Python Coding Challenge – Transfer Grouping by Country and Period

Problem Statement

You are provided with a list of applicants, where each applicant has a history of transfer transactions. Each transaction includes the following information:

- `country`: The country the transfer was made from (e.g., `"USA"`, `"GE"`)
- `period`: A numerical value indicating the time period (e.g., `1`, `2`, `3`)
- `amountgel`: Amount of the transfer in GEL (Georgian Lari)
- `source`: Source label for the transfer (e.g., `"A"`, `"B"`, `"M"`)

Task

Write a function `process_applicant_transfers(applicants)` that processes the list of applicants and performs the following for each applicant:

1. Group the transactions by (`country`, `period`) pair
2. For each (`country`, `period`) group:
   - Compute the total `amountgel` (sum of all amounts)
   - Collect all unique `source` values and join them alphabetically using `/` (e.g., `"A/B"`)
3. For each applicant, return:
   - The `applicant_id`
   - A list of grouped transfer records, each with:

     python

     ```
     {
         "country": ...,    # str
         "period": ...,     # int
         "amountgel": ...,  # float (sum)
         "source": ...      # str (joined sources)
     }
     ```

4. The grouped transfer records for each applicant must be sorted by `country` (alphabetically) and then by `period` (ascending).

Input Format

python

```python
applicants = [
    {
        "applicant_id": "APP_001",
        "transfers": [
            {"country": "USA", "period": 1, "amountgel": 100.0, "source": "A"},
            {"country": "USA", "period": 1, "amountgel": 50.0, "source": "B"},
            {"country": "GE", "period": 2, "amountgel": 200.0, "source": "M"},
            {"country": "USA", "period": 2, "amountgel": 75.0, "source": "A"},
            {"country": "GE", "period": 1, "amountgel": 120.0, "source": "B"},
        ]
    },
    {
        "applicant_id": "APP_002",
        "transfers": [
            {"country": "UK", "period": 1, "amountgel": 300.0, "source": "C"},
            {"country": "UK", "period": 1, "amountgel": 100.0, "source": "A"},
        ]
    }
]
```

Expected Output

python

```python
[
    {
        "applicant_id": "APP_001",
        "grouped_transfers": [
            {"country": "GE", "period": 1, "amountgel": 120.0, "source": "B"},
            {"country": "GE", "period": 2, "amountgel": 200.0, "source": "M"},
            {"country": "USA", "period": 1, "amountgel": 150.0, "source": "A/B"},
            {"country": "USA", "period": 2, "amountgel": 75.0, "source": "A"}
        ]
    },
    {
        "applicant_id": "APP_002",
        "grouped_transfers": [
            {"country": "UK", "period": 1, "amountgel": 400.0, "source": "A/C"}
        ]
    }
]
```

Requirements

1. Function signature: `def process_applicant_transfers(applicants) -> list:`
2. Handle edge cases (empty transfers, missing fields, etc.)
3. Consider efficiency for large datasets
4. Write clean, readable code