In [85]:
```
import pandas as pd
import json
import re
```

In [86]:
```
with open('data/cleaned_cards_with_placeholders.json', 'r') as file:
    cards = json.load(file)

data = pd.DataFrame(cards)
```

In [87]:
```
import json

def assign_core_label(row, current_labels):
    labels = []
    counter = row['counter']

    defensive_labels = ['Blocker', 'Counter', 'Opponents Turn', 'Protection', 'S
    aggressive_labels = ['Rush', 'Double Attack', 'Banish', 'Summon', 'Don Ramp'
    support_labels = ['Draw', 'Searcher', 'Buff Power', 'Debuff Power', 'Cost Re

    # Determine the label that fits most criteria
    label_counts = {'Defensive': 0, 'Aggressive': 0, 'Support': 0}

    # Count occurrences of each label type in current_labels
    for label in current_labels:
        if label in defensive_labels:
            label_counts['Defensive'] += 1
        elif label in aggressive_labels:
            label_counts['Aggressive'] += 1
        elif label in support_labels:
            label_counts['Support'] += 1

    # Add the label with the highest count
    if label_counts['Defensive'] >= label_counts['Aggressive'] and label_counts[
        labels.append('Defensive')
    elif label_counts['Aggressive'] >= label_counts['Defensive'] and label_count
        labels.append('Aggressive')
    else:
        labels.append('Support')

    # 2K counter is always considered defensive
    if counter == 2000 and 'Defensive' not in labels:
        labels.append('Defensive')

    return labels

def assign_cost_label(cost):
    if cost <= 3:
        return ['Low Cost']
    elif 4 <= cost <= 6:
        return ['Mid Cost']
    else:
        return ['High Cost']

def assign_effect_key_label(effect):
    effect = effect.lower()
    labels = []
    effect_keywords = {
        '<rush>': 'Rush',
```

```
        '<banish>': 'Banish',
        '<double attack>': 'Double Attack',
        '<blocker>': 'Blocker',
        # '[activate main]': 'Main',
        # "[opponents turn]": "Opponents Turn",
        '[counter]': 'Counter',
        # '[when attacking]': 'When Attacking',
        # '[on your opponents attack]': 'On Opponents Attack',
        # '[once per turn]': 'Once Per Turn',
        # '[on ko]': 'On KO',
        # '[on play]': 'On Play',
        '[trigger]': 'Trigger',
        # '[end of your turn]': 'End of Your Turn',
        # '[on block]': 'On Block',
    }
    for keyword, label in effect_keywords.items():
        if keyword in effect:
            labels.append(label)
    return labels

def assign_utility_label(effect):
    if effect == '':
        return ['Vanilla']
    effect = effect.lower()
    labels = []
    utility_mapping = {
        "Removal": [r"ko .* opponents character", r"trash .* opponents character
        "Board Wipe": [r"ko all", r"trash all"],
        "Buff Power": [r"gain \+\s?\d+\s? power", r"gains \+\s?\d+\s? power"],
        "Debuff Power": [r"\-\s?\d+\s? power", r"lose \d+ power"],
        "Draw": [r"draw"],
        "Discard": [r"discard .* from your hand", r"trash .* card from your hand
        "Hand Destruction": [r"opponent trashes \d+ card"],
        "Mill": [r"trash the top \d+ cards of your deck", r"trash .* of your dec
        "Cost Reduction": [r"\-\d+\s?cost"],
        "Increase Cost": [r"\+\d+\s?cost"],
        "Gain Life": [r"put .* on top of your life area", r"add .* to the top of
        "Take Life": [r"trash.*from the top of your life"],
        "Searcher": [r"look at .* reveal .* add .* to your hand"],
        "Rearrange Deck": [r"look at.*of your deck.*rearrange them in any order"
        "Rearrange Life": [r"look at.*life area", r"look at.*life card"],
        "Protection": [r"would be kod.*instead", r"cant be kod", r"cannot be kod
        "Summon": [r"play "],
        "Don Ramp": [r"add up to \d+ don!! from your don!! deck", r"add \d+ don!
        "Don Minus": [r"don!!\s?-"],
        "Trash Interaction": [r"from your trash"],
        # "Leader Locked": [r"if your leader"],
        "Stun": [r"not become active"],
        "Restand Don": [r"of your don!! cards as active"],
        "Restand Character": [r"character as active", r"your rested characters .
        "Rest Character": [r"rest .* opponents character"],
    }

    for label, patterns in utility_mapping.items():
        for pattern in patterns:
            if re.search(pattern, effect) and label not in labels:
                labels.append(label)

    return labels
```

```
def classify_card(row):
    labels = []

    labels += assign_effect_key_label(row['effect'].lower())
    labels += assign_utility_label(row['effect'].lower())

    if row['type'] != 1:
        labels += assign_cost_label(row['cost'])
        # labels += assign_core_label(row, labels)

    return labels

def export_cards_by_label(cards, label):
    cards_by_label = []
    for card in cards:
        if label in card['labels']:
            cards_by_label.append(card)
    with open(f'data/spec/{label}.json', 'w') as file:
        json.dump(cards_by_label, file, indent=2)
```

In [88]:
```
for card in cards:
    card['labels'] = classify_card(card)
```

In [89]:
```
aggro = "Aggro"
control = "Control"
midrange = "Midrange"
combo = "Combo"

archetype_map = {
    'OP01-003': midrange,
    'OP01-001': aggro,
    'OP01-061': control,
    'OP01-002': combo,
    'OP01-031': midrange,
    'OP01-060': aggro,
    'OP01-062': combo,
    "OP01-091": control,
    'ST01-001': midrange,
    'ST02-001': midrange,
    'ST03-001': control,
    'ST04-001': control,
    'ST05-001': midrange,
    'OP02-049': combo,
    'OP02-071': midrange,
    'OP02-093': control,
    'OP02-025': aggro,
    'OP02-001': midrange,
    'OP02-072': control,
    'OP02-002': control,
    'OP02-026': midrange,
    'P-011': aggro,
    'P-047': midrange,
    'P-076': control,
    'ST06-001': control,
    'ST07-001': combo,
    'OP03-099': aggro,
    'OP03-076': control,
    'OP03-022': aggro,
    'OP03-001': combo,
```

```
    'OP03-021': combo,
    'OP03-040': combo,
    'OP03-058': midrange,
    'OP03-077': control,
    'ST08-001': control,
    'ST09-001': midrange,
    'OP04-058': combo,
    'OP04-001': aggro,
    'OP04-039': control,
    'OP04-019': control,
    'OP04-020': control,
    'OP04-040': combo,
    'ST10-001': combo,
    'ST10-002': aggro,
    'ST10-003': combo,
    'OP05-001': control,
    'OP05-002': aggro,
    'OP05-022': combo,
    'OP05-041': control,
    'OP05-060': midrange,
    'OP05-098': combo,
    'ST11-001': midrange,
    'ST12-001': aggro,
    'OP06-001': combo,
    'OP06-080': control,
    'OP06-020': combo,
    'OP06-021': control,
    'OP06-022': aggro,
    'OP06-042': aggro,
    'EB01-001': combo,
    'EB01-021': combo,
    'EB01-040': control,
    'OP07-001': aggro,
    'OP07-019': control,
    'OP07-038': control,
    'OP07-059': combo,
    'OP07-079': control,
    'OP07-097': combo,
    'ST13-001': aggro,
    'ST13-002': combo,
    'ST13-003': combo,
    'ST14-001': control,
    'OP08-001': aggro,
    'OP08-002': combo,
    'OP08-021': combo,
    'OP08-057': control,
    'OP08-058': combo,
    'OP08-098': midrange
}

def add_archteype(cards):
    for card in cards:
        if card.get('type') == 'Leader':
            card['archetype'] = archetype_map.get(card.get('id'), '')
        else:
            card['archetype'] = ''
    return cards

cards = add_archteype(cards)
```

In [90]:
```
#store labeled data
with open('data/labeled_cards.json', 'w') as file:
    json.dump(cards, file, indent=2)
```

In [91]:
```
export_cards_by_label(cards, 'Searcher')
```

In [92]:
```
export_cards_by_label(cards, 'Leader Locked')
```

In [93]:
```
# export leader cards
leader_cards = []
for card in cards:
    if card['type'] == 'Leader':
        leader_cards.append(card)
with open('data/leader_cards.json', 'w') as file:
    json.dump(leader_cards, file, indent=2)
```