In [1]:
```
import json
```

In [2]:
```
with open('data/cards.json', 'r') as file:
    cards = json.load(file)

unique_cards = {}
for card in cards:
    if (card['al'] is None or card['al'] == '') and card['cid'] not in unique_ca
        unique_cards[card['cid']] = card

filtered_cards = list(unique_cards.values())
```

In [3]:
```
def cleanSpecialCharacters(text):
    text = text.replace('\u2022', '')
    text = text.replace('\u00e8', 'e')
    text = text.replace('\u2460', '(1 DON!!)')
    text = text.replace('\u2461', '(2 DON!!)')
    text = text.replace('\u2462', '(3 DON!!)')
    text = text.replace('\u2463', '(4 DON!!)')
    text = text.replace('\u2464', '(5 DON!!)')
    text = text.replace('\u2465', '(6 DON!!)')
    text = text.replace('\u2466', '(7 DON!!)')
    text = text.replace('\u2467', '(8 DON!!)')
    text = text.replace('\u2468', '(9 DON!!)')
    text = text.replace('\u2469', '(10 DON!!)')
    text = text.replace('\u2605', ' ')

    # Preserve special keywords by temporarily replacing them with placeholders
    tokens = {
        "<Rush>": "__TOKEN_RUSH__",
        "<Banish>": "__TOKEN_BANISH__",
        "<Double Attack>": "__TOKEN_DOUBLE_ATTACK__",
        "<Blocker>": "__TOKEN_BLOCKER__",
        "[Activate Main]": "__TOKEN_ACTIVATE_MAIN__",
        "[Opponents Turn]": "__TOKEN_OPPONENTS_TURN__",
        "[Counter]": "__TOKEN_COUNTER__",
        "[When Attacking]": "__TOKEN_WHEN_ATTACKING__",
        "[On Your Opponents Attack]": "__TOKEN_ON_YOUR_OPPONENTS_ATTACK__",
        "[Once per Turn]": "__TOKEN_ONCE_PER_TURN__",
        "[On KO]": "__TOKEN_ON_KO__",
        "[On Play]": "__TOKEN_ON_PLAY__",
        "[Trigger]": "__TOKEN_TRIGGER__",
        "[End of Your Turn]": "__TOKEN_END_OF_YOUR_TURN__",
        "[On Block]": "__TOKEN_ON_BLOCK__"
    }
    for token, placeholder in tokens.items():
        text = text.replace(token, placeholder)

    # Remove unwanted special characters
    text = text.replace('{', '')
    text = text.replace('}', '')
    text = text.replace('[', '')
    text = text.replace(']', '')
    text = text.replace('\"', '')

    # Restore preserved keywords in lower case
    for token, placeholder in tokens.items():
        text = text.replace(placeholder, token)
```

```
    return text

def cleanEffect(effect):
    effect = effect.strip()

    effect = effect.replace('>(', '> (')
    effect = effect.replace('K.O.', 'KO')
    effect = effect.replace('K.O', 'KO')
    effect = effect.replace('\'', '')
    effect = effect.replace('’', '')
    effect = effect.replace('“', '')
    effect = effect.replace('”', '')
    effect = effect.replace('—', '-')
    effect = effect.replace('–', '-')
    effect = effect.replace('\{\{', '{')
    effect = effect.replace('\}\}', '}')

    # remove reminder texts
    effect = effect.replace('(After your opponent declares an attack, you may re
    effect = effect.replace('(When your opponent attacks, by resting this card,
    effect = effect.replace('(During your opponents attack, you may rest this ca
    effect = effect.replace('(When your opponent attacks, by resting this card y
    effect = effect.replace('(When your opponent attacks, you may rest this card
    effect = effect.replace('(During your opponents attack, you may rest this ca
    effect = effect.replace('(This Character can attack the turn it enters play.
    effect = effect.replace('(This Character can attack the turn it enters play.
    effect = effect.replace('(This character can attack the turn it comes into p
    effect = effect.replace('(This card can attack on the turn in which it is pl
    effect = effect.replace('(When this card deals damage, the target card is tr
    effect = effect.replace('(When this card deals damage, the life card is tras
    effect = effect.replace('(When this character deals damage, any Trigger effe
    effect = effect.replace('(This card deals 2 damage.)', '')
    effect = effect.replace('(This card deals 2 damage)', '')
    effect = effect.replace('(The damage this character deals to your opponents
    effect = effect.replace('(Return the specified amount of DON!! from your fie
    effect = effect.replace('(Return 1 of your DON!! cards to your DON!! deck.)'
    effect = effect.replace('(You may return the specified number of DON!! cards
    effect = effect.replace('(You may return the specified number of DON!! cards
    effect = effect.replace('(Rest the indicated number of Don!! in your Cost Ar
    effect = effect.replace('(Rest the designated number of DON!! in your Cost A
    effect = effect.replace('(Rest the designated amount of DON!! cards in your
    effect = effect.replace('(You may rest the designated number for DON!! in yo
    effect = effect.replace('(You may rest the specified number of DON!! cards i
    effect = effect.replace('(You may rest the specified number of DON!! cards i
    effect = effect.replace('(You may rest the specified number of DON!! cards i

    # inconsistent keyword
    effect = effect.replace('[Activate: Main]', '[Activate Main]')
    effect = effect.replace('[Activate:Main]', '[Activate Main]')
    effect = effect.replace('[Main]', '[Activate Main]')
    effect = effect.replace('(Once per Turn)', '[Once per Turn]')
    effect = effect.replace('(Once per turn)', '[Once per Turn]')
    effect = effect.replace('(When Attacking)', '[When Attacking]')
    effect = effect.replace('(When Attacking)', '[When Attacking]')
    effect = effect.replace('[On Opponents Attack]', '[On Your Opponents Attack]
    effect = effect.replace('[On Opponent Attacks]', '[On Your Opponents Attack]
    effect = effect.replace('[End of your turn]', '[End of Your Turn]')

    # make rest DON!! cards consistent
```

```
    effect = effect.replace('(1)', '(1 DON!!)')
    effect = effect.replace('(2)', '(2 DON!!)')
    effect = effect.replace('(3)', '(3 DON!!)')
    effect = effect.replace('(4)', '(4 DON!!)')
    effect = effect.replace('(5)', '(5 DON!!)')
    effect = effect.replace('(6)', '(6 DON!!)')
    effect = effect.replace('(7)', '(7 DON!!)')
    effect = effect.replace('(8)', '(8 DON!!)')
    effect = effect.replace('(9)', '(9 DON!!)')
    effect = effect.replace('(10)', '(10 DON!!)')

    # remove extra spaces and dot spaces
    effect = effect.replace('  ', ' ')
    effect = effect.replace(' .', '.')
    effect = effect.replace(' :', ':')
    effect = effect.replace(' ,', ',')

    effect = cleanSpecialCharacters(effect)

    return effect

def cleanName(name):
    name = name.strip()
    name = cleanSpecialCharacters(name)

    return name

def cleanTraits(traits):
    traitsArray = traits.split('/')
    cleanedTraits = []
    for trait in traitsArray:
        trait = trait.strip()
        trait = cleanSpecialCharacters(trait)

        # has to be completely identical to the trait
        if trait == "Straw Hat Pirates":
            trait = "Straw Hat Crew"
        elif trait == "Alabasta Kingdom":
            trait = "Alabasta"
        elif trait == "Arlong Crew":
            trait = "Arlong Pirates"
        elif trait == "Big Mom P":
            trait = "Big Mom Pirates"
        elif trait == "Fish Man":
            trait = "Fish-Man"
        elif trait == "Galley-la Company":
            trait = "Galley-La Company"
        elif trait == "Giants":
            trait = "Giant"
        elif trait == "Gran Tesoro":
            trait = "Grantesoro"
        elif trait == "Kingdom of Germa" or trait == "Kingdom of GERMA":
            trait = "GERMA 66"
        elif trait == "Kuja":
            trait = "Kuja Pirates"
        elif trait == "Mink Tribe":
            trait = "Minks"
        elif trait == "Mountain Bandit":
            trait = "Mountain Bandits"
        elif trait == "New Giant Warrior Pirates":
```

```
            trait = "New Giant Pirates"
        elif trait == "Red Haired Pirates" or trait == "Red Hair Pirates":
            trait = "Red-Haired Pirates"
        elif trait == "Supernova":
            trait = "Supernovas"
        elif trait == "The Akazaya Nine":
            trait = "Nine Red Scabbards"
        elif trait == "The Seven Warlords" or trait == "Seven Warlords of the Se
            trait = "The Seven Warlords of the Sea"
        elif trait == "The Sun Pir" or trait == "Sun Pirates":
            trait = "The Sun Pirates"
        elif trait == "Thriller Bark Pira" or trait == "Thriller Bark Pirate":
            trait = "Thriller Bark Pirates"
        elif trait == "The Tontattas":
            trait = "Tontatta"
        elif trait == "Water Seven":
            trait = "Water 7"

        cleanedTraits.append(trait)

    return cleanedTraits

def cleanColor(color):
    color_map = {
        '1': ['Red'],
        '4': ['Purple'],
        '5': ['DON'],
        '6': ['Blue'],
        '7': ['Green'],
        '8': ['Blue', 'Green'],
        '9': ['Red', 'Blue'],
        '10': ['Red', 'Green'],
        '11': ['Blue', 'Purple'],
        '12': ['Black'],
        '13': ['Purple', 'Black'],
        '14': ['Red', 'Black'],
        '15': ['Green', 'Blue'],
        '16': ['Yellow'],
        '17': ['Green', 'Yellow'],
        '18': ['Black', 'Yellow'],
        '19': ['Black', 'Blue'],
        '20': ['Black', 'Green'],
        '21': ['Purple', 'Yellow'],
        '22': ['Red', 'Yellow'],
        '23': ['Blue', 'Yellow'],
        '24': ['Red', 'Purple'],
        '25': ['Purple', 'Green']
    }

    colors = color_map.get(str(color), [])

    return colors

def cleanType(card_type):
    type_map = {
        '1': 'Leader',
        '2': 'Character',
        '3': 'Event',
        '4': 'Stage'
    }
```

```
    return type_map.get(card_type, [])

def cleanAttribute(attribute):
    attribute_map = {
        '1': ['Slash'],
        '2': ['Strike'],
        '3': ['Ranged'],
        '4': ['Wisdom'],
        '5': ['Special'],
        '6': ['Slash', 'Strike'],
        '7': ['Slash', 'Special'],
        '8': ['Strike', 'Ranged'],
        '9': ['Strike', 'Special']
    }

    return attribute_map.get(attribute, [])
```

In [4]:

```
cleaned_cards = []
for card in filtered_cards:
    cleaned_card = {
        "id": card['cid'],
        'name': card['n'],
        'type': card['t'],
        'cost': card['cs'],
        'power': card.get('p', None),
        'effect': card.get('e', None),
        'traits': card['tr'],
        'color': card['col'],
        'attribute': card['a'],
        'rarity': card['r'],
        'life': card.get('l', None),
        'counter': card.get('cp', None),
    }

    if cleaned_card['effect']:
        cleaned_card['effect'] = cleanEffect(cleaned_card['effect'])

    if cleaned_card['name']:
        cleaned_card['name'] = cleanName(cleaned_card['name'])

    if cleaned_card['traits']:
        cleaned_card['traits'] = cleanTraits(cleaned_card['traits'])

    if cleaned_card['color']:
        cleaned_card['color'] = cleanColor(cleaned_card['color'])

    if cleaned_card['type']:
        cleaned_card['type'] = cleanType(cleaned_card['type'])

    if cleaned_card['attribute']:
        cleaned_card['attribute'] = cleanAttribute(cleaned_card['attribute'])

    cleaned_cards.append(cleaned_card)

with open('data/cleaned_cards.json', 'w') as file:
    json.dump(cleaned_cards, file, indent=2)
```

In [5]:

```python
with open('data/cleaned_cards.json', 'r') as file:
    cleaned_cards = json.load(file)

# Replace null values in numerical fields with 0
for card in cleaned_cards:
    card['power'] = int(card['power']) if card['power'] is not None else 0
    card['life'] = int(card['life']) if card['life'] is not None else 0
    card['cost'] = int(card['cost']) if card['cost'] is not None else 0
    card['counter'] = int(card['counter']) if card['counter'] is not None else 0
    card['rarity'] = int(card['rarity']) if card['rarity'] is not None else 999

# Replace null values in categorical fields with empty strings
for card in cleaned_cards:
    card['effect'] = card['effect'] if card['effect'] is not None else ''
    card['traits'] = card['traits'] if card['traits'] is not None else ''

# Save the cleaned data to a new JSON file
with open('data/cleaned_cards_with_placeholders.json', 'w') as file:
    json.dump(cleaned_cards, file, indent=2)
```