7506-2021 Parcialito de Pandas

Román Vázquez Lareu

TOTAL POINTS

90 / 100

QUESTION 1

1 Punto a 50 / 50

✓ **- 0 pts** **Ok**

La respuesta dada no debería depender del dataset armado para probar.

El formato de respuesta era solamente con los consumers (por ejemplo, mostrando solamente su id o su target). En este caso, los ids se muestran como índices en el formato de respuesta dado.

QUESTION 2

2 Punto b 40 / 50

✓ **- 10 pts** **Genera una columna de mas para sumar en el groupby, pudiendo usar un count o size.**

1. No es necesario agregar una columna y sumar, se puede usar count sobre cualquier columna
2. Codigo incompleto no se puede leer.

```
import pandas as pd
import numpy as np


event_log = pd.DataFrame({"event_id":["A01","A02","A03","A04","A05","A06","A05","A06"],
                     "event_type_id":["C01","C02","C03","C02","C03","C02","C03","C02"],
                     "event_status":["ERROR","DELAYED","ERROR","CAPTURED","CAPTURED","DELAYED","ERROR","ERROR"],
                     "country_id":["BR","ARG","BR","BR","USA","BR","USA","BR"],
                     "event_producer_id":["B01","B02","A03","B01","B01","B03","B01","B03"],
                     "event_date":["2020-9-13","2020-9-12","2020-9-13","2020-9-12","2020-9-15","2020-9-16","2020-9-15","2020-9-16"]})

event_types = pd.DataFrame({"event_type_id":["C01","C02","C03","C04","C05","C06","C07","C08"],
                     "event_type_name":["A","B","B","B","B","B","B","B"],
                     "event_consumer_id":["D01","D02","D03","D04","D05","D06","D07","D08"],
                     "event_consumer_target":["erp","dynamodb","salesforce","dynamodb","salesforce","erp","erp","erp"]
                     })
```

```
event_log
```

	event_id	event_type_id	event_status	country_id	event_producer_id	event_date
0	A01	C01	ERROR	BR	B01	2020-9-13
1	A02	C02	DELAYED	ARG	B02	2020-9-12
2	A03	C03	ERROR	BR	A03	2020-9-13
3	A04	C02	CAPTURED	BR	B01	2020-9-12
4	A05	C03	CAPTURED	USA	B01	2020-9-15
5	A06	C02	DELAYED	BR	B03	2020-9-16
6	A05	C03	ERROR	USA	B01	2020-9-15
7	A06	C02	ERROR	BR	B03	2020-9-16

```
event_types
```

	event_type_id	event_type_name	event_consumer_id	event_consumer_target
0	C01	A	D01	erp
1	C02	B	D02	dynamodb
2	C03	B	D03	salesforce
3	C04	B	D04	dynamodb
4	C05	B	D05	salesforce
5	C06	B	D06	erp
6	C07	B	D07	erp
7	C08	B	D08	erp

Top 5 de Consumers que han tenido la mayor cantidad de eventos que resultaron en un event_status de ERROR.

```
#me quedo con los eventos que terminaron en ERROR. Quedan los eventos y tipos de eventos asociados a error
event_log_error = event_log[event_log["event_status"]=="ERROR"]
event_log_error = event_log_error[["event_type_id","event_id"]]
event_log_error
```

	event_type_id	event_id
0	C01	A01
2	C03	A03
6	C03	A05
7	C02	A06

```
#lo cruzo con event_types para ver los event_consumer_id
event_log_error_consumer_id = event_log_error.merge(event_types,how="inner")
event_log_error_consumer_id
```

	event_type_id	event_id	event_type_name	event_consumer_id	event_consumer_target
0	C01	A01	A	D01	erp
1	C03	A03	B	D03	salesforce
2	C03	A05	B	D03	salesforce
3	C02	A06	B	D02	dynamodb

```
#cuento las ocurrencias de event_cada consumer_id, por ser dataset de juguete me quedo con los 2 con mas errores
event_log_error_consumer_id["event_consumer_id"].value_counts().nlargest(2)
```

```
D03    2
D02    1
Name: event_consumer_id, dtype: int64
```

De los eventos ocurridos para el country_id: BR indicar la cantidad de eventos totales por cada evento ocurridos por event_consumer_target.

```
#me quedo con los eventos ocurridos en BR
event_log_br = event_log[event_log["country_id"] == "BR"]
event_log_br = event_log_br[["event_id","event_type_id","event_status"]]
event_log_br
```

	event_id	event_type_id	event_status
0	A01	C01	ERROR
2	A03	C03	ERROR
3	A04	C02	CAPTURED
5	A06	C02	DELAYED
7	A06	C02	ERROR

```
#cruzo con event_type para quedarme con los tipos de eventos en BR
event_log_br_type = event_log_br.merge(event_types,how="inner")
event_log_br_type
```

	event_id	event_type_id	event_status	event_type_name	event_consumer_id	event_consum
0	A01	C01	ERROR	A	D01	
1	A03	C03	ERROR	B	D03	
2	A04	C02	CAPTURED	B	D02	
3	A06	C02	DELAYED	B	D02	
4	A06	C02	ERROR	B	D02	

```
#me quedo con las columnas que quiero
event_log_br_type = event_log_br_type[["event_consumer_target","event_status"]]
event_log_br_type
```

	event_consumer_target	event_status
0	erp	ERROR
1	salesforce	ERROR
2	dynamodb	CAPTURED
3	dynamodb	DELAYED
4	dynamodb	ERROR

```
event_log_br_type["auxiliar"] = np.repeat([1],len(event_log_br_type))
event_log_br_type
```

1

	event_consumer_target	event_status	auxiliar
0	erp	ERROR	1
1	salesforce	ERROR	1
2	dynamodb	CAPTURED	1
3	dynamodb	DELAYED	1
4	dynamodb	ERROR	1

1 Punto a **50 / 50**

✓ **- 0 pts** **Ok**

La respuesta dada no debería depender del dataset armado para probar.

El formato de respuesta era solamente con los consumers (por ejemplo, mostrando solamente su id o su target). En este caso, los ids se muestran como índices en el formato de respuesta dado.

	event_type_id	event_id	event_type_name	event_consumer_id	event_consumer_target
0	C01	A01	A	D01	erp
1	C03	A03	B	D03	salesforce
2	C03	A05	B	D03	salesforce
3	C02	A06	B	D02	dynamodb

```
#cuento las ocurrencias de event_cada consumer_id, por ser dataset de juguete me quedo con los 2 con mas errores
event_log_error_consumer_id["event_consumer_id"].value_counts().nlargest(2)
```

```
D03    2
D02    1
Name: event_consumer_id, dtype: int64
```

De los eventos ocurridos para el country_id: BR indicar la cantidad de eventos totales por cada evento ocurridos por event_consumer_target.

```
#me quedo con los eventos ocurridos en BR
event_log_br = event_log[event_log["country_id"] == "BR"]
event_log_br = event_log_br[["event_id","event_type_id","event_status"]]
event_log_br
```

	event_id	event_type_id	event_status
0	A01	C01	ERROR
2	A03	C03	ERROR
3	A04	C02	CAPTURED
5	A06	C02	DELAYED
7	A06	C02	ERROR

```
#cruzo con event_type para quedarme con los tipos de eventos en BR
event_log_br_type = event_log_br.merge(event_types,how="inner")
event_log_br_type
```

	event_id	event_type_id	event_status	event_type_name	event_consumer_id	event_consum
0	A01	C01	ERROR	A	D01	
1	A03	C03	ERROR	B	D03	
2	A04	C02	CAPTURED	B	D02	
3	A06	C02	DELAYED	B	D02	
4	A06	C02	ERROR	B	D02	

```
#me quedo con las columnas que quiero
event_log_br_type = event_log_br_type[["event_consumer_target","event_status"]]
event_log_br_type
```

	event_consumer_target	event_status
0	erp	ERROR
1	salesforce	ERROR
2	dynamodb	CAPTURED
3	dynamodb	DELAYED
4	dynamodb	ERROR

```
event_log_br_type["auxiliar"] = np.repeat([1],len(event_log_br_type))
event_log_br_type
```

1

	event_consumer_target	event_status	auxiliar
0	erp	ERROR	1
1	salesforce	ERROR	1
2	dynamodb	CAPTURED	1
3	dynamodb	DELAYED	1
4	dynamodb	ERROR	1

```
#Hago pivot table donde si cae en la misma celda, suma 1.
event_log_br_type_totales = event_log_br_type.pivot_table(index="event_status",columns="event_consumer_target",values="auxiliar",ag
event_log_br_type_totales.columns.name = None
event_log_br_type_totales.reset_index(inplace=True)

#Si el evento debe quedar como columna y no como indice, la linea de abajo no hace falta
event_log_br_type_totales.set_index("event_status")

event_log_br_type_totales.fillna(0)
```

	event_status	dynamodb	erp	salesforce
0	CAPTURED	1.0	0.0	0.0
1	DELAYED	1.0	0.0	0.0
2	ERROR	1.0	1.0	1.0

✓ 0s completed at 7:55 PM

2 Punto b **40 / 50**

✓ **- 10 pts** **Genera una columna de mas para sumar en el groupby, pudiendo usar un count o size.**

1. No es necesario agregar una columna y sumar, se puede usar count sobre cualquier columna
2. Codigo incompleto no se puede leer.