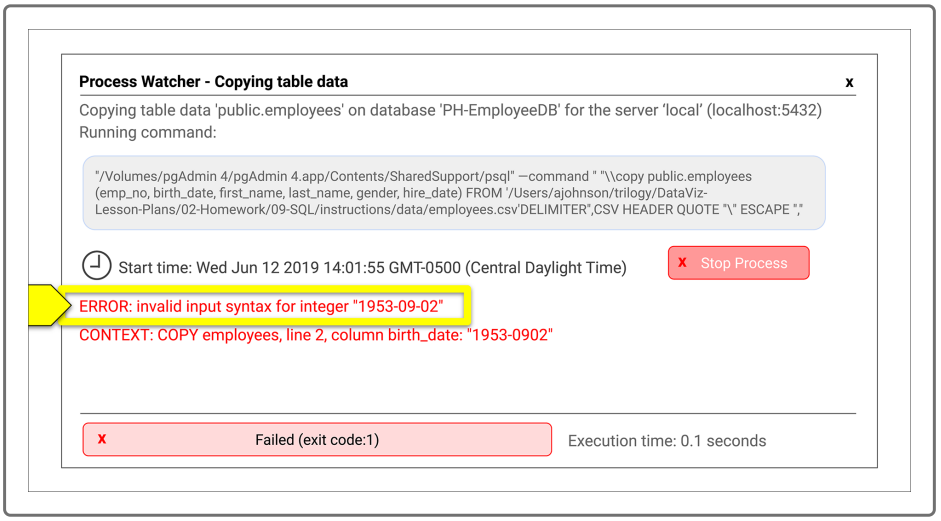
1. The old table will still exist and clutter up the fresh database.
2. The database doesn't follow the original layout in the entity relationship diagram.

The best solution is to delete the current, unusable table using the `DROP TABLE` statement, then create a new one with updated code.

Imagine that we created the Employees table with the wrong data type assigned to the birth_date column. When we tried to import the CSV data to the table, we encountered the following error:



That error doesn't tell us much more than that the import failed, but clicking on "More details" is enlightening.



Now the error tells us exactly where the import failed: The birth_date column should be a date instead of an integer. The correct resolution is to remove, or drop, the current table then recreate it using the proper data type.

Drop a Table

To drop the table, the following code is used:

```
DROP TABLE employees CASCADE;
```

- `DROP TABLE employees` tells Postgres that we want to remove the Employees table from the database completely.
- `CASCADE;` indicates that we also want to remove the connections to other tables in the database.

IMPORTANT

> Even without data, by adding foreign keys that reference other tables, we've created a network of data connections. Not every table will need the `CASCADE` constraint, but it will come up when you need to drop a table that already has a defined relationship with another. Any table that does not reference a foreign key can be dropped without the `CASCADE` constraint.

After the table has been dropped, we can adjust the `CREATE TABLE employees` code block to match the CSV's data types. After re-executing the code to create that table, Postgres will add the new table to the database. Now we're ready to finish the data import.