Pulling data from databases

DATA PROCESSING IN SHELL

Susan Sun
Data Person

datacamp

sql2csv: documentation

`sql2csv` :

- executes an SQL query on a large variety of SQL databases (e.g. MS SQL, MySQL, Oracle, PostgreSQL, SQLite)
- outputs the result to a CSV file

Documentation:

```
sql2csv -h
```

```
usage: sql2csv [-h] [-v] [-l] [-V] [--db CONNECTION_STRING] [--query QUERY]
               [-e ENCODING] [-H]
               [FILE]
```

sql2csv: querying against the database

Sample syntax:

```
sql2csv --db "sqlite:///SpotifyDatabase.db" \
        --query "SELECT * FROM Spotify_Popularity" \
        > Spotify_Popularity.csv
```

1. Establishing database connection:

- `--db` is followed by the database connection string
- **SQLite**: starts with `sqlite:///` and ends with `.db`
- **Postgres** & **MySQL**: starts with `postgres:///` or `mysql:///` and with **no** `.db`

sql2csv: querying against the database

Sample syntax:

```
sql2csv --db "sqlite:///SpotifyDatabase.db" \
        --query "SELECT * FROM Spotify_Popularity" \
        > Spotify_Popularity.csv
```

2. Querying against the database:

- `--query` is followed by the SQL query string
- Use SQL syntax compatible with the database
- Write query in one line with no line breaks

sql2csv: querying against the database

Sample syntax:

```
sql2csv --db "sqlite:///SpotifyDatabase.db" \
        --query "SELECT * FROM Spotify_Popularity" \
        > Spotify_Popularity.csv
```

3. Saving the output:

- `>` : re-directs output to new local CSV file
- Otherwise, will only print query results to console

Let's practice!

DATA PROCESSING IN SHELL

Manipulating data using SQL syntax

DATA PROCESSING IN SHELL

Susan Sun
Data Person

csvsql: documentation

`csvsql` :

- applies SQL statements to one or more CSV files
- creates an in-memory SQL database that temporarily hosts the file being processed
- suitable for small to medium files only

Documentation:

```
csvsql -h
```

```
usage: csvsql [-h] [-d DELIMITER] [-t] [-q QUOTECHAR] [-u {0,1,2,3}] [-b]
              [-p ESCAPECHAR] [-z FIELD_SIZE_LIMIT] [-e ENCODING] [-L LOCALE]
```

csvsql: applying SQL to a local CSV file

Sample syntax:

```
ls
```

```
Spotify_MusicAttributes.csv
```

csvsql: applying SQL to a local CSV file

Sample syntax:

```
csvsql --query "SELECT * FROM Spotify_MusicAttributes LIMIT 1" \
    Spotify_MusicAttributes.csv
```

```
track_id,danceability,duration_ms,instrumentalness,loudness,tempo,time_signature
118GQ70Sp6pMqn6w1oKuki,0.787,124016.0,0.784,-10.457,119.988,4.0
```

csvsql: applying SQL to a local CSV file

Sample syntax:

```
csvsql --query "SELECT * FROM Spotify_MusicAttributes LIMIT 1" \
    data/Spotify_MusicAttributes.csv | csvlook
```

```
| track_id                | danceability | duration_ms | instrumentalness ...
| ----------------------- | ------------ | ----------- | ---------------- ...
| 118GQ70Sp6pMqn6w1oKuki |        0.787 |     124,016 |            0.784 ...
```

csvsql: applying SQL to a local CSV file

Sample syntax:

```
csvsql --query "SELECT * FROM Spotify_MusicAttributes LIMIT 1" \
    data/Spotify_MusicAttributes.csv > OneSongFile.csv
```

```
ls
```

```
OneSongFile.csv
```

csvsql: joining CSVs using SQL syntax

Sample syntax:

```
csvsql --query "SELECT * FROM file_a INNER JOIN file_b..." file_a.csv file_b.csv
```

Note:

- SQL Query must be written in one line, no breaks
- Indicate CSV files in order of appearance in SQL

Let's practice!

DATA PROCESSING IN SHELL

Pushing data back to database

DATA PROCESSING IN SHELL

Susan Sun
Data Person

csvsql: documentation

`csvsql`:

- execute SQL statements directly on a database
- supports both creating tables and inserting data.

More option arguments:

- `--insert`
- `--db`
- `--no-inference` & `--no-constraints`

csvsql: pushing data back to database

Sample syntax:

```
csvsql --db "sqlite:///SpotifyDatabase.db" \
       --insert Spotify_MusicAttributes.csv
```

Note:

1. Line break is used to keep code clean and readable
2. Use three forward slashes to initiate database name
3. End with file extension `.db` for SQLITE database

csvsql: pushing data back to database

Sample syntax:

```
csvsql --db "sqlite:///SpotifyDatabase.db" \
       --insert Spotify_MusicAttributes.csv
```

Documentation:

```
csvsql -h
```

```
--insert  In addition to creating the table, also insert the
          data into the table. Only valid when --db is specified.
```

csvsql: pushing data back to database

Sample syntax:

```
csvsql --no-inference --no-constraints \
       --db "sqlite:///SpotifyDatabase.db" \
       --insert Spotify_MusicAttributes.csv
```

Documentation:

```
csvsql -h
```

```
--no-inference   Disable type inference when parsing the input.
--no-constraints Generate a schema without length limits or null checks.
```

Let's practice!

DATA PROCESSING IN SHELL