Process Mining: Process Discovery

Dependencies

In [2]:
```
import pm4py
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from IPython.display import display
```

Event Log Import

In [ ]:
```
file_path = r""
complete_log_df = pm4py.read_xes(file_path)
print("Log loaded. Type:", type(complete_log_df))
```

```
c:\Users\compt\AppData\Local\Programs\Python\Python310\lib\site-packages\pm4py\util\dt_parsing\parser.py:82: UserWarning: ISO8601 strings are not fully supported with strpfromiso for Python versions below 3.11
  warnings.warn(
parsing log, completed traces ::   0%|          | 0/6449 [00:00<?, ?it/s]
Log loaded. Type: <class 'pandas.core.frame.DataFrame'>
```

Variants: Complete Log

In [4]:
```
def get_n_print_variants(event_log, log_name):
    variant_dictonary = pm4py.get_variants(
        event_log,
        activity_key='concept:name',
        case_id_key='case:concept:name',
        timestamp_key='time:timestamp'
    )

    print(f"The {sum(variant_dictonary.values())} traces in the {log_name} have {len(variant_dic

    return variant_dictonary
```

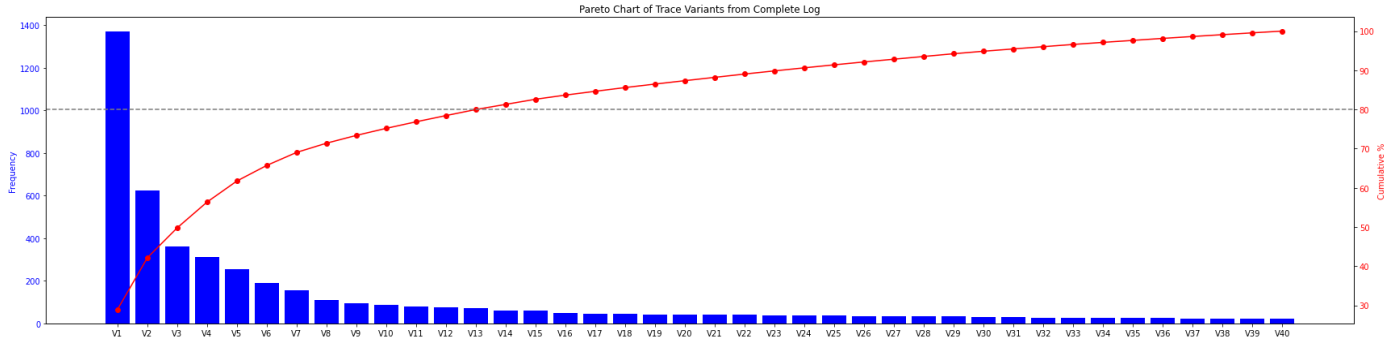
In [5]:
```
variants_complete_log = get_n_print_variants(complete_log_df, log_name="Complete Log")
```

```
The 6449 traces in the Complete Log have 753 variants
```

In [6]:
```
def top_N_variants_Pareto_chart(variants, Top_N, log_name):

    variant_stats = sorted(variants.items(), key=lambda x: x[1], reverse=True)

    top_variant_stats = variant_stats[:Top_N]

    x = [f"V{i+1}" for i in range(len(top_variant_stats))]
    y = [v[1] for v in top_variant_stats]

    cumulative = np.cumsum(y)
    cumulative_percent = cumulative / cumulative[-1] * 100

    fig, ax1 = plt.subplots(figsize=(24, 6))

    ax1.bar(x, y, color='blue')
```

```
    ax1.set_ylabel('Frequency', color='blue')
    ax1.tick_params(axis='y', labelcolor='blue')

    ax2 = ax1.twinx()
    ax2.plot(x, cumulative_percent, color='red', marker='o')
    ax2.set_ylabel('Cumulative %', color='red')
    ax2.tick_params(axis='y', labelcolor='red')
    ax2.axhline(80, color='gray', linestyle='--')

    plt.title(f"Pareto Chart of Trace Variants from {log_name}")
    plt.xticks(rotation=90)
    plt.tight_layout()
    plt.show()
```

In [7]:
```
top_N_variants_Pareto_chart(variants_complete_log, Top_N=40, log_name="Complete Log")
```



In [8]:
```
def display_variants_head_or_tail(variants, head_or_tail, Top_N):

    total_cases = sum(variants.values())

    data = []
    for i, (variant_str, count) in enumerate(sorted(variants.items(), key=lambda x: x[1], revers
        variant_id = f"V{i+1}"
        percent = (count / total_cases) * 100
        data.append({
            "variant_id": variant_id,
            "variant_str": variant_str,
            "count": count,
            "percent": round(percent, 2)
        })

    variant_df = pd.DataFrame(data)

    if head_or_tail == "head":
        sorted_df = variant_df.sort_values(by='count', ascending=False).head(Top_N)
    elif head_or_tail == "tail":
        sorted_df = variant_df.sort_values(by='count', ascending=True).head(Top_N)

    styled_df = sorted_df.style.set_properties(**{
        'text-align': 'center'
        }).set_table_styles([
            {'selector': 'th', 'props': [('text-align', 'center')]},
            {'selector': 'td:nth-child(1)', 'props': [('text-align', 'Left')]}
        ]).hide(axis="index")

    display(styled_df)
```

Top 5 Variants: Complete Log

In [9]:
```
display_variants_head_or_tail(variants_complete_log, head_or_tail="head", Top_N=5)
```

variant_id	variant_str	count	percent
V1	('Permit SUBMITTED by EMPLOYEE', 'Permit APPROVED by ADMINISTRATION', 'Permit FINAL_APPROVED by SUPERVISOR', 'Start trip', 'End trip', 'Declaration SUBMITTED by EMPLOYEE', 'Declaration APPROVED by ADMINISTRATION', 'Declaration FINAL_APPROVED by SUPERVISOR', 'Request Payment', 'Payment Handled')	1369	21.230000
V2	('Permit SUBMITTED by EMPLOYEE', 'Permit APPROVED by ADMINISTRATION', 'Permit APPROVED by BUDGET OWNER', 'Permit FINAL_APPROVED by SUPERVISOR', 'Start trip', 'End trip', 'Declaration SUBMITTED by EMPLOYEE', 'Declaration APPROVED by ADMINISTRATION', 'Declaration APPROVED by BUDGET OWNER', 'Declaration FINAL_APPROVED by SUPERVISOR', 'Request Payment', 'Payment Handled')	624	9.680000
V3	('Permit SUBMITTED by EMPLOYEE', 'Permit FINAL_APPROVED by SUPERVISOR', 'Start trip', 'End trip', 'Declaration SUBMITTED by EMPLOYEE', 'Declaration FINAL_APPROVED by SUPERVISOR', 'Request Payment', 'Payment Handled')	361	5.600000
V4	('Permit SUBMITTED by EMPLOYEE', 'Permit APPROVED by ADMINISTRATION', 'Permit FINAL_APPROVED by SUPERVISOR', 'Start trip', 'End trip', 'Declaration SUBMITTED by EMPLOYEE', 'Declaration REJECTED by ADMINISTRATION', 'Declaration REJECTED by EMPLOYEE', 'Declaration SUBMITTED by EMPLOYEE', 'Declaration APPROVED by ADMINISTRATION', 'Declaration FINAL_APPROVED by SUPERVISOR', 'Request Payment', 'Payment Handled')	311	4.820000
V5	('Permit SUBMITTED by EMPLOYEE', 'Permit APPROVED by PRE_APPROVER', 'Permit FINAL_APPROVED by SUPERVISOR', 'Start trip', 'End trip', 'Declaration SUBMITTED by EMPLOYEE', 'Declaration APPROVED by PRE_APPROVER', 'Declaration FINAL_APPROVED by SUPERVISOR', 'Request Payment', 'Payment Handled')	254	3.940000

Bottom 5 Variants: Complete Log

In [10]:
```
display_variants_head_or_tail(variants_complete_log, head_or_tail="tail", Top_N=5)
```

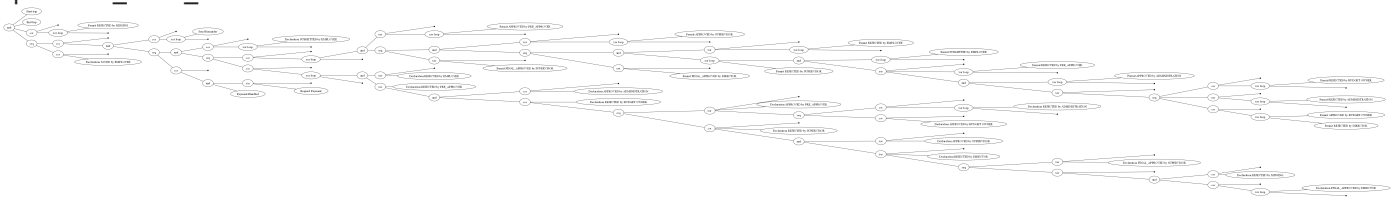
variant_id	variant_str	count	percent
V738	('Permit SUBMITTED by EMPLOYEE', 'Permit APPROVED by SUPERVISOR', 'Permit FINAL_APPROVED by DIRECTOR', 'Permit REJECTED by MISSING', 'Permit SUBMITTED by EMPLOYEE', 'Permit FINAL_APPROVED by SUPERVISOR', 'Start trip', 'End trip', 'Declaration SUBMITTED by EMPLOYEE', 'Declaration FINAL_APPROVED by SUPERVISOR', 'Request Payment', 'Payment Handled')	1	0.020000
V539	('Permit SUBMITTED by EMPLOYEE', 'Permit APPROVED by ADMINISTRATION', 'Permit FINAL_APPROVED by SUPERVISOR', 'Start trip', 'End trip', 'Send Reminder', 'Declaration SUBMITTED by EMPLOYEE', 'Declaration REJECTED by ADMINISTRATION', 'Declaration SUBMITTED by EMPLOYEE', 'Declaration REJECTED by EMPLOYEE', 'Declaration SUBMITTED by EMPLOYEE', 'Declaration APPROVED by ADMINISTRATION', 'Declaration FINAL_APPROVED by SUPERVISOR', 'Request Payment', 'Payment Handled')	1	0.020000
V538	('Start trip', 'End trip', 'Permit SUBMITTED by EMPLOYEE', 'Permit APPROVED by ADMINISTRATION', 'Permit APPROVED by BUDGET OWNER', 'Permit REJECTED by SUPERVISOR', 'Permit REJECTED by EMPLOYEE', 'Permit SUBMITTED by EMPLOYEE', 'Permit APPROVED by ADMINISTRATION', 'Permit APPROVED by BUDGET OWNER', 'Permit FINAL_APPROVED by SUPERVISOR', 'Send Reminder', 'Declaration SUBMITTED by EMPLOYEE', 'Declaration APPROVED by ADMINISTRATION', 'Declaration APPROVED by BUDGET OWNER', 'Declaration FINAL_APPROVED by SUPERVISOR', 'Request Payment', 'Payment Handled')	1	0.020000
V537	('Permit SUBMITTED by EMPLOYEE', 'Permit APPROVED by ADMINISTRATION', 'Permit APPROVED by BUDGET OWNER', 'Permit APPROVED by SUPERVISOR', 'Permit FINAL_APPROVED by DIRECTOR', 'Start trip', 'Declaration SUBMITTED by EMPLOYEE', 'End trip', 'Declaration REJECTED by ADMINISTRATION', 'Declaration REJECTED by EMPLOYEE', 'Declaration SUBMITTED by EMPLOYEE', 'Declaration APPROVED by ADMINISTRATION', 'Declaration APPROVED by BUDGET OWNER', 'Declaration APPROVED by SUPERVISOR', 'Declaration FINAL_APPROVED by DIRECTOR', 'Request Payment', 'Payment Handled')	1	0.020000
V536	('Permit SUBMITTED by EMPLOYEE', 'Permit APPROVED by ADMINISTRATION', 'Permit FINAL_APPROVED by SUPERVISOR', 'Declaration SUBMITTED by EMPLOYEE', 'Declaration APPROVED by ADMINISTRATION', 'Declaration FINAL_APPROVED by SUPERVISOR', 'Request Payment', 'Start trip', 'Payment Handled', 'End trip')	1	0.020000

Process Tree Discovery: Inductive Miner

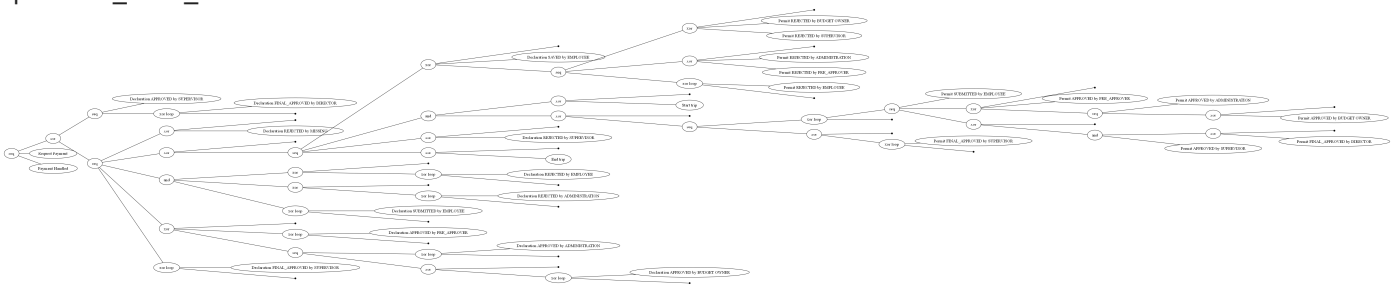
In [11]:
```python
def discover_multiple_process_trees(event_log, thresholds, show=False):

    process_trees = []

    for threshold in thresholds:

        process_tree = pm4py.discover_process_tree_inductive(
            event_log,
            activity_key='concept:name',
            case_id_key='case:concept:name',
            timestamp_key='time:timestamp',
            noise_threshold= threshold
        )

        if show:
            print(f"process_tree_{threshold}")
            pm4py.view_process_tree(process_tree, format='png')

        process_trees.append((f"process_tree_{threshold}", process_tree))

    return process_trees
```

In [12]:
```python
complete_process_trees = discover_multiple_process_trees(complete_log_df, thresholds=[0.0, 0.1,
```

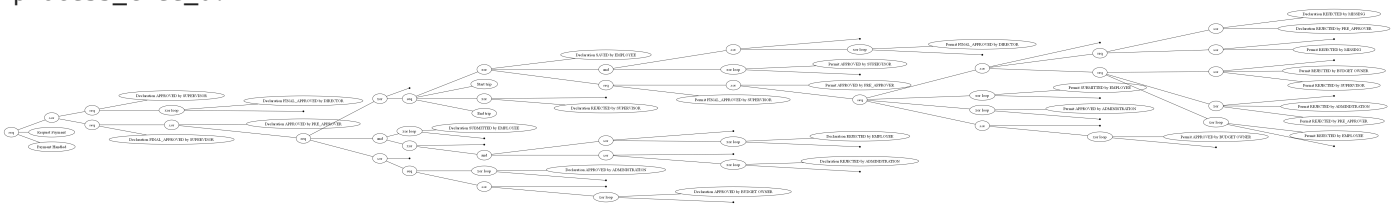
process_tree_0.0



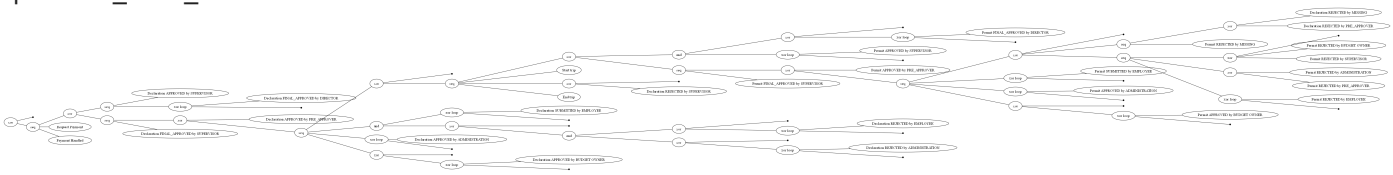
process_tree_0.1



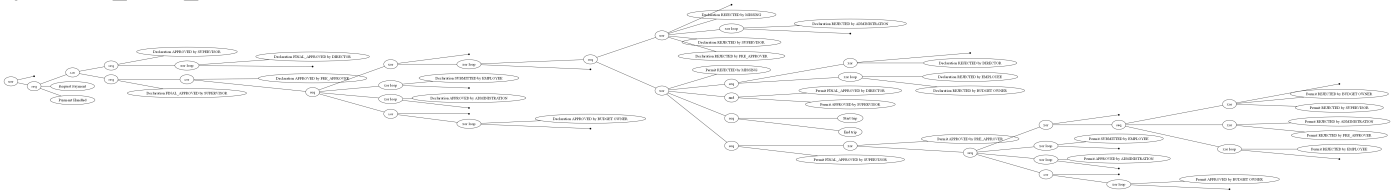
process_tree_0.2



process_tree_0.3



process_tree_0.4

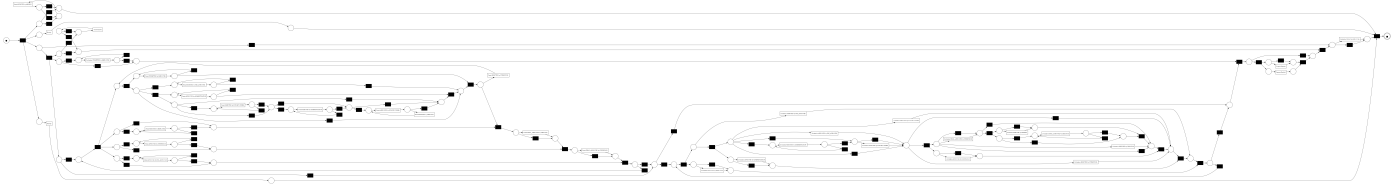


Process Tree Convertion to Petri Net

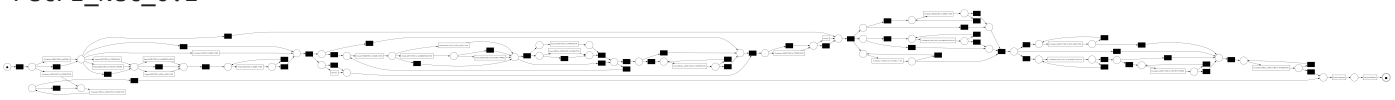
In [13]:
```
def convert_multiple_Process_Trees_to_Petri_Nets(process_trees, show=False):

    petri_nets = []

    for name, process_tree in process_trees:
        net, initial_marking, final_marking = pm4py.convert_to_petri_net(process_tree)
        petri_nets.append((f"Petri_Net_{name[-3:]}", net, initial_marking, final_marking))

        if show:
            print(f"Petri_Net_{name[-3:]}")
            pm4py.view_petri_net(net, initial_marking, final_marking, format='png')

    return petri_nets
```

In [14]:
```
complete_Petri_Nets = convert_multiple_Process_Trees_to_Petri_Nets(complete_process_trees, show=
```

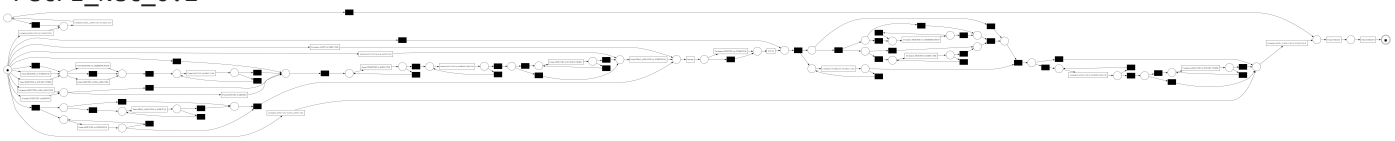
Petri_Net_0.0



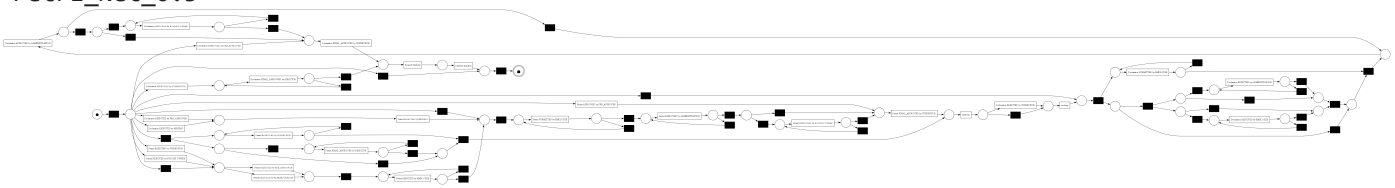
Petri_Net_0.1



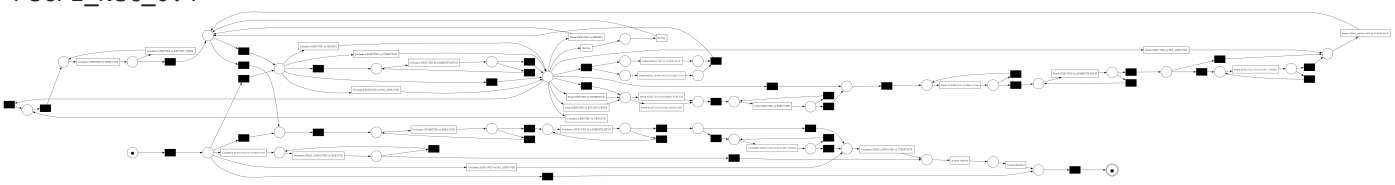
Petri_Net_0.2



Petri_Net_0.3



Petri_Net_0.4



Petri Net Evaluation

In [ ]:

```
class PetriNetEvaluator:
    def __init__(self, event_log, net, im, fm):
        self.event_log = event_log
        self.net = net
        self.im = im
        self.fm = fm

    def evaluate_fitness(self):
        fitness = pm4py.fitness_token_based_replay(
            self.event_log,
            self.net,
            self.im,
            self.fm,
            activity_key='concept:name',
            case_id_key='case:concept:name',
            timestamp_key='time:timestamp'
        )
        return fitness

    def evaluate_precision(self):
        precision = pm4py.precision_token_based_replay(
            self.event_log,
            self.net,
            self.im,
            self.fm,
            activity_key='concept:name',
            case_id_key='case:concept:name',
            timestamp_key='time:timestamp'
        )
        return precision

    def evaluate_generalization(self):
```

```python
        from pm4py.algo.evaluation.generalization import algorithm as generalization_factory
        generalization = generalization_factory.apply(self.event_log, self.net, self.im, self.fm
        return generalization

    def evaluate_simplicity(self):
        from pm4py.algo.evaluation.simplicity import algorithm as simplicity_factory
        simplicity = simplicity_factory.apply(self.net)
        return simplicity

    def evaluate_all(self):
        full_fitness = self.evaluate_fitness()
        fitness_avg = full_fitness.get("average_trace_fitness")
        fitness_perc = full_fitness.get("perc_fit_traces")
        precision = self.evaluate_precision()
        generalization = self.evaluate_generalization()
        simplicity = self.evaluate_simplicity()
        f1_score = 2 * (precision * fitness_avg) / (precision + fitness_avg)

        results = {
            "fitness_avg": fitness_avg,
            "fitness_perc_fit_traces": fitness_perc,
            "precision": precision,
            "generalization": generalization,
            "simplicity": simplicity,
            "f1_score": f1_score
        }
        return results
```

In [ ]:

```python
results = []

for name, net, im, fm in complete_Petri_Nets:
    evaluator = PetriNetEvaluator(complete_log_df, net, im, fm)
    metrics = evaluator.evaluate_all()
    metrics["model_name"] = name
    results.append(metrics)

df = pd.DataFrame(results)

cols = ["model_name", "fitness_avg", "fitness_perc_fit_traces", "precision", "generalization", "
df = df[cols]

styled_df = df.style.set_properties(**{
    'text-align': 'center'
}).set_table_styles([
    {'selector': 'th', 'props': [('text-align', 'center')]},
    {'selector': 'td:nth-child(1)', 'props': [('text-align', 'Left')]}
]).hide(axis="index")

display(styled_df)
```

```
replaying log with TBR, completed traces ::   0%|          | 0/753 [00:00<?, ?it/s]
replaying log with TBR, completed traces ::   0%|          | 0/4077 [00:00<?, ?it/s]
replaying log with TBR, completed traces ::   0%|          | 0/753 [00:00<?, ?it/s]
replaying log with TBR, completed traces ::   0%|          | 0/753 [00:00<?, ?it/s]
replaying log with TBR, completed traces ::   0%|          | 0/4077 [00:00<?, ?it/s]
replaying log with TBR, completed traces ::   0%|          | 0/753 [00:00<?, ?it/s]
replaying log with TBR, completed traces ::   0%|          | 0/753 [00:00<?, ?it/s]
replaying log with TBR, completed traces ::   0%|          | 0/4077 [00:00<?, ?it/s]
replaying log with TBR, completed traces ::   0%|          | 0/753 [00:00<?, ?it/s]
replaying log with TBR, completed traces ::   0%|          | 0/753 [00:00<?, ?it/s]
replaying log with TBR, completed traces ::   0%|          | 0/4077 [00:00<?, ?it/s]
replaying log with TBR, completed traces ::   0%|          | 0/753 [00:00<?, ?it/s]
replaying log with TBR, completed traces ::   0%|          | 0/753 [00:00<?, ?it/s]
replaying log with TBR, completed traces ::   0%|          | 0/4077 [00:00<?, ?it/s]
```

```
replaying log with TBR, completed traces ::   0%|          | 0/753 [00:00<?, ?it/s]
```

model_name	fitness_avg	fitness_perc_fit_traces	precision	generalization	simplicity	f1_score
Petri_Net_0.0	0.971365	6.946813	0.225696	0.876672	0.622222	0.366286
Petri_Net_0.1	0.980931	66.072259	0.356559	0.911931	0.641509	0.523010
Petri_Net_0.2	0.941366	45.154287	0.338725	0.860729	0.640000	0.498189
Petri_Net_0.3	0.934303	45.154287	0.326071	0.858766	0.658031	0.483426
Petri_Net_0.4	0.954289	38.920763	0.216777	0.864635	0.649718	0.353299

Choice of Normative Model

Petri_Net_0.1, Petri_Net_0.2 and Petri_Net_0.3 have similar F1 score, however the latter has higher simplicity. To decide which Petri Net to choose as our normative model, let's look at their graphs.

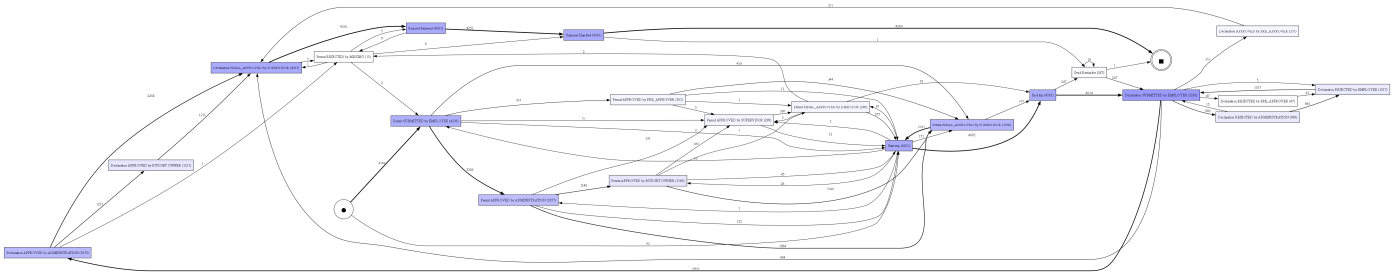
Direct Follow Graphs

In [ ]:
```python
def extract_confirming_log_from_Petri_Net(event_log, net, im, fm):

    from pm4py.objects.conversion.log import converter as log_converter
    from pm4py.algo.conformance.tokenreplay import algorithm as token_replay

    event_log_list = log_converter.apply(event_log, parameters={
        log_converter.Variants.TO_EVENT_LOG.value.Parameters.CASE_ID_KEY: "case:concept:name"
    })

    replay_results = token_replay.apply(event_log_list, net, im, fm)

    conforming_trace_indices = [
        idx for idx, result in enumerate(replay_results) if  result["trace_is_fit"]
    ]

    print("Number of conforming traces:", len(conforming_trace_indices))

    conforming_case_ids = [
        trace.attributes["concept:name"]
        for idx, trace in enumerate(event_log_list)
        if replay_results[idx]["trace_is_fit"]
    ]

    df_conforming = event_log[event_log["case:concept:name"].isin(conforming_case_ids)].copy()

    return df_conforming
```

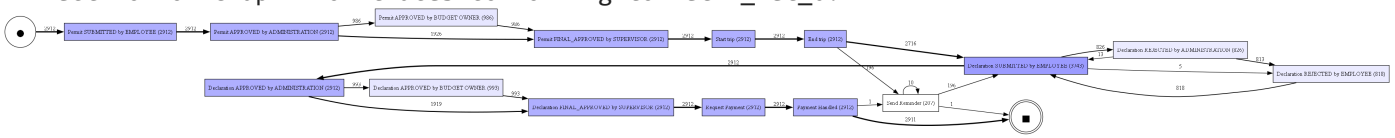
In [24]:
```python
conforming_log_01_df = extract_confirming_log_from_Petri_Net(complete_log_df, complete_Petri_Net
dfg_01, start_activities_01, end_activities_01 = pm4py.discover_dfg(conforming_log_01_df, case_i

conforming_log_02_df = extract_confirming_log_from_Petri_Net(complete_log_df, complete_Petri_Net
dfg_02, start_activities_02, end_activities_02 = pm4py.discover_dfg(conforming_log_02_df, case_i

conforming_log_03_df = extract_confirming_log_from_Petri_Net(complete_log_df, complete_Petri_Net
dfg_03, start_activities_03, end_activities_03 = pm4py.discover_dfg(conforming_log_03_df, case_i

print("Direct Follow Graph from traces conforming to Petri_Net_0.1")
pm4py.view_dfg(dfg_01, start_activities_01, end_activities_01, format='png')
print("Direct Follow Graph from traces conforming to Petri_Net_0.2")
pm4py.view_dfg(dfg_02, start_activities_02, end_activities_02, format='png')
```

```
print("Direct Follow Graph from traces conforming to Petri_Net_0.3")
pm4py.view_dfg(dfg_03, start_activities_03, end_activities_03, format='png')
```

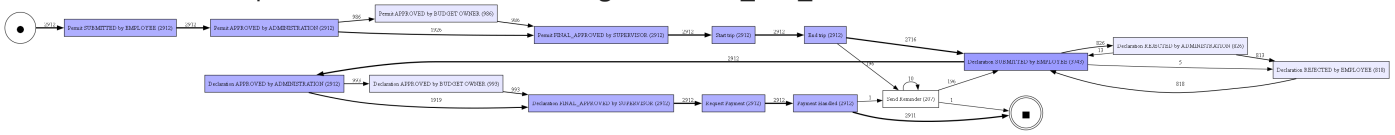
```
replaying log with TBR, completed traces ::   0%|          | 0/753 [00:00<?, ?it/s]
Number of conforming traces: 4261
replaying log with TBR, completed traces ::   0%|          | 0/753 [00:00<?, ?it/s]
Number of conforming traces: 2912
replaying log with TBR, completed traces ::   0%|          | 0/753 [00:00<?, ?it/s]
Number of conforming traces: 2912
Direct Follow Graph from traces conforming to Petri_Net_0.1
```



```
Direct Follow Graph from traces conforming to Petri_Net_0.2
```



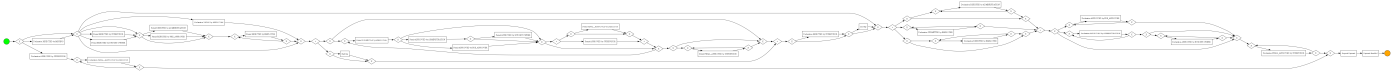
```
Direct Follow Graph from traces conforming to Petri_Net_0.3
```



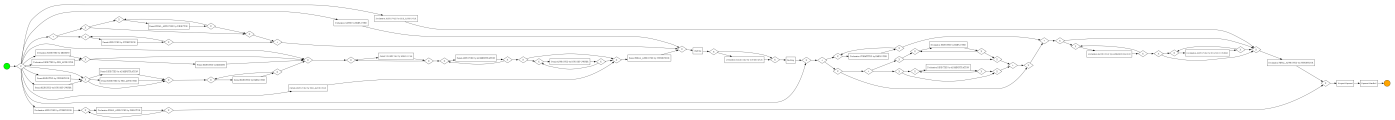
BPMN Graphs

In [26]:
```
bpmn_graph_01 = pm4py.convert_to_bpmn(complete_Petri_Nets[1][1], complete_Petri_Nets[1][2], comp
bpmn_graph_02 = pm4py.convert_to_bpmn(complete_Petri_Nets[2][1], complete_Petri_Nets[2][2], comp
bpmn_graph_03 = pm4py.convert_to_bpmn(complete_Petri_Nets[3][1], complete_Petri_Nets[3][2], comp

print("BPNM from Petri_Net_0.1")
pm4py.view_bpmn(bpmn_graph_01, format='png')
print("BPNM from Petri_Net_0.2")
pm4py.view_bpmn(bpmn_graph_02, format='png')
print("BPNM from Petri_Net_0.3")
pm4py.view_bpmn(bpmn_graph_03, format='png')
```

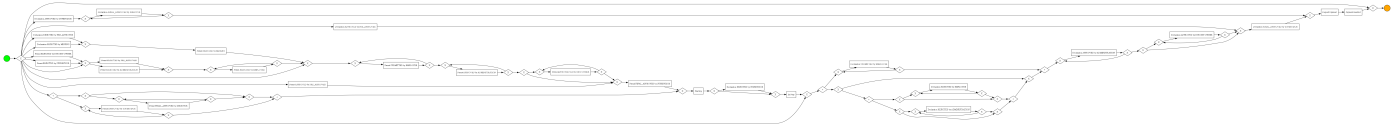
```
BPNM from Petri_Net_0.1
```



```
BPNM from Petri_Net_0.2
```



```
BPNM from Petri_Net_0.3
```



Altough Petri_Net_0.1 has a higher F1-Score, the graphs show that Petri_Net_0.2 and 0.3 have a much more streamlined structure. For this reason, the choice of Normative model is the **Petri_Net_0.2** since it has the highest F1-score compared to Petri_Net_0.3.

Export Normative Model

In [ ]:
```
#file_path = r""
#pm4py.write_pnml(complete_Petri_Nets[2][1], complete_Petri_Nets[2][2], complete_Petri_Nets[2][3
```

Process Mining: Conformance Checking

Dependencies

In [20]:
```
%matplotlib inline
```

In [2]:
```
import pm4py
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from IPython.display import display
```

Event Log Import

In [ ]:
```
file_path = r""
complete_log_df = pm4py.read_xes(file_path)
print("Log loaded. Type:", type(complete_log_df))
```

```
c:\Users\compt\AppData\Local\Programs\Python\Python310\lib\site-packages\pm4py\util\dt_parsing\parser.py:82: UserWarning: ISO8601 strings are not fully supported with strpfromiso for Python versions below 3.11
  warnings.warn(
parsing log, completed traces ::   0%|          | 0/6449 [00:00<?, ?it/s]
Log loaded. Type: <class 'pandas.core.frame.DataFrame'>
```

Normative Model Import

In [4]:
```
file_path = r"C:\Users\compt\Desktop\Process Mining\normative-model.pnml"
normative_petri_net = pm4py.read_pnml(file_path)
```

Extraction of Conforming and Non-Conforming Traces by Token Replay

In [ ]:
```
def separate_conforming_n_non_conforming_log_from_Petri_Net(event_log, net, im, fm):

    from pm4py.objects.conversion.log import converter as log_converter
    from pm4py.algo.conformance.tokenreplay import algorithm as token_replay

    event_log_list = log_converter.apply(event_log, parameters={
        log_converter.Variants.TO_EVENT_LOG.value.Parameters.CASE_ID_KEY: "case:concept:name"
    })

    replay_results = token_replay.apply(event_log_list, net, im, fm)

    conforming_trace_indices = [
        idx for idx, result in enumerate(replay_results) if result["trace_is_fit"]
    ]

    non_conforming_trace_indices = [
        idx for idx, result in enumerate(replay_results) if not result["trace_is_fit"]
    ]

    conforming_case_ids = [
        trace.attributes["concept:name"]
```

```
        for idx, trace in enumerate(event_log_list)
        if replay_results[idx]["trace_is_fit"]
    ]

    df_conforming = event_log[event_log["case:concept:name"].isin(conforming_case_ids)].copy()
    df_non_conforming = event_log[~event_log["case:concept:name"].isin(conforming_case_ids)].cop

    print("Number of conforming traces:", len(conforming_trace_indices))
    print("Number of non-conforming traces:", len(non_conforming_trace_indices))

    return df_conforming, df_non_conforming
```

In [6]:
```
conforming_log_df, non_conforming_log_df = separate_conforming_n_non_conforming_log_from_Petri_N
```

```
replaying log with TBR, completed traces ::   0%|          | 0/753 [00:00<?, ?it/s]
Number of conforming traces: 2912
Number of non-conforming traces: 3537
```

Comparaison of First & Last Activities

In [7]:
```
def display_first_n_last_activities(event_log):

    first_activities_list = (
        event_log.sort_values(by=["case:concept:name", "time:timestamp"])
        .groupby("case:concept:name")
        .head(1)["concept:name"]
        .value_counts()
    )

    last_activities_list = (
        event_log.sort_values(by=["case:concept:name", "time:timestamp"])
        .groupby("case:concept:name")
        .tail(1)["concept:name"]
        .value_counts()
    )

    first_activities_df = first_activities_list.reset_index()
    first_activities_df.columns = ["First Activity", "Count"]
    last_activities_df = last_activities_list.reset_index()
    last_activities_df.columns = ["Last Activity", "Count"]

    styled_first_df = first_activities_df.style.set_properties(**{
        'text-align': 'center'
        }).set_table_styles([
            {'selector': 'th', 'props': [('text-align', 'center')]},
            {'selector': 'td:nth-child(1)', 'props': [('text-align', 'Left')]}
        ]).hide(axis="index")

    styled_last_df = last_activities_df.style.set_properties(**{
        'text-align': 'center'
        }).set_table_styles([
            {'selector': 'th', 'props': [('text-align', 'center')]},
            {'selector': 'td:nth-child(1)', 'props': [('text-align', 'Left')]}
        ]).hide(axis="index")

    display(styled_first_df)
    display(styled_last_df)
```

In [8]:
```
print("First & Last Activities (Conforming):")
display_first_n_last_activities(conforming_log_df)
print("First & Last Activities (Non-Conforming):")
display_first_n_last_activities(non_conforming_log_df)
```

First & Last Activities (Conforming):

First Activity	Count
Permit SUBMITTED by EMPLOYEE	2912

Last Activity	Count
Payment Handled	2911
Send Reminder	1

First & Last Activities (Non-Conforming):

First Activity	Count
Permit SUBMITTED by EMPLOYEE	2382
Start trip	740
Declaration SUBMITTED by EMPLOYEE	407
Declaration SAVED by EMPLOYEE	8

Last Activity	Count
Payment Handled	2735
End trip	593
Declaration REJECTED by EMPLOYEE	130
Declaration SAVED by EMPLOYEE	54
Declaration REJECTED by MISSING	11
Permit REJECTED by MISSING	8
Request Payment	3
Send Reminder	1
Declaration REJECTED by SUPERVISOR	1
Declaration FINAL_APPROVED by SUPERVISOR	1

In [9]:
```
def filter_traces_by_first_n_last_activities(event_log, first_activities, last_activities):

    df_sorted = event_log.sort_values(by=["case:concept:name", "time:timestamp"])

    first_events = df_sorted.groupby("case:concept:name").head(1)
    last_events = df_sorted.groupby("case:concept:name").tail(1)

    matching_first_case_ids = first_events[first_events["concept:name"].isin(first_activities)][
    matching_last_case_ids = last_events[last_events["concept:name"].isin(last_activities)]["cas

    matching_case_ids = set(matching_first_case_ids).intersection(set(matching_last_case_ids))

    filtered_log = df_sorted[df_sorted["case:concept:name"].isin(matching_case_ids)].copy()

    return filtered_log
```

In [10]:
```
filtered_non_conforming_log_df = filter_traces_by_first_n_last_activities(non_conforming_log_df,
num_traces = filtered_non_conforming_log_df["case:concept:name"].nunique()
print(f"Number of traces: {num_traces}")
```

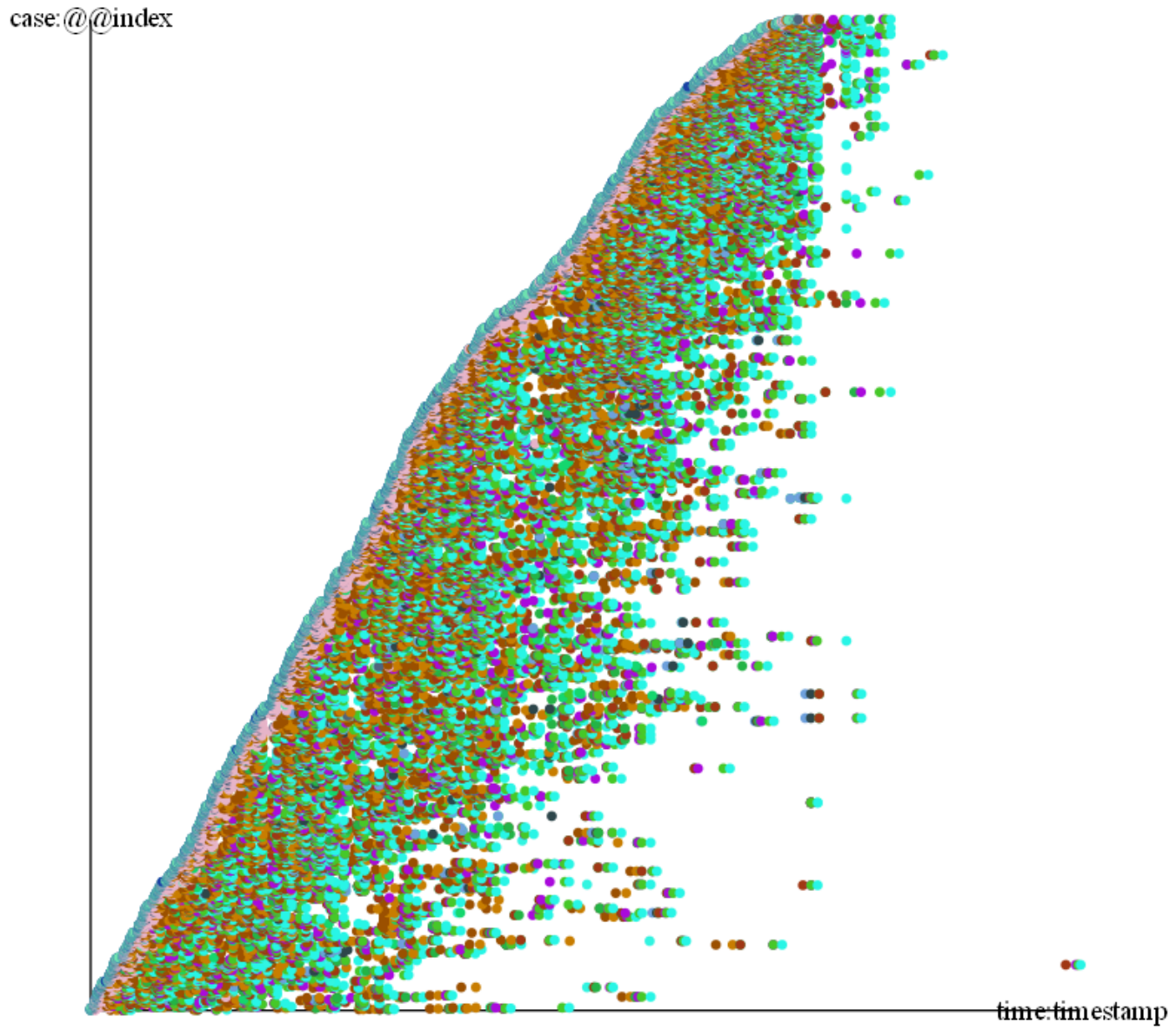
```
Number of traces: 2106
```

Comparing Conforming & Non-Conforming Traces

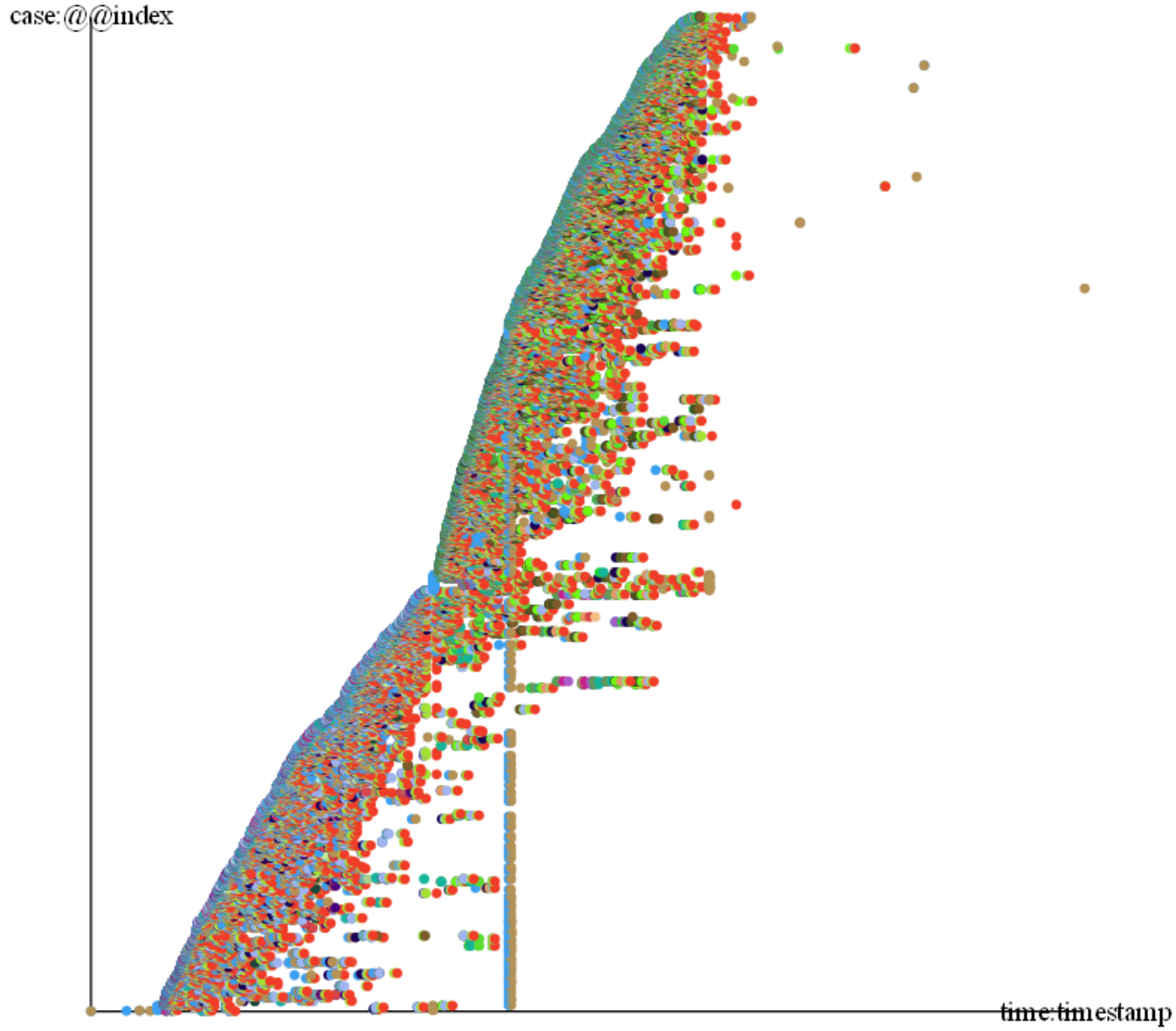
Dotted Charts

In [11]:
```
print("Conforming Traces")
pm4py.view_dotted_chart(conforming_log_df, show_legend = False)
print("Non-Conforming Traces")
pm4py.view_dotted_chart(non_conforming_log_df, show_legend = False)
print("Filtered Non-Conforming Traces")
pm4py.view_dotted_chart(filtered_non_conforming_log_df, show_legend = False)
```

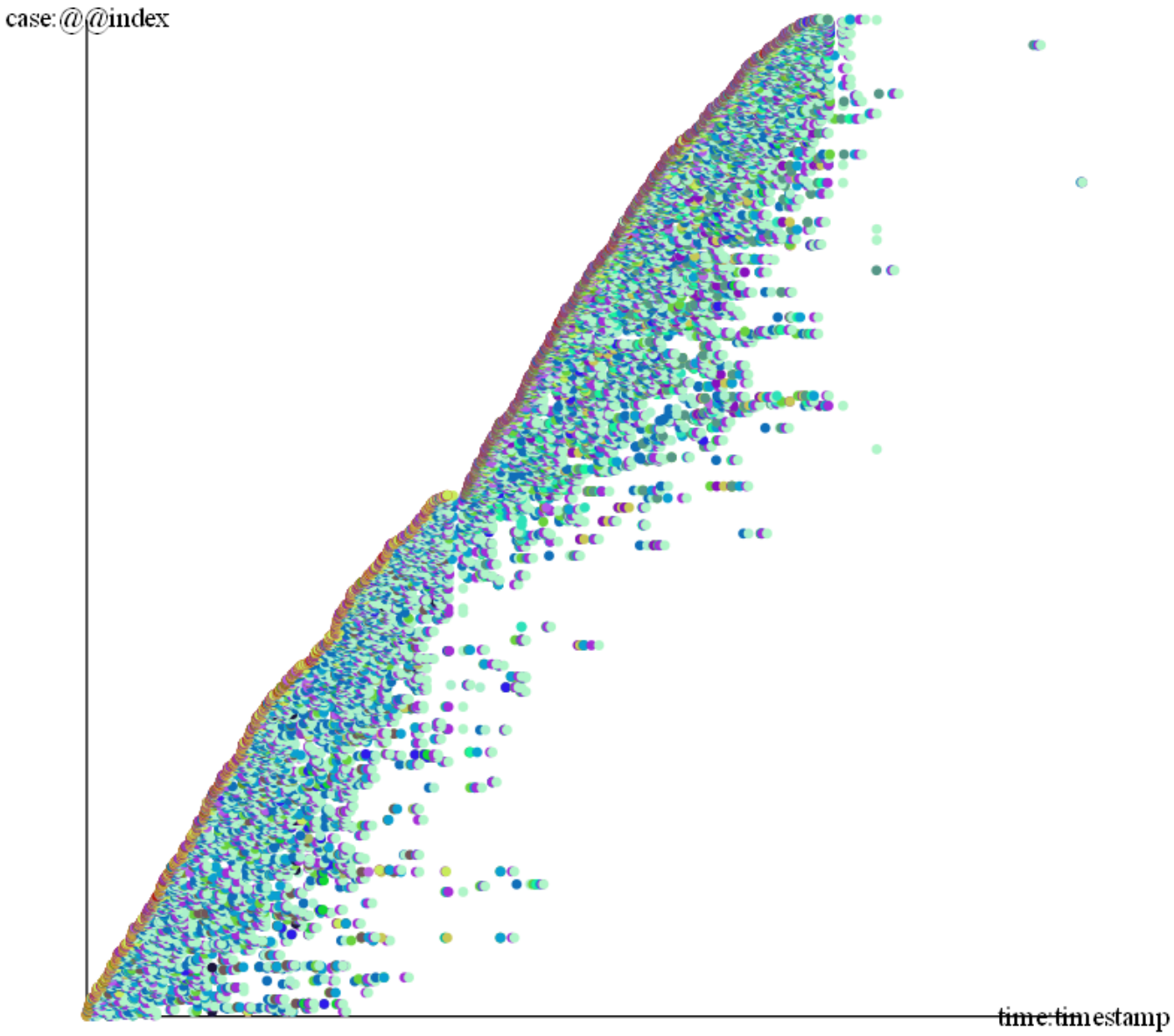
```
Conforming Traces
```




```
Non-Conforming Traces
```



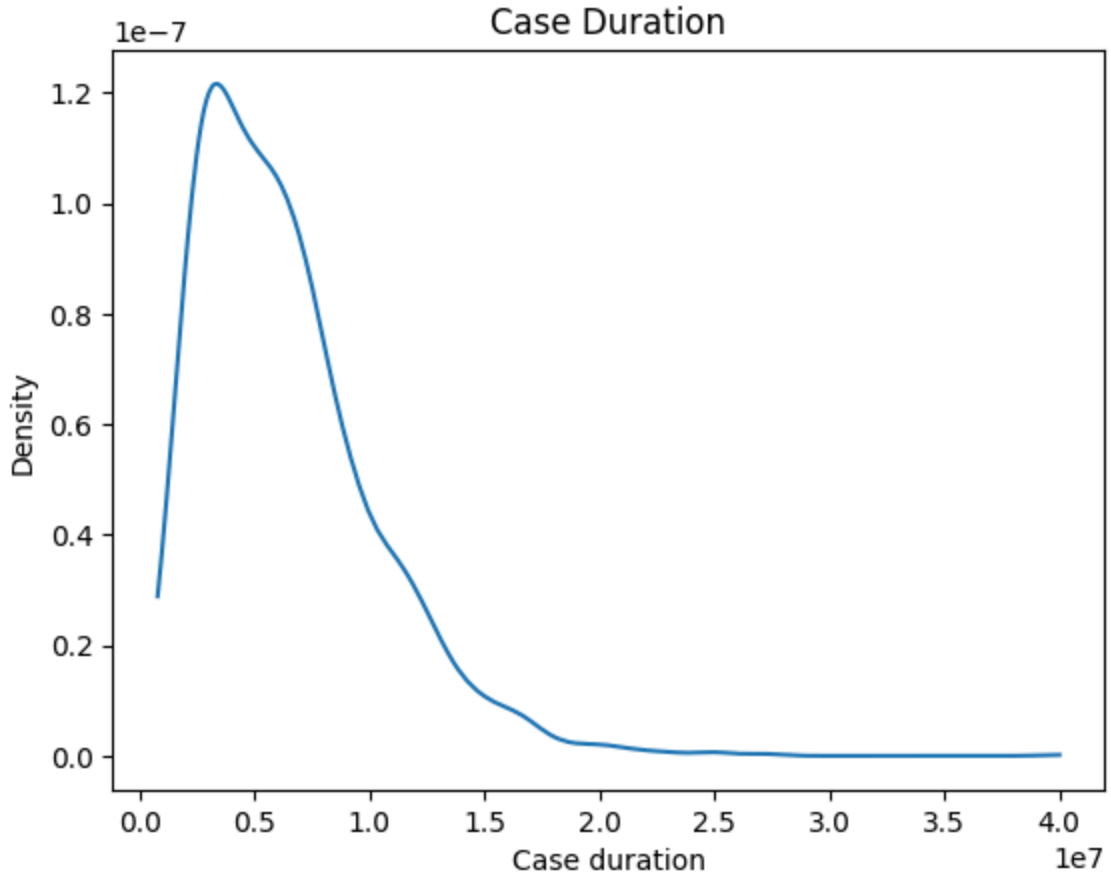
Filtered Non-Conforming Traces

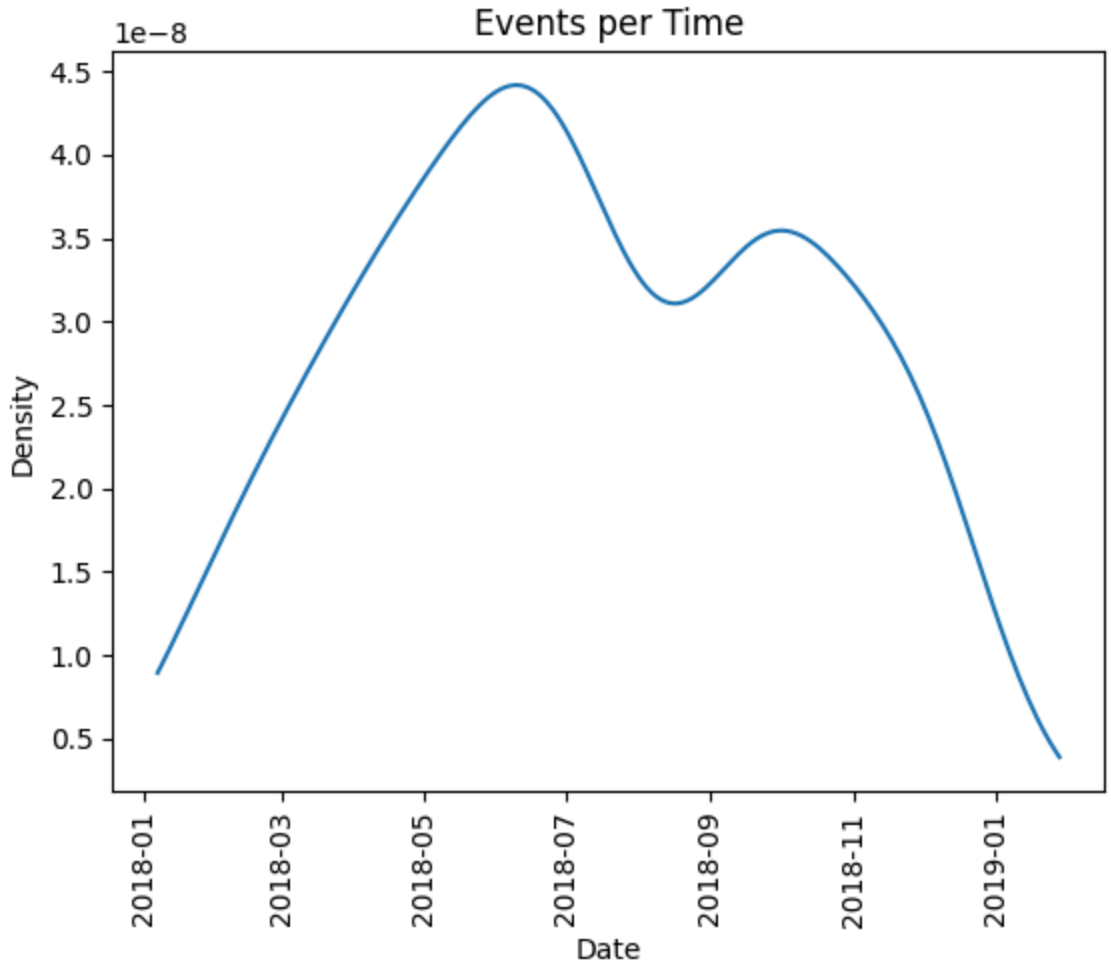


Trace Duration and Density of Events per Time

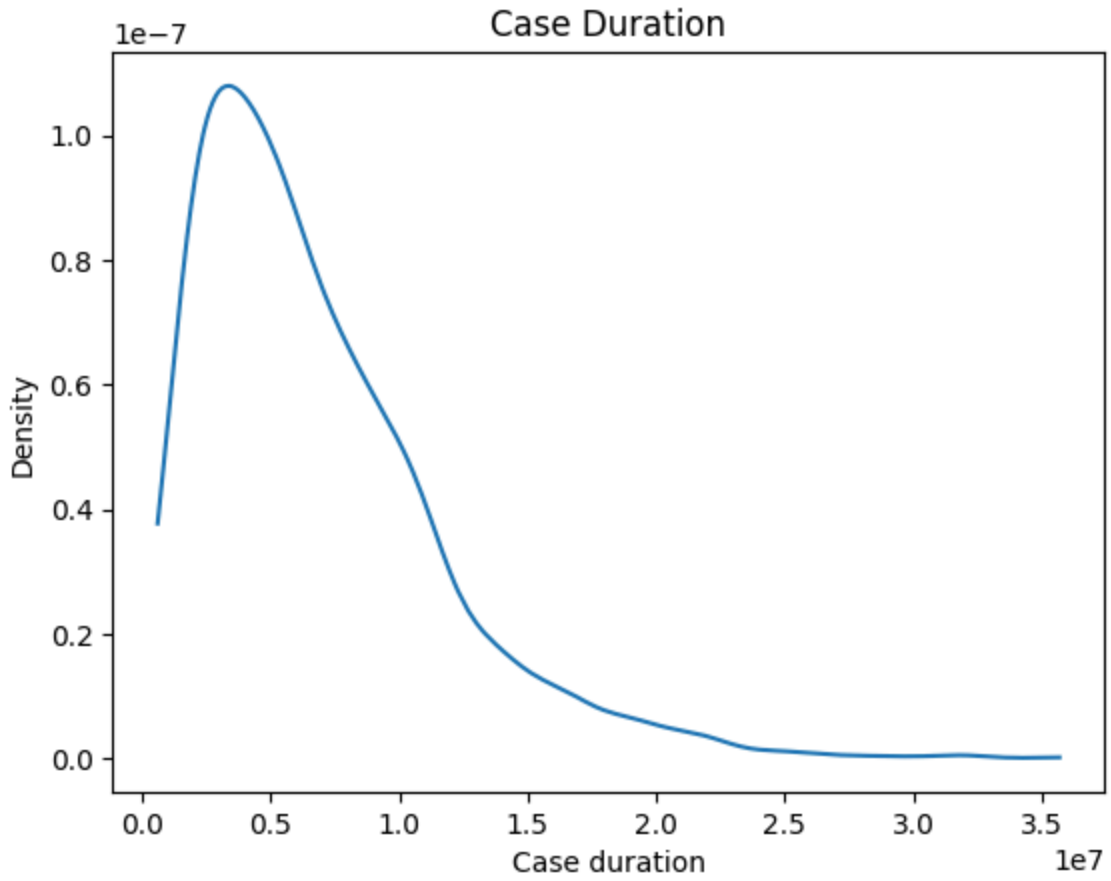
In [12]:
```
print("Conforming Traces")
pm4py.view_case_duration_graph(conforming_log_df)
pm4py.view_events_per_time_graph(conforming_log_df)
print("Filtered Non-Conforming Traces")
pm4py.view_case_duration_graph(filtered_non_conforming_log_df)
pm4py.view_events_per_time_graph(filtered_non_conforming_log_df)
```

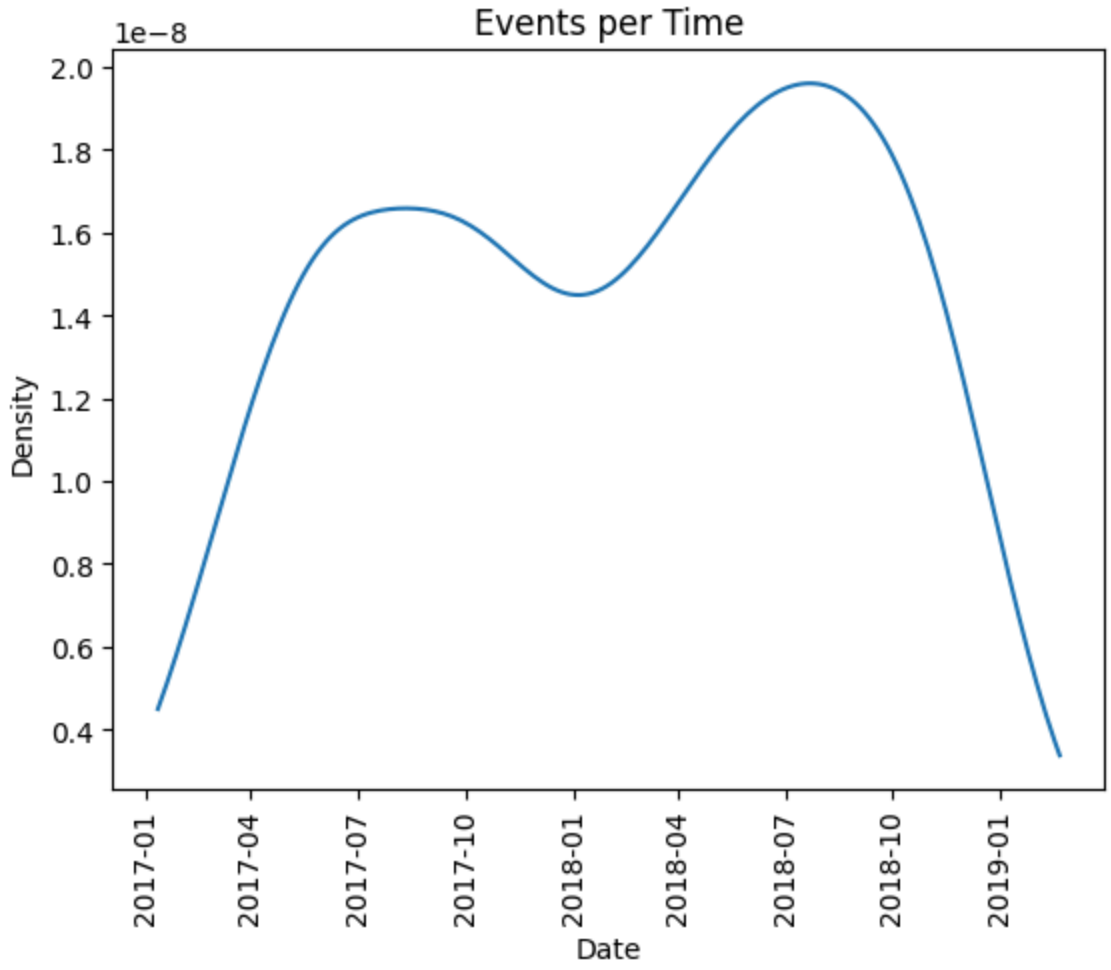
```
Conforming Traces
```





Filtered Non-Conforming Traces





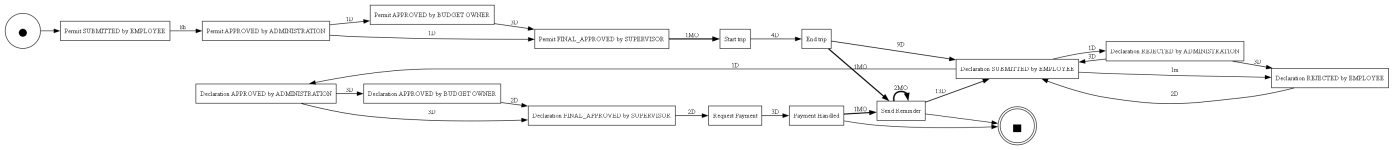
Aggregate Performance

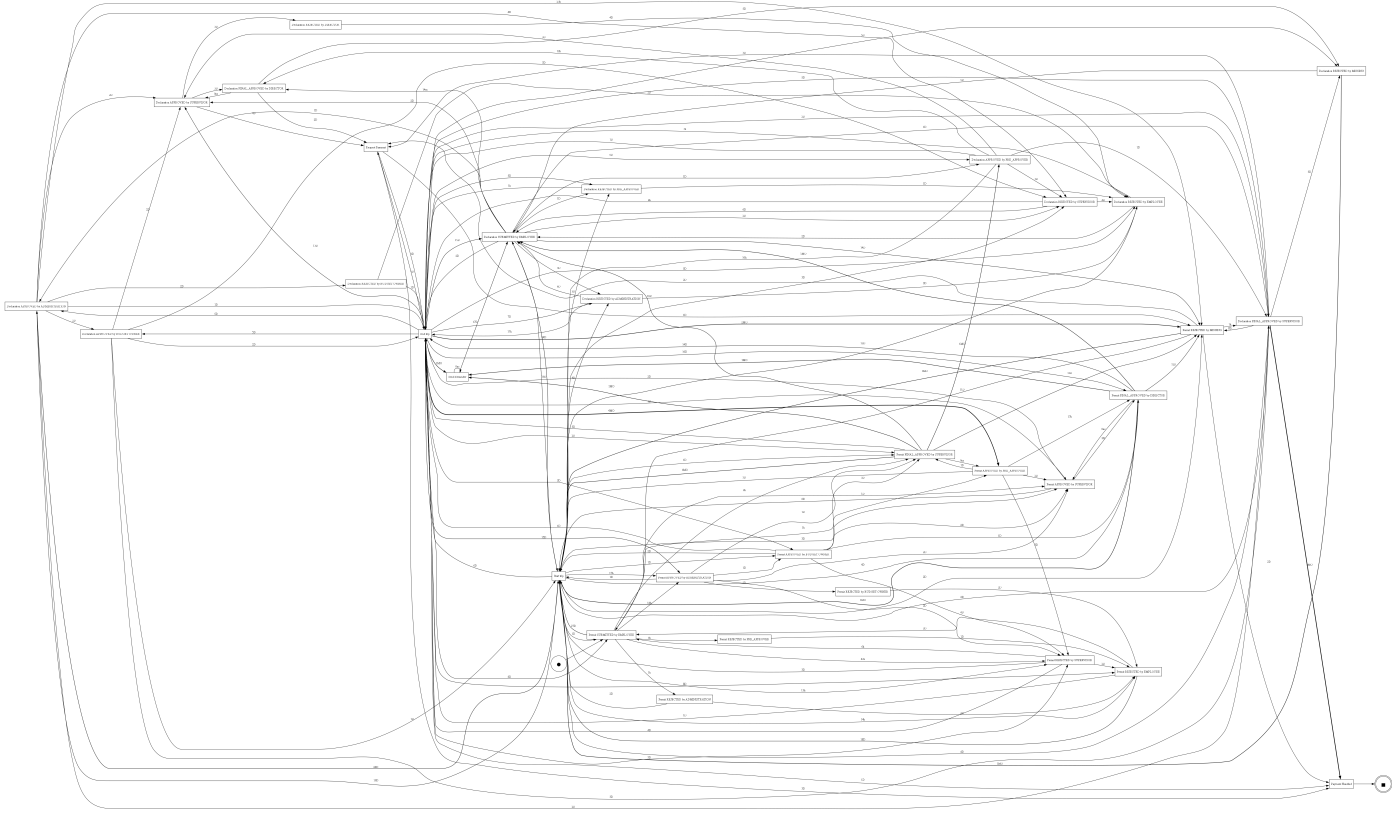
In [25]:
```
def compare_logs_performance(conforming_log_df, non_conforming_log_df ):
    avg_length_conforming = conforming_log_df.groupby("case:concept:name").size().mean()
    avg_length_non_conforming = non_conforming_log_df.groupby("case:concept:name").size().mean()

    conforming_durations = (
        conforming_log_df.groupby("case:concept:name")["time:timestamp"].agg(["min", "max"])
    )
    conforming_durations["duration_days"] = (
        (conforming_durations["max"] - conforming_durations["min"]).dt.total_seconds() / (60 * 6
    )
    avg_duration_conforming = conforming_durations["duration_days"].mean()
    std_duration_conforming = conforming_durations["duration_days"].std()
    median_duration_conforming = conforming_durations["duration_days"].median()

    non_conforming_durations = (
        non_conforming_log_df.groupby("case:concept:name")["time:timestamp"].agg(["min", "max"])
    )
    non_conforming_durations["duration_days"] = (
        (non_conforming_durations["max"] - non_conforming_durations["min"]).dt.total_seconds() /
    )
    avg_duration_non_conforming = non_conforming_durations["duration_days"].mean()
    std_duration_non_conforming = non_conforming_durations["duration_days"].std()
    median_duration_non_conforming = non_conforming_durations["duration_days"].median()

    summary_stats = pd.DataFrame({
        "Log Type": ["Conforming", "Filtered Non-Conforming"],
        "Avg Trace Length (events)": [avg_length_conforming, avg_length_non_conforming],
        "Avg Trace Duration (days)": [avg_duration_conforming, avg_duration_non_conforming],
        "Median Trace Duration (days)": [median_duration_conforming, median_duration_non_conform
        "Std Dev Trace Duration (days)": [std_duration_conforming, std_duration_non_conforming]
    })

    styled_summary = summary_stats.style.format({
        "Avg Trace Length (events)": "{:.2f}",
        "Avg Trace Duration (days)": "{:.2f}",
        "Median Trace Duration (days)": "{:.2f}",
        "Std Dev Trace Duration (days)": "{:.2f}"
    }).set_properties(**{'text-align': 'center'}).set_table_styles([
        {'selector': 'th', 'props': [('text-align', 'center')]},
        {'selector': 'td:nth-child(1)', 'props': [('text-align', 'left')]}
    ]).hide(axis="index")

    display(styled_summary)
```

In [26]:
```
compare_logs_performance(conforming_log_df, filtered_non_conforming_log_df)
```

Log Type	Avg Trace Length (events)	Avg Trace Duration (days)	Median Trace Duration (days)	Std Dev Trace Duration (days)
Conforming	11.60	73.86	65.26	44.91
Filtered Non-Conforming	11.51	78.43	65.96	54.61

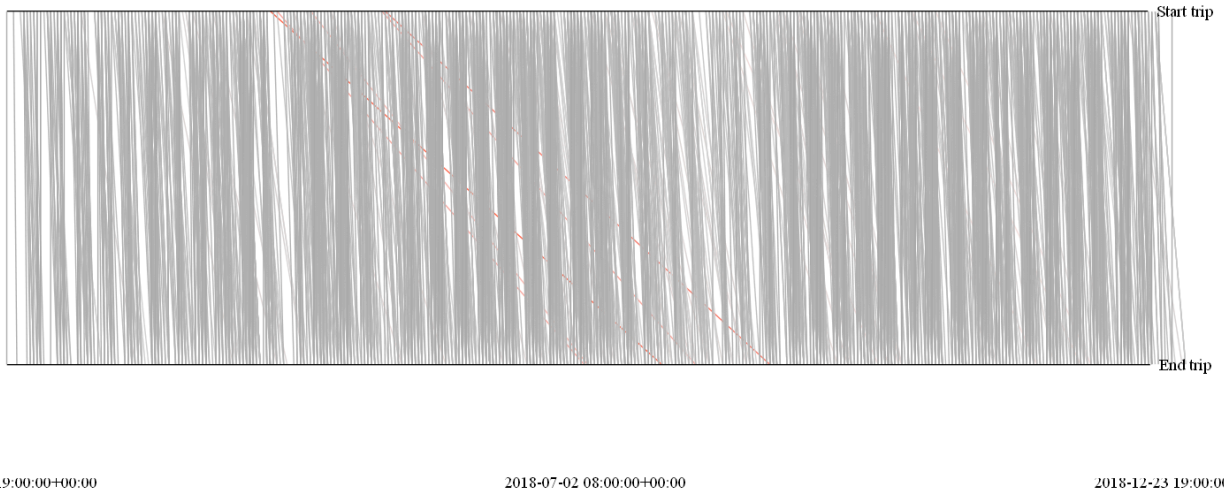
In [29]:
```
performance_conforming_dfg, start_c_activities, end_c_activities = pm4py.discover_performance_df
performance_filtered_non_conforming_dfg, start_fnc_activities, end_fnc_activities = pm4py.discov
pm4py.view_performance_dfg(performance_conforming_dfg, start_c_activities, end_c_activities, for
pm4py.view_performance_dfg(performance_filtered_non_conforming_dfg, start_fnc_activities, end_fn
```



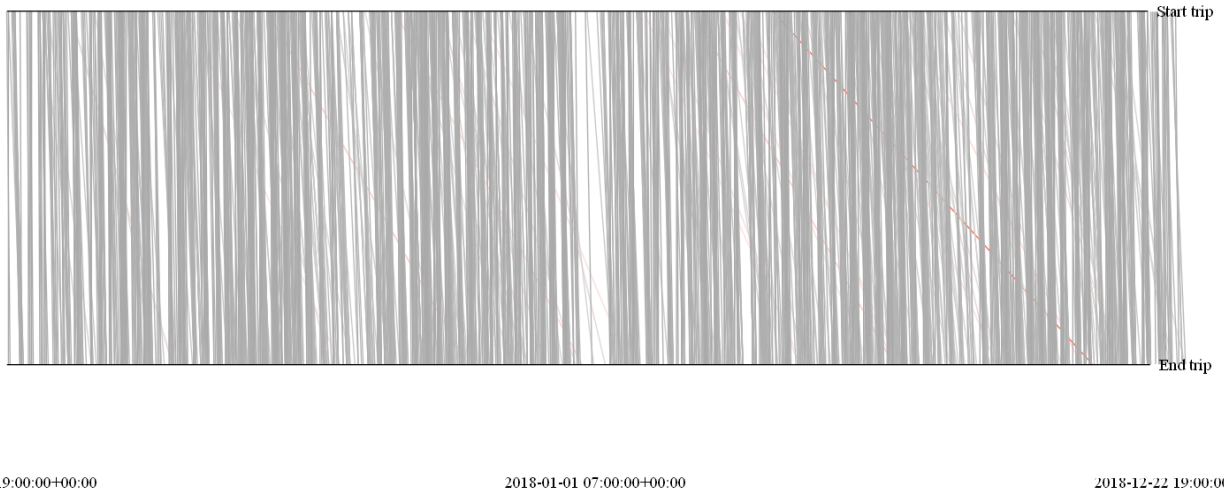


In [32]:
```
pm4py.view_performance_spectrum(conforming_log_df, ['Start trip', 'End trip'], format='png', act
pm4py.view_performance_spectrum(filtered_non_conforming_log_df, ['Start trip', 'End trip'], form
```

```
c:\Users\compt\AppData\Local\Programs\Python\Python310\lib\site-packages\pm4py\algo\discovery\pe
rformance_spectrum\variants\dataframe.py:77: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/in
dexing.html#returning-a-view-versus-a-copy
  dataframe[activity_key] = dataframe[activity_key].astype("string")
```




```
c:\Users\compt\AppData\Local\Programs\Python\Python310\lib\site-packages\pm4py\algo\discovery\pe
rformance_spectrum\variants\dataframe.py:77: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/in
dexing.html#returning-a-view-versus-a-copy
  dataframe[activity_key] = dataframe[activity_key].astype("string")
```



Variants Analysis

In [14]:
```
def get_n_print_variants(event_log, log_name):
    variant_dictonary = pm4py.get_variants(
        event_log,
        activity_key='concept:name',
        case_id_key='case:concept:name',
        timestamp_key='time:timestamp'
    )

    print(f"The {sum(variant_dictonary.values())} traces in the {log_name} have {len(variant_dic

    return variant_dictonary
```

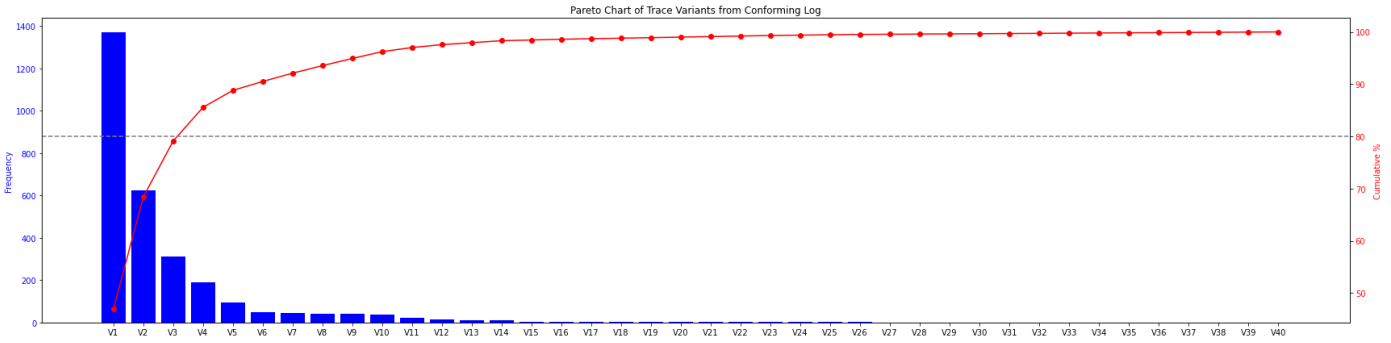
In [15]:
```
variants_conforming_log = get_n_print_variants(conforming_log_df, log_name="Conforming Log")
variants_filtered_non_conforming_log = get_n_print_variants(filtered_non_conforming_log_df, log_
```

```
The 2912 traces in the Conforming Log have 40 variants
The 2106 traces in the Filtered Non-Conforming Log have 391 variants
```

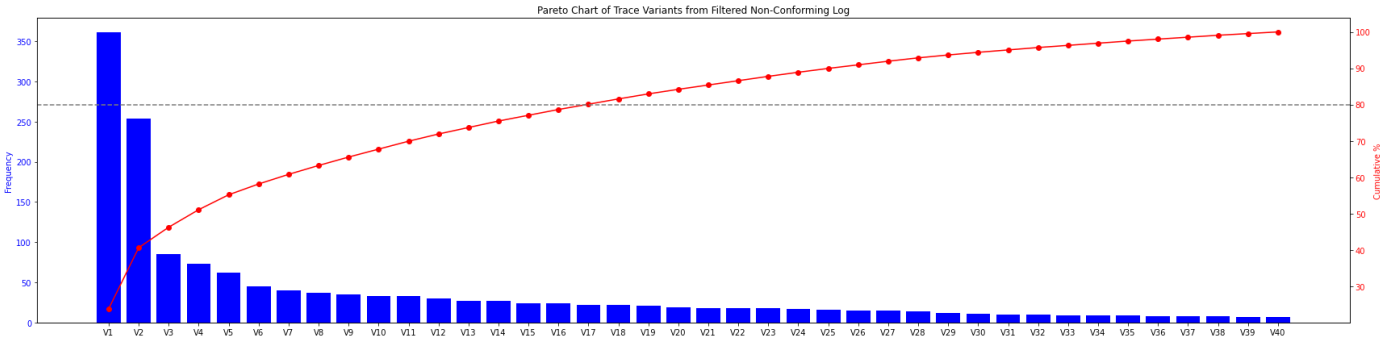
In [16]:
```
def top_N_variants_Pareto_chartd(variants, Top_N, log_name):

    variant_stats = sorted(variants.items(), key=lambda x: x[1], reverse=True)

    top_variant_stats = variant_stats[:Top_N]

    x = [f"V{i+1}" for i in range(len(top_variant_stats))]
    y = [v[1] for v in top_variant_stats]

    cumulative = np.cumsum(y)
    cumulative_percent = cumulative / cumulative[-1] * 100

    fig, ax1 = plt.subplots(figsize=(24, 6))

    ax1.bar(x, y, color='blue')
    ax1.set_ylabel('Frequency', color='blue')
    ax1.tick_params(axis='y', labelcolor='blue')

    ax2 = ax1.twinx()
    ax2.plot(x, cumulative_percent, color='red', marker='o')
    ax2.set_ylabel('Cumulative %', color='red')
    ax2.tick_params(axis='y', labelcolor='red')
    ax2.axhline(80, color='gray', linestyle='--')

    plt.title(f"Pareto Chart of Trace Variants from {log_name}")
    plt.xticks(rotation=90)
```

```
    plt.tight_layout()
    plt.show()
```

In [22]:
```
top_N_variants_Pareto_chartd(variants_conforming_log, Top_N=40, log_name="Conforming Log")
```



In [23]:
```
top_N_variants_Pareto_chartd(variants_filtered_non_conforming_log, Top_N=40, log_name="Filtered
```



Saving Event Logs

In [ ]:
```
#conforming_log_df.to_pickle("conforming-log.pkl")
#filtered_non_conforming_log_df.to_pickle("filtered-non-conforminlog.pkl")
```

Feature Extraction: Categorical

Dependencies

In [1]:
```
import pm4py
import numpy as np
import pandas as pd
```

Event Log Import

In [2]:
```
non_conforming_log_df = pd.read_pickle("filtered-non-conforminlog.pkl")
conforming_log_df  = pd.read_pickle("conforming-log.pkl")
```

Normative Model Import

In [3]:
```
file_path = r"C:\Users\compt\Desktop\Process Mining\normative-model.pnml"
normative_petri_net = pm4py.read_pnml(file_path)
```

Feature Extraction from Log

Built-In pm4py Extraction Function

In [4]:
```
def extract_initial_features(event_log):
    features_df = pm4py.extract_features_dataframe(
        event_log,
        activity_key='concept:name',
        case_id_key='case:concept:name',
        timestamp_key='time:timestamp',
        str_tr_attr=[],
        num_tr_attr=["Amount", "RequestedAmount", "OriginalAmount", "Permit RequestedBudget", "A
        str_ev_attr=['org:role'],
        include_case_id=True
    )

    return features_df
```

In [5]:
```
non_conf_init_features_df = extract_initial_features(non_conforming_log_df)
conf_init_features_df = extract_initial_features(conforming_log_df)
```

Standardize Numerical Values

In [6]:
```
def convert_floats_to_int64(df):
    df_copy = df.copy()

    float_cols = df_copy.select_dtypes(include=['float32', 'float64']).columns

    for col in float_cols:
        df_copy[col] = df_copy[col].astype('int64')

    return df_copy
```

In [7]:
```
non_conf_init_features_df = convert_floats_to_int64(non_conf_init_features_df)
```

```
conf_init_features_df = convert_floats_to_int64(conf_init_features_df)
```

Maintain Uniform Dataframes (conforming vs non-conforming)

In [8]:
```
def add_n_fill_missing_columns(df_1, df_2):
    all_columns = sorted(set(df_1.columns).union(df_2.columns))

    df_1_all_columns = df_1.reindex(columns=all_columns, fill_value=0)
    df_2_all_columns = df_2.reindex(columns=all_columns, fill_value=0)

    return df_1_all_columns, df_2_all_columns
```

In [9]:
```
non_conf_full_init_features_df, conf_init_full_features_df = add_n_fill_missing_columns(non_conf
```

Convert Numerical Values to Bins

In [10]:
```
def numerical_to_bin(
    df_list,
    columns_to_bin,
    n_bins=4,
    bin_labels=None,
    strategy='quantile',
    drop_original=True,
    custom_bins=None
):

    if bin_labels is None:
        bin_labels = [f'bin_{i+1}' for i in range(n_bins)]

    if custom_bins is None:
        custom_bins = {}

    combined = pd.concat(df_list, axis=0)

    bin_edges = {}

    for col in columns_to_bin:
        if strategy == 'quantile':
            _, bins = pd.qcut(combined[col], q=n_bins, retbins=True, duplicates='drop')
        elif strategy == 'uniform':
            if col in custom_bins:
                bins = custom_bins[col]
            else:
                _, bins = pd.cut(combined[col], bins=n_bins, retbins=True)
        else:
            raise ValueError("strategy must be 'quantile' or 'uniform'")

        bin_edges[col] = bins

        updated_dfs = []
        for df in df_list:
            df_copy = df.copy()
            df_copy[col + '_bin'] = pd.cut(
                df_copy[col],
                bins=bins,
                labels=bin_labels[:len(bins)-1],
                include_lowest=True
            )

            if drop_original:
                df_copy = df_copy.drop(columns=col)
```

```
            updated_dfs.append(df_copy)

        df_list = updated_dfs

    return df_list, bin_edges
```

In [11]:
```
dfs = [non_conf_full_init_features_df, conf_init_full_features_df]

numerical_cols = [
    'case:Permit RequestedBudget',
    'case:AdjustedAmount',
    'case:OriginalAmount',
    'case:Amount',
    'case:RequestedAmount'
]

[non_conf_binned_df, conf_binned_df], bin_info = numerical_to_bin(
    df_list = dfs,
    columns_to_bin = numerical_cols,
    n_bins = 4,
    bin_labels = ['very_low', 'low', 'high', 'very_high'],
    strategy = 'quantile',
    drop_original = True
)
```

Convert Encoding (0/1) to Categorical

In [12]:
```
def one_hot_to_catergorical(df):

    role_cols = [col for col in df.columns if col.startswith("org:role_")]

    features_categorical = df.copy()

    for col in role_cols:
        features_categorical[col] = df[col].map(lambda x: 'present' if x == 1 else 'absent')

    return features_categorical
```

In [13]:
```
conf_cat_df = one_hot_to_catergorical(conf_binned_df)
non_conf_cat_df = one_hot_to_catergorical(non_conf_binned_df)
```

Extract Case Level Categorical Attributes

In [14]:
```
def add_categorical_features(event_log, df):
    trace_categorical_attributes_df = event_log.groupby("case:concept:name").agg({
        "case:Permit BudgetNumber": "first",
        "case:Permit OrganizationalEntity": "first",
        "case:Permit ProjectNumber": "first",
        "case:BudgetNumber": "first"
    }).reset_index()

    categorical_features_df = pd.merge(df, trace_categorical_attributes_df, on="case:concept:nam

    return categorical_features_df
```

In [15]:
```
non_conf_cat_features_df = add_categorical_features(non_conforming_log_df, non_conf_cat_df)
conf_cat_features_df = add_categorical_features(conforming_log_df, conf_cat_df)
```

Extract Event Frequency

In [16]:
```
def add_event_frequency_features(event_log, df):
    frequency_table = event_log.groupby(["case:concept:name", "concept:name"]).size().unstack(fi
    frequency_table = frequency_table.reset_index()

    event_freqency_features_df = pd.merge(df, frequency_table, on="case:concept:name", how="left

    return event_freqency_features_df
```

In [17]:
```
non_conf_ev_freq_features_df = add_event_frequency_features(non_conforming_log_df, non_conf_cat_
conf_ev_freq_features_df = add_event_frequency_features(conforming_log_df, conf_cat_features_df)
```

Maintain Uniform Dataframes (conforming vs non-conforming)

In [18]:
```
non_conf_full_ev_freq_features_df, conf_full_ev_freq_features_df = add_n_fill_missing_columns(no
```

Convert Numerical to Categorical Values

In [19]:
```
def map_values_to_labels(df, columns, mappings):

    df_copy = df.copy()

    value_map = dict(mappings)

    for col in columns:
        if col in df_copy.columns:
            df_copy[col] = df_copy[col].map(value_map)
        else:
            print(f"Column '{col}' not found in DataFrame.")

    return df_copy
```

In [20]:
```
numeric_cols = non_conf_full_ev_freq_features_df.select_dtypes(include='int64').columns.tolist()
value_map = [(0, 'Never'), (1, 'Once'), (2, 'Twice'), (3, 'Three times'), (4, 'Four Times'), (5,

non_conf_cat_freq_features_df = map_values_to_labels(non_conf_full_ev_freq_features_df, numeric_
conf_cat_freq_features_df = map_values_to_labels(conf_full_ev_freq_features_df, numeric_cols, va
```

Performance Metrics

Fitness

In [21]:
```
def add_trace_fitness_metric(event_log, df):

    from pm4py.objects.conversion.log import converter as log_converter

    fitness_scores = []

    for case_id in df["case:concept:name"]:
        trace_df = event_log[event_log["case:concept:name"] == case_id]

        sublog = log_converter.apply(trace_df, variant=log_converter.Variants.TO_EVENT_LOG)

        fitness = pm4py.fitness_token_based_replay(
        sublog,
        normative_petri_net[0],
        normative_petri_net[1],
        normative_petri_net[2],
        activity_key='concept:name',
```

```
        case_id_key='case:concept:name',
        timestamp_key='time:timestamp'
        )

        fitness_value = fitness.get("average_trace_fitness")
        fitness_scores.append((case_id, fitness_value))

    fitness_df = pd.DataFrame(fitness_scores, columns=["case:concept:name", "token_fitness"])

    features_fit_df = df.merge(fitness_df, on="case:concept:name", how="left")

    return features_fit_df
```

In [22]:
```
non_conf_fit_features_df = add_trace_fitness_metric(non_conforming_log_df, non_conf_cat_freq_fea
conf_fit_features_df = add_trace_fitness_metric(conforming_log_df, conf_cat_freq_features_df)
```

Convert Numerical Values to Bins

In [23]:
```
dfs = [non_conf_fit_features_df, conf_fit_features_df]

num_cols = ['token_fitness']

[non_conf_fit_binned_df2, conf_fit_binned_df2], bin_info2 = numerical_to_bin(
    df_list = dfs,
    columns_to_bin = num_cols,
    n_bins = 4,
    bin_labels = ['very_low', 'low', 'high', 'very_high'],
    strategy = 'quantile',
    drop_original = True
)
```

In [24]:
```
dfs = [non_conf_fit_features_df, conf_fit_features_df]

custom_bins = {
    'token_fitness': [0.7, 0.8, 0.85, 0.95, 1.01]
}

[non_conf_fit_binned_df, conf_fit_binned_df], bin_info = numerical_to_bin(
    df_list = dfs,
    columns_to_bin = ['token_fitness'],
    n_bins = 4,
    bin_labels = ['very_low', 'low', 'high', 'very_high'],
    strategy = 'uniform',
    custom_bins = custom_bins,
    drop_original = True
)
```

Trace Duration & Event Count

In [25]:
```
def add_trace_duration_n_event_count(event_log, df):

    trace_duration_df = event_log.groupby("case:concept:name")["time:timestamp"].agg(
        trace_start="min",
        trace_end="max"
    ).reset_index()

    trace_duration_df["trace_duration_days"] = (trace_duration_df["trace_end"] - trace_duration_

    trace_event_count_df = event_log.groupby("case:concept:name").size().reset_index(name="num_e

    features_duration_df = df.merge(trace_duration_df[["case:concept:name", "trace_duration_days
```

```
    features_ev_count_df = features_duration_df.merge(trace_event_count_df, on="case:concept:nam

    return features_ev_count_df
```

In [26]:
```
non_conf_duration_n_count_features_df = add_trace_duration_n_event_count(non_conforming_log_df,
conf_duration_n_count_features_df = add_trace_duration_n_event_count(conforming_log_df, conf_fit
```

Convert Numerical Values to Bins

In [27]:
```
dfs = [non_conf_duration_n_count_features_df, conf_duration_n_count_features_df]

[non_conf_duration_n_count_binned_df, conf_duration_n_count_binned_df], bin_info = numerical_to_
    df_list = dfs,
    columns_to_bin = ['trace_duration_days', 'num_events'],
    n_bins = 4,
    bin_labels = ['very_low', 'low', 'high', 'very_high'],
    strategy = 'quantile',
    drop_original=True
)
```

In [28]:
```
from pandas.api.types import CategoricalDtype

def apply_ordered_binning_dtype(df, suffix='_bin', categories=None):

    if categories is None:
        categories = ['very_low', 'low', 'high', 'very_high']

    ordinal_type = CategoricalDtype(categories=categories, ordered=True)
    df_copy = df.copy()

    for col in df_copy.columns:
        if col.endswith(suffix):
            df_copy[col] = df_copy[col].astype(ordinal_type)

    return df_copy
```

In [29]:
```
non_conf_final_features_df = apply_ordered_binning_dtype(non_conf_duration_n_count_binned_df)
conf_final_features_df = apply_ordered_binning_dtype(conf_duration_n_count_binned_df)
```

Exporting Features

In [30]:
```
print(f"Non-conforming feature set is of shape {non_conf_final_features_df.shape}")
print(f"Conforming feature set is of shape {conf_final_features_df.shape}")
```

```
Non-conforming feature set is of shape (2106, 53)
Conforming feature set is of shape (2912, 53)
```

In [218…
```
non_conf_final_features_df.to_pickle("non_conforming_categorical_features.pkl")
conf_final_features_df.to_pickle("conforming_categorical_features.pkl")
```

Feature Extraction: Numerical

Dependencies

In [1]:
```
import pm4py
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Event Log Import

In [2]:
```
non_conforming_log_df = pd.read_pickle("filtered-non-conforminlog.pkl")
conforming_log_df  = pd.read_pickle("conforming-log.pkl")
```

Normative Model Import

In [3]:
```
file_path = r"C:\Users\compt\Desktop\Process Mining\normative-model.pnml"
normative_petri_net = pm4py.read_pnml(file_path)
```

Feature Extraction from Log

Built-In pm4py Extraction Function

In [4]:
```
def extract_initial_features(event_log):
    features_df = pm4py.extract_features_dataframe(
        event_log,
        activity_key='concept:name',
        case_id_key='case:concept:name',
        timestamp_key='time:timestamp',
        str_tr_attr=[],
        num_tr_attr=["Amount", "RequestedAmount", "OriginalAmount", "Permit RequestedBudget", "A
        str_ev_attr=['org:role'],
        include_case_id=True
    )

    return features_df
```

In [5]:
```
non_conf_init_features_df = extract_initial_features(non_conforming_log_df)
conf_init_features_df = extract_initial_features(conforming_log_df)
```

Extract Case Level Categorical Attributes

In [6]:
```
def add_categorical_features(event_log, df):
    trace_categorical_attributes_df = event_log.groupby("case:concept:name").agg({
        "case:Permit BudgetNumber": "first",
        "case:Permit OrganizationalEntity": "first",
        "case:Permit ProjectNumber": "first",
        "case:BudgetNumber": "first"
    }).reset_index()

    categorical_features_df = pd.merge(df, trace_categorical_attributes_df, on="case:concept:nam
```

```
    return categorical_features_df
```

In [7]:
```
non_conf_cat_features_df = add_categorical_features(non_conforming_log_df, non_conf_init_feature
conf_cat_features_df = add_categorical_features(conforming_log_df, conf_init_features_df)
```

Extract Event Frequency

In [8]:
```
def add_event_frequency_features(event_log, df):
    frequency_table = event_log.groupby(["case:concept:name", "concept:name"]).size().unstack(fi
    frequency_table = frequency_table.reset_index()

    event_freqency_features_df = pd.merge(df, frequency_table, on="case:concept:name", how="left

    return event_freqency_features_df
```

In [9]:
```
non_conf_ev_freq_features_df = add_event_frequency_features(non_conforming_log_df, non_conf_cat_
conf_ev_freq_features_df = add_event_frequency_features(conforming_log_df, conf_cat_features_df)
```

Performance Metrics

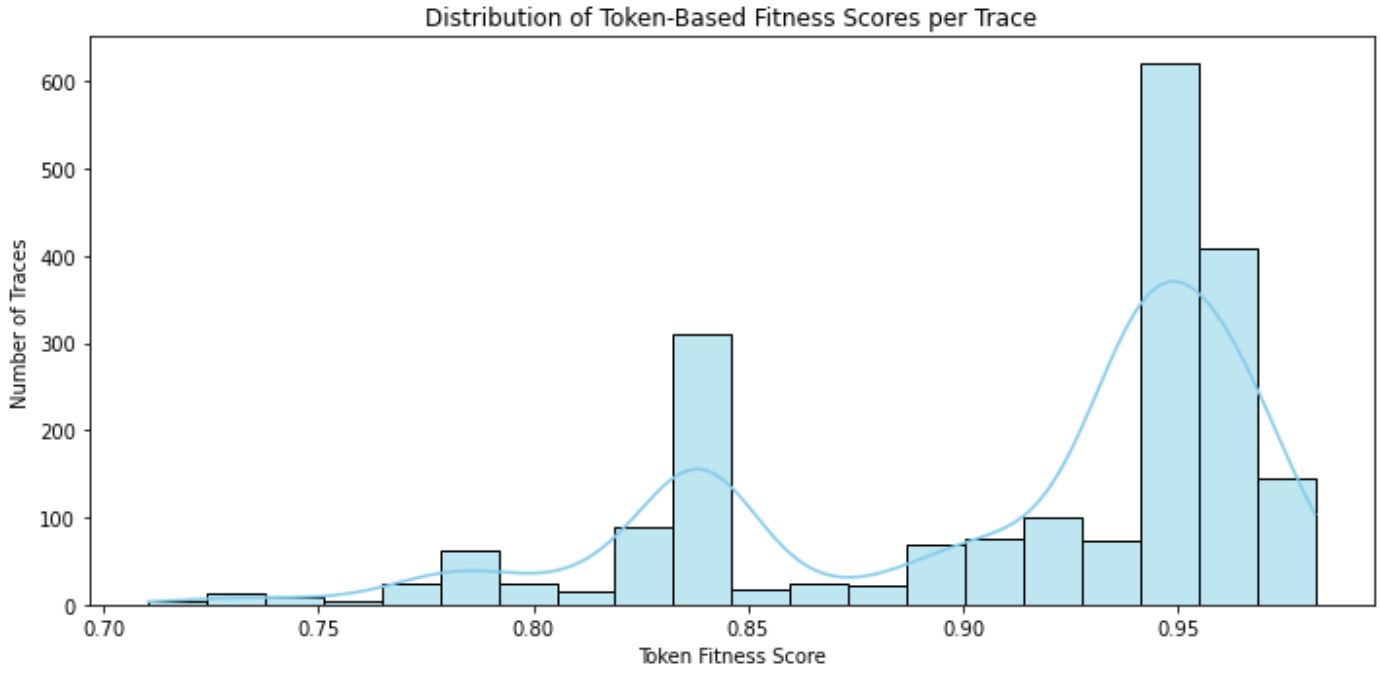
Fitness

In [10]:
```
def add_trace_fitness_metric(event_log, df):

    from pm4py.objects.conversion.log import converter as log_converter

    fitness_scores = []

    for case_id in df["case:concept:name"]:
        trace_df = event_log[event_log["case:concept:name"] == case_id]

        sublog = log_converter.apply(trace_df, variant=log_converter.Variants.TO_EVENT_LOG)

        fitness = pm4py.fitness_token_based_replay(
        sublog,
        normative_petri_net[0],
        normative_petri_net[1],
        normative_petri_net[2],
        activity_key='concept:name',
        case_id_key='case:concept:name',
        timestamp_key='time:timestamp'
        )

        fitness_value = fitness.get("average_trace_fitness")
        fitness_scores.append((case_id, fitness_value))

    fitness_df = pd.DataFrame(fitness_scores, columns=["case:concept:name", "token_fitness"])

    features_fit_df = df.merge(fitness_df, on="case:concept:name", how="left")

    return features_fit_df
```

In [11]:
```
non_conf_ev_freq_features_df = add_trace_fitness_metric(non_conforming_log_df, non_conf_ev_freq_
conf_ev_freq_features_df = add_trace_fitness_metric(conforming_log_df, conf_ev_freq_features_df)
```

In [12]:
```
plt.figure(figsize=(10, 5))
sns.histplot(non_conf_ev_freq_features_df["token_fitness"], bins=20, kde=True, color="skyblue")
```

```
plt.title("Distribution of Token-Based Fitness Scores per Trace")
plt.xlabel("Token Fitness Score")
plt.ylabel("Number of Traces")
plt.tight_layout()
plt.show()
```



Trace Duration & Event Count

In [13]:
```
def add_trace_duration_n_event_count(event_log, df):

    trace_duration_df = event_log.groupby("case:concept:name")["time:timestamp"].agg(
        trace_start="min",
        trace_end="max"
    ).reset_index()

    trace_duration_df["trace_duration_days"] = (trace_duration_df["trace_end"] - trace_duration_

    trace_event_count_df = event_log.groupby("case:concept:name").size().reset_index(name="num_e

    features_duration_df = df.merge(trace_duration_df[["case:concept:name", "trace_duration_days
    features_ev_count_df = features_duration_df.merge(trace_event_count_df, on="case:concept:nam

    return features_ev_count_df
```

In [14]:
```
non_conf_duration_n_count_features_df = add_trace_duration_n_event_count(non_conforming_log_df,
conf_duration_n_count_features_df = add_trace_duration_n_event_count(conforming_log_df, conf_ev_
```

Encode Categoricl Values

In [15]:
```
def encode_categorical_to_label(df):

    from sklearn.preprocessing import LabelEncoder

    label_cols = ['case:Permit BudgetNumber',
                  'case:Permit OrganizationalEntity',
                  'case:Permit ProjectNumber',
                  'case:BudgetNumber']
```

```
    le = LabelEncoder()

    for col in label_cols:
        df[col + '_le'] = le.fit_transform(df[col])
        df.drop(columns=col, inplace=True)

    return df
```

In [16]:
```
non_conf_complete_features_df = encode_categorical_to_label(non_conf_duration_n_count_features_d
conf_complete_features_df = encode_categorical_to_label(conf_duration_n_count_features_df)
```

Maintain Uniform Dataframes (conforming vs non-conforming)

In [17]:
```
all_columns = set(non_conf_complete_features_df.columns).union(conf_complete_features_df.columns

non_conf_final_features_df = non_conf_complete_features_df.reindex(columns=all_columns, fill_val
conf_final_features_df = conf_complete_features_df.reindex(columns=all_columns, fill_value=0)
```

In [18]:
```
print(f"Non-conforming feature set is of shape {non_conf_final_features_df.shape}")
print(f"Conforming feature set is of shape {conf_final_features_df.shape}")
```

```
Non-conforming feature set is of shape (2106, 53)
Conforming feature set is of shape (2912, 53)
```

Export Features

In [25]:
```
non_conf_final_features_df.to_pickle("non_conforming_numerical_features.pkl")
conf_final_features_df.to_pickle("conforming_numerical_features.pkl")
```

Clustering

Dependencies

In [1]:
```
import pm4py
import numpy as np
import pandas as pd
from sklearn.metrics import silhouette_score
import networkx as nx
import matplotlib.pyplot as plt
import seaborn as sns
from IPython.display import display
```

Feature Sets Import

Categorical

In [2]:
```
non_conforming_categorical_features = pd.read_pickle("non_conforming_categorical_features.pkl")
non_conforming_categorical_features = non_conforming_categorical_features.drop(columns='case:con
```

Numerical

In [3]:
```
non_conforming_numerical_features = pd.read_pickle("non_conforming_numerical_features.pkl")
non_conf_idx = non_conforming_numerical_features[["case:concept:name"]].copy()
non_conforming_numerical_features = non_conforming_numerical_features.drop(columns='case:concept
```

Log Import

In [4]:
```
non_conforming_log = pd.read_pickle("filtered-non-conforminlog.pkl")
```

Modeling

In [5]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_num_scaled = scaler.fit_transform(non_conforming_numerical_features)
```

KMeans

In [6]:
```
from sklearn.cluster import KMeans

kmeans_3 = KMeans(n_clusters=3, random_state=42)
kmeans_3_labels = kmeans_3.fit_predict(X_num_scaled)

kmeans_4 = KMeans(n_clusters=4, random_state=42)
kmeans_4_labels = kmeans_4.fit_predict(X_num_scaled)

kmeans_5 = KMeans(n_clusters=5, random_state=42)
kmeans_5_labels = kmeans_5.fit_predict(X_num_scaled)
```

DBSCAN

In [7]:
```
from sklearn.cluster import DBSCAN

dbscan = DBSCAN(eps=0.5, min_samples=5)
dbscan_labels = dbscan.fit_predict(X_num_scaled)
```

Louvain (numerical)

In [ ]:
```
from sklearn.metrics.pairwise import cosine_similarity
from networkx.algorithms.community import louvain_communities

similarity_matrix = cosine_similarity(X_num_scaled)

G_num = nx.from_numpy_array(similarity_matrix)

louvain_num_clusters = louvain_communities(G_num, seed=42)
```

Louvain (categorical)

In [9]:
```
from sklearn.preprocessing import OneHotEncoder
from sklearn.metrics import pairwise_distances

encoder = OneHotEncoder()
X_cat_encoded = encoder.fit_transform(non_conforming_categorical_features).toarray()

jaccard_sim = 1 - pairwise_distances(X_cat_encoded, metric='jaccard')

G_cat = nx.from_numpy_array(jaccard_sim)
louvain_cat_clusters = louvain_communities(G_cat, seed=42)
```

```
c:\Users\compt\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\metrics\pairwis
e.py:2361: DataConversionWarning: Data was converted to boolean for metric jaccard
  warnings.warn(msg, DataConversionWarning)
```

Evaluation

In [10]:
```
def evaluate_clustering(name, labels, data=None, is_graph=False, graph=None):

    labels = np.array(labels)

    if is_graph:
        from networkx.algorithms.community.quality import modularity
        communities = [set(np.where(labels == i)[0]) for i in np.unique(labels)]
        mod_score = modularity(graph, communities)
        sil_score = None
    else:
        mod_score = None
        if len(np.unique(labels)) > 1:
            sil_score = silhouette_score(data, labels)
        else:
            sil_score = None

    cluster_sizes = pd.Series(labels).value_counts().to_dict()

    return {
        'Method': name,
        'Clusters': len(np.unique(labels)),
        'Silhouette': sil_score,
        'Modularity': mod_score,
```

```
        'Cluster Sizes': cluster_sizes
    }
```

In [11]:
```
results = []

results.append(evaluate_clustering('KMeans_3', kmeans_3_labels, data=X_num_scaled))
results.append(evaluate_clustering('KMeans_4', kmeans_4_labels, data=X_num_scaled))
results.append(evaluate_clustering('KMeans_5', kmeans_5_labels, data=X_num_scaled))

results.append(evaluate_clustering('DBSCAN', dbscan_labels, data=X_num_scaled))

louvain_num_labels = np.zeros(len(X_num_scaled), dtype=int)
for i, group in enumerate(louvain_num_clusters):
    for node in group:
        louvain_num_labels[node] = i
results.append(evaluate_clustering('Louvain (Numerical)', louvain_num_labels, is_graph=True, gra

louvain_cat_labels = np.zeros(len(X_cat_encoded), dtype=int)
for i, group in enumerate(louvain_cat_clusters):
    for node in group:
        louvain_cat_labels[node] = i
results.append(evaluate_clustering('Louvain (Categorical)', louvain_cat_labels, is_graph=True, g

comparison_df = pd.DataFrame(results)

comparison_df
```

Out[11]:

	Method	Clusters	Silhouette	Modularity	Cluster Sizes
0	KMeans_3	3	0.201716	NaN	{1: 964, 2: 877, 0: 265}
1	KMeans_4	4	0.177984	NaN	{1: 910, 2: 716, 0: 269, 3: 211}
2	KMeans_5	5	0.199633	NaN	{1: 902, 4: 744, 3: 212, 2: 129, 0: 119}
3	DBSCAN	12	-0.359259	NaN	{-1: 2032, 7: 14, 8: 11, 6: 7, 3: 6, 2: 6, 5: ...
4	Louvain (Numerical)	3	NaN	4.215688	{2: 918, 0: 792, 1: 396}
5	Louvain (Categorical)	3	NaN	0.058864	{1: 918, 0: 641, 2: 547}

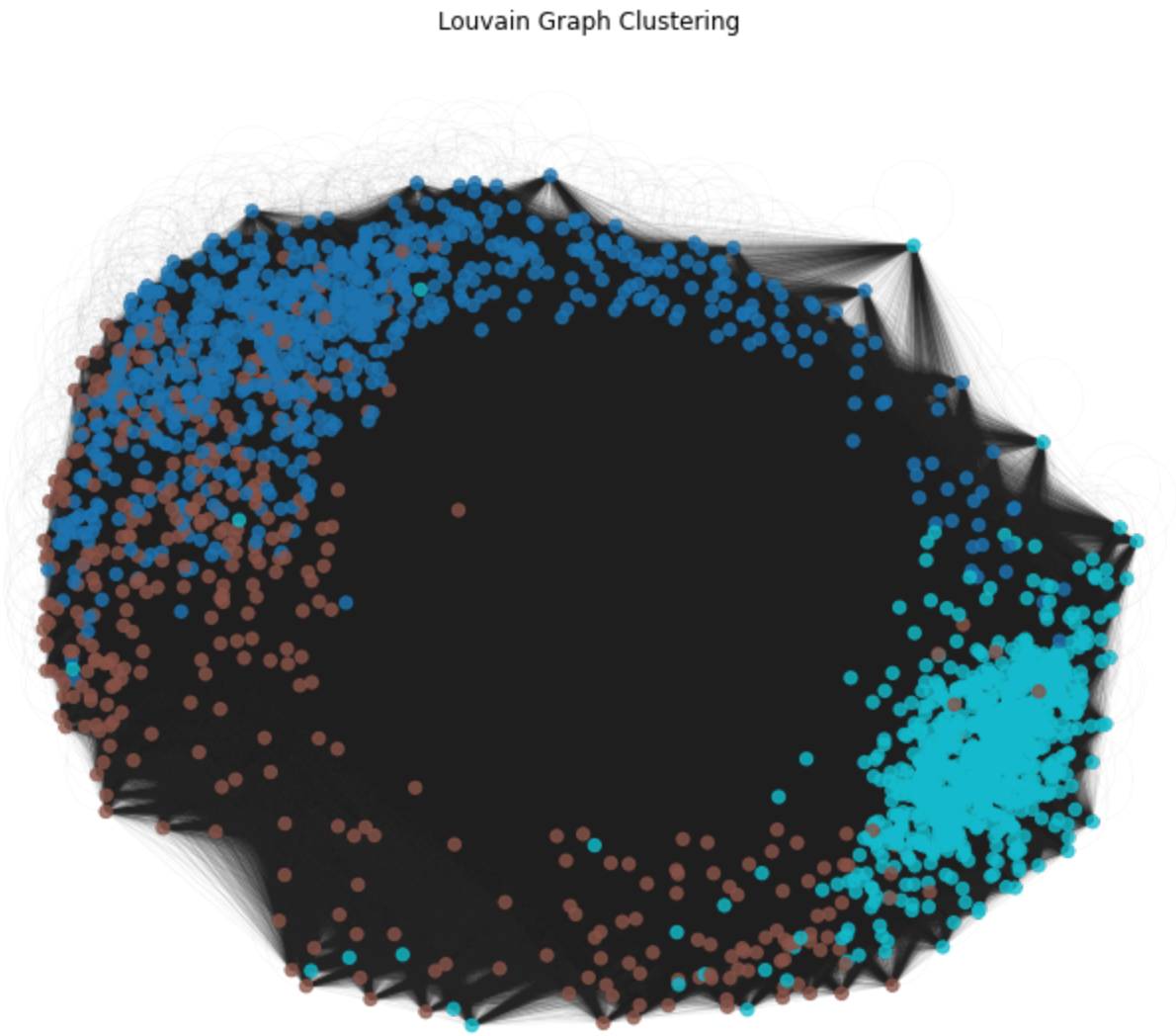
Graph Visualization

In [ ]:
```
node_labels = {}
for i, community in enumerate(louvain_num_clusters):
    for node in community:
        node_labels[node] = i

node_colors = [node_labels[n] for n in G_num.nodes()]
```

In [ ]:
```
node_cat_labels = {}
for i, community in enumerate(louvain_cat_clusters):
    for node in community:
        node_cat_labels[node] = i

node_cat_colors = [node_cat_labels[n] for n in G_cat.nodes()]
```

In [ ]:
```
plt.figure(figsize=(12, 10), facecolor='white')
pos = nx.spring_layout(G_num, seed=42)

nx.draw_networkx_nodes(G_num, pos, node_color=node_colors, cmap=plt.cm.tab10, node_size=40, alph
nx.draw_networkx_edges(G_num, pos, alpha=0.05, width=0.5)
```

```
plt.title("Louvain Graph Clustering")
plt.axis("off")
plt.show()
```



In [ ]:
```
cluster_label_series = pd.Series(node_labels).sort_index()

non_conforming_numerical_features['cluster'] = cluster_label_series.values

non_conforming_categorical_features['cluster'] = cluster_label_series.values
```

Export Clusters

In [17]:
```
non_conforming_numerical_features.to_pickle("clustered_non_conforming_numerical2.pkl")
non_conforming_categorical_features.to_pickle("clustered_non_conforming_categorical2.pkl")
```

Create & Export Sub-Eventlogs From Clusters

In [ ]:
```
non_conf_idx["cluster"] = cluster_label_series.values

non_conforming_log_with_clusters = non_conforming_log.merge(
    non_conf_idx,
    on='case:concept:name',
    how='inner'
)
```

```
cluster_0_log = non_conforming_log_with_clusters[
    non_conforming_log_with_clusters['cluster'] == 0
]

cluster_1_log = non_conforming_log_with_clusters[
    non_conforming_log_with_clusters['cluster'] == 1
]

cluster_2_log = non_conforming_log_with_clusters[
    non_conforming_log_with_clusters['cluster'] == 2
]

non_conforming_cluster_0_log = cluster_0_log.drop(columns='cluster')
non_conforming_cluster_1_log = cluster_1_log.drop(columns='cluster')
non_conforming_cluster_2_log = cluster_2_log.drop(columns='cluster')
```

In [36]:

```
non_conforming_cluster_0_log.to_pickle("non_conforming_cluster_0_log.pkl")
non_conforming_cluster_1_log.to_pickle("non_conforming_cluster_1_log.pkl")
non_conforming_cluster_2_log.to_pickle("non_conforming_cluster_2_log.pkl")
```

Rule Extraction

Dependencies

In [ ]:
```
import pm4py
import numpy as np
import pandas as pd
import shap
from xgboost import XGBClassifier
import networkx as nx
import matplotlib.pyplot as plt
import seaborn as sns
```

Import Feature Sets

In [2]:
```
non_conforming_numerical_features = pd.read_pickle("clustered_non_comforming_numerical.pkl")
non_conforming_categorical_features = pd.read_pickle("clustered_non_comforming_categorical.pkl")

conforming_numerical_features = pd.read_pickle("conforming_numerical_features.pkl")
conforming_categorical_features = pd.read_pickle("conforming_categorical_features.pkl")
```

In [3]:
```
conforming_numerical_features = conforming_numerical_features.drop(columns='case:concept:name')
conforming_categorical_features = conforming_categorical_features.drop(columns='case:concept:nam
```

Adding Cluster Value to Conforming Traces

In [4]:
```
conforming_numerical_features["cluster"] = 3
conforming_categorical_features["cluster"] = 3
```

In [5]:
```
X_num = pd.concat([non_conforming_numerical_features.drop("cluster", axis=1),
                   conforming_numerical_features.drop("cluster", axis=1)], axis=0)

y_num = pd.concat([non_conforming_numerical_features["cluster"],
                   conforming_numerical_features["cluster"]], axis=0)
```

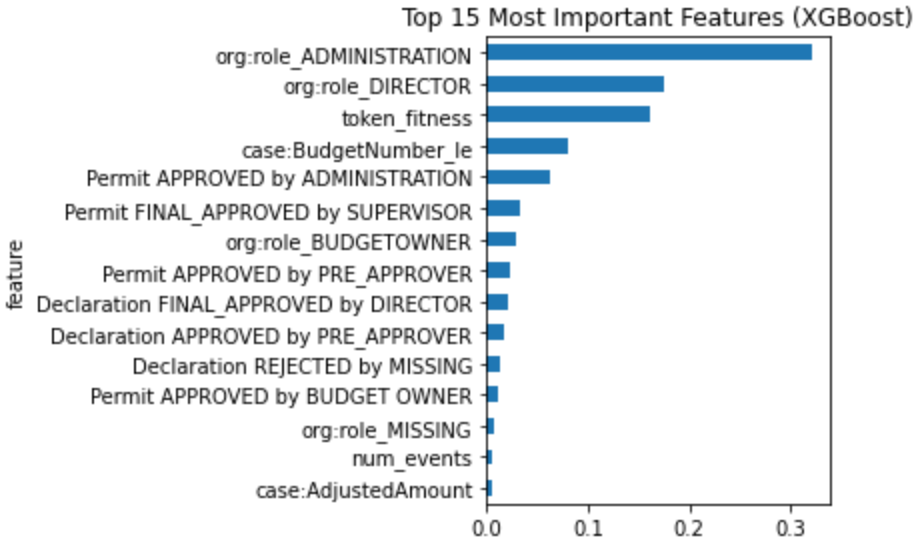
In [6]:
```
df_rules = pd.concat([non_conforming_categorical_features,
                   conforming_categorical_features], axis=0)
```

In [7]:
```
model = XGBClassifier(random_state=42)
model.fit(X_num, y_num)

y_pred = model.predict(X_num)

feature_names = X_num.columns
importances = model.feature_importances_

importance_df = pd.DataFrame({
    'feature': feature_names,
    'importance': importances
}).sort_values(by='importance', ascending=False)

print(importance_df.head(10))
```

```
                                  feature  importance
36                org:role_ADMINISTRATION    0.322575
37                     org:role_DIRECTOR    0.175484
6                          token_fitness    0.161391
28                   case:BudgetNumber_le    0.080931
31      Permit APPROVED by ADMINISTRATION    0.061774
7     Permit FINAL_APPROVED by SUPERVISOR    0.033511
10                   org:role_BUDGETOWNER    0.028380
20        Permit APPROVED by PRE_APPROVER    0.022555
8   Declaration FINAL_APPROVED by DIRECTOR    0.021531
30   Declaration APPROVED by PRE_APPROVER    0.017444
```

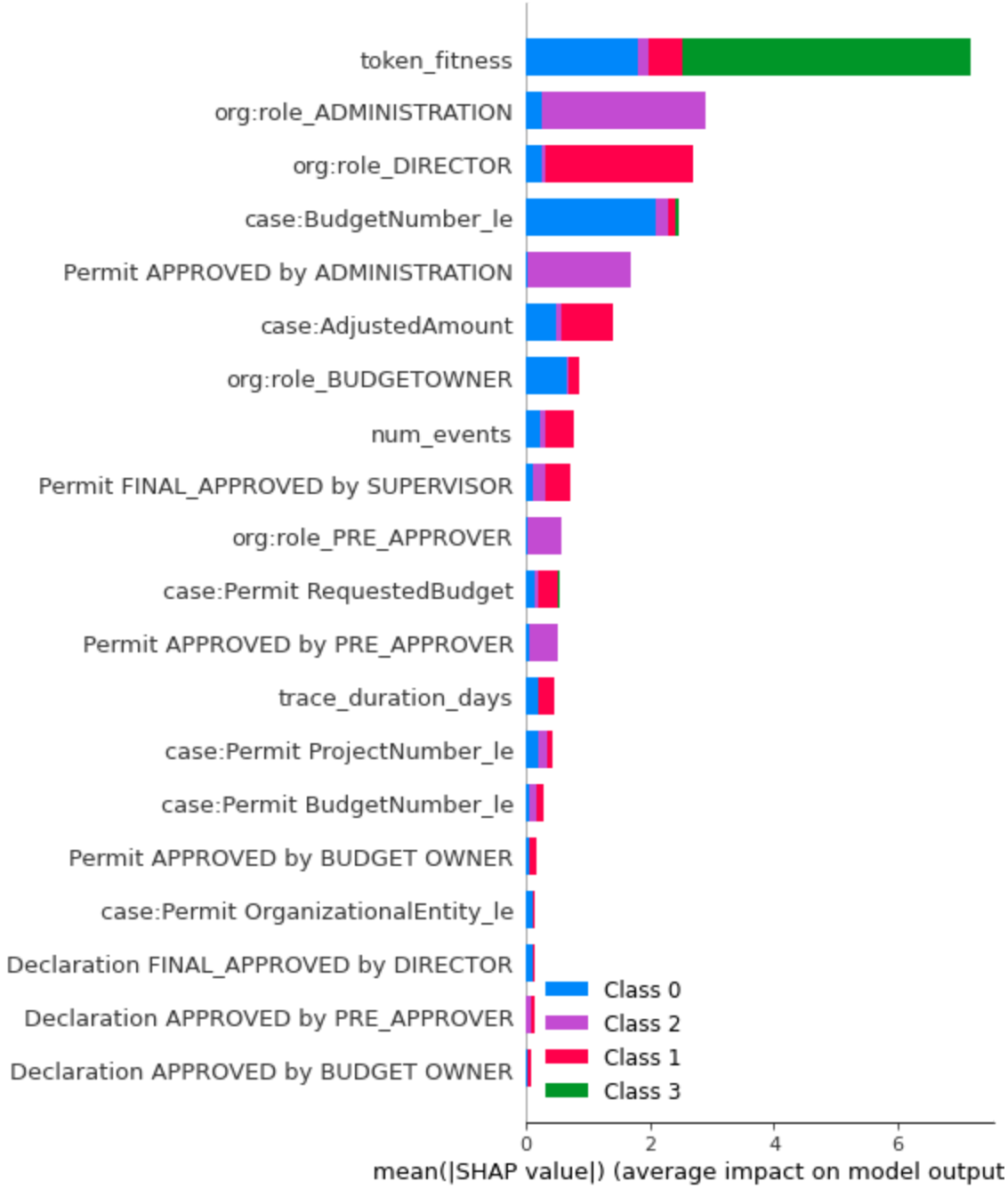
In [8]:
```
plt.figure(figsize=(10, 6))
importance_df.head(15).plot(kind='barh', x='feature', y='importance', legend=False)
plt.title("Top 15 Most Important Features (XGBoost)")
plt.gca().invert_yaxis()
plt.tight_layout()
plt.show()
```

```
<Figure size 720x432 with 0 Axes>
```



In [9]:
```
shap.initjs()
```

In [10]:
```
explainer = shap.TreeExplainer(model)
shap_values = explainer.shap_values(X_num)

shap.summary_plot(shap_values, X_num, plot_type="bar")
```



In [59]:
```
from sklearn.tree import DecisionTreeClassifier, export_text

tree = DecisionTreeClassifier(max_depth=4)
tree.fit(X_num, y_num)

print(export_text(tree, feature_names=list(X_num.columns)))
```

```
|--- token_fitness <= 0.99
|   |--- org:role_ADMINISTRATION <= 0.50
|   |   |--- Permit FINAL_APPROVED by SUPERVISOR <= 0.50
|   |   |   |--- num_events <= 10.50
|   |   |   |   |--- class: 1
|   |   |   |--- num_events >  10.50
|   |   |   |   |--- class: 2
|   |   |--- Permit FINAL_APPROVED by SUPERVISOR >  0.50
|   |   |   |--- case:AdjustedAmount <= 4213.87
|   |   |   |   |--- class: 2
|   |   |   |--- case:AdjustedAmount >  4213.87
|   |   |   |   |--- class: 1
|   |--- org:role_ADMINISTRATION >  0.50
|   |   |--- org:role_DIRECTOR <= 0.50
|   |   |   |--- Permit APPROVED by PRE_APPROVER <= 0.50
|   |   |   |   |--- class: 0
|   |   |   |--- Permit APPROVED by PRE_APPROVER >  0.50
|   |   |   |   |--- class: 2
|   |   |--- org:role_DIRECTOR >  0.50
|   |   |   |--- org:role_BUDGETOWNER <= 0.50
|   |   |   |   |--- class: 1
|   |   |   |--- org:role_BUDGETOWNER >  0.50
|   |   |   |   |--- class: 0
|--- token_fitness >  0.99
|   |--- class: 3
```

In [ ]:
```python
df_rules["cluster"] = df_rules["cluster"].astype(str)
```

In [ ]:
```python
from sklearn.utils import resample

cluster_counts = df_rules['cluster'].explode().value_counts()
print("Cluster distribution in consequents:")
print(cluster_counts)

cluster_3_data = df_rules[df_rules['cluster'] == '3']
other_clusters_data = df_rules[df_rules['cluster'] != '3']

desired_size = cluster_counts.min()

undersampled_cluster_3 = resample(cluster_3_data,
                                  replace=False,
                                  n_samples=desired_size,
                                  random_state=42)

balanced_df_rules = pd.concat([undersampled_cluster_3, other_clusters_data])

new_cluster_counts = balanced_df_rules['cluster'].explode().value_counts()
print("New cluster distribution in consequents:")
print(new_cluster_counts)
```

```
Cluster distribution in consequents:
cluster
3    2912
2     918
0     792
1     396
Name: count, dtype: int64
New cluster distribution in consequents:
cluster
2    918
0    792
3    396
1    396
Name: count, dtype: int64
```

In [42]:
```python
df_encoded = pd.get_dummies(balanced_df_rules)

min_support_threshold_for_filtering = 0.05

item_support = df_encoded.sum() / len(df_encoded)

items_to_keep = item_support[item_support >= min_support_threshold_for_filtering].index
print(f"Original number of items: {df_encoded.shape[1]}")
print(f"Number of items after filtering: {len(items_to_keep)}")

df_filtered = df_encoded[items_to_keep]
```

```
Original number of items: 1494
Number of items after filtering: 106
```

In [43]:
```python
from mlxtend.frequent_patterns import fpgrowth

frequent_itemsets = fpgrowth(df_filtered, min_support=0.1, use_colnames=True, max_len=3)
```

In [44]:
```python
from mlxtend.frequent_patterns import association_rules

rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.6)
```

```
c:\Users\compt\AppData\Local\Programs\Python\Python310\lib\site-packages\mlxtend\frequent_patter
ns\association_rules.py:186: RuntimeWarning: invalid value encountered in divide
  cert_metric = np.where(certainty_denom == 0, 0, certainty_num / certainty_denom)
```

Consequent-Based Association Rule Summarization

In [45]:
```python
excluded_suffixes = ("absent", "Never")

def contains_excluded_suffix(itemset):
  if not itemset:
    return False
  for item in itemset:
    if isinstance(item, str) and item.endswith(excluded_suffixes):
      return True
  return False

rows_to_remove_mask = rules['antecedents'].apply(contains_excluded_suffix) | \
                      rules['consequents'].apply(contains_excluded_suffix)

rules_filtered = rules[~rows_to_remove_mask]

print(f"Original number of rules: {len(rules)}")
print(f"Number of rules after filtering suffixes {excluded_suffixes}: {len(rules_filtered)}")
```

```
Original number of rules: 185911
Number of rules after filtering suffixes ('absent', 'Never'): 19894
```

In [46]:
```python
unique_consequents = rules['consequents'].apply(
    lambda x: {item for item in x if "cluster" in str(item)}
).explode().dropna().unique()

print("Unique consequents containing 'cluster':")
for consequent in unique_consequents:
    print(consequent)
```

```
Unique consequents containing 'cluster':
cluster_2
cluster_0
cluster_1
```

In [ ]:
```python
rules_clusters = rules_filtered[rules_filtered['consequents'].apply(
    lambda x: len(x) == 1 and list(x)[0].startswith('cluster_')
)].copy()

rules_clusters = rules_clusters.reset_index(drop=True)

print(f"Number of rules after filtering exclusive cluster consequents: {len(rules_clusters)}")
```

```
Number of rules after filtering exclusive cluster consequents: 194
```

In [51]:
```python
rules_clusters['consequents'].value_counts()
```

Out[51]:
```
consequents
(cluster_2)    82
(cluster_1)    58
(cluster_0)    54
Name: count, dtype: int64
```

In [ ]:
```python
def is_single_cluster_consequent(x, cluster_name):
    return len(x) == 1 and cluster_name in x

rules_cluster_0 = rules_clusters[rules_clusters['consequents'].apply(
    lambda x: is_single_cluster_consequent(x, 'cluster_0')
)].copy()

rules_cluster_1 = rules_clusters[rules_clusters['consequents'].apply(
    lambda x: is_single_cluster_consequent(x, 'cluster_1')
)].copy()

rules_cluster_2 = rules_clusters[rules_clusters['consequents'].apply(
    lambda x: is_single_cluster_consequent(x, 'cluster_2')
)].copy()
```

Visualization

In [68]:
```python
from collections import Counter
import itertools

TOP_N_RULES_FOR_PLOTS = 50
TOP_M_FEATURES_FOR_BAR = 15
TOP_M_FEATURES_FOR_HEATMAP = 10
SORT_BY_METRIC = 'lift'

try:
    rules_cluster_0.head()
    rules_cluster_1.head()
    rules_cluster_2.head()
```

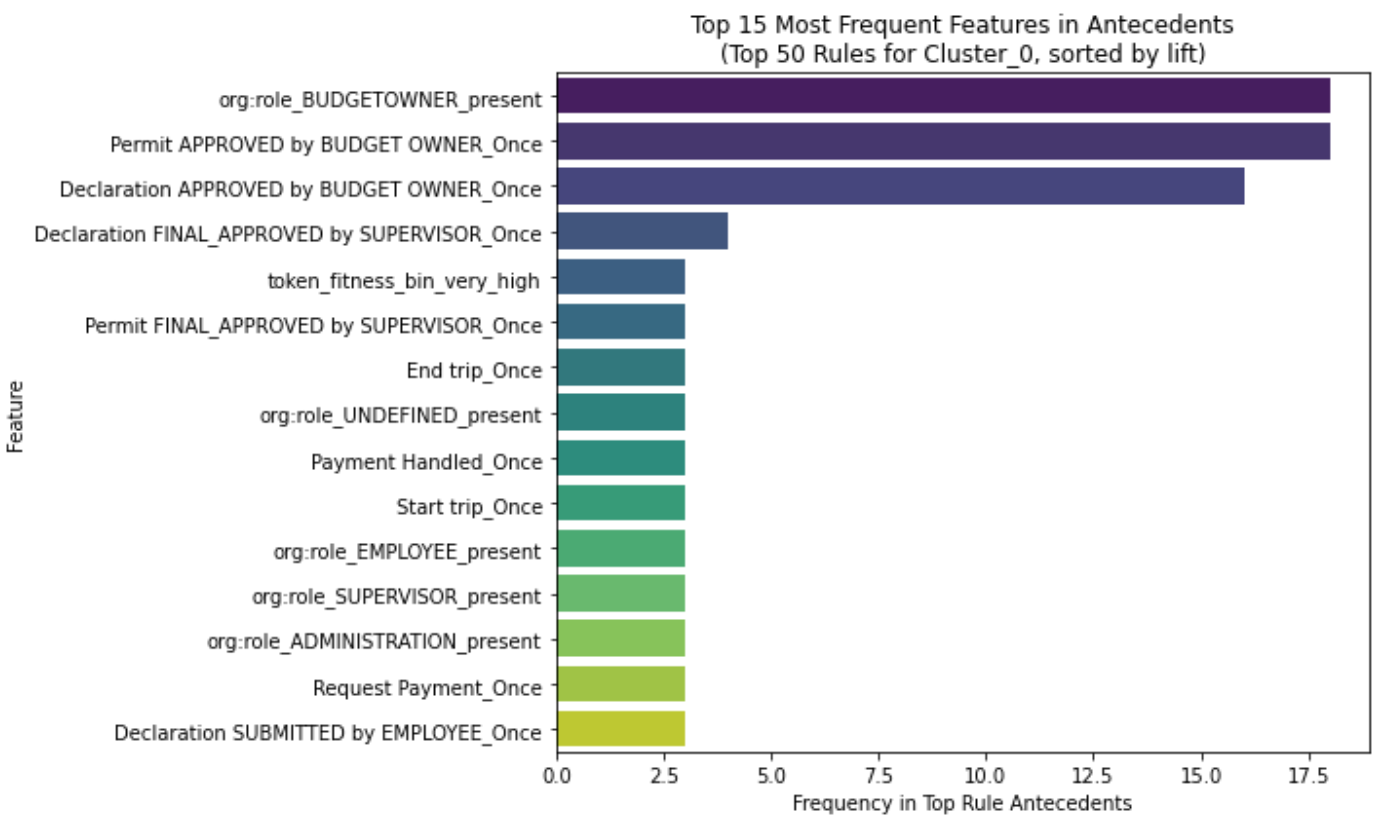
```
    print("Using existing rules DataFrames.")

    if 'cluster' not in rules_cluster_0.columns:
      rules_cluster_0 = rules_cluster_0.assign(cluster='Cluster_0')
    if 'cluster' not in rules_cluster_1.columns:
      rules_cluster_1 = rules_cluster_1.assign(cluster='Cluster_1')
    if 'cluster' not in rules_cluster_2.columns:
      rules_cluster_2 = rules_cluster_2.assign(cluster='Cluster_2')

except NameError:
    print("Creating dummy rules DataFrames for demonstration.")
    dummy_data = {
        'antecedents': [frozenset({f'feature_{i}', f'feature_{j}'}) for i in range(5) for j in r
        'consequents': [frozenset({'Cluster_X'})] * 10,
        'support': np.random.rand(10) * 0.1,
        'confidence': np.random.rand(10) * 0.4 + 0.6,
        'lift': np.random.rand(10) * 3 + 1,
    }
    rules_cluster_0 = pd.DataFrame(dummy_data)
    rules_cluster_0['consequents'] = rules_cluster_0['consequents'].apply(lambda x: frozenset({'
    rules_cluster_0['cluster'] = 'Cluster_0'

    rules_cluster_1 = pd.DataFrame(dummy_data)
    rules_cluster_1['antecedents'] = rules_cluster_1['antecedents'].apply(lambda s: frozenset({i
    rules_cluster_1['consequents'] = rules_cluster_1['consequents'].apply(lambda x: frozenset({'
    rules_cluster_1['confidence'] = np.random.rand(10) * 0.3 + 0.5
    rules_cluster_1['lift'] = np.random.rand(10) * 2 + 1.5
    rules_cluster_1['cluster'] = 'Cluster_1'

    rules_cluster_2 = pd.DataFrame(dummy_data)
    rules_cluster_2['antecedents'] = rules_cluster_2['antecedents'].apply(lambda s: frozenset({i
    rules_cluster_2['consequents'] = rules_cluster_2['consequents'].apply(lambda x: frozenset({'
    rules_cluster_2['support'] = np.random.rand(10) * 0.05
    rules_cluster_2['lift'] = np.random.rand(10) * 1 + 1
    rules_cluster_2['cluster'] = 'Cluster_2'


cluster_rules_dfs = {
    "Cluster_0": rules_cluster_0,
    "Cluster_1": rules_cluster_1,
    "Cluster_2": rules_cluster_2,
}

def plot_feature_frequency(rules_df, cluster_name, top_n_rules, top_m_features, sort_by):
    """Plots a bar chart of the most frequent features in rule antecedents."""
    if rules_df.empty:
        print(f"No rules for {cluster_name} to plot feature frequency.")
        return

    top_rules = rules_df.sort_values(by=sort_by, ascending=False).head(top_n_rules)
    if top_rules.empty:
        print(f"Not enough rules for {cluster_name} after sorting/selecting top {top_n_rules}.")
        return

    antecedent_items = list(itertools.chain.from_iterable(top_rules['antecedents']))
    item_counts = Counter(antecedent_items)

    if not item_counts:
        print(f"No antecedent items found in the top rules for {cluster_name}.")
        return

    features_df = pd.DataFrame(item_counts.items(), columns=['feature', 'count'])
    features_df = features_df.sort_values(by='count', ascending=False).head(top_m_features)
```

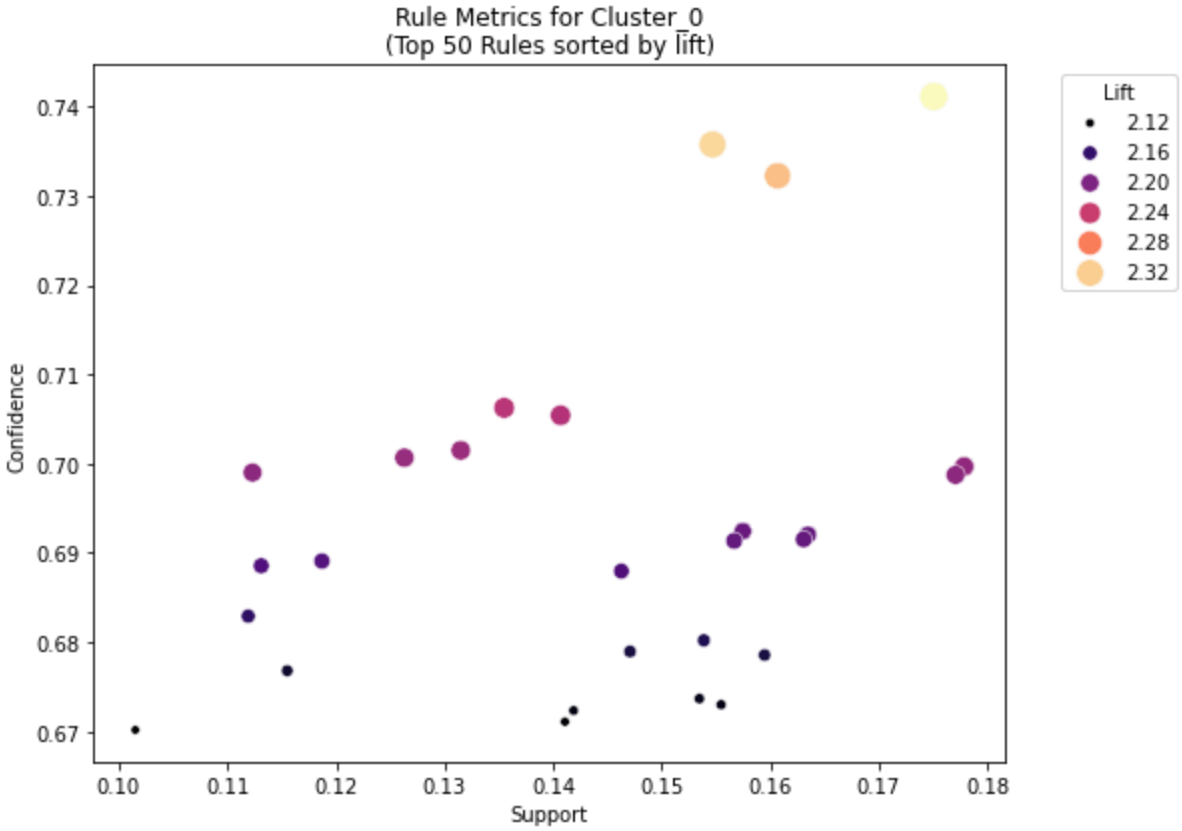
```
    plt.figure(figsize=(10, max(5, len(features_df)*0.4)))
    sns.barplot(x='count', y='feature', data=features_df, hue='feature', palette='viridis', lege
    plt.title(f'Top {len(features_df)} Most Frequent Features in Antecedents\n(Top {top_n_rules}
    plt.xlabel('Frequency in Top Rule Antecedents')
    plt.ylabel('Feature')
    plt.tight_layout()
    plt.show()

def plot_rule_metrics_scatter(rules_df, cluster_name, top_n_rules, sort_by):
    """Plots a scatter plot of rule metrics (Support vs Confidence, colored by Lift)."""
    if rules_df.empty:
        print(f"No rules for {cluster_name} to plot scatter metrics.")
        return

    top_rules = rules_df.sort_values(by=sort_by, ascending=False).head(top_n_rules)
    if top_rules.empty:
        print(f"Not enough rules for {cluster_name} after sorting/selecting top {top_n_rules}.")
        return

    plt.figure(figsize=(10, 6))
    scatter = sns.scatterplot(
        data=top_rules,
        x='support',
        y='confidence',
        hue='lift',
        size='lift',
        palette='magma',
        sizes=(20, 200),
        legend='auto'
    )
    plt.title(f'Rule Metrics for {cluster_name}\n(Top {len(top_rules)} Rules sorted by {sort_by}
    plt.xlabel('Support')
    plt.ylabel('Confidence')
    plt.legend(title='Lift', bbox_to_anchor=(1.05, 1), loc='upper left')
    plt.tight_layout(rect=[0, 0, 0.85, 1])
    plt.show()

def display_top_rules_table(rules_df, cluster_name, top_n_rules, sort_by):
    """Prints the top N rules in a formatted way."""
    if rules_df.empty:
        print(f"No rules for {cluster_name} to display.")
        return

    print(f"\n--- Top {top_n_rules} Rules for {cluster_name} (Sorted by {sort_by}) ---")
    top_rules = rules_df.sort_values(by=sort_by, ascending=False).head(top_n_rules)

    if top_rules.empty:
        print(f"Not enough rules found after sorting/selecting top {top_n_rules}.")
        return

    print(top_rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']].to_string()
    print("-" * (len(cluster_name) + 30))


def plot_metric_distributions(all_rules_list, cluster_names):
    if not all_rules_list:
        print("No rules data provided for distribution plotting.")
        return

    combined_rules = pd.concat(all_rules_list, ignore_index=True)

    if combined_rules.empty:
        print("Combined rules DataFrame is empty. Cannot plot distributions.")
        return
```

```
    plt.figure(figsize=(12, 5))

    plt.subplot(1, 2, 1)
    sns.boxplot(x='cluster', y='confidence', data=combined_rules, palette='Set2', order=cluster_
    plt.title('Confidence Distribution by Cluster')
    plt.xlabel('Cluster')
    plt.ylabel('Confidence')
    plt.xticks(rotation=15, ha='right')

    plt.subplot(1, 2, 2)
    sns.boxplot(x='cluster', y='lift', data=combined_rules, palette='Set2', order=cluster_names)
    plt.title('Lift Distribution by Cluster')
    plt.xlabel('Cluster')
    plt.ylabel('Lift')
    plt.xticks(rotation=15, ha='right')

    plt.suptitle('Comparison of Rule Metric Distributions')
    plt.tight_layout(rect=[0, 0, 1, 0.96])
    plt.show()

print("\nGenerating visualizations per cluster...")
for cluster_name, rules_df in cluster_rules_dfs.items():
    print(f"\n--- {cluster_name} ---")
    display_top_rules_table(rules_df, cluster_name, TOP_N_RULES_FOR_PLOTS, SORT_BY_METRIC)
    plot_feature_frequency(rules_df, cluster_name, TOP_N_RULES_FOR_PLOTS, TOP_M_FEATURES_FOR_BAR
    plot_rule_metrics_scatter(rules_df, cluster_name, TOP_N_RULES_FOR_PLOTS, SORT_BY_METRIC)
    plot_feature_cooccurrence_heatmap(rules_df, cluster_name, TOP_N_RULES_FOR_PLOTS, TOP_M_FEATU

print("\nGenerating comparison visualizations...")
all_rules_list = [df for df in cluster_rules_dfs.values() if not df.empty]
cluster_names_list = [name for name, df in cluster_rules_dfs.items() if not df.empty]

plot_metric_distributions(all_rules_list, cluster_names_list)
```

```
Using existing rules DataFrames.

Generating visualizations per cluster...

--- Cluster_0 ---

--- Top 50 Rules for Cluster_0 (Sorted by lift) ---
                                                                                         antecedents
consequents   support  confidence      lift
10                (Declaration FINAL_APPROVED by SUPERVISOR_Once, org:role_BUDGETOWNER_present)
(cluster_0)  0.175060    0.741117  2.341255
59        (Declaration FINAL_APPROVED by SUPERVISOR_Once, Permit APPROVED by BUDGET OWNER_Once)
(cluster_0)  0.154676    0.735741  2.324274
26  (Declaration FINAL_APPROVED by SUPERVISOR_Once, Declaration APPROVED by BUDGET OWNER_Once)
(cluster_0)  0.160671    0.732240  2.313214
28                     (token_fitness_bin_very_high, Declaration APPROVED by BUDGET OWNER_Once)
(cluster_0)  0.135492    0.706250  2.231108
12                                  (org:role_BUDGETOWNER_present, token_fitness_bin_very_high)
(cluster_0)  0.140687    0.705411  2.228457
13                     (Permit FINAL_APPROVED by SUPERVISOR_Once, org:role_BUDGETOWNER_present)
(cluster_0)  0.131495    0.701493  2.216079
65                          (Permit APPROVED by BUDGET OWNER_Once, token_fitness_bin_very_high)
(cluster_0)  0.126299    0.700665  2.213465
5                                                (org:role_BUDGETOWNER_present, End trip_Once)
(cluster_0)  0.177858    0.699686  2.210370
3                                    (org:role_BUDGETOWNER_present, org:role_UNDEFINED_present)
(cluster_0)  0.177858    0.699686  2.210370
6                                           (Payment Handled_Once, org:role_BUDGETOWNER_present)
(cluster_0)  0.177858    0.699686  2.210370
7                                                (Start trip_Once, org:role_BUDGETOWNER_present)
(cluster_0)  0.177858    0.699686  2.210370
4                                     (org:role_BUDGETOWNER_present, org:role_EMPLOYEE_present)
(cluster_0)  0.177858    0.699686  2.210370
0                                                                (org:role_BUDGETOWNER_present)
(cluster_0)  0.177858    0.699686  2.210370
2                                   (org:role_BUDGETOWNER_present, org:role_SUPERVISOR_present)
(cluster_0)  0.177858    0.699686  2.210370
8                               (org:role_ADMINISTRATION_present, org:role_BUDGETOWNER_present)
(cluster_0)  0.177858    0.699686  2.210370
34                    (Declaration FINAL_APPROVED by SUPERVISOR_Once, num_events_bin_very_high)
(cluster_0)  0.112310    0.699005  2.208220
9                                           (Request Payment_Once, org:role_BUDGETOWNER_present)
(cluster_0)  0.177058    0.698738  2.207377
51                           (Permit APPROVED by BUDGET OWNER_Once, org:role_UNDEFINED_present)
(cluster_0)  0.157474    0.692443  2.187490
48                                                        (Permit APPROVED by BUDGET OWNER_Once)
(cluster_0)  0.157474    0.692443  2.187490
50                          (Permit APPROVED by BUDGET OWNER_Once, org:role_SUPERVISOR_present)
(cluster_0)  0.157474    0.692443  2.187490
53                                         (Permit APPROVED by BUDGET OWNER_Once, End trip_Once)
(cluster_0)  0.157474    0.692443  2.187490
54                                 (Payment Handled_Once, Permit APPROVED by BUDGET OWNER_Once)
(cluster_0)  0.157474    0.692443  2.187490
52                            (Permit APPROVED by BUDGET OWNER_Once, org:role_EMPLOYEE_present)
(cluster_0)  0.157474    0.692443  2.187490
57                         (org:role_BUDGETOWNER_present, Permit APPROVED by BUDGET OWNER_Once)
(cluster_0)  0.157474    0.692443  2.187490
56                      (org:role_ADMINISTRATION_present, Permit APPROVED by BUDGET OWNER_Once)
(cluster_0)  0.157474    0.692443  2.187490
55                                       (Start trip_Once, Permit APPROVED by BUDGET OWNER_Once)
(cluster_0)  0.157474    0.692443  2.187490
23                 (org:role_ADMINISTRATION_present, Declaration APPROVED by BUDGET OWNER_Once)
(cluster_0)  0.163469    0.692047  2.186241
20                                  (End trip_Once, Declaration APPROVED by BUDGET OWNER_Once)
```

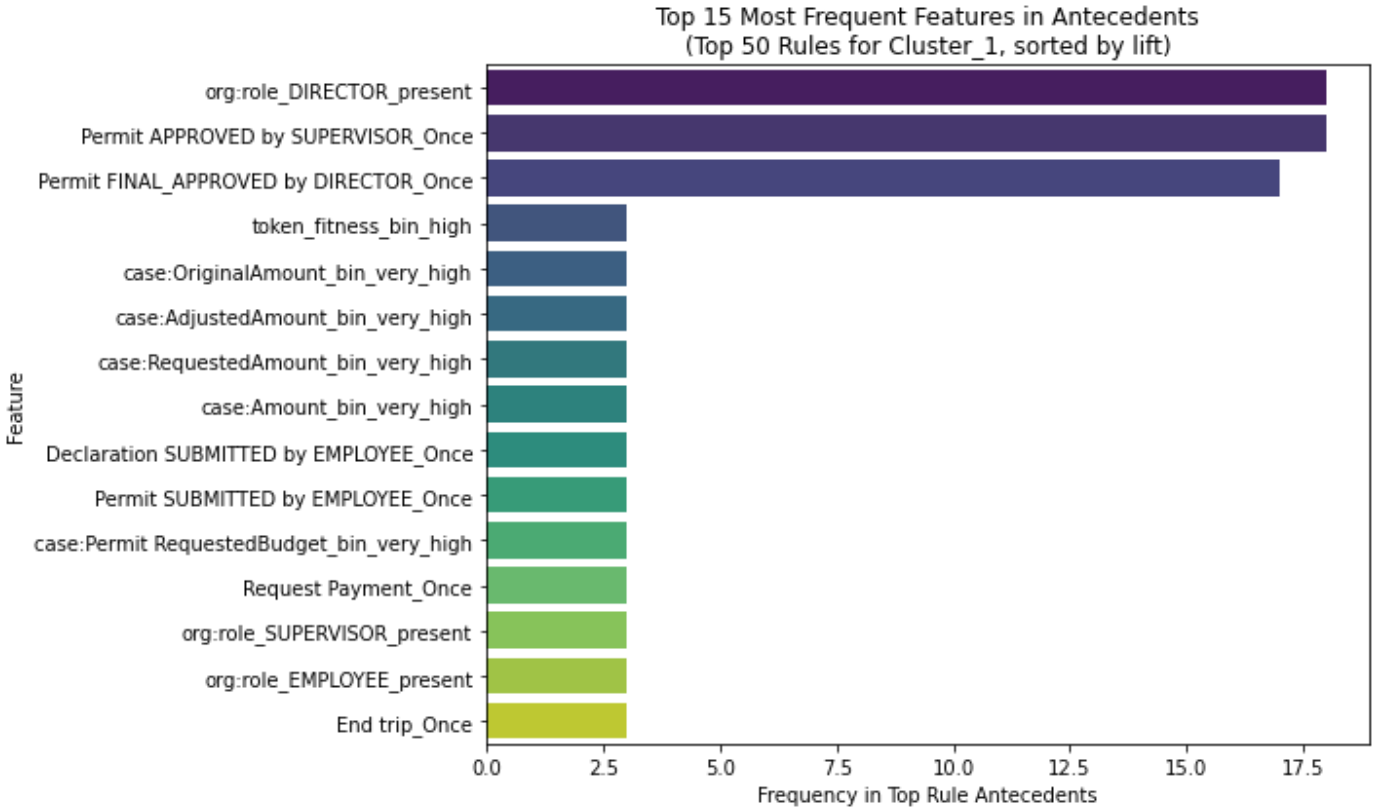
```
(cluster_0)  0.163469    0.692047  2.186241
16                                                (Declaration APPROVED by BUDGET OWNER_Once)
(cluster_0)  0.163469    0.692047  2.186241
21                           (Payment Handled_Once, Declaration APPROVED by BUDGET OWNER_Once)
(cluster_0)  0.163469    0.692047  2.186241
17                     (org:role_SUPERVISOR_present, Declaration APPROVED by BUDGET OWNER_Once)
(cluster_0)  0.163469    0.692047  2.186241
19                       (Declaration APPROVED by BUDGET OWNER_Once, org:role_EMPLOYEE_present)
(cluster_0)  0.163469    0.692047  2.186241
22                                (Start trip_Once, Declaration APPROVED by BUDGET OWNER_Once)
(cluster_0)  0.163469    0.692047  2.186241
24                    (org:role_BUDGETOWNER_present, Declaration APPROVED by BUDGET OWNER_Once)
(cluster_0)  0.163469    0.692047  2.186241
18                      (org:role_UNDEFINED_present, Declaration APPROVED by BUDGET OWNER_Once)
(cluster_0)  0.163469    0.692047  2.186241
25                           (Request Payment_Once, Declaration APPROVED by BUDGET OWNER_Once)
(cluster_0)  0.163070    0.691525  2.184592
58                                (Request Payment_Once, Permit APPROVED by BUDGET OWNER_Once)
(cluster_0)  0.156675    0.691358  2.184063
29       (Permit FINAL_APPROVED by SUPERVISOR_Once, Declaration APPROVED by BUDGET OWNER_Once)
(cluster_0)  0.118705    0.689095  2.176914
64            (Permit FINAL_APPROVED by SUPERVISOR_Once, Permit APPROVED by BUDGET OWNER_Once)
(cluster_0)  0.113110    0.688564  2.175238
61           (Permit APPROVED by BUDGET OWNER_Once, Declaration APPROVED by BUDGET OWNER_Once)
(cluster_0)  0.146283    0.687970  2.173360
30         (Declaration SUBMITTED by EMPLOYEE_Once, Declaration APPROVED by BUDGET OWNER_Once)
(cluster_0)  0.111910    0.682927  2.157428
27    (Declaration APPROVED by ADMINISTRATION_Once, Declaration APPROVED by BUDGET OWNER_Once)
(cluster_0)  0.153877    0.680212  2.148852
60              (Permit APPROVED by ADMINISTRATION_Once, Permit APPROVED by BUDGET OWNER_Once)
(cluster_0)  0.147082    0.678967  2.144918
11                 (org:role_BUDGETOWNER_present, Declaration APPROVED by ADMINISTRATION_Once)
(cluster_0)  0.159472    0.678571  2.143669
1                       (Declaration SUBMITTED by EMPLOYEE_Once, org:role_BUDGETOWNER_present)
(cluster_0)  0.115508    0.676815  2.138120
15                      (Permit APPROVED by ADMINISTRATION_Once, org:role_BUDGETOWNER_present)
(cluster_0)  0.153477    0.673684  2.128230
14                           (Permit SUBMITTED by EMPLOYEE_Once, org:role_BUDGETOWNER_present)
(cluster_0)  0.155476    0.673010  2.126101
62                    (Permit SUBMITTED by EMPLOYEE_Once, Permit APPROVED by BUDGET OWNER_Once)
(cluster_0)  0.141886    0.672348  2.124010
63         (Permit APPROVED by BUDGET OWNER_Once, Declaration APPROVED by ADMINISTRATION_Once)
(cluster_0)  0.141087    0.671103  2.120074
49              (Declaration SUBMITTED by EMPLOYEE_Once, Permit APPROVED by BUDGET OWNER_Once)
(cluster_0)  0.101519    0.670185  2.117174
--------------------------------------
```

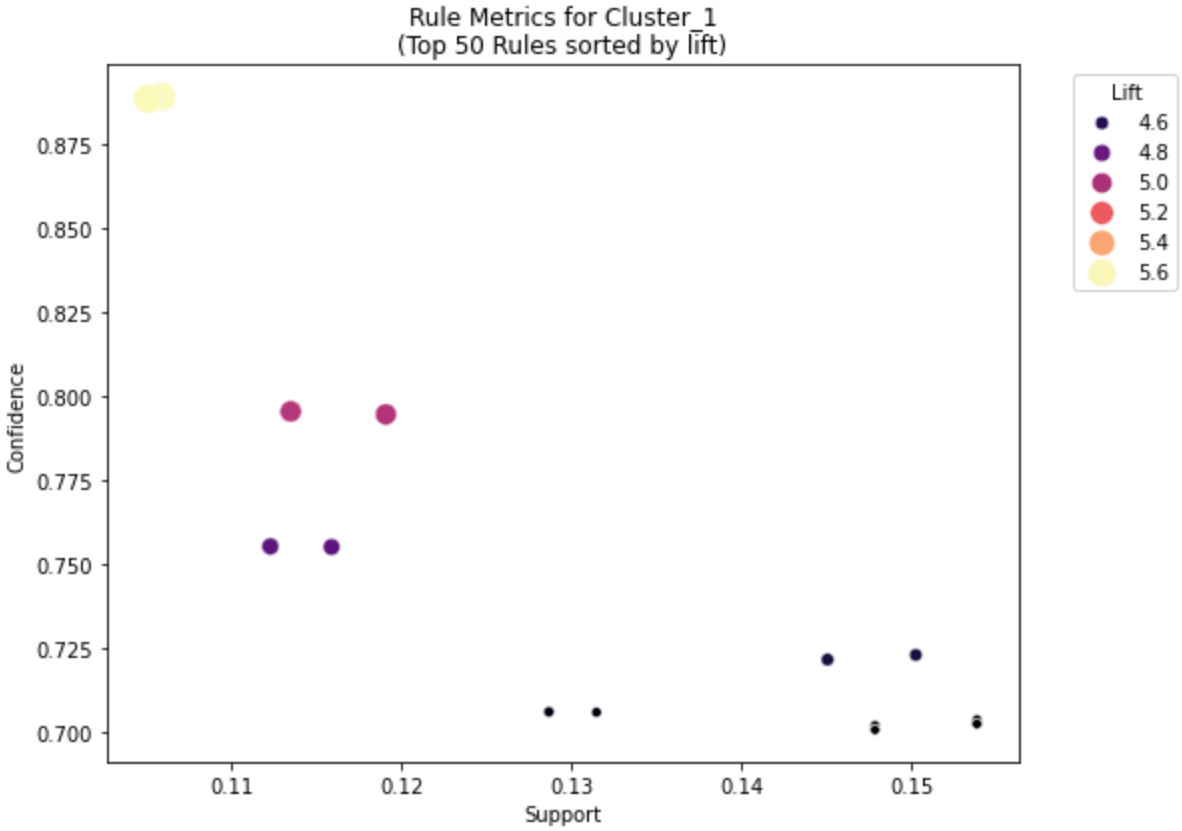

Top 15 Most Frequent Features in Antecedents
(Top 50 Rules for Cluster_0, sorted by lift)
org:role_BUDGETOWNER_present
Permit APPROVED by BUDGET OWNER_Once
Declaration APPROVED by BUDGET OWNER_Once
Declaration FINAL_APPROVED by SUPERVISOR_Once
token_fitness_bin_very_high
Permit FINAL_APPROVED by SUPERVISOR_Once
End trip_Once
org:role_UNDEFINED_present
Payment Handled_Once
Start trip_Once
org:role_EMPLOYEE_present
org:role_SUPERVISOR_present
org:role_ADMINISTRATION_present
Request Payment_Once
Declaration SUBMITTED by EMPLOYEE_Once
Feature
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Frequency in Top Rule Antecedents


Rule Metrics for Cluster_0
(Top 50 Rules sorted by lift)
Confidence
0.74
0.73
0.72
0.71
0.70
0.69
0.68
0.67
0.10
0.11
0.12
0.13
0.14
0.15
0.16
0.17
0.18
Support
Lift
2.12
2.16
2.20
2.24
2.28
2.32

```
--- Cluster_1 ---

--- Top 50 Rules for Cluster_1 (Sorted by lift) ---
                                                                  antecedents  conseq
uents   support  confidence      lift
126                               (org:role_DIRECTOR_present, token_fitness_bin_high)  (clust
er_1)  0.105915    0.889262  5.618517
127                  (Permit FINAL_APPROVED by DIRECTOR_Once, token_fitness_bin_high)  (clust
er_1)  0.105116    0.888514  5.613790
128                      (Permit APPROVED by SUPERVISOR_Once, token_fitness_bin_high)  (clust
er_1)  0.105116    0.888514  5.613790
113           (Permit APPROVED by SUPERVISOR_Once, case:OriginalAmount_bin_very_high)  (clust
er_1)  0.113509    0.795518  5.026229
109       (Permit FINAL_APPROVED by DIRECTOR_Once, case:AdjustedAmount_bin_very_high)  (clust
er_1)  0.113509    0.795518  5.026229
107          (Permit APPROVED by SUPERVISOR_Once, case:RequestedAmount_bin_very_high)  (clust
er_1)  0.113509    0.795518  5.026229
110           (Permit APPROVED by SUPERVISOR_Once, case:AdjustedAmount_bin_very_high)  (clust
er_1)  0.113509    0.795518  5.026229
112       (Permit FINAL_APPROVED by DIRECTOR_Once, case:OriginalAmount_bin_very_high)  (clust
er_1)  0.113509    0.795518  5.026229
106      (Permit FINAL_APPROVED by DIRECTOR_Once, case:RequestedAmount_bin_very_high)  (clust
er_1)  0.113509    0.795518  5.026229
115                 (case:Amount_bin_very_high, Permit FINAL_APPROVED by DIRECTOR_Once)  (clust
er_1)  0.113509    0.795518  5.026229
116                     (case:Amount_bin_very_high, Permit APPROVED by SUPERVISOR_Once)  (clust
er_1)  0.113509    0.795518  5.026229
114                            (case:Amount_bin_very_high, org:role_DIRECTOR_present)  (clust
er_1)  0.119105    0.794667  5.020848
108                      (org:role_DIRECTOR_present, case:AdjustedAmount_bin_very_high)  (clust
er_1)  0.119105    0.794667  5.020848
111                      (org:role_DIRECTOR_present, case:OriginalAmount_bin_very_high)  (clust
er_1)  0.119105    0.794667  5.020848
105                     (org:role_DIRECTOR_present, case:RequestedAmount_bin_very_high)  (clust
er_1)  0.119105    0.794667  5.020848
119      (Permit APPROVED by SUPERVISOR_Once, Declaration SUBMITTED by EMPLOYEE_Once)  (clust
er_1)  0.112310    0.755376  4.772605
118  (Declaration SUBMITTED by EMPLOYEE_Once, Permit FINAL_APPROVED by DIRECTOR_Once)  (clust
er_1)  0.112310    0.755376  4.772605
117               (org:role_DIRECTOR_present, Declaration SUBMITTED by EMPLOYEE_Once)  (clust
er_1)  0.115907    0.755208  4.771544
82                     (org:role_DIRECTOR_present, Permit SUBMITTED by EMPLOYEE_Once)  (clust
er_1)  0.150280    0.723077  4.568531
101       (Permit FINAL_APPROVED by DIRECTOR_Once, Permit SUBMITTED by EMPLOYEE_Once)  (clust
er_1)  0.145084    0.721670  4.559642
91            (Permit APPROVED by SUPERVISOR_Once, Permit SUBMITTED by EMPLOYEE_Once)  (clust
er_1)  0.145084    0.721670  4.559642
104    (Permit APPROVED by SUPERVISOR_Once, case:Permit RequestedBudget_bin_very_high)  (clust
er_1)  0.128697    0.706140  4.461523
103  (Permit FINAL_APPROVED by DIRECTOR_Once, case:Permit RequestedBudget_bin_very_high)  (clust
er_1)  0.128697    0.706140  4.461523
102            (org:role_DIRECTOR_present, case:Permit RequestedBudget_bin_very_high)  (clust
er_1)  0.131495    0.706009  4.460691
81                              (org:role_DIRECTOR_present, Request Payment_Once)  (clust
er_1)  0.153877    0.703839  4.446984
75                        (org:role_DIRECTOR_present, org:role_SUPERVISOR_present)  (clust
er_1)  0.153877    0.702555  4.438869
72                                                     (org:role_DIRECTOR_present)  (clust
er_1)  0.153877    0.702555  4.438869
77                          (org:role_DIRECTOR_present, org:role_EMPLOYEE_present)  (clust
er_1)  0.153877    0.702555  4.438869
78                                     (org:role_DIRECTOR_present, End trip_Once)  (clust
er_1)  0.153877    0.702555  4.438869
80                                   (org:role_DIRECTOR_present, Start trip_Once)  (clust
```

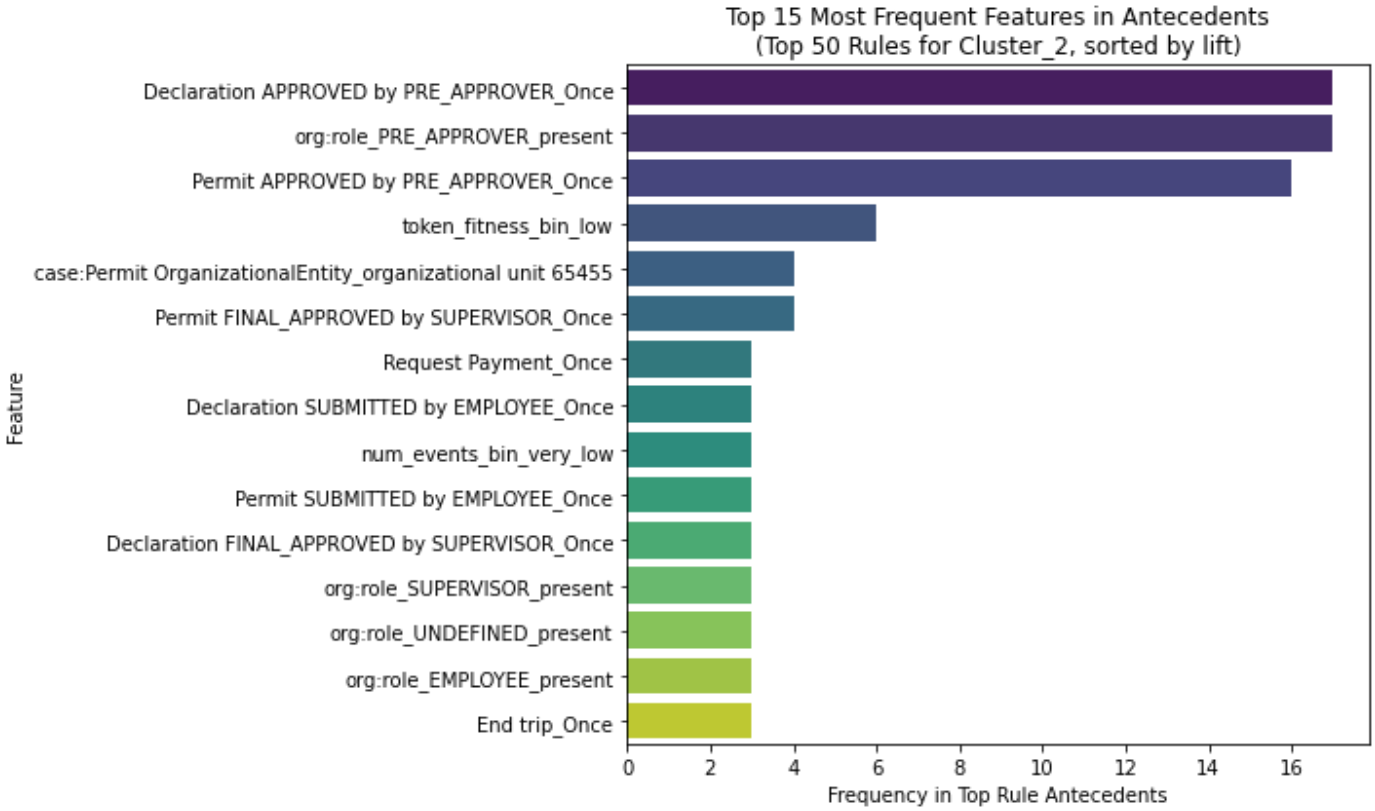
```
er_1)  0.153877    0.702555  4.438869
76                              (org:role_DIRECTOR_present, org:role_UNDEFINED_present)  (clust
er_1)  0.153877    0.702555  4.438869
79                                    (org:role_DIRECTOR_present, Payment Handled_Once)  (clust
er_1)  0.153877    0.702555  4.438869
100                      (Request Payment_Once, Permit FINAL_APPROVED by DIRECTOR_Once)  (clust
er_1)  0.147882    0.702087  4.435915
90                              (Permit APPROVED by SUPERVISOR_Once, Request Payment_Once)  (clust
er_1)  0.147882    0.702087  4.435915
85                         (Permit APPROVED by SUPERVISOR_Once, org:role_UNDEFINED_present)  (clust
er_1)  0.147882    0.700758  4.427514
84                        (Permit APPROVED by SUPERVISOR_Once, org:role_SUPERVISOR_present)  (clust
er_1)  0.147882    0.700758  4.427514
87                                     (Permit APPROVED by SUPERVISOR_Once, End trip_Once)  (clust
er_1)  0.147882    0.700758  4.427514
86                           (Permit APPROVED by SUPERVISOR_Once, org:role_EMPLOYEE_present)  (clust
er_1)  0.147882    0.700758  4.427514
83                           (Permit APPROVED by SUPERVISOR_Once, org:role_DIRECTOR_present)  (clust
er_1)  0.147882    0.700758  4.427514
74                                                (Permit FINAL_APPROVED by DIRECTOR_Once)  (clust
er_1)  0.147882    0.700758  4.427514
73                                                    (Permit APPROVED by SUPERVISOR_Once)  (clust
er_1)  0.147882    0.700758  4.427514
89                                    (Permit APPROVED by SUPERVISOR_Once, Start trip_Once)  (clust
er_1)  0.147882    0.700758  4.427514
97                                    (Permit FINAL_APPROVED by DIRECTOR_Once, End trip_Once)  (clust
er_1)  0.147882    0.700758  4.427514
96                           (Permit FINAL_APPROVED by DIRECTOR_Once, org:role_EMPLOYEE_present)  (clust
er_1)  0.147882    0.700758  4.427514
95                          (Permit FINAL_APPROVED by DIRECTOR_Once, org:role_UNDEFINED_present)  (clust
er_1)  0.147882    0.700758  4.427514
94                         (Permit FINAL_APPROVED by DIRECTOR_Once, org:role_SUPERVISOR_present)  (clust
er_1)  0.147882    0.700758  4.427514
93                           (org:role_DIRECTOR_present, Permit FINAL_APPROVED by DIRECTOR_Once)  (clust
er_1)  0.147882    0.700758  4.427514
92                 (Permit APPROVED by SUPERVISOR_Once, Permit FINAL_APPROVED by DIRECTOR_Once)  (clust
er_1)  0.147882    0.700758  4.427514
88                              (Permit APPROVED by SUPERVISOR_Once, Payment Handled_Once)  (clust
er_1)  0.147882    0.700758  4.427514
99                                (Start trip_Once, Permit FINAL_APPROVED by DIRECTOR_Once)  (clust
er_1)  0.147882    0.700758  4.427514
---------------------------------------
```

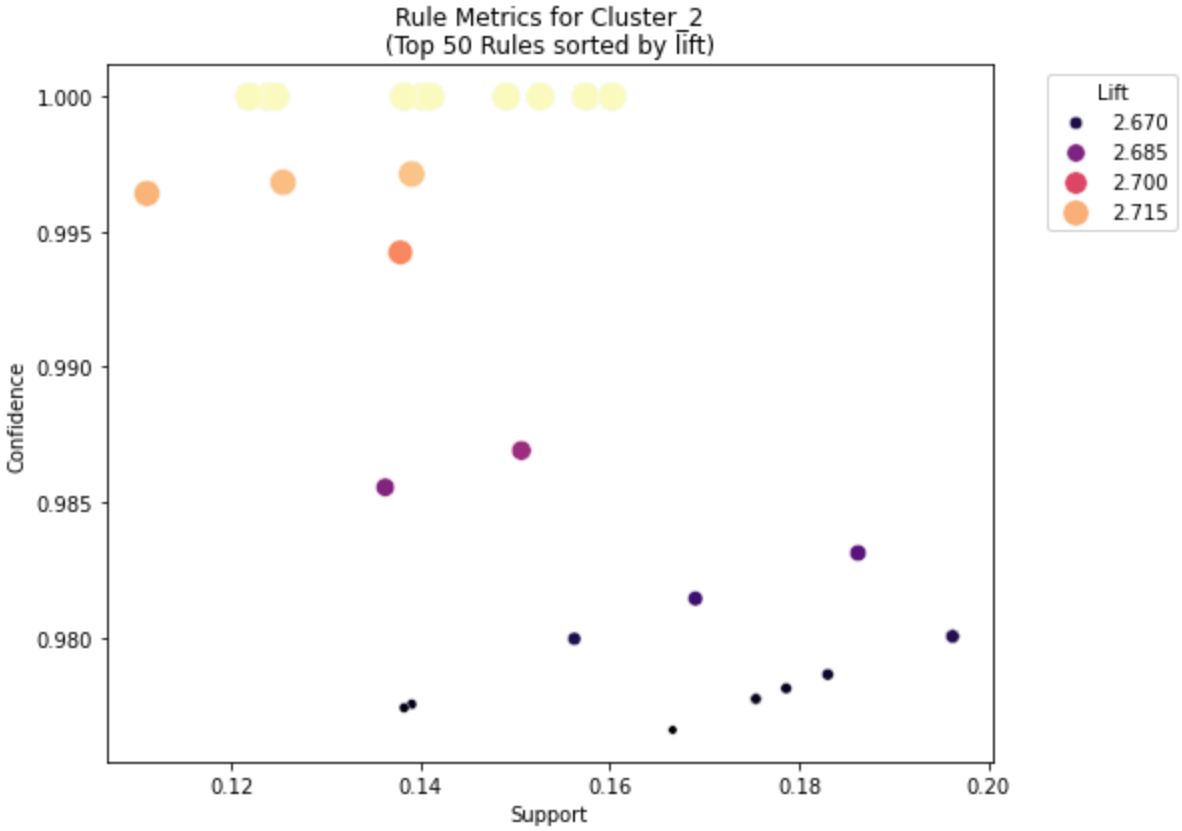

Top 15 Most Frequent Features in Antecedents
(Top 50 Rules for Cluster_1, sorted by lift)
org:role_DIRECTOR_present
Permit APPROVED by SUPERVISOR_Once
Permit FINAL_APPROVED by DIRECTOR_Once
token_fitness_bin_high
case:OriginalAmount_bin_very_high
case:AdjustedAmount_bin_very_high
case:RequestedAmount_bin_very_high
case:Amount_bin_very_high
Declaration SUBMITTED by EMPLOYEE_Once
Permit SUBMITTED by EMPLOYEE_Once
case:Permit RequestedBudget_bin_very_high
Request Payment_Once
org:role_SUPERVISOR_present
org:role_EMPLOYEE_present
End trip_Once
Feature
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Frequency in Top Rule Antecedents


Rule Metrics for Cluster_1
(Top 50 Rules sorted by lift)
Lift
4.6
4.8
5.0
5.2
5.4
5.6
Confidence
0.875
0.850
0.825
0.800
0.775
0.750
0.725
0.700
0.11
0.12
0.13
0.14
0.15
Support

```
--- Cluster_2 ---

--- Top 50 Rules for Cluster_2 (Sorted by lift) ---

antecedents  consequents   support  confidence      lift
179                                        (Request Payment_Once, Declaration APPROVED by PRE_AP
PROVER_Once)  (cluster_2)  0.160272    1.000000  2.725490
178                                (org:role_PRE_APPROVER_present, Declaration APPROVED by PRE_AP
PROVER_Once)  (cluster_2)  0.160272    1.000000  2.725490
193                                   (token_fitness_bin_low, Declaration APPROVED by PRE_AP
PROVER_Once)  (cluster_2)  0.138289    1.000000  2.725490
192  (case:Permit OrganizationalEntity_organizational unit 65455, Declaration APPROVED by PRE_AP
PROVER_Once)  (cluster_2)  0.123901    1.000000  2.725490
191                                    (Declaration APPROVED by PRE_APPROVER_Once, num_events_b
in_very_low)  (cluster_2)  0.124700    1.000000  2.725490
190                       (Declaration SUBMITTED by EMPLOYEE_Once, Declaration APPROVED by PRE_AP
PROVER_Once)  (cluster_2)  0.140288    1.000000  2.725490
189                         (Permit APPROVED by PRE_APPROVER_Once, Declaration APPROVED by PRE_AP
PROVER_Once)  (cluster_2)  0.141087    1.000000  2.725490
188                            (Permit SUBMITTED by EMPLOYEE_Once, Declaration APPROVED by PRE_AP
PROVER_Once)  (cluster_2)  0.149081    1.000000  2.725490
187                     (Permit FINAL_APPROVED by SUPERVISOR_Once, Declaration APPROVED by PRE_AP
PROVER_Once)  (cluster_2)  0.152678    1.000000  2.725490
186                 (Declaration FINAL_APPROVED by SUPERVISOR_Once, Declaration APPROVED by PRE_AP
PROVER_Once)  (cluster_2)  0.157474    1.000000  2.725490
185                                  (Declaration APPROVED by PRE_APPROVER_Once, org:role_SUPERVI
SOR_present)  (cluster_2)  0.160272    1.000000  2.725490
184                                   (Declaration APPROVED by PRE_APPROVER_Once, org:role_UNDEFI
NED_present)  (cluster_2)  0.160272    1.000000  2.725490
183                                    (Declaration APPROVED by PRE_APPROVER_Once, org:role_EMPLO
YEE_present)  (cluster_2)  0.160272    1.000000  2.725490
182                                              (Declaration APPROVED by PRE_APPROVER_Once, En
d trip_Once)  (cluster_2)  0.160272    1.000000  2.725490
181                                          (Payment Handled_Once, Declaration APPROVED by PRE_AP
PROVER_Once)  (cluster_2)  0.160272    1.000000  2.725490
180                                               (Start trip_Once, Declaration APPROVED by PRE_AP
PROVER_Once)  (cluster_2)  0.160272    1.000000  2.725490
144                                            (token_fitness_bin_low, Permit APPROVED by PRE_AP
PROVER_Once)  (cluster_2)  0.121902    1.000000  2.725490
143                                                 (token_fitness_bin_low, org:role_PRE_APPRO
VER_present)  (cluster_2)  0.138289    1.000000  2.725490
177                                                                (Declaration APPROVED by PRE_AP
PROVER_Once)  (cluster_2)  0.160272    1.000000  2.725490
161                                                   (org:role_PRE_APPROVER_present, num_events_b
in_very_low)  (cluster_2)  0.139089    0.997135  2.717681
176                                         (Permit APPROVED by PRE_APPROVER_Once, num_events_b
in_very_low)  (cluster_2)  0.125500    0.996825  2.716838
145                       (case:Permit OrganizationalEntity_organizational unit 65455, token_fitn
ess_bin_low)  (cluster_2)  0.111111    0.996416  2.715721
142                                          (token_fitness_bin_low, Permit FINAL_APPROVED by SUPE
RVISOR_Once)  (cluster_2)  0.137890    0.994236  2.709781
160              (case:Permit OrganizationalEntity_organizational unit 65455, org:role_PRE_APPRO
VER_present)  (cluster_2)  0.150679    0.986911  2.689816
175       (case:Permit OrganizationalEntity_organizational unit 65455, Permit APPROVED by PRE_AP
PROVER_Once)  (cluster_2)  0.136291    0.985549  2.686104
156                                (org:role_PRE_APPROVER_present, Permit FINAL_APPROVED by SUPE
RVISOR_Once)  (cluster_2)  0.186251    0.983122  2.679490
171                          (Permit FINAL_APPROVED by SUPERVISOR_Once, Permit APPROVED by PRE_AP
PROVER_Once)  (cluster_2)  0.169065    0.981439  2.674901
155                                                    (org:role_PRE_APPROVER_present, Request P
ayment_Once)  (cluster_2)  0.196243    0.980040  2.671089
153                                                    (org:role_PRE_APPROVER_present, Payment H
andled_Once)  (cluster_2)  0.196243    0.980040  2.671089
152                                                                 (org:role_PRE_APPROVER_present, En
```

d trip_Once) (cluster_2) 0.196243   0.980040 2.671089
150   (org:role_PRE_APPROVER_present, org:role_UNDEFI
NED_present) (cluster_2) 0.196243   0.980040 2.671089
151   (org:role_PRE_APPROVER_present, org:role_EMPLO
YEE_present) (cluster_2) 0.196243   0.980040 2.671089
148   (org:role_PRE_APPRO
VER_present) (cluster_2) 0.196243   0.980040 2.671089
149   (org:role_PRE_APPROVER_present, org:role_SUPERVI
SOR_present) (cluster_2) 0.196243   0.980040 2.671089
154   (org:role_PRE_APPROVER_present, Star
t trip_Once) (cluster_2) 0.196243   0.980040 2.671089
159   (org:role_PRE_APPROVER_present, Declaration SUBMITTED by EM
PLOYEE_Once) (cluster_2) 0.156275   0.979950 2.670844
157   (org:role_PRE_APPROVER_present, Permit SUBMITTED by EM
PLOYEE_Once) (cluster_2) 0.183054   0.978632 2.667253
158   (org:role_PRE_APPROVER_present, Declaration FINAL_APPROVED by SUPE
RVISOR_Once) (cluster_2) 0.178657   0.978118 2.665851
166   (Permit APPROVED by PRE_APPROVER_Once, En
d trip_Once) (cluster_2) 0.175460   0.977728 2.664789
167   (Payment Handled_Once, Permit APPROVED by PRE_AP
PROVER_Once) (cluster_2) 0.175460   0.977728 2.664789
165   (Permit APPROVED by PRE_APPROVER_Once, org:role_EMPLO
YEE_present) (cluster_2) 0.175460   0.977728 2.664789
164   (Permit APPROVED by PRE_APPROVER_Once, org:role_UNDEFI
NED_present) (cluster_2) 0.175460   0.977728 2.664789
169   (Permit APPROVED by PRE_APPROVER_Once, Request P
ayment_Once) (cluster_2) 0.175460   0.977728 2.664789
170   (org:role_PRE_APPROVER_present, Permit APPROVED by PRE_AP
PROVER_Once) (cluster_2) 0.175460   0.977728 2.664789
168   (Start trip_Once, Permit APPROVED by PRE_AP
PROVER_Once) (cluster_2) 0.175460   0.977728 2.664789
162   (Permit APPROVED by PRE_AP
PROVER_Once) (cluster_2) 0.175460   0.977728 2.664789
163   (Permit APPROVED by PRE_APPROVER_Once, org:role_SUPERVI
SOR_present) (cluster_2) 0.175460   0.977728 2.664789
174   (Declaration SUBMITTED by EMPLOYEE_Once, Permit APPROVED by PRE_AP
PROVER_Once) (cluster_2) 0.139089   0.977528 2.664243
139   (token_fitness_bin_low, Declaration FINAL_APPROVED by SUPE
RVISOR_Once) (cluster_2) 0.138289   0.977401 2.663897
172   (Permit APPROVED by PRE_APPROVER_Once, Permit SUBMITTED by EM
PLOYEE_Once) (cluster_2) 0.166667   0.976581 2.661661






```
Generating comparison visualizations...
```

```
C:\Users\compt\AppData\Local\Temp\ipykernel_7976\2282116101.py:145: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign t
he `x` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x='cluster', y='confidence', data=combined_rules, palette='Set2', order=cluster_na
mes)
C:\Users\compt\AppData\Local\Temp\ipykernel_7976\2282116101.py:152: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign t
he `x` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x='cluster', y='lift', data=combined_rules, palette='Set2', order=cluster_names)
```