Python Intern Task - CSV Report Processing

An important part of the operations of an AdTech platform is making sure metrics reported by different systems don't have significant discrepancies. These systems usually have peculiar report interfaces, so we need some automation to obtain data that we can compare.

We have an ad campaign where ads are displayed in multiple countries. A single ad view is called an *impression* and users *click* on ads. Ads are designed and targeted to maximize *click to impression rate* (*CTR*). CTR is calculated by dividing the number of clicks by the number of impressions and is usually written as a percentage.

A third-party ad server company offers CSV reports with daily impression numbers and CTRs for our campaign, aggregated per state (or any other ISO 3166-2 subdivision). Our system has reports aggregated by country.

Your task is to write a Python script that reads a CSV file of such third-party data. Based on standard input, the script should output a CSV file with a report aggregated by date and country.

Input format: UTF-8 or UTF-16 CSV file (with any kind of line endings), with columns: date (MM/DD/YYYY), state name, number of impressions and CTR percentage.

Output format: UTF-8 CSV file with Unix line endings, with columns: date (YYYY-MM-DD), three letter country code (or XXX for unknown states), number of impressions, number of clicks (rounded, assuming the CTR is exact). Rows are sorted lexicographically by date followed by the country code.

If the input file has errors, appropriate error messages should be written to the standard error (stderr). An output CSV should be produced if the errors are not critical (if it's not clear which ones are critical, the solution should include an explanation of your reasoning).

Tips:

- UTF-16 has a byte order mark, otherwise it would be ambiguous. Just like UTF-8, it has a variable width, but this detail shouldn't be noticeable in the solution.
- While a single date represents different times in different regions, here they are all UTC.
- Region names and codes used are provided by https://pypi.org/project/pycountry/
- We use Python 3.7; you can use external libraries available on PyPI (like pycountry) if you submit a requirements.txt listing them with versions (these are output by `pip freeze`). If a different Python version is needed, please provide a Dockerfile setting up a proper environment for running the script.
- CSV isn't standard and has many dialects, but here we can assume that all fields have no quoting, no commas and are separated literally by commas.

What we'd like to see:

- Readable code (consider comments and docstrings that would help understand and maintain it)
- Error handling

Example input:

```
01/21/2019,Mandiana,883,0.38%
01/21/2019,Lola,76,0.78%
01/21/2019,Fāryāb,919,0.67%
01/22/2019,Lola,201,0.82%
01/22/2019,Beroun,139,0.61%
01/22/2019,Mandiana,1050,0.93%
01/23/2019,😺,777,0.22%
01/23/2019,Gaoual,72,0.7%
01/23/2019,Lola,521,0.19%
01/24/2019,Beroun,620,0.1%
01/24/2019,Unknown,586,0.86%
01/24/2019,😺,1082,0.68%
```

Example output:

```
2019-01-21,AFG,919,6
2019-01-21,GIN,959,4
2019-01-22,CZE,139,1
2019-01-22,GIN,1251,12
2019-01-23,GIN,593,2
2019-01-23,XXX,777,2
2019-01-24,CZE,620,1
2019-01-24,XXX,1668,12
```

Good luck!