sample_test

October 17, 2023

1 Sample Test

1.1 Framework 1: Pandas

[1]:
```
import pandas as pd
```

[2]:
```
df_1 = pd.read_csv('dataset1.csv')
df_1.head()
```

[2]:

	invoice_id	legal_entity	counter_party	rating	status	value
0	1	L1	C1	1	ARAP	10
1	2	L2	C2	2	ARAP	20
2	3	L3	C3	4	ACCR	30
3	4	L1	C4	6	ARAP	40
4	5	L2	C5	4	ACCR	50

[3]:
```
df_2 = pd.read_csv('dataset2.csv')
df_2.head()
```

[3]:

	counter_party	tier
0	C1	1
1	C2	2
2	C3	3
3	C4	4
4	C5	5

1.1.1 Join dataset1 with dataset2 and get tier

[4]:
```
df = df_1.merge(df_2, on='counter_party')
df.head(n=10)
```

[4]:

	invoice_id	legal_entity	counter_party	rating	status	value	tier
0	1	L1	C1	1	ARAP	10	1
1	7	L1	C1	2	ARAP	10	1
2	13	L1	C1	3	ARAP	20	1
3	2	L2	C2	2	ARAP	20	2
4	8	L2	C2	3	ACCR	40	2
5	3	L3	C3	4	ACCR	30	3

```
6           9          L3          C3       3   ACCR     80     3
7          14          L2          C3       2   ACCR     52     3
8          15          L3          C3       4   ACCR     35     3
9          16          L1          C3       6   ARAP      5     3
```

1.1.2 Generate below output file

legal_entity, counterparty, tier, max(rating by counterparty), sum(value where status=ARAP), sum(value where status=ACCR)

[5]:
```
df_groups = df.groupby(["legal_entity", "counter_party", "tier"])

# Find max rating
g1 = df_groups.agg({"rating": "max"}).rename(columns={"rating": "max(rating by␣
 ↪counterparty)"})

# Find sum of values where status=Filter
g2 = df_groups.agg({"value": lambda x: x[df["status"] == "ARAP"].sum()}).rename(
    columns={"value": "sum(value where status=ARAP)"}
)
g3 = df_groups.agg({"value": lambda x: x[df["status"] == "ACCR"].sum()}).rename(
    columns={"value": "sum(value where status=ACCR)"}
)

# Merge aggregations
result_df = pd.concat(objs=[g1, g2, g3], axis=1).reset_index()

result_df.to_csv('result_dataset_pandas.csv', index=False)
result_df.head()
```

[5]:
```
  legal_entity counter_party  tier  max(rating by counterparty)  \
0           L1            C1     1                            3
1           L1            C3     3                            6
2           L1            C4     4                            6
3           L2            C2     2                            3
4           L2            C3     3                            2

   sum(value where status=ARAP)  sum(value where status=ACCR)
0                            40                             0
1                             5                             0
2                            40                           100
3                            20                            40
4                             0                            52
```

1.1.3 Also create new record to add total for each of legal entity, counterparty & tier.

[6]:
```
result_df['value_total'] = result_df['sum(value where status=ARAP)'] +␣
↪result_df['sum(value where status=ACCR)']
result_df.head(n=10)
```

[6]:

	legal_entity	counter_party	tier	max(rating by counterparty)	sum(value where status=ARAP)	sum(value where status=ACCR)	value_total
0	L1	C1	1	3	40	0	40
1	L1	C3	3	6	5	0	5
2	L1	C4	4	6	40	100	140
3	L2	C2	2	3	20	40	60
4	L2	C3	3	2	0	52	52
5	L2	C5	5	6	1000	115	1115
6	L3	C3	3	4	0	145	145
7	L3	C6	6	6	145	60	205

1.2 Framework 2: Apache Beam

[7]:
```
import apache_beam as beam
from apache_beam.dataframe import convert
import pandas as pd
import typing
```

1.2.1 Schemas and functions

[8]:
```
class InvoicesJoined(typing.NamedTuple):
  invoice_id: int
  legal_entity: str
  counter_party: str
  rating: int
  status: str
  value: int
  tier: int

class InvoicesJoinedAggregated(typing.NamedTuple):
  legal_entity: str
  counter_party: str
```

```
  tier: int
  max_rating_by_counterparty: int
  sum_value_where_status_is_ARAP: int
  sum_value_where_status_is_ACCR: int
  value_sum: int

def join_counter_party(e):
    _, mapping = e
    d1, d2 = mapping["dataset1"], mapping["dataset2"]
    return [{**v, **d2[0]} for v in d1]
```

1.2.2 Pipeline

[9]:
```
with beam.Pipeline() as pipeline:
    pcollection1 = (
        pipeline
        | "Dataset1" >> beam.Create(["dataset1.csv"])
        | "Dataset1 read" >> beam.Map(pd.read_csv)
        | "Dataset1 to_dict" >> beam.FlatMap(lambda df: df.to_dict("records"))
    )
    pcollection2 = (
        pipeline
        | "Dataset2" >> beam.Create(["dataset2.csv"])
        | "Dataset2 read" >> beam.Map(pd.read_csv)
        | "Dataset2 to_dict" >> beam.FlatMap(lambda df: df.to_dict("records"))
    )

    # Create kv pairs for dataset grouping
    pcol1 = pcollection1 | "d1 key" >> beam.Map(lambda x: (x["counter_party"],␣
 ↪x))
    pcol2 = pcollection2 | "d2 key" >> beam.Map(lambda x: (x["counter_party"],␣
 ↪x))

    # Join datasets
    joined_dicts = (
        {"dataset1": pcol1, "dataset2": pcol2}
        | beam.CoGroupByKey()
        | beam.FlatMap(join_counter_party)
        | beam.Map(lambda e: beam.Row(**e)).with_input_types(InvoicesJoined)
    )
    joined_dicts | "Print joined datasets" >> beam.Map(lambda x: print(x))

    # Perform aggregations
    aggregated_rows = joined_dicts | beam.GroupBy(
        "legal_entity", "counter_party", "tier"
    ).aggregate_field(
        "rating",
```

```
        max,
        "max_rating_by_counterparty",
    ).aggregate_field(
        lambda x: x.value if x.status == "ARAP" else 0,
        sum,
        "sum_value_where_status_is_ARAP",
    ).aggregate_field(
        lambda x: x.value if x.status == "ACCR" else 0,
        sum,
        "sum_value_where_status_is_ACCR",
    ).aggregate_field(
        lambda x: x.value if x.status in ["ARAP", "ACCR"] else 0,
        sum,
        "value_sum",
    )

    # Apply output schema to aggregated rows for dataframe conversion
    aggregated_rows_typed = aggregated_rows | beam.Map(
        lambda x: x._asdict()
    ).with_output_types(InvoicesJoinedAggregated)
    aggregated_rows_typed | beam.Map(lambda x: print(x))

    # Save results as dataframe
    df = convert.to_dataframe(aggregated_rows_typed)
    df.to_csv("./result_dataset_beam.csv", index=False)
```

```
Row(invoice_id=1, legal_entity='L1', counter_party='C1', rating=1,
status='ARAP', value=10, tier=1)
Row(invoice_id=7, legal_entity='L1', counter_party='C1', rating=2,
status='ARAP', value=10, tier=1)
Row(invoice_id=13, legal_entity='L1', counter_party='C1', rating=3,
status='ARAP', value=20, tier=1)
Row(invoice_id=2, legal_entity='L2', counter_party='C2', rating=2,
status='ARAP', value=20, tier=2)
Row(invoice_id=8, legal_entity='L2', counter_party='C2', rating=3,
status='ACCR', value=40, tier=2)
Row(invoice_id=3, legal_entity='L3', counter_party='C3', rating=4,
status='ACCR', value=30, tier=3)
Row(invoice_id=9, legal_entity='L3', counter_party='C3', rating=3,
status='ACCR', value=80, tier=3)
Row(invoice_id=14, legal_entity='L2', counter_party='C3', rating=2,
status='ACCR', value=52, tier=3)
Row(invoice_id=15, legal_entity='L3', counter_party='C3', rating=4,
status='ACCR', value=35, tier=3)
Row(invoice_id=16, legal_entity='L1', counter_party='C3', rating=6,
status='ARAP', value=5, tier=3)
Row(invoice_id=4, legal_entity='L1', counter_party='C4', rating=6,
```

```
status='ARAP', value=40, tier=4)
Row(invoice_id=10, legal_entity='L1', counter_party='C4', rating=5,
status='ACCR', value=100, tier=4)
Row(invoice_id=5, legal_entity='L2', counter_party='C5', rating=4,
status='ACCR', value=50, tier=5)
Row(invoice_id=11, legal_entity='L2', counter_party='C5', rating=6,
status='ARAP', value=1000, tier=5)
Row(invoice_id=17, legal_entity='L2', counter_party='C5', rating=3,
status='ACCR', value=65, tier=5)
Row(invoice_id=6, legal_entity='L3', counter_party='C6', rating=6,
status='ACCR', value=60, tier=6)
Row(invoice_id=12, legal_entity='L3', counter_party='C6', rating=4,
status='ARAP', value=80, tier=6)
Row(invoice_id=18, legal_entity='L3', counter_party='C6', rating=5,
status='ARAP', value=65, tier=6)
{'legal_entity': 'L1', 'counter_party': 'C1', 'tier': 1,
'max_rating_by_counterparty': 3, 'sum_value_where_status_is_ARAP': 40,
'sum_value_where_status_is_ACCR': 0, 'value_sum': 40}
{'legal_entity': 'L2', 'counter_party': 'C2', 'tier': 2,
'max_rating_by_counterparty': 3, 'sum_value_where_status_is_ARAP': 20,
'sum_value_where_status_is_ACCR': 40, 'value_sum': 60}
{'legal_entity': 'L3', 'counter_party': 'C3', 'tier': 3,
'max_rating_by_counterparty': 4, 'sum_value_where_status_is_ARAP': 0,
'sum_value_where_status_is_ACCR': 145, 'value_sum': 145}
{'legal_entity': 'L2', 'counter_party': 'C3', 'tier': 3,
'max_rating_by_counterparty': 2, 'sum_value_where_status_is_ARAP': 0,
'sum_value_where_status_is_ACCR': 52, 'value_sum': 52}
{'legal_entity': 'L1', 'counter_party': 'C3', 'tier': 3,
'max_rating_by_counterparty': 6, 'sum_value_where_status_is_ARAP': 5,
'sum_value_where_status_is_ACCR': 0, 'value_sum': 5}
{'legal_entity': 'L1', 'counter_party': 'C4', 'tier': 4,
'max_rating_by_counterparty': 6, 'sum_value_where_status_is_ARAP': 40,
'sum_value_where_status_is_ACCR': 100, 'value_sum': 140}
{'legal_entity': 'L2', 'counter_party': 'C5', 'tier': 5,
'max_rating_by_counterparty': 6, 'sum_value_where_status_is_ARAP': 1000,
'sum_value_where_status_is_ACCR': 115, 'value_sum': 1115}
{'legal_entity': 'L3', 'counter_party': 'C6', 'tier': 6,
'max_rating_by_counterparty': 6, 'sum_value_where_status_is_ARAP': 145,
'sum_value_where_status_is_ACCR': 60, 'value_sum': 205}
```