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
```

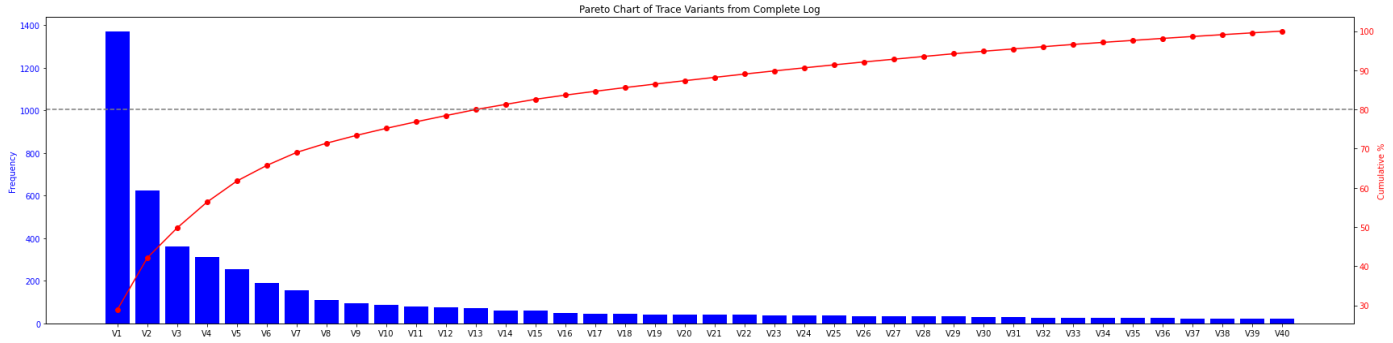
In [5]:
```
variants_complete_log = get_n_print_variants(complete_log_df, log_name="Complete Log")
```

```
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)
```

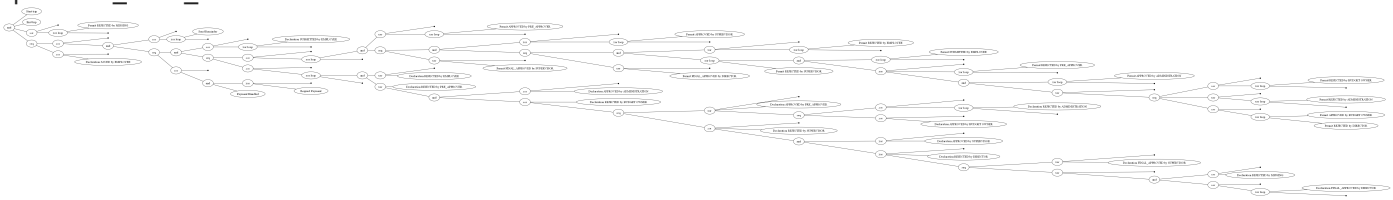
variant_id	variant_str	count	percent
V738	('Permit SUBMITTED by EMPLOYEE', 'Permit APPROVED by SUPERVISOR', 'Permit FINAL_APPROVED by DIRECTOR', 'Permit REJECTED by MISSING', 'Permit SUBMITTED by EMPLOYEE', 'Permit FINAL_APPROVED by SUPERVISOR', 'Start trip', 'End trip', 'Declaration SUBMITTED by EMPLOYEE', '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

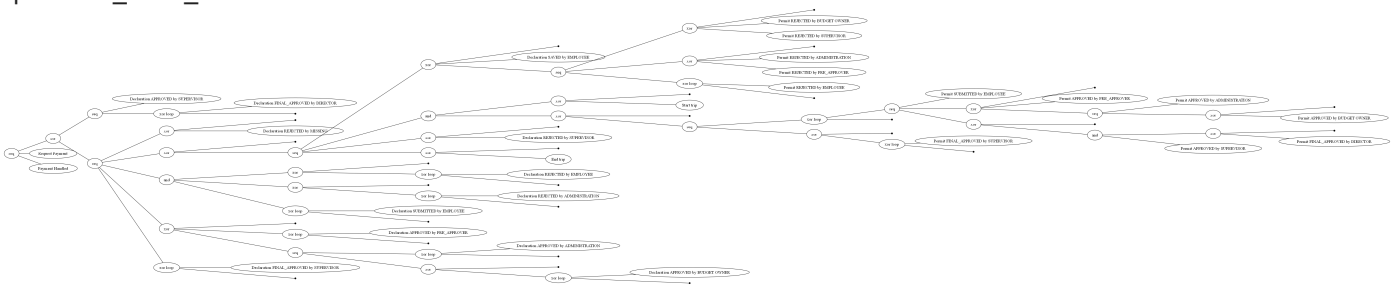
In [11]:
```python
def discover_multiple_process_trees(event_log, thresholds, show=False):

    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,
```

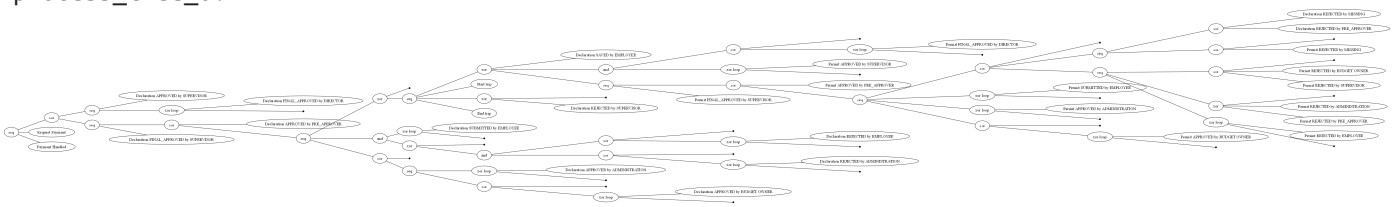
process_tree_0.0



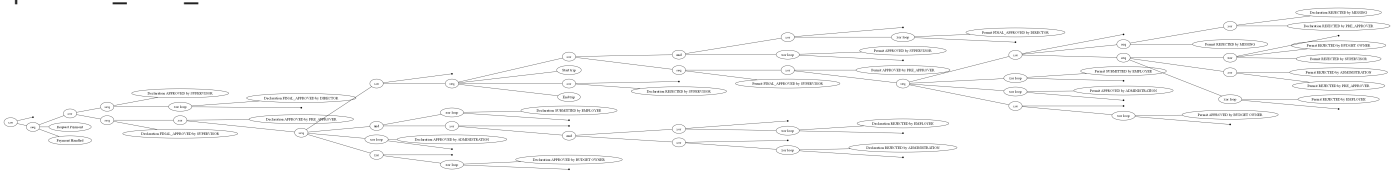
process_tree_0.1



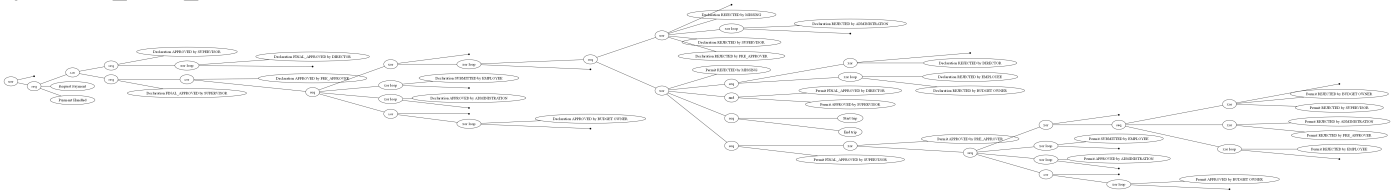
process_tree_0.2



process_tree_0.3



process_tree_0.4

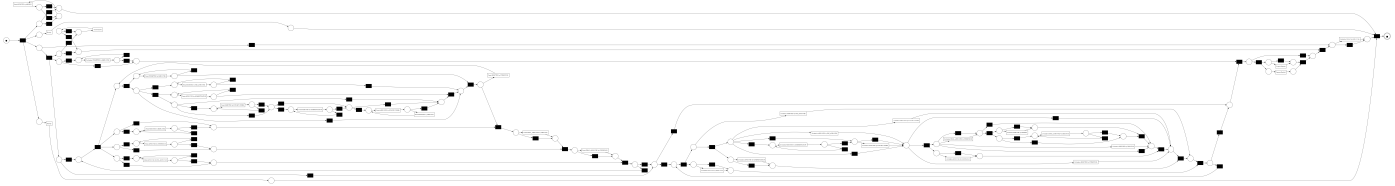


Process Tree Convertion to Petri Net

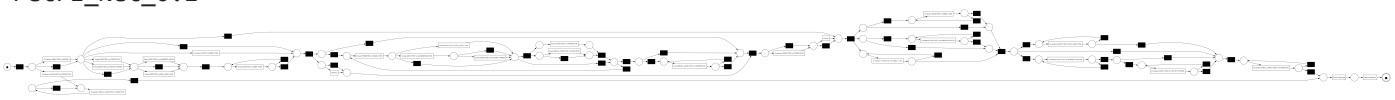
In [13]:
```
def convert_multiple_Process_Trees_to_Petri_Nets(process_trees, show=False):

    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=
```

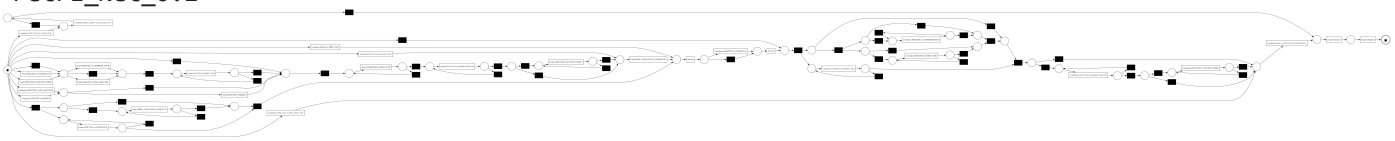
Petri_Net_0.0



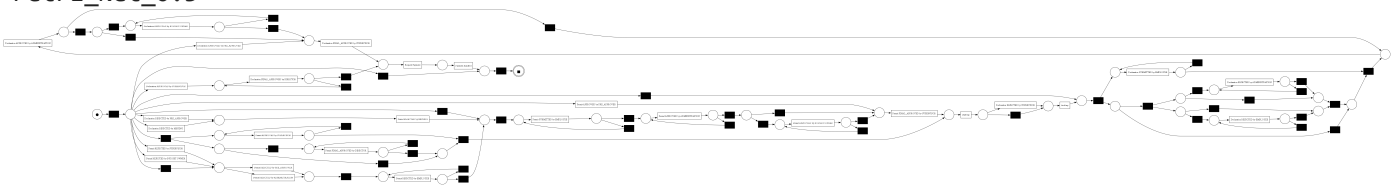
Petri_Net_0.1



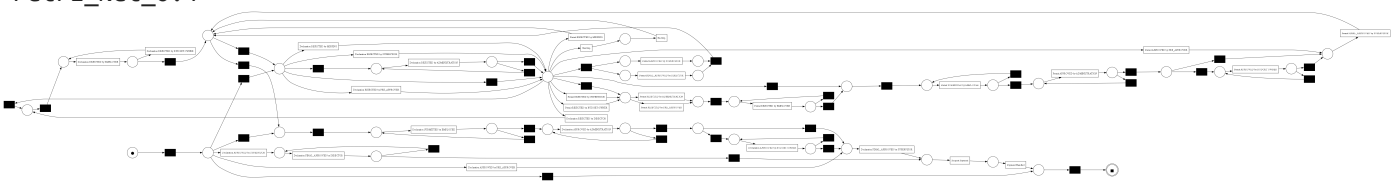
Petri_Net_0.2



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):
```

```
        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 [ ]:

```
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.

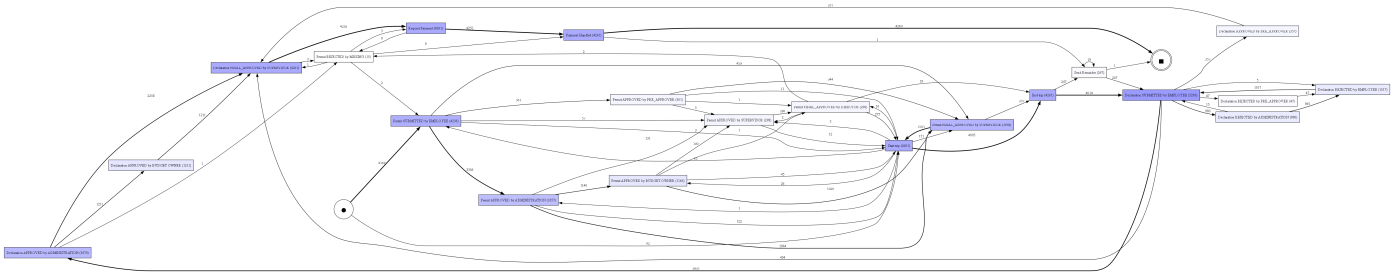
Direct Follow Graphs

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
```

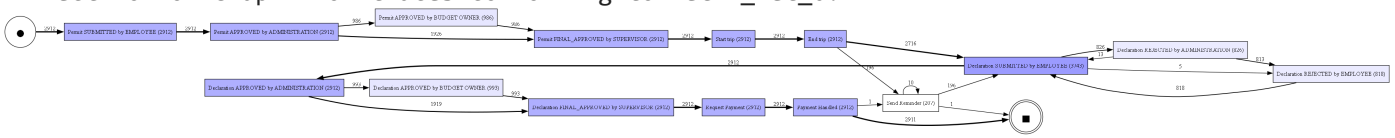
In [24]:
```python
conforming_log_01_df = extract_confirming_log_from_Petri_Net(complete_log_df, complete_Petri_Net
dfg_01, start_activities_01, end_activities_01 = pm4py.discover_dfg(conforming_log_01_df, case_i

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')
```

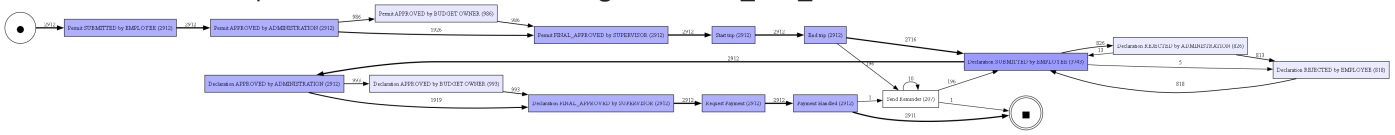
```
replaying log with TBR, completed traces ::   0%|          | 0/753 [00:00<?, ?it/s]
Number of conforming traces: 4261
replaying log with TBR, completed traces ::   0%|          | 0/753 [00:00<?, ?it/s]
Number of conforming traces: 2912
replaying log with TBR, completed traces ::   0%|          | 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
```



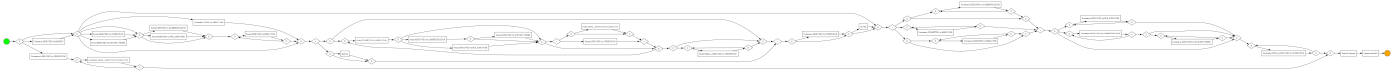
```
Direct Follow Graph from traces conforming to Petri_Net_0.3
```



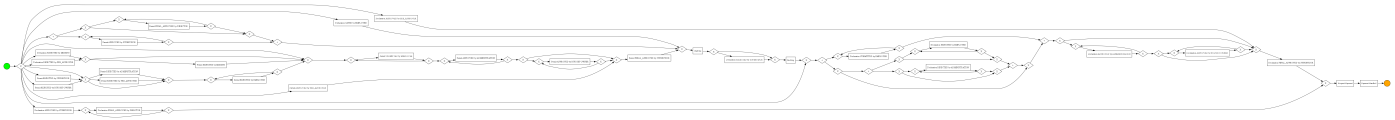
BPMN Graphs

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')
```

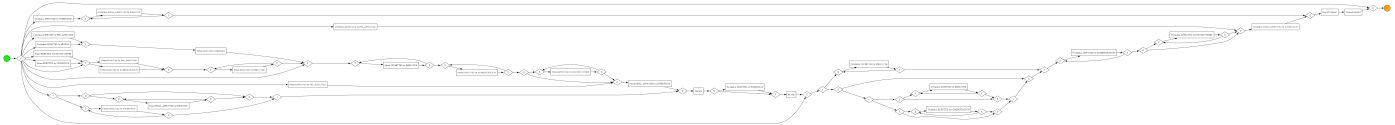
```
BPNM from Petri_Net_0.1
```



```
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}")
```

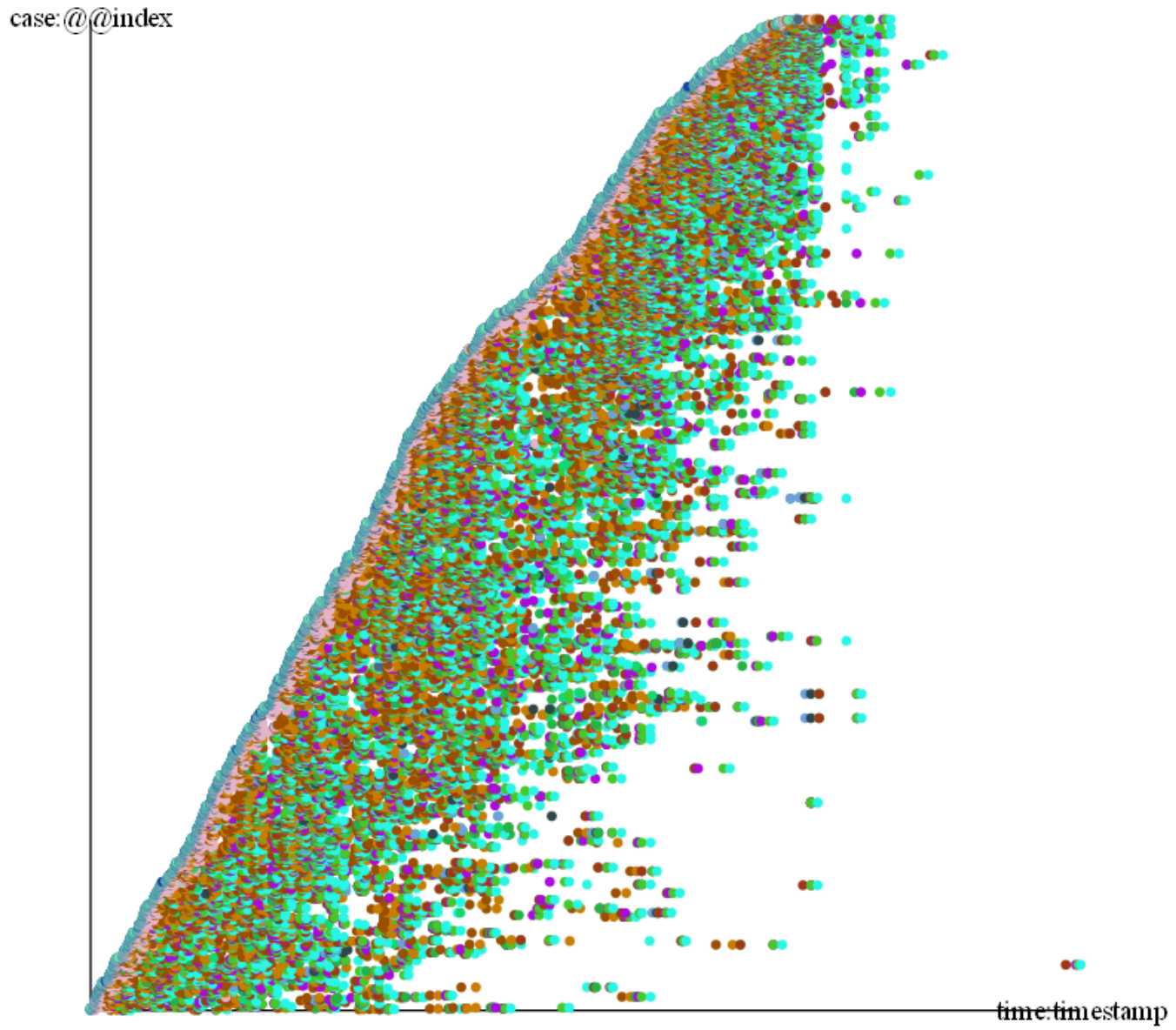
```
Number of traces: 2106
```

Comparing Conforming & Non-Conforming Traces

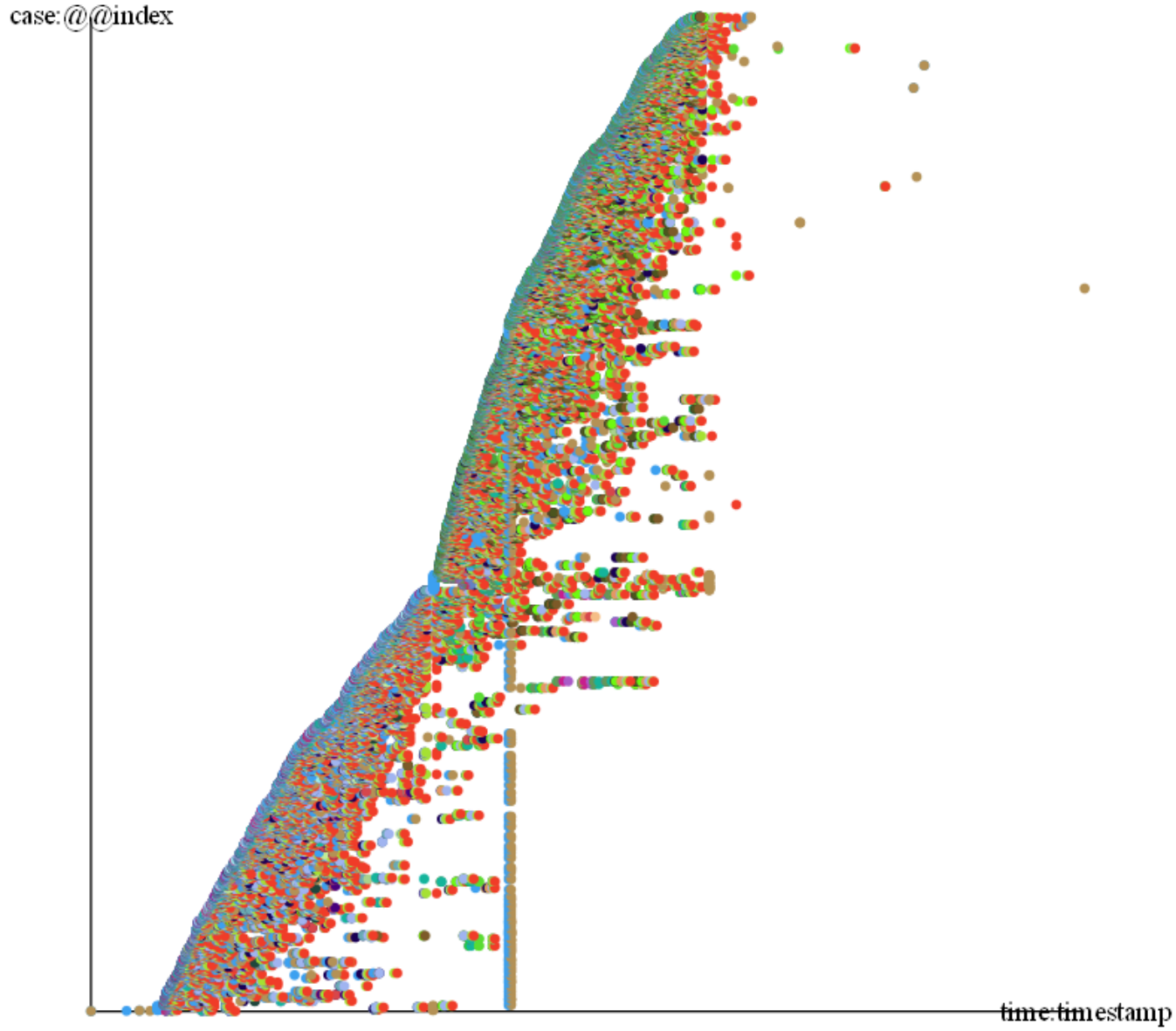
Dotted Charts

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)
```

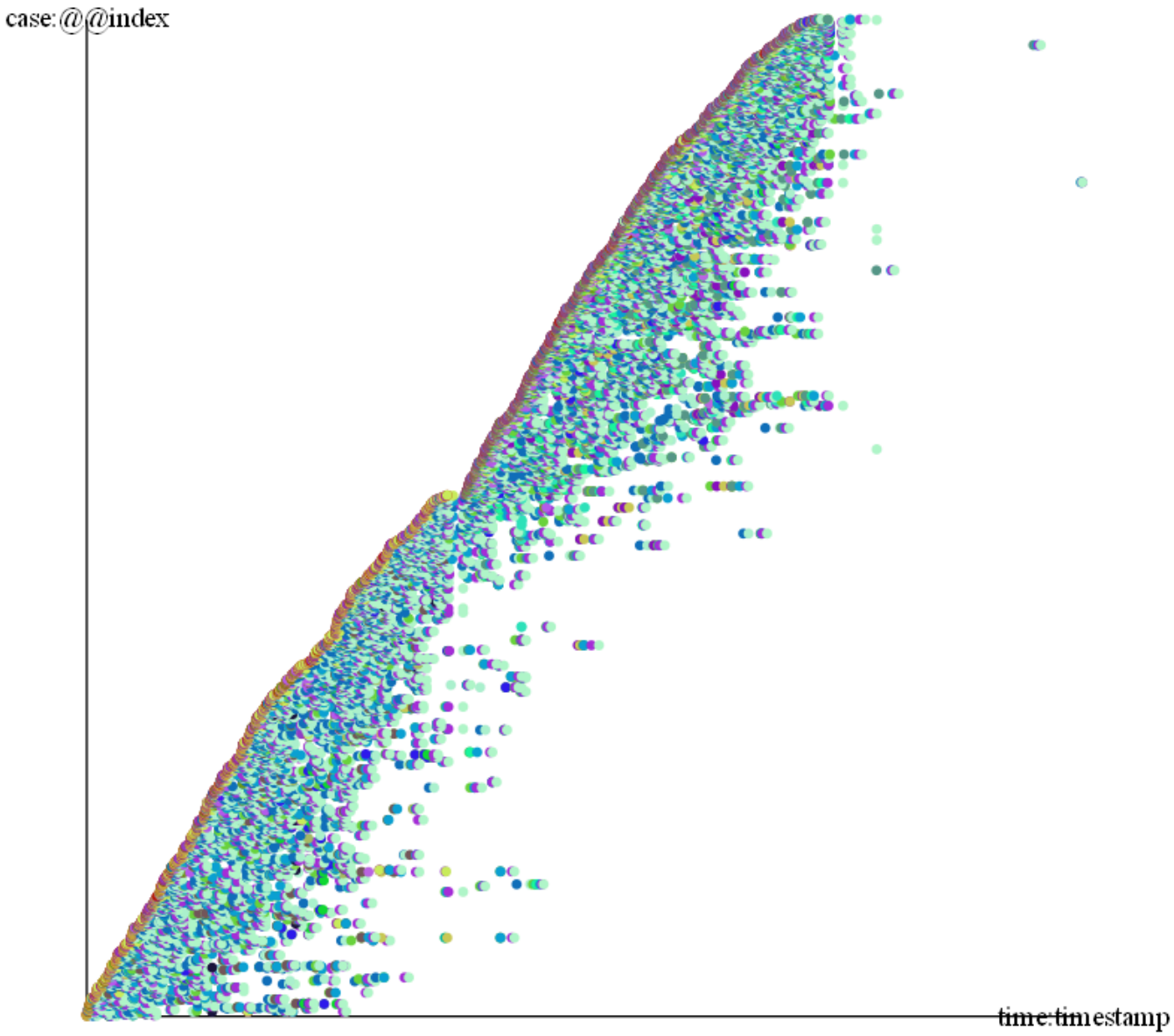
```
Conforming Traces
```




```
Non-Conforming Traces
```



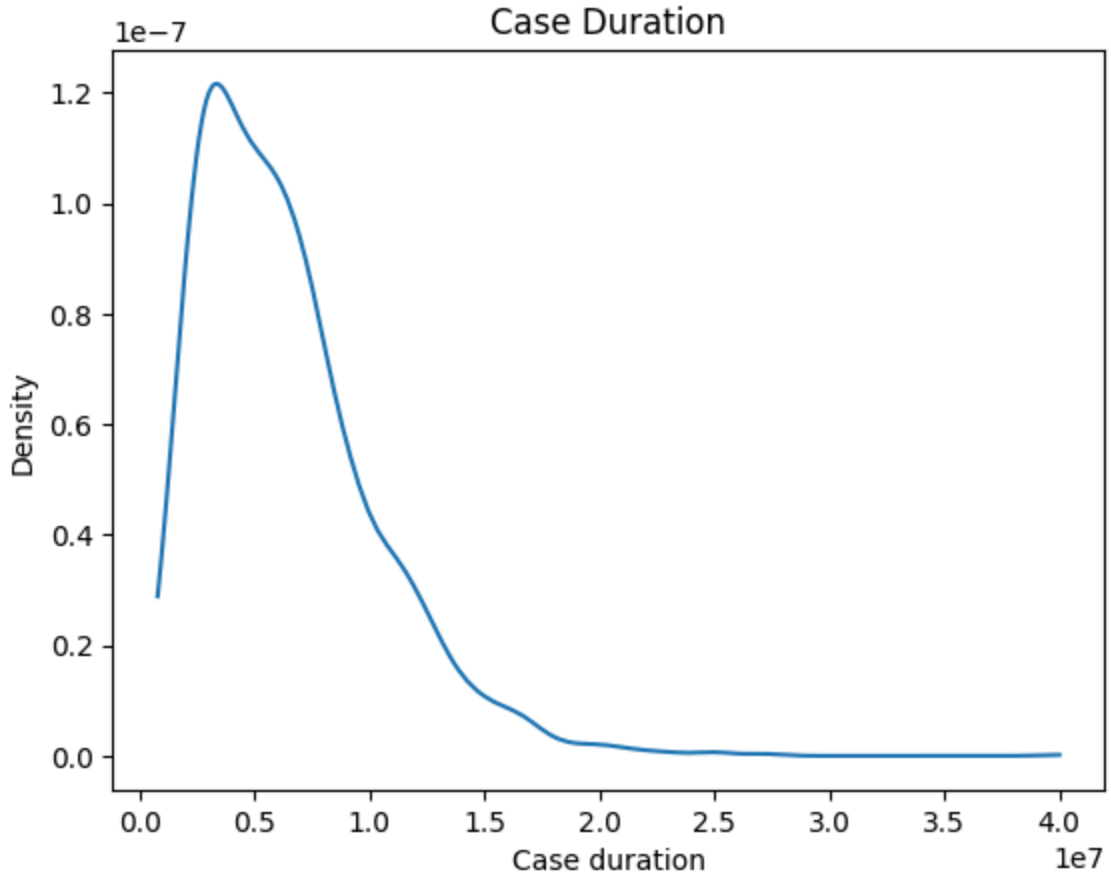
Filtered Non-Conforming Traces

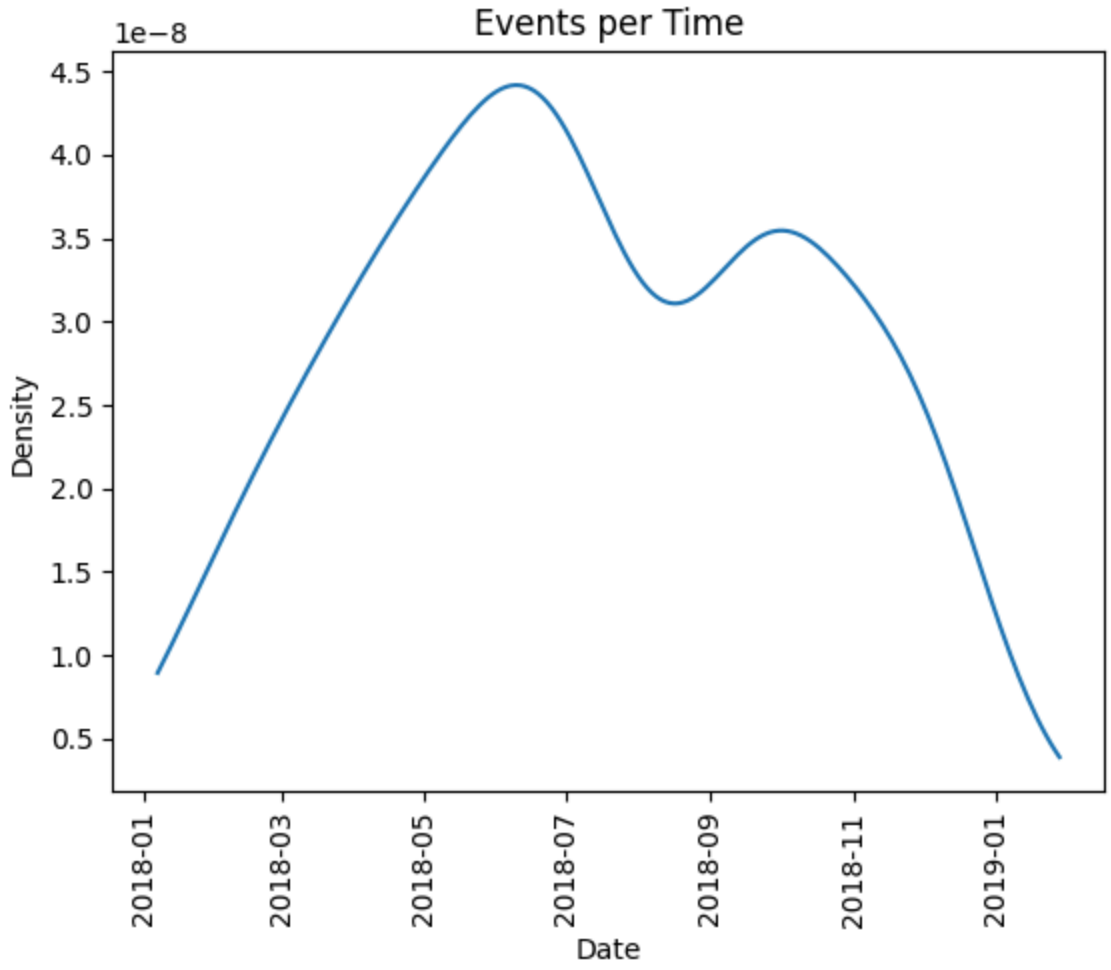


Trace Duration and Density of Events per Time

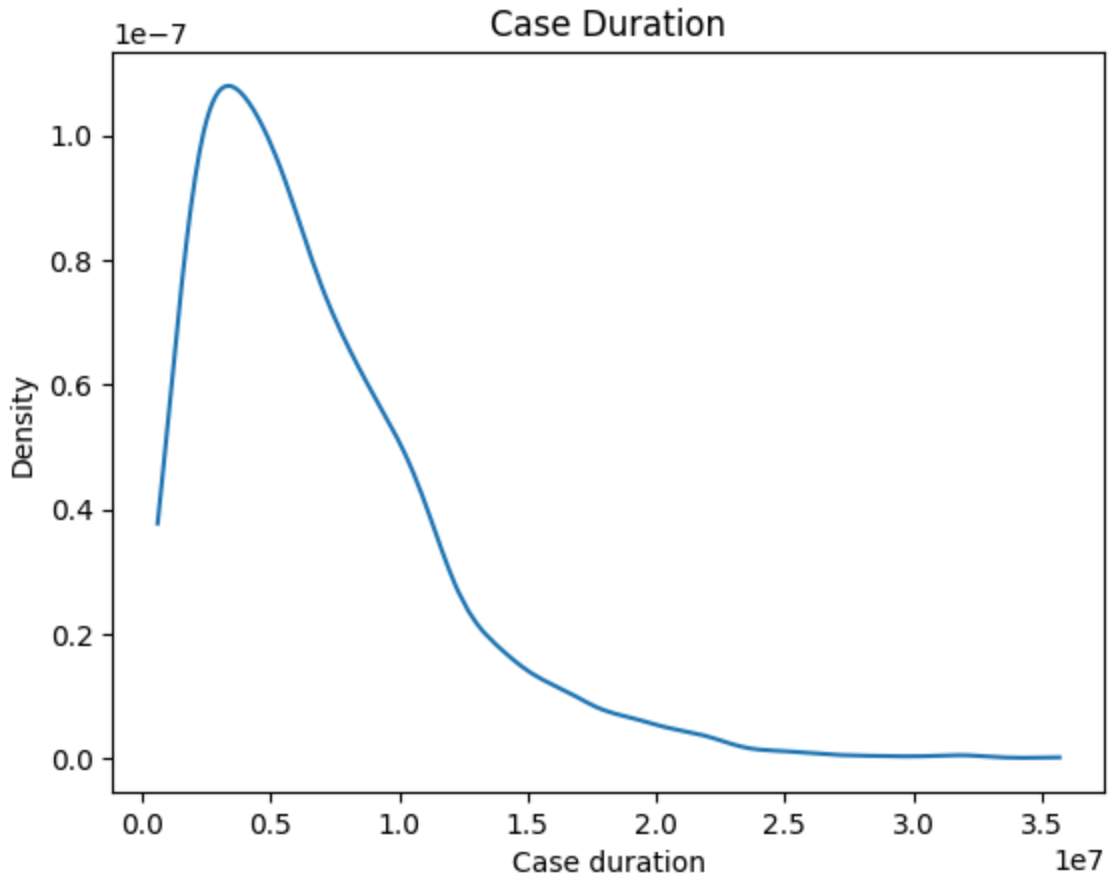
In [12]:
```
print("Conforming Traces")
pm4py.view_case_duration_graph(conforming_log_df)
pm4py.view_events_per_time_graph(conforming_log_df)
print("Filtered Non-Conforming Traces")
pm4py.view_case_duration_graph(filtered_non_conforming_log_df)
pm4py.view_events_per_time_graph(filtered_non_conforming_log_df)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

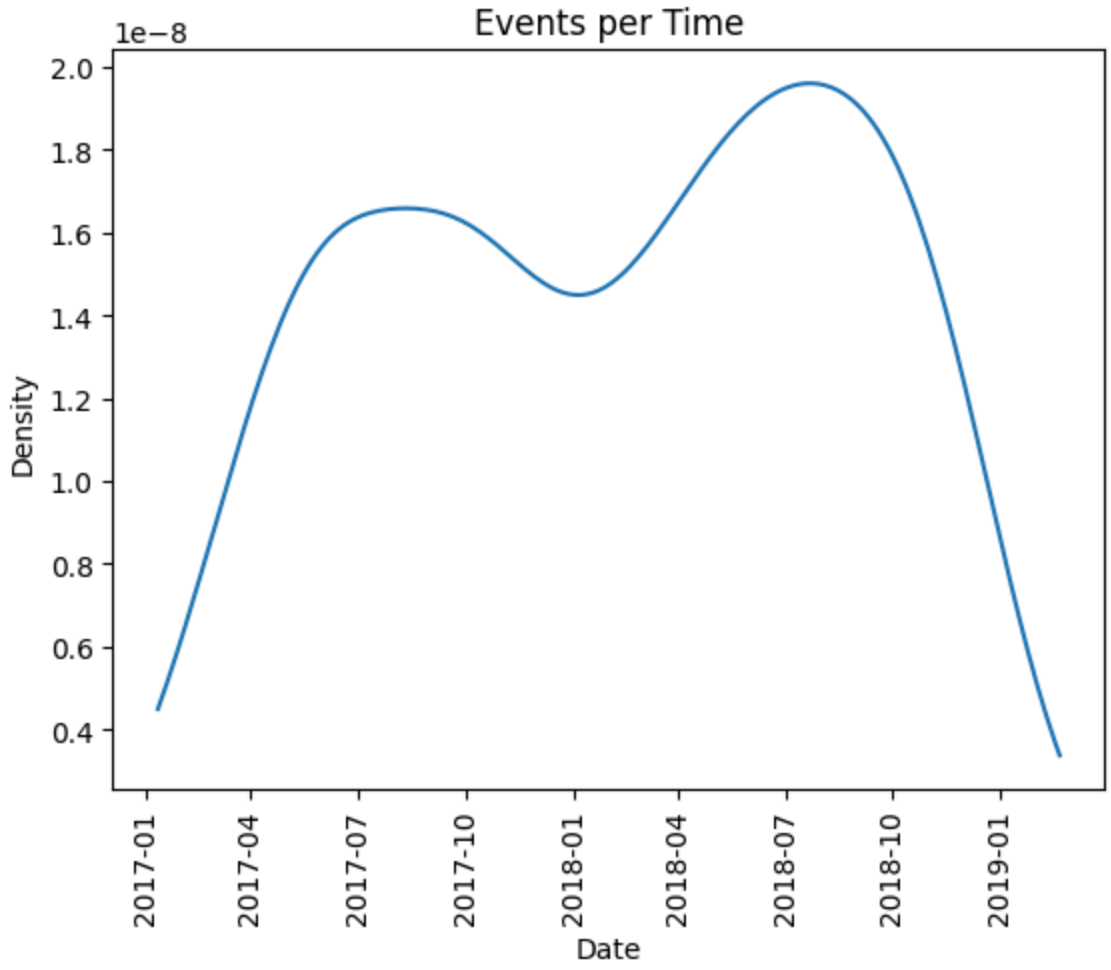
```
Conforming Traces
```





Filtered Non-Conforming Traces





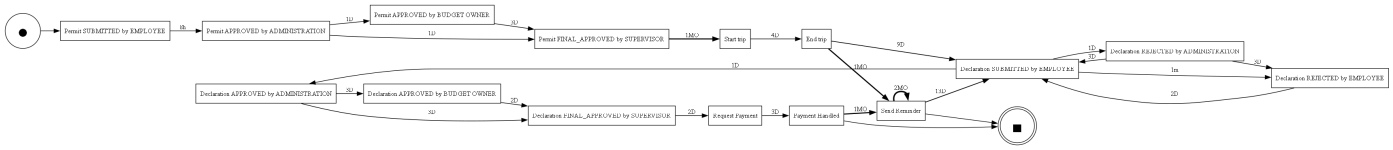
Aggregate Performance

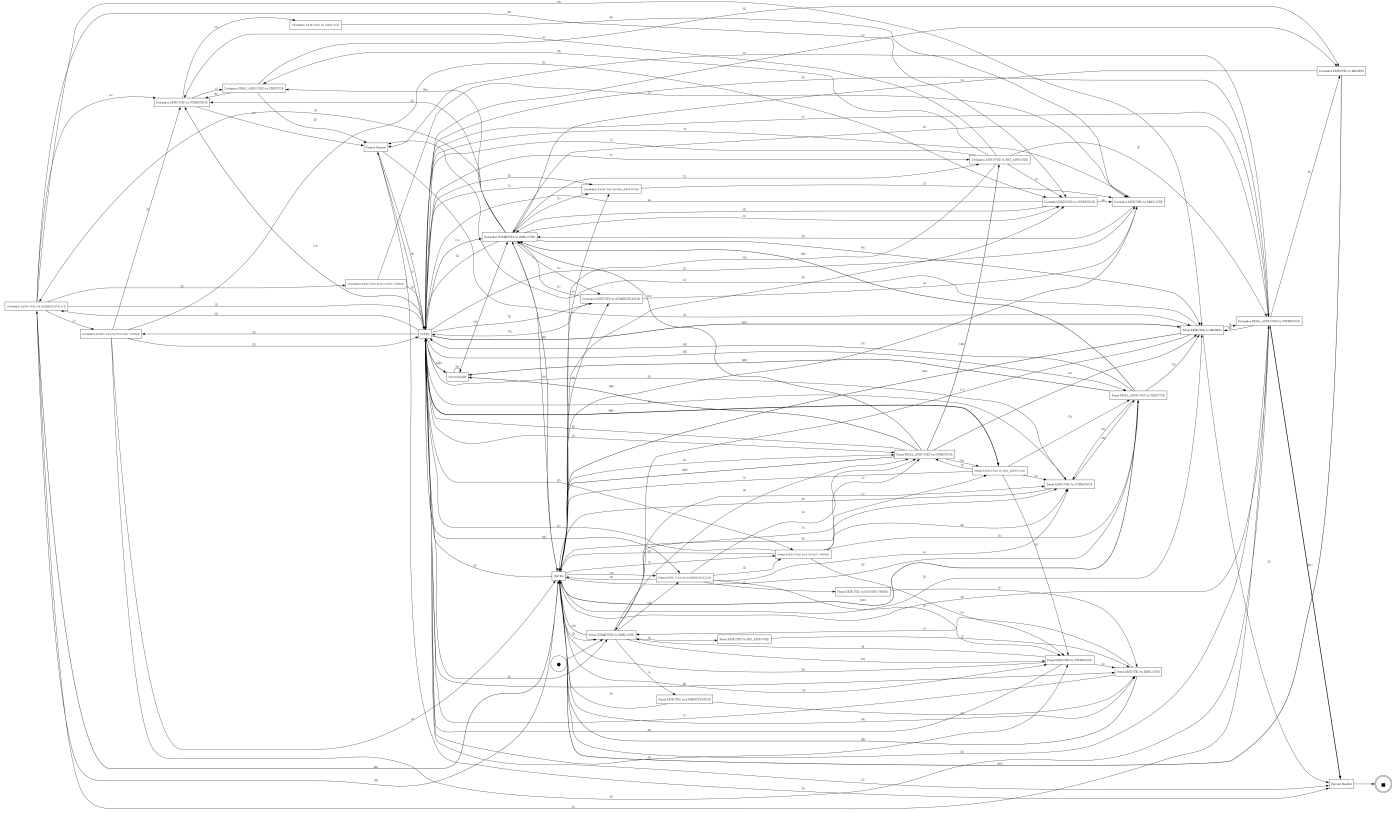
In [25]:
```
def compare_logs_performance(conforming_log_df, non_conforming_log_df ):
    avg_length_conforming = conforming_log_df.groupby("case:concept:name").size().mean()
    avg_length_non_conforming = non_conforming_log_df.groupby("case:concept:name").size().mean()

    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

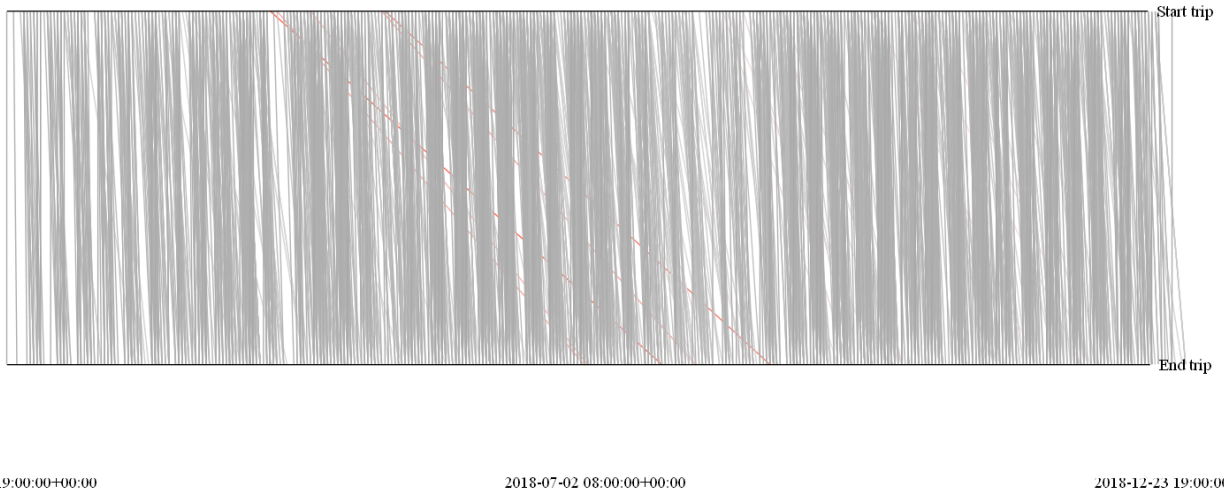
In [29]:
```
performance_conforming_dfg, start_c_activities, end_c_activities = pm4py.discover_performance_df
performance_filtered_non_conforming_dfg, start_fnc_activities, end_fnc_activities = pm4py.discov
pm4py.view_performance_dfg(performance_conforming_dfg, start_c_activities, end_c_activities, for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



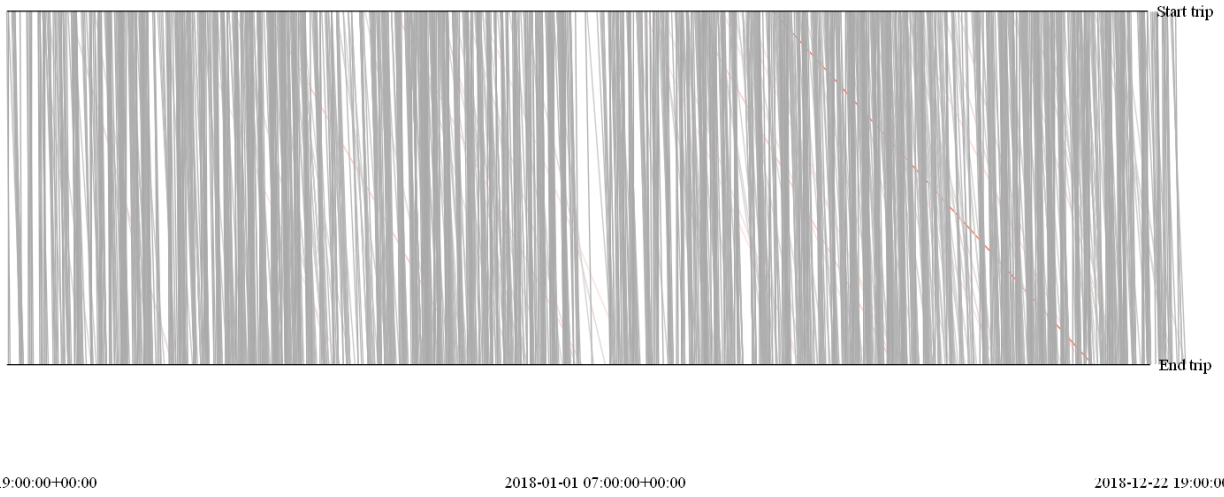


In [32]:
```
pm4py.view_performance_spectrum(conforming_log_df, ['Start trip', 'End trip'], format='png', act
pm4py.view_performance_spectrum(filtered_non_conforming_log_df, ['Start trip', 'End trip'], form
```

```
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
```

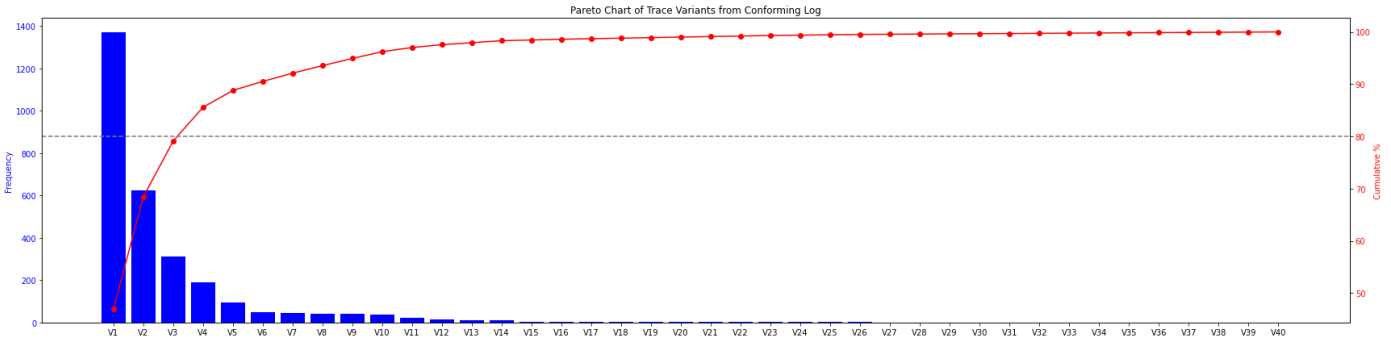
In [15]:
```
variants_conforming_log = get_n_print_variants(conforming_log_df, log_name="Conforming Log")
variants_filtered_non_conforming_log = get_n_print_variants(filtered_non_conforming_log_df, log_
```

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

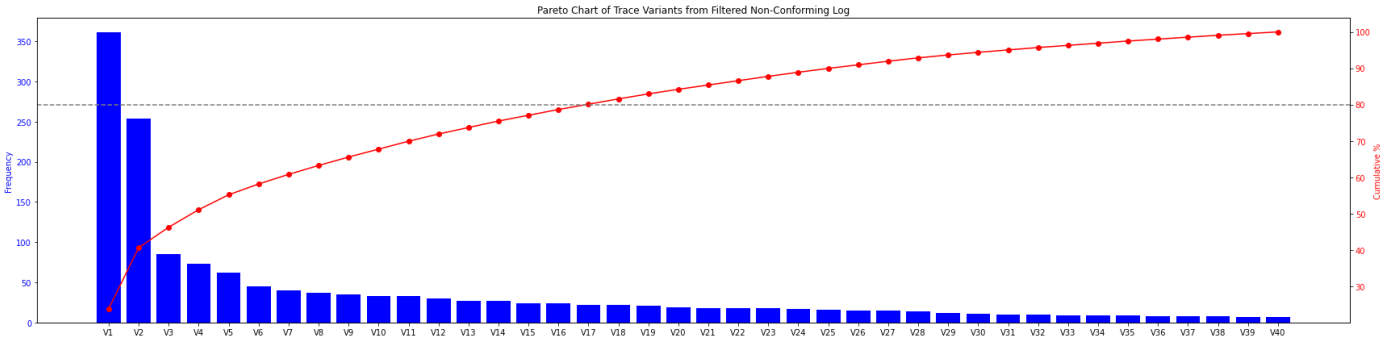
In [16]:
```
def top_N_variants_Pareto_chartd(variants, Top_N, log_name):

    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

Dependencies

In [1]:
```
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 [2]:
```
non_conforming_log_df = pd.read_pickle("filtered-non-conforminlog.pkl")
```

Normative Model Import

In [ ]:
```
file_path = r""
normative_petri_net = pm4py.read_pnml(file_path)
```

Feature Extraction from Log

In [4]:
```
features_df = pm4py.extract_features_dataframe(
    non_conforming_log_df,
    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", "Adjus
    str_ev_attr=['org:role'],
    include_case_id=True
)
```

In [5]:
```
trace_attributes_df = non_conforming_log_df.groupby("case:concept:name").agg({
    "case:Permit BudgetNumber": "first",
    "case:Permit OrganizationalEntity": "first",
    "case:Permit ProjectNumber": "first",
    "case:BudgetNumber": "first"
}).reset_index()

features_df["case:concept:name"] = trace_attributes_df["case:concept:name"]
features_tr_att_df = pd.merge(features_df, trace_attributes_df, on="case:concept:name", how="lef
```

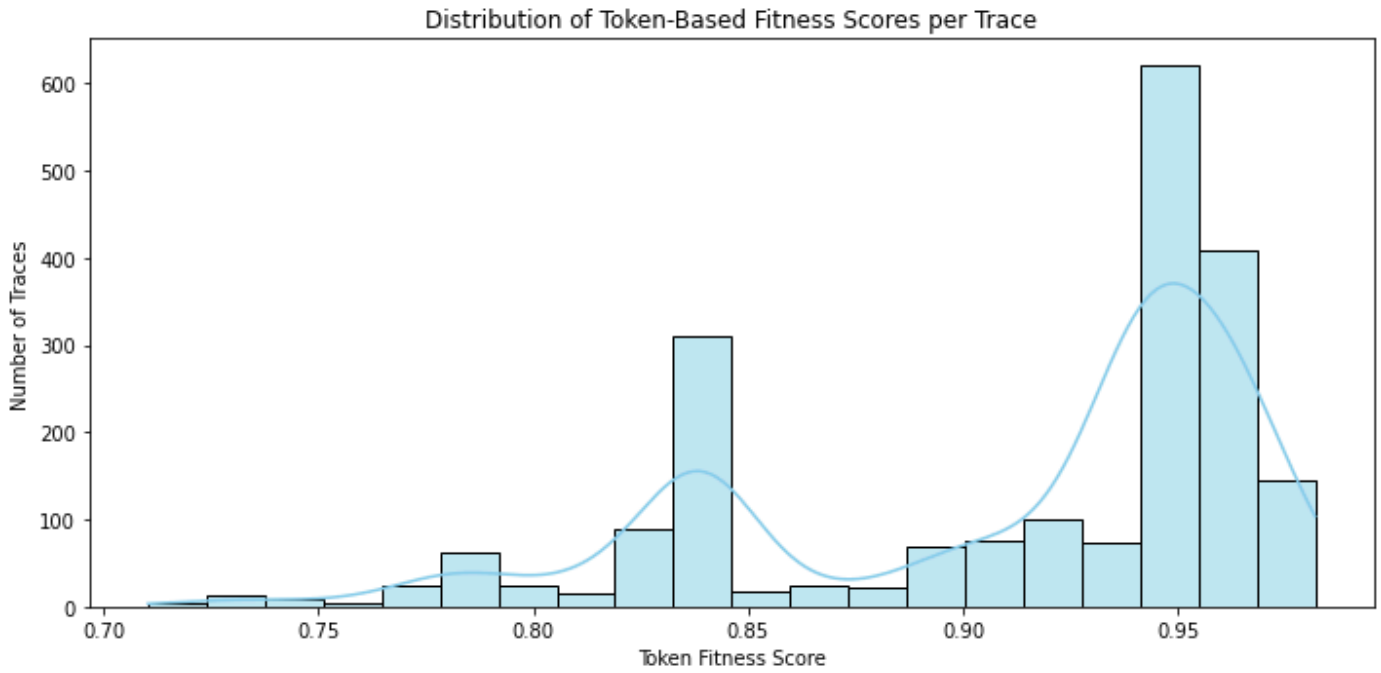
In [6]:
```
frequency_table = non_conforming_log_df.groupby(["case:concept:name", "concept:name"]).size().un
frequency_table = frequency_table.reset_index()

features_freq_df = pd.merge(features_tr_att_df, frequency_table, on="case:concept:name", how="le
```

Performance Metrics

Fitness

In [8]:
```
from pm4py.objects.conversion.log import converter as log_converter

fitness_scores = []

for case_id in features_freq_df["case:concept:name"]:
    trace_df = non_conforming_log_df[non_conforming_log_df["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 = features_freq_df.merge(fitness_df, on="case:concept:name", how="left")
```

In [9]:
```
plt.figure(figsize=(10, 5))
sns.histplot(features_fit_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 [10]:
```
trace_duration_df = non_conforming_log_df.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_df["

trace_event_count_df = non_conforming_log_df.groupby("case:concept:name").size().reset_index(nam

features_duration_df = features_fit_df.merge(trace_duration_df[["case:concept:name", "trace_dura
features_ev_count_df = features_duration_df.merge(trace_event_count_df, on="case:concept:name",
```

In [11]:
```
features_ev_count_df.head()
```

Out[11]:

	case:concept:name	case:AdjustedAmount	case:OriginalAmount	org:role_EMPLOYEE	org:role_ADMINISTRATION
0	declaration 10069	71.195831	71.195831	1.0	1.0
1	declaration 10089	3146.344971	3146.344971	1.0	1.0
2	declaration 10455	586.435608	586.435608	1.0	1.0
3	declaration 10472	679.303223	679.303223	1.0	1.0
4	declaration 10552	997.438721	997.438721	1.0	1.0

5 rows × 53 columns